



**Diseño y programación de software  
multiplataforma  
Ciclo 02 2024**

**Docente: Ing. Alexander Sigüenza**

**“Proyecto de Catedra”**

**Presentado por:**

**David Eduardo Rodríguez Vigil – RV202840**

**El Salvador, 6 de diciembre de 2024**

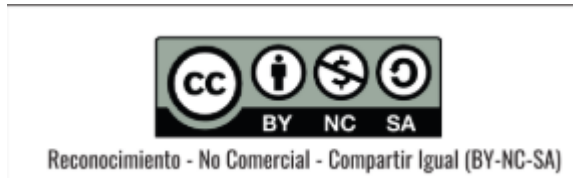
Link del trello:

<https://trello.com/invite/b/67511f1210be0fb69d7c39f0/ATTIf3f68861f8612c0a2ff97388145f5e1321E3880A/catedra-dps>

Link del repositorio de github:

[https://github.com/eduardovigil/Comunidad\\_Eventos](https://github.com/eduardovigil/Comunidad_Eventos)

El tipo de licencias creative common que elegi es la siguiente:



Debido a que esta es la que más se relaciona con el proyecto que estoy realizando con respecto a la comunidad de eventos.

Link del video de YouTube:

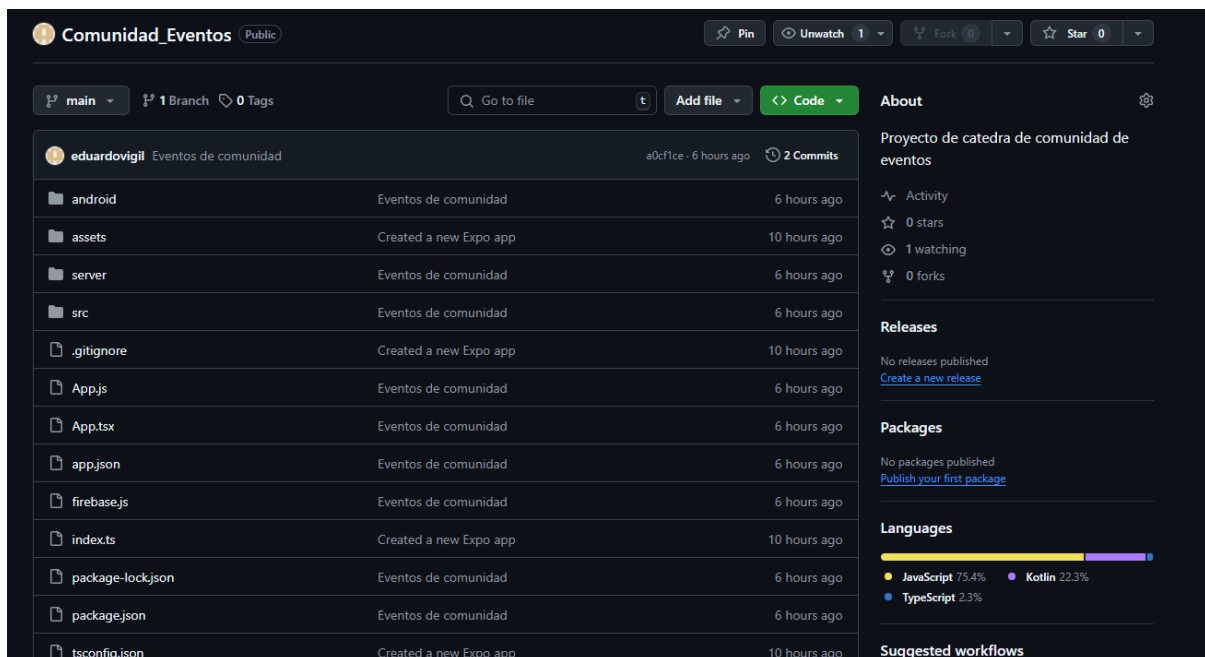
<https://youtu.be/gps9A6uVMas>

Link de los mockups:

<https://drive.google.com/drive/folders/14FxqYnTHI4RgBdzcStZNRfCNbm24WZ0U?usp=sharing>

## Manual de usuario.

Como primer paso se descarga del repositorio github el proyecto de comunidad de eventos.



