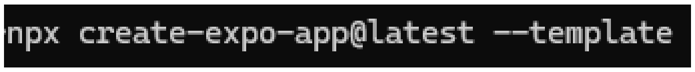
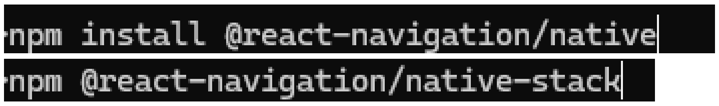
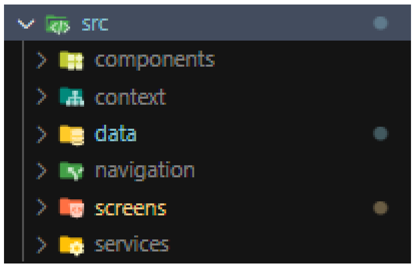
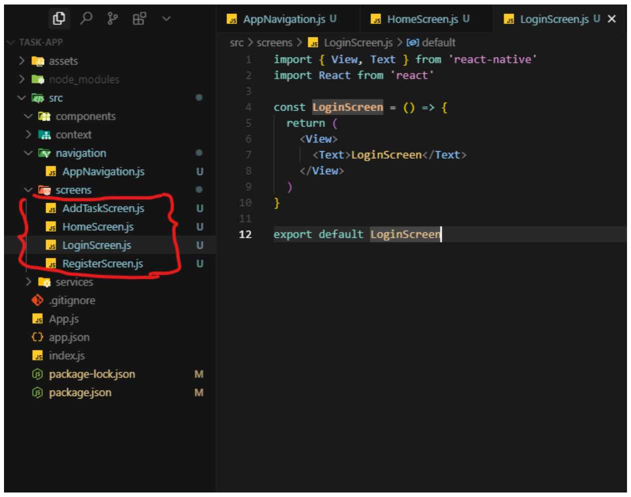
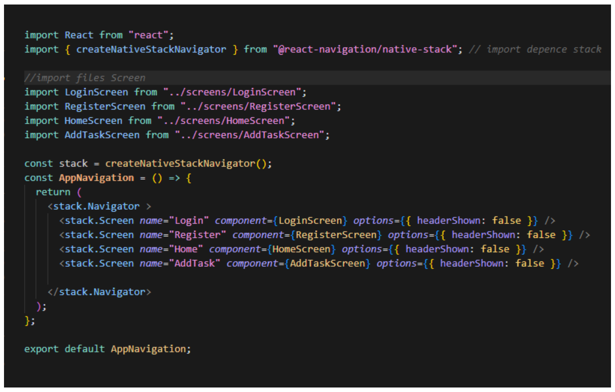
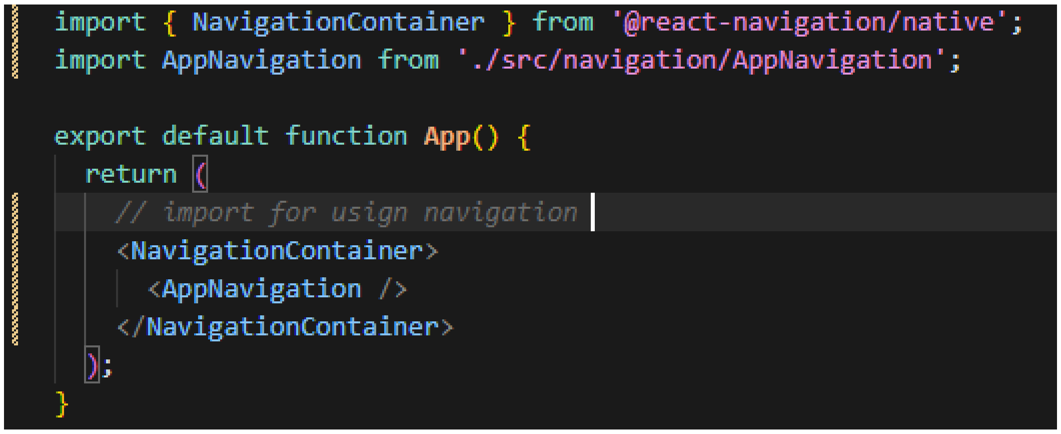
Create app todo list parcial

