

Step by step, practical point

- The project is created in Expo using the command “npx create-expo-app@latest” and adding the “--template” statement.

```
p\Desarrollo Movil>npx create-expo-app@latest --template
```

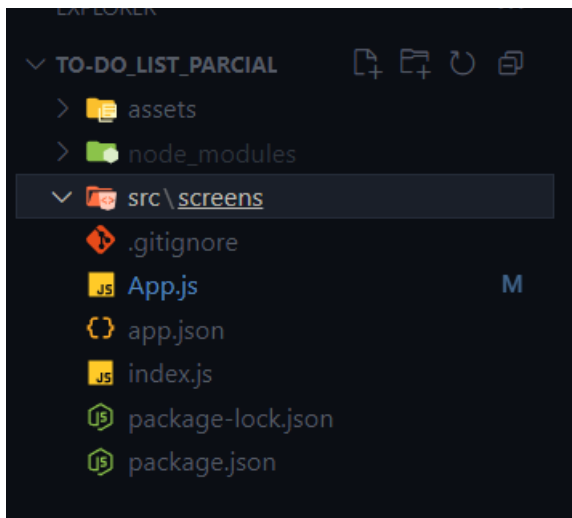
- Template: Blank is selected and the project name App expo is assigned.

```
✓ Choose a template: » Blank  
? What is your app named? » To-Do_List_Parcial
```

- We access our created project with “cd project_name” and execute the line “code .” to open the program

```
p\Desarrollo Movil>cd To-Do_List_Parcial  
  
p\Desarrollo Movil\To-Do_List_Parcial>code .
```

- We create the SRC folder to store the elements of our app Project



- We generate the structure of each of the requested screens and your styles

```
HomeScreen.js U X LoginScreen.js U RegisterScreen.js U
src > screens > HomeScreen.js > Styles > buttonText
1 import React from "react";
2 import { View, Text, TouchableOpacity,StyleSheet } from "react-native";
3
4
5 const HomeScreen = ()=> {
6   return (
7     <View style={{ flex: 1, justifyContent: "center", alignItems: "center" }}>
8       <Text>Home Screen</Text>
9       <TouchableOpacity onPress={() => alert("Button Pressed!")}>
10         <Text style={{ color: "blue", marginTop: 20 }}>Press Me</Text>
11       </TouchableOpacity>
12     </View>
13   );
14 }
15
16 const Styles = StyleSheet.create({
17   container: {
18     flex: 1,
19     justifyContent: 'center',
20     alignItems: 'center',
21   },
22   button: {
23     backgroundColor: '#007BFF',
24     padding: 10,
25     borderRadius: 5,
26   },
27   buttonText: {
28     color: '#FFFFFF',
29     fontSize: 16,
30   },
31 });
```

- The AppNavigator.js file is generated to have control over the created screens

```
src > navigation > AppNavigator.js > ...
1  // navigation/AppNavigator.js
2  import React from 'react';
3  import { NavigationContainer } from '@react-navigation/native';
4  import { createStackNavigator } from '@react-navigation/stack';
5
6  import LoginScreen from '../screens/LoginScreen';
7  import RegisterScreen from '../screens/RegisterScreen';
8  import HomeScreen from '../screens/HomeScreen';
9  import AddTaskScreen from '../screens/AddTaskScreen';
10
11  const Stack = createStackNavigator();
12
13  const AppNavigator = () => {
14    return (
15      <NavigationContainer>
16        <Stack.Navigator initialRouteName="Login">
17          <Stack.Screen name="Login" component={LoginScreen} />
18          <Stack.Screen name="Register" component={RegisterScreen} />
19          <Stack.Screen name="Home" component={HomeScreen} />
20          <Stack.Screen name="AddTask" component={AddTaskScreen} />
21        </Stack.Navigator>
22      </NavigationContainer>
23    );
24  }
25
26  export default AppNavigator;
```