Después de descargar el proyecto, se tiene que descomprimir y abrir la terminal desde el proyecto al realizar esto y que la terminal se abra se tiene que realizar un npm install para poder instalar todas las dependencias del proyecto que se han utilizado en el proyecto.

```
added 65 packages, and audited 969 packages in 35s

74 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

added 3 packages, and audited 972 packages in 4s

74 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

added 17 packages, and audited 989 packages in 7s

74 packages are looking for funding
  run `npm fund` for details


found 0 vulnerabilities
> Installing 1 SDK 52.0.0 compatible native module and 1 other package using npm
> npm install --save expo-random
npm warn deprecated expo-random@14.0.1: This package is now deprecated in favor of expo-crypto, which provides the same
functionality. To migrate, replace all imports from expo-random with imports from expo-crypto.

added 6 packages, and audited 995 packages in 6s

74 packages are looking for funding
  run `npm fund` for details
```

Continuando pondremos el comando de npx expo start para poner en marcha el proyecto de react-native, en el cual podremos elegir el emulador por preferencia que queramos o utilizarlo desde nuestro dispositivo descargando la aplicación de expo Go.

```
PS C:\Users\Eduardo\Desktop\ComunidadeEventos> npx expo start --clear
Starting project at C:\Users\Eduardo\Desktop\ComunidadeEventos
Starting Metro Bundler
warning: Bundler cache is empty, rebuilding (this may take a minute)
The following packages should be updated for best compatibility with the installed expo version:
  @react-native-async-storage/async-storage@1.24.0 - expected version: 1.23.1
Your project may not work correctly until you install the expected versions of the packages.



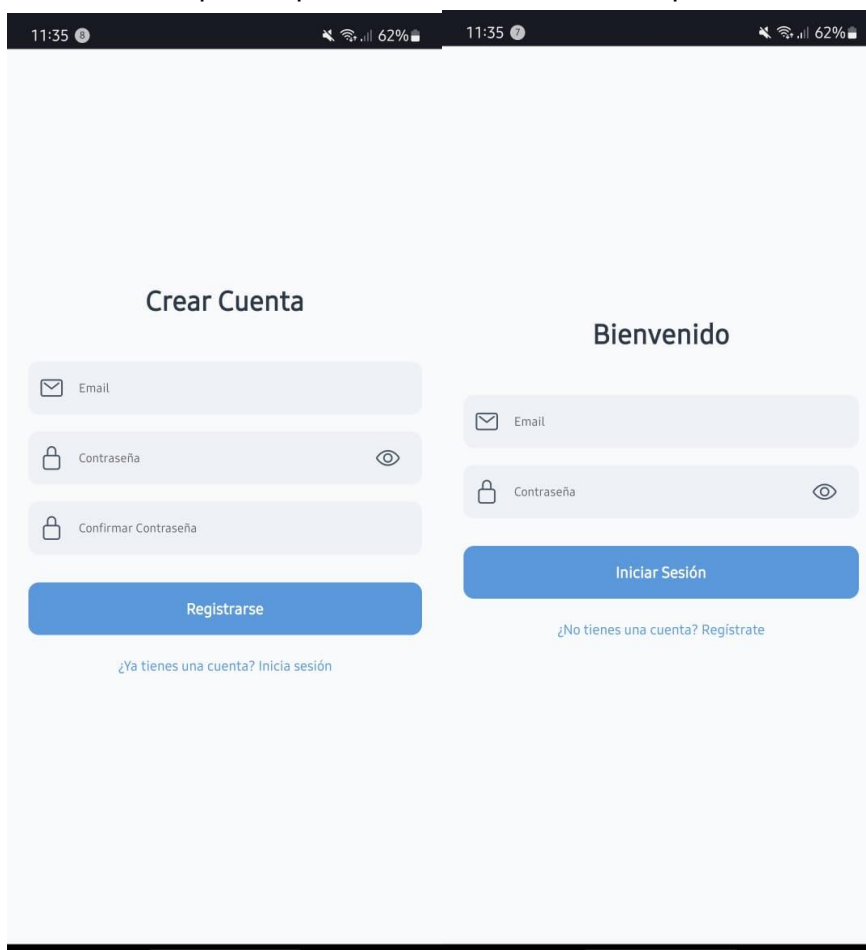
> Metro waiting on exp://192.168.0.5:8081
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

> Using Expo Go
> Press s | switch to development build

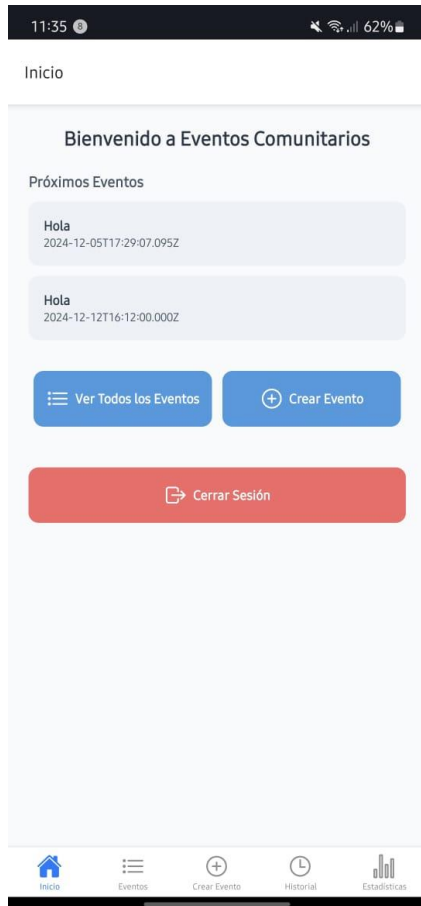
> Press a | open Android
> Press w | open web

> Press j | open debugger
> Press r | reload app
> Press m | toggle menu
```