1. Créate expo proyect
   In the terminal app, execute next commant
   
   ```
   npx create-expo-app@latest --template
   ```
2. Install dependeces for navigation
   
   ```
   npm install @react-navigation/native
   npm @react-navigation/native-stack
   ```
3. Create estructure for the app
   
   ```
   src
     components
     context
     data
     navigation
     screens
     services
   ```
4. Create files for the app
   

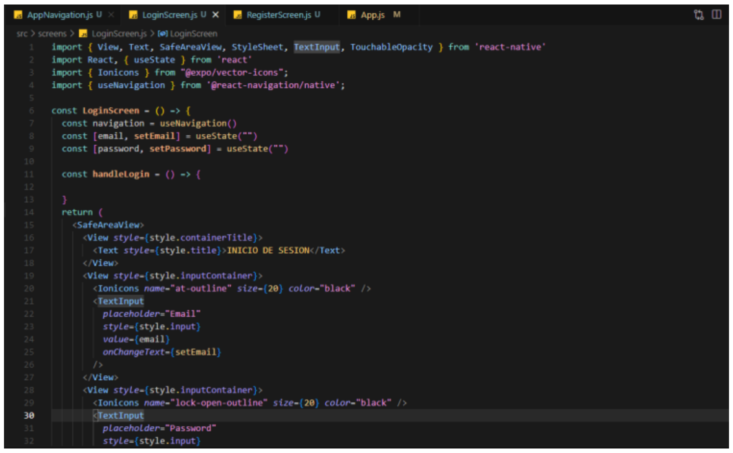
   ```
   TASK-APP
     assets
     node_modules
     src
       components
       context
       navigation
         AppNavigation.js        U
       screens
         AddTaskScreen.js        U
         HomeScreen.js           U
         LoginScreen.js          U
         RegisterScreen.js       U
       services
     .gitignore
     App.js
     app.json
     index.js
     package-lock.json           M
     package.json                M
   ```
   ```
   import { View, Text } from 'react-native'
   import React from 'react'

   const LoginScreen = () => {
     return (
       <View>
         <Text>LoginScreen</Text>
       </View>
     )
   }

   export default LoginScreen
   ```
5. Create the file for the navigation configuration in the application
   
   ```
   import React from "react";
   import { createNativeStackNavigator } from "@react-navigation/native-stack"; // import depence stack

   //import files Screen
   import LoginScreen from "../screens/LoginScreen";
   import RegisterScreen from "../screens/RegisterScreen";
   import HomeScreen from "../screens/HomeScreen";
   import AddTaskScreen from "../screens/AddTaskScreen";

   const stack = createNativeStackNavigator();
   const AppNavigation = () => {
     return (
       <stack.Navigator >
         <stack.Screen name="Login" component={LoginScreen} options={{ headerShown: false }} />
         <stack.Screen name="Register" component={RegisterScreen} options={{ headerShown: false }} />
         <stack.Screen name="Home" component={HomeScreen} options={{ headerShown: false }} />
         <stack.Screen name="AddTask" component={AddTaskScreen} options={{ headerShown: false }} />

       </stack.Navigator>
     );
   };

   export default AppNavigation;
   ```