- The “.env” file is created, which contains the information to consume the Firebase API.

```
1  EXPO_PUBLIC_API_KEY="AIzaSyAiuGv13Z6wJ-a5dg-r1JlNp0XDnXPC6BY"
2  EXPO_PUBLIC_AUTH_DOMAIN="proyecto-latest.firebaseio.com"
3  EXPO_PUBLIC_PROJECT_ID="proyecto-latest"
4  EXPO_PUBLIC_STORAGE_BUCKET="proyecto-latest.firebaseio.com"
5  EXPO_PUBLIC_MESSAGING_SENDER_ID="763292271569"
6  EXPO_PUBLIC_APP_ID="1:763292271569:web:b7471b1bf67dc8e0a256f0"
7  EXPO_PUBLIC_MEASUREMENT_ID="G-DLCC4GK7KT"
8  EXPO_PUBLIC_KEY_GOOGLELG="project-763292271569"
9
```

- The “firebaseConfig.js” file is created to consume the firebase service in our project

```
✓ import { initializeApp } from 'firebase/app';
import { getAuth } from 'firebase/auth';
// import { API_KEY, AUTH_DOMAIN, PROJECT_ID, STORAGE_BUCKET, MESSAGING_SENDER_ID, APP_ID } from '@env';

✓ const firebaseConfig = {
  apiKey: process.env.EXPO_PUBLIC_API_KEY,
  authDomain: process.env.EXPO_PUBLIC_AUTH_DOMAIN,
  projectId: process.env.EXPO_PUBLIC_PROJECT_ID,
  storageBucket: process.env.EXPO_PUBLIC_STORAGE_BUCKET,
  messagingSenderId: process.env.EXPO_PUBLIC_MESSAGING_SENDER_ID,
  appId: process.env.EXPO_PUBLIC_APP_ID,
  measurementId: process.env.EXPO_PUBLIC_MEASUREMENT_ID
};

const app = initializeApp(firebaseConfig);
const auth = getAuth(app);

export { auth };
```

- In the “AddTaskScreen” file we create the logic to save the new tasks

```
src > screens > AddTaskScreen.js > AddTaskScreen > handleSaveTask
4 import { useRoute } from '@react-navigation/native';
5
6 const AddTaskScreen = ({ navigation }) => {
7   const [title, setTitle] = useState('');
8   const [description, setDescription] = useState('');
9   const route = useRoute();
10
11   const handleSaveTask = async () => {
12     if (!title.trim()) {
13       alert('El título es obligatorio.');
```

- And in the “HomeScreen” file we call, through the Async function, the tasks stored in an array

```
src > screens > JS HomeScreen.js > HomeScreen
6
7  const HomeScreen = ({ navigation }) => {
8    const [tasks, setTasks] = useState([]);
9
10   const loadTasks = useCallback(async () => {
11     try {
12       const tasksJSON = await AsyncStorage.getItem('tasks');
13       const loadedTasks = tasksJSON ? JSON.parse(tasksJSON) : [];
14       setTasks(loadedTasks);
15     } catch (error) {
16       alert.error('Error al cargar las tareas:', error);
17     }
18   }, []);
19
20   const saveTasks = useCallback(async (newTasks) => {
21     try {
22       await AsyncStorage.setItem('tasks', JSON.stringify(newTasks));
23     } catch (error) {
24       alert.error('Error al guardar la tarea:', error);
25     }
26   }, []);
27
```

- We call the “FlatList” component to traverse the list we have in our task array

```
return (
  <View style={styles.container}>
    <Text style={styles.title}>Mis Tareas</Text>
    <FlatList
      data={tasks}
      renderItem={renderItem}
      keyExtractor={(item) => item.id}
      ListEmptyComponent={<Text style={styles.emptyText}>No hay tareas.</Text>}
      contentContainerStyle={{ paddingBottom: 100 }}
    />
    <TouchableOpacity style={styles.addButton} onPress={navigateAddTask}>
      <Feather name="plus" size={28} color="#fff" />
    </TouchableOpacity>
  </View>
);
```