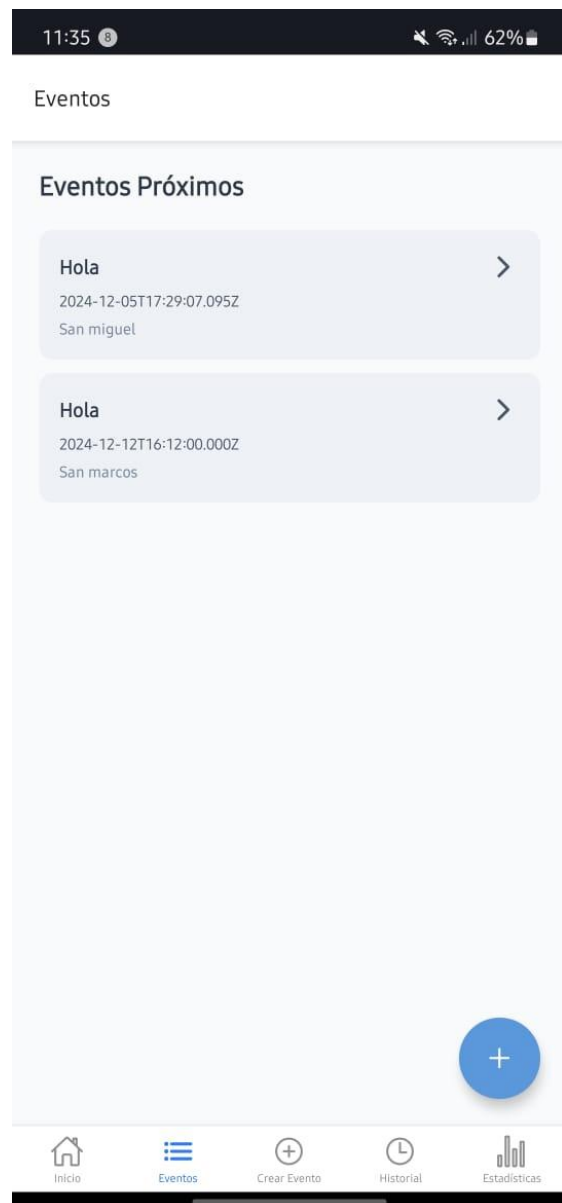
Al momento de activar la aplicación se presentará la pestaña de inicio de sesión y registro en las cuales los usuarios podrán ingresar su correo electrónico como una contraseña para poder entrar en la aplicación de comunidad eventos.



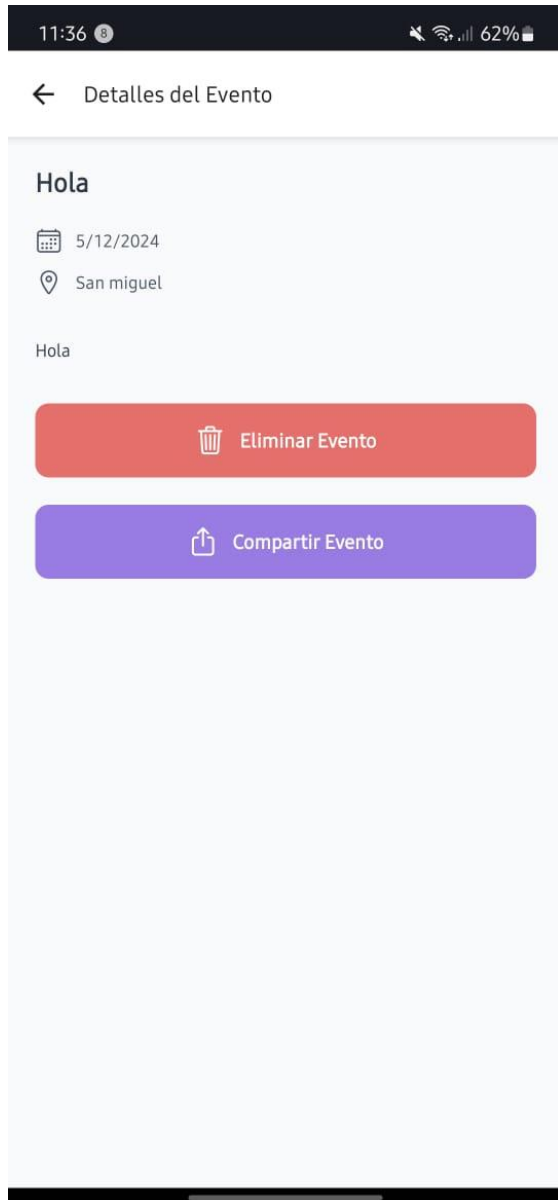
Al momento de crear una cuenta en la aplicación se mostrará la siguiente pantalla de bienvenida a los usuarios donde se les mostrara los eventos próximos como también los botones de crear evento como la lista de eventos que se aproximan además tendremos el menú en la parte inferior de la pantalla donde habrá diferentes opciones relacionadas con los eventos.