6. Using navigation in the origin app



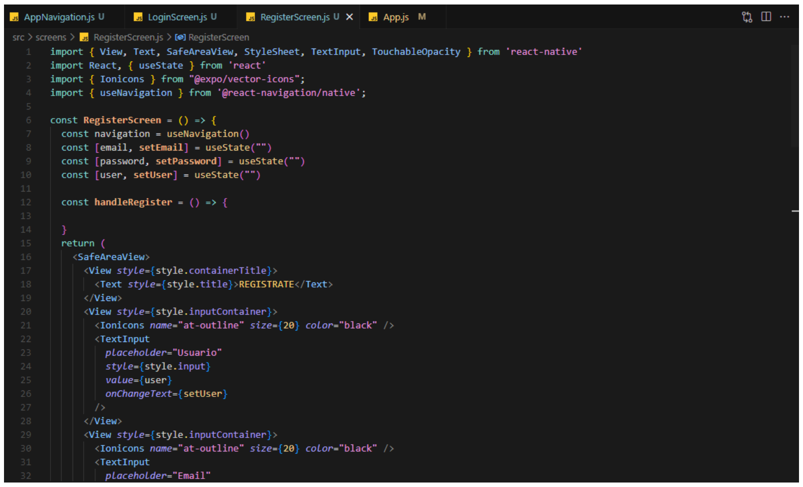
```
import { NavigationContainer } from '@react-navigation/native';
import AppNavigation from './src/navigation/AppNavigation';

export default function App() {
  return (
    // import for usign navigation
    <NavigationContainer>
      <AppNavigation />
    </NavigationContainer>
  );
}
```

7. create view for loginScreen



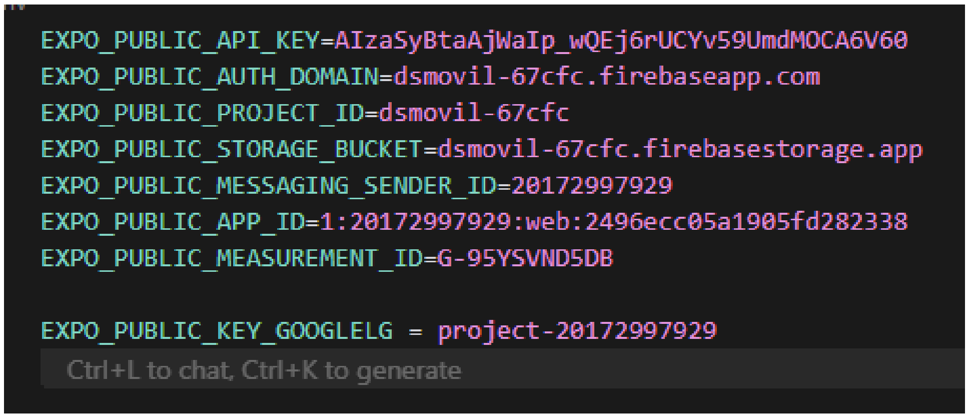
```
import { View, Text, SafeAreaView, StyleSheet, TextInput, TouchableOpacity } from 'react-native'
import React, { useState } from 'react'
import { Ionicons } from "@expo/vector-icons";
import { useNavigation } from '@react-navigation/native';

const LoginScreen = () => {
  const navigation = useNavigation()
  const [email, setEmail] = useState("")
  const [password, setPassword] = useState("")

  const handleLogin = () => {

  }
  return (
    <SafeAreaView>
      <View style={style.containerTitle}>
        <Text style={style.title}>INICIO DE SESION</Text>
      </View>
      <View style={style.inputContainer}>
        <Ionicons name="at-outline" size={20} color="black" />
        <TextInput
          placeholder="Email"
          style={style.input}
          value={email}
          onChangeText={setEmail}
        />
      </View>
      <View style={style.inputContainer}>
        <Ionicons name="lock-open-outline" size={20} color="black" />
        <TextInput
          placeholder="Password"
          style={style.input}
```

8. créate view for Register Screen



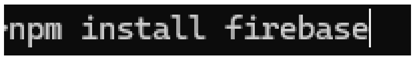
```
import { View, Text, SafeAreaView, StyleSheet, TextInput, TouchableOpacity } from 'react-native'
import React, { useState } from 'react'
import { Ionicons } from "@expo/vector-icons";
import { useNavigation } from '@react-navigation/native';

const RegisterScreen = () => {
  const navigation = useNavigation()
  const [email, setEmail] = useState("")
  const [password, setPassword] = useState("")
  const [user, setUser] = useState("")

  const handleRegister = () => {

  }
  return (
    <SafeAreaView>
      <View style={style.containerTitle}>
        <Text style={style.title}>REGISTRATE</Text>
      </View>
      <View style={style.inputContainer}>
        <Ionicons name="at-outline" size={20} color="black" />
        <TextInput
          placeholder="Usuario"
          style={style.input}
          value={user}
          onChangeText={setUser}
        />
      </View>
      <View style={style.inputContainer}>
        <Ionicons name="at-outline" size={20} color="black" />
        <TextInput
          placeholder="Email"
```