Si el usuario presiona el botón de ver todos los eventos se desplegará la siguiente pestaña en la cual se visualizará todos los eventos que se tengan programados próximamente y en el cual si se les presiona desplegará la información de estos eventos.



Si el usuario presiona algún de los eventos ya mostrados en la pantalla se le mostrar la siguiente pantalla con la información del evento y con dos botones de eliminación y compartir el evento además se presenta dos capturas desde si el usuario crea el evento o asístete al evento.



7:18 3

62%

7:18 2

62%

← Detalles del Evento

← Detalles del Evento

fj

5/12/2024

hd

zbzjxj

Editar Evento

Eliminar Evento

Finalizar Evento

Compartir Evento

#### Asistentes

Aún no hay asistentes confirmados.

#### Deja un comentario

Escribe tu comentario aquí



Enviar Comentario

Hola

5/12/2024

Hello

Bfjddjd

Cancelar Asistencia

Compartir Evento

#### Asistentes

AGdAI8fSLXenJw6o62eF7i7N3cM2

#### Comentarios

holacdce





Si el usuario en la pantalla de bienvenida presiona el botón de Crear un evento se le presentara la siguiente pantalla en la cual se necesita que rellene con información que se le pide del evento en el cual al momento de terminar de rellenar se puede presionar el botón de crear evento en el cual creara el evento y se mostrara en la pantalla de ver todos los eventos.

11:35

62%

Crear Evento

Crear Nuevo Evento

Título del evento

5/12/2024

Ubicación

Descripción

Crear Evento

Inicio

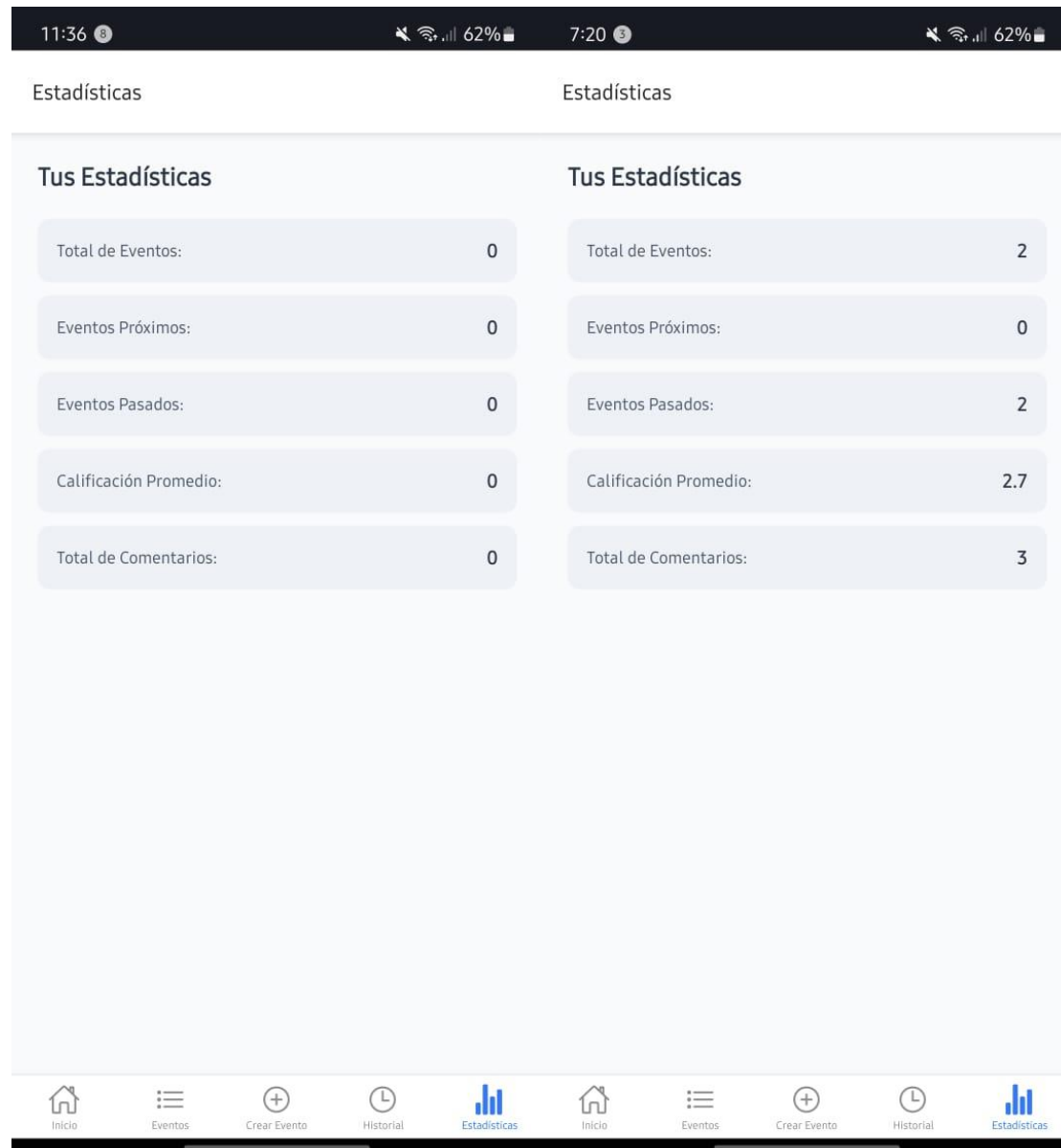
Eventos

Crear Evento

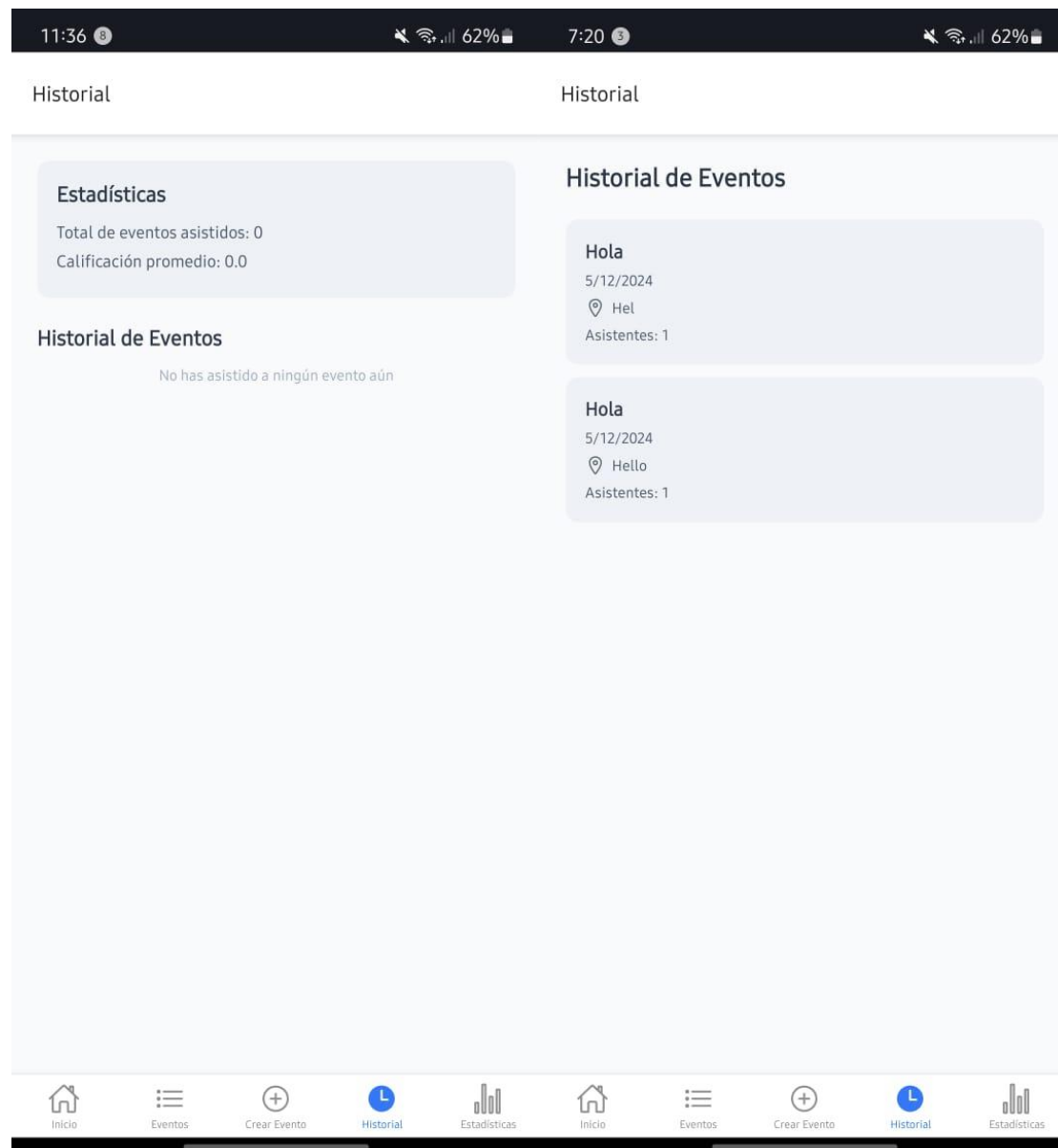
Historial

Estadísticas

Ademas de estas opciones, tenemos la opción de menú inferior en el cual podemos ingresar a las estadísticas y al historial de los eventos que el usuario a asistido, en el cual si ingresa a las estadísticas se le presentara la siguiente pantalla en la cual se mostrar toda la información de los eventos como también comentarios del evento



Si el usuario presiona en el menú la opción de historial se le presentara la siguiente pantalla para ver los eventos pasados como también los comentarios y calificaciones de estos eventos.



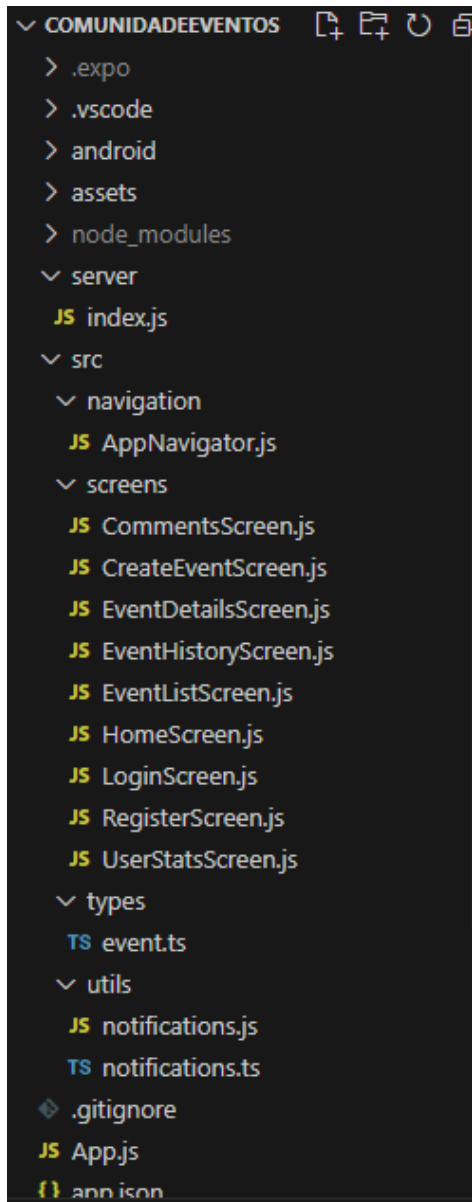
Estructura del código fuente.

Enlace del repositorio:

[https://github.com/eduardovigil/Comunidad\\_Eventos](https://github.com/eduardovigil/Comunidad_Eventos)

Estructura del proyecto:

En esta primera imagen se presenta las diferentes carpetas que se tiene en el proyecto y la estructura que se ha utilizado en este proyecto.



En esta siguiente imagen se presenta la estructura del código de las diferentes pantallas que se utilizan en el proyecto.

App.txs

```
1  import React, { useEffect, useRef } from 'react';
2  import { Platform } from 'react-native';
3  import { SafeAreaProvider } from 'react-native-safe-area-context';
4  import AppNavigator from './src/navigation/AppNavigator';
5  import * as Notifications from 'expo-notifications';
6  import * as Device from 'expo-device';
7
8  Notifications.setNotificationHandler({
9    handleNotification: async () => ({
10      shouldShowAlert: true,
11      shouldPlaySound: true,
12      shouldSetBadge: false,
13    }),
14  });
15
16  async function registerForPushNotificationsAsync() {
17    let token;
18
19    if (Device.isDevice) {
20      const { status: existingStatus } = await Notifications.getPermissionsAsync();
21      let finalStatus = existingStatus;
22      if (existingStatus !== 'granted') {
23        const { status } = await Notifications.requestPermissionsAsync();
24        finalStatus = status;
25      }
26      if (finalStatus !== 'granted') {
27        alert('Failed to get push token for push notification!');
28        return;
29      }
30      token = (await Notifications.getExpoPushTokenAsync()).data;
```

En esta captura se muestra el código del archivo en el cual se realizar las diferentes funciones y los diferentes enlaces entre los archivos como tambien utiliza la navegación que se a realizado para este proyecto.