9. créate file .env for the environment variables

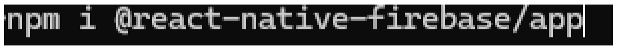


```
EXPO_PUBLIC_API_KEY=AIzaSyBtaAjWaIp_wQEj6rUCYv59UmdMOCA6V60
EXPO_PUBLIC_AUTH_DOMAIN=dsmovil-67cfc.firebaseapp.com
EXPO_PUBLIC_PROJECT_ID=dsmovil-67cfc
EXPO_PUBLIC_STORAGE_BUCKET=dsmovil-67cfc.firebasestorage.app
EXPO_PUBLIC_MESSAGING_SENDER_ID=20172997929
EXPO_PUBLIC_APP_ID=1:20172997929:web:2496ecc05a1905fd282338
EXPO_PUBLIC_MEASUREMENT_ID=G-95YSVND5DB

EXPO_PUBLIC_KEY_GOOGLELG = project-20172997929
```

10. Install dependeces for using firebase

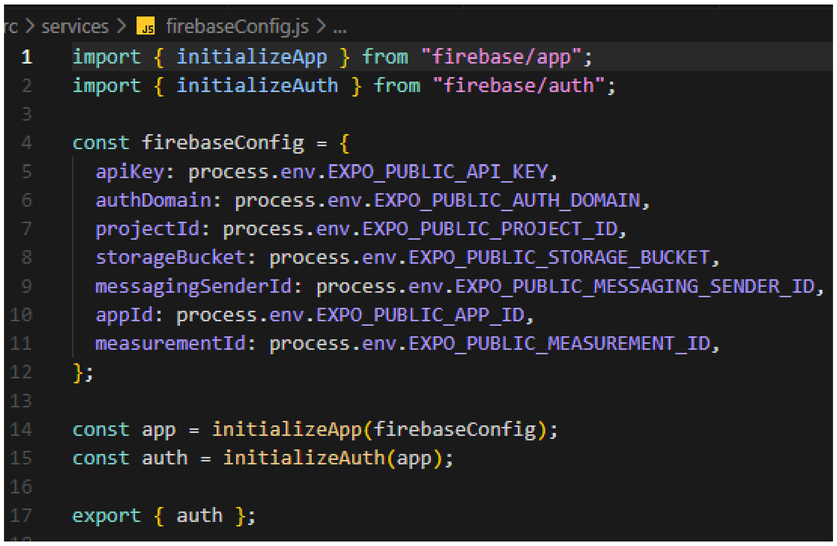


```
npm install firebase
```



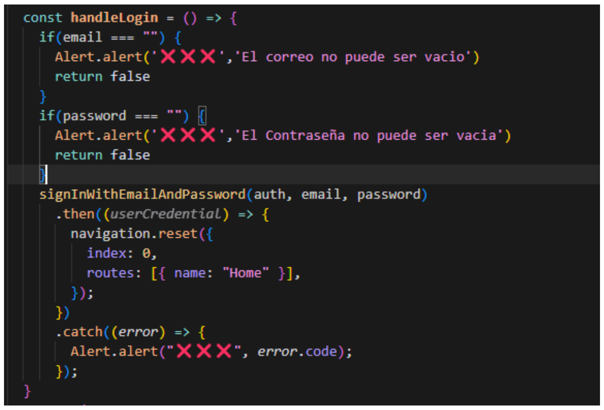
```
npm i @react-native-firebase/app
```