## LoginScreen.js

```
1  import React, { useState } from 'react';
2  import { View, Text, TextInput, TouchableOpacity, StyleSheet, Alert } from 'react-native';
3  import { signInWithEmailAndPassword } from '@firebase/auth';
4  import { auth } from '../..//firebase';
5  import { Ionicons } from '@expo/vector-icons';
6
7  export default function LoginScreen({ navigation }) {
8    const [email, setEmail] = useState('');
9    const [password, setPassword] = useState('');
10   const [isLoading, setIsLoading] = useState(false);
11   const [showPassword, setShowPassword] = useState(false);
12
13   const handleLogin = async () => {
14     if (!email || !password) {
15       Alert.alert('Error', 'Por favor, ingresa tu email y contraseña.');
```

```
16     return;
17   }
18
19   setIsLoading(true);
20   try {
21     await signInWithEmailAndPassword(auth, email, password);
22     navigation.navigate('Main');
```

```
23   } catch (error) {
24     console.error('Error de inicio de sesión:', error);
25     Alert.alert('Error', 'Inicio de sesión fallido. Por favor, verifica tus credenciales.');
```

```
26   } finally {
27     setIsLoading(false);
28   }
29   };
```

En esta imagen podemos apreciar el código de login en el cual se muestra una estructura de las dependencias y de las validaciones para poder ingresar a la aplicación.

## RegisterScreen.js

```
1  import React, { useState } from 'react';
2  import { View, Text, TextInput, TouchableOpacity, StyleSheet, Alert } from 'react-native';
3  import { createUserWithEmailAndPassword } from '@firebase/auth';
4  import { auth } from '../../firebase';
5  import { Ionicons } from '@expo/vector-icons';
6
7  export default function RegisterScreen({ navigation }) {
8    const [email, setEmail] = useState('');
9    const [password, setPassword] = useState('');
10   const [confirmPassword, setConfirmPassword] = useState('');
11   const [isLoading, setIsLoading] = useState(false);
12   const [showPassword, setShowPassword] = useState(false);
13
14   const handleRegister = async () => {
15     if (!email || !password || !confirmPassword) {
16       Alert.alert('Error', 'Por favor, completa todos los campos.');
```

```
17     return;
18   }
19
20   if (password !== confirmPassword) {
21     Alert.alert('Error', 'Las contraseñas no coinciden.');
```

```
22     return;
23   }
24
25   setIsLoading(true);
26   try {
27     await createUserWithEmailAndPassword(auth, email, password);
28     Alert.alert('Éxito', 'Cuenta creada exitosamente', [
29       { text: 'OK', onPress: () => navigation.navigate('Login') }
30     ]);
```

En esta imagen se presenta el código del registro a la aplicación en donde se muestran las validaciones necesarias como también se presentan las dependencias utilizadas.

## AppNavigator.js

```
src > navigation > JS AppNavigator.js > ...
28 function MainTabs() {
31   screenOptions={({ route }) => ({
32     tabBarIcon: ({ focused, color, size }) => {
43     } else if (route.name === 'UserStats') {
44       iconName = focused ? 'stats-chart' : 'stats-chart-outline';
45     }
46   }
47   return <Ionicons name={iconName} size={size} color={color} />;
48 },
49 )}
50 >
51 <Tab.Screen name="Home" component={HomeScreen} options={{ title: 'Inicio' }} />
52 <Tab.Screen name="EventList" component={EventListScreen} options={{ title: 'Eventos' }} />
53 <Tab.Screen name="CreateEvent" component={CreateEventScreen} options={{ title: 'Crear Evento' }} />
54 <Tab.Screen name="EventHistory" component={EventHistoryScreen} options={{ title: 'Historial' }} />
55 <Tab.Screen name="UserStats" component={UserStatsScreen} options={{ title: 'Estadísticas' }} />
56 </Tab.Navigator>
57 );
58 }
59
60 export default function AppNavigator() {
61   return (
62     <NavigationContainer>
63       <Stack.Navigator screenOptions={{ headerShown: false }}>
64         <Stack.Screen name="Auth" component={AuthStack} />
65         <Stack.Screen name="Main" component={MainTabs} />
66         <Stack.Screen
67           name="EventDetails"
68           component={EventDetailsScreen}
69           options={{ headerShown: true, title: 'Detalles del Evento' }}