11. Create file for config firebase



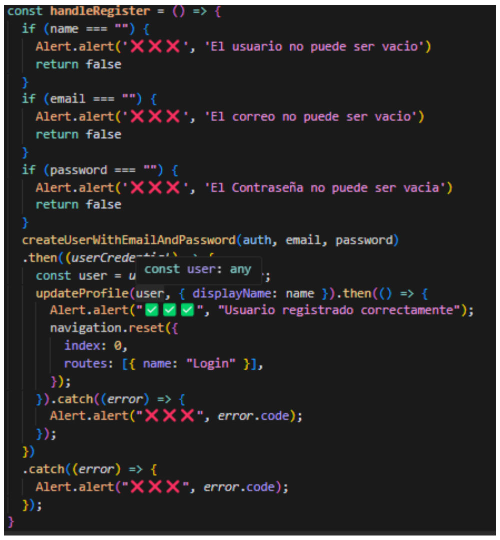
```
import { initializeApp } from "firebase/app";
import { initializeAuth } from "firebase/auth";

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 = initializeAuth(app);

export { auth };
```

12. Create a Firebase login function



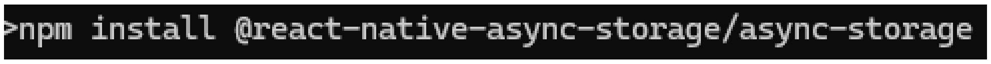
```
const handleLogin = () => {
  if(email === "") {
    Alert.alert('❌❌❌','El correo no puede ser vacio')
    return false
  }
  if(password === "") {
    Alert.alert('❌❌❌','El Contraseña no puede ser vacia')
    return false
  }
  signInWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
      navigation.reset({
        index: 0,
        routes: [{ name: "Home" }],
      });
    })
    .catch((error) => {
      Alert.alert("❌❌❌", error.code);
    });
}
```

13. Créate a firebase register function



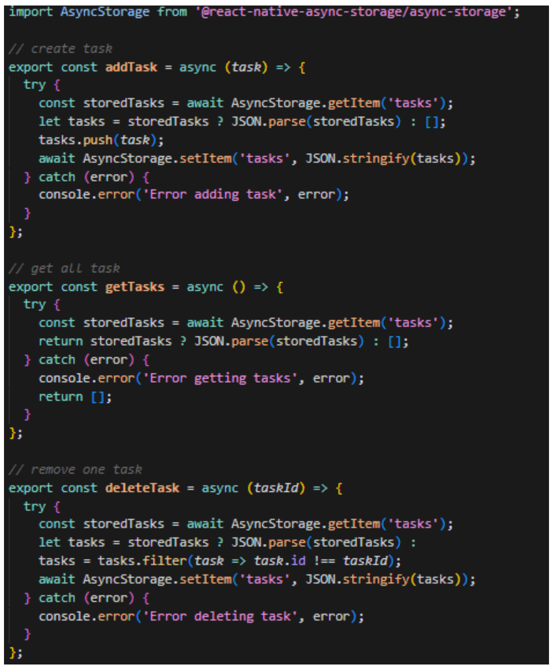
```
const handleRegister = () => {
  if (name === "") {
    Alert.alert('❌❌❌', 'El usuario no puede ser vacio')
    return false
  }
  if (email === "") {
    Alert.alert('❌❌❌', 'El correo no puede ser vacio')
    return false
  }
  if (password === "") {
    Alert.alert('❌❌❌', 'El Contraseña no puede ser vacia')
    return false
  }
  createUserWithEmailAndPassword(auth, email, password)
  .then((userCrede
    const user = u const user: any ;
    updateProfile(user, { displayName: name }).then(() => {
      Alert.alert("✅✅✅", "Usuario registrado correctamente");
      navigation.reset({
        index: 0,
        routes: [{ name: "Login" }],
      });
    }).catch((error) => {
      Alert.alert("❌❌❌", error.code);
    });
  })
  .catch((error) => {
    Alert.alert("❌❌❌", error.code);
  });
}
```

14. Install dependece for async storage