```

En esta imagen se presenta la navegación que se tiene en el proyecto y como se puede navegar en la aplicación y las opciones que este tiene para los usuarios.

```
src > screens > JS CreateEventScreen.js > ...
8 export default function CreateEventScreen({ navigation }) {
46   return (
47     <ScrollView style={styles.container}>
48       <Text style={styles.title}>Crear Nuevo Evento</Text>
49
50       <View style={styles.inputContainer}>
51         <Ionicons name="pencil-outline" size={24} color="#4A5568" style={styles.icon} />
52         <TextInput
53           style={styles.input}
54           placeholder="Título del evento"
55           value={title}
56           onChangeText={setTitle}
57         />
58       </View>
59
60       <TouchableOpacity style={styles.datePickerButton} onPress={() => setShowDatePicker(true)}>
61         <Ionicons name="calendar-outline" size={24} color="#4A5568" style={styles.icon} />
62         <Text style={styles.datePickerButtonText}>
63           {date.toLocaleDateString()}
64         </Text>
65       </TouchableOpacity>
66       {showDatePicker && (
67         <DateTimePicker
68           value={date}
69           mode="date"
70           display="default"
71           onChange={onChangeDate}
72         />
73     )}

```



src > screens > JS EventDetailsScreen.js > ...

```
7   export default function EventDetailsScreen({ route, navigation }) {
11     useEffect(() => {
12       const fetchEvent = async () => {
25       };
26
27       fetchEvent();
28     }, [eventId]);
29
30     const handleDeleteEvent = async () => {
31       Alert.alert(
32         'Confirmar eliminación',
33         '¿Estás seguro de que quieres eliminar este evento?',
34         [
35           { text: 'Cancelar', style: 'cancel' },
36           {
37             text: 'Eliminar',
38             style: 'destructive',
39             onPress: async () => {
40               try {
41                 await deleteDoc(doc(db, 'events', eventId));
42                 Alert.alert('Éxito', 'Evento eliminado correctamente');
43                 navigation.goBack();
44               } catch (error) {
45                 console.error('Error deleting event:', error);
46                 Alert.alert('Error', 'No se pudo eliminar el evento');
47               }
48             }
49           },
50         ],
51       );
52     };
53   }
54 }
```

src > screens > JS EventHistoryScreen.js > ...

```
6   export default function EventHistoryScreen() {
14     const fetchPastEvents = async () => {
20     };
21
22     const renderEventItem = ({ item }) => (
23       <View style={styles.eventItem}>
24         <Text style={styles.eventTitle}>{item.title}</Text>
25         <Text style={styles.eventDate}>{new Date(item.date).toLocaleDateString()}</Text>
26       </View>
27     );
28
29     return (
30       <View style={styles.container}>
31         <View style={styles.statisticsContainer}>
32           <Text style={styles.statisticsTitle}>Estadísticas</Text>
33           <Text style={styles.statisticsText}>Total de eventos asistidos: {statistics.totalEvents}</Text>
34           <Text style={styles.statisticsText}>Calificación promedio: {statistics.averageRating.toFixed(1)}</Text>
35         </View>
36         <Text style={styles.historyTitle}>Historial de Eventos</Text>
37         <FlatList
38           data={pastEvents}
39           renderItem={renderEventItem}
40           keyExtractor={({ item }) => item.id}
41           ListEmptyComponent={<Text style={styles.emptyText}>No has asistido a ningún evento aún</Text>}
42         />
43       </View>
44     );
45   }
46 }
```

```

src > screens > JS EventListScreen.js > ...
 7  export default function EventListScreen({ navigation }) {
33    const renderEventItem = ({ item }) => (
35      style={styles.eventItem}
36      onPress={() => navigation.navigate('EventDetails', { eventId: item.id })}
37    >
38      <View style={styles.eventHeader}>
39        <Text style={styles.eventTitle}>{item.title}</Text>
40        <Ionicons name="chevron-forward" size={24} color="#4A5568" />
41      </View>
42      <Text style={styles.eventDate}>{item.date}</Text>
43      <Text style={styles.eventLocation}>{item.location}</Text>
44    </TouchableOpacity>
45  );
46
47  return (
48    <View style={styles.container}>
49      <Text style={styles.title}>Eventos Próximos</Text>
50      {events.length > 0 ? (
51        <FlatList
52          data={events}
53          renderItem={renderEventItem}
54          keyExtractor={item => item.id}
55          refreshControl={
56            <RefreshControl refreshing={refreshing} onRefresh={onRefresh} />
57          }
58        />
59      ) : (
60        <Text style={styles.noEventsText}>No hay eventos disponibles.</Text>
61      )}
62    </TouchableOpacity>

```

En estas imágenes se presenta el código de la funcionalidad de los eventos desde la creación de los eventos como también el listado y como se presentan en el historial y en las estadísticas, además de presentar la estructura que se realizará para obtener o ingresar estos eventos.

Además de cómo se calcula las estadísticas como también como se muestran los historiales de los eventos pasados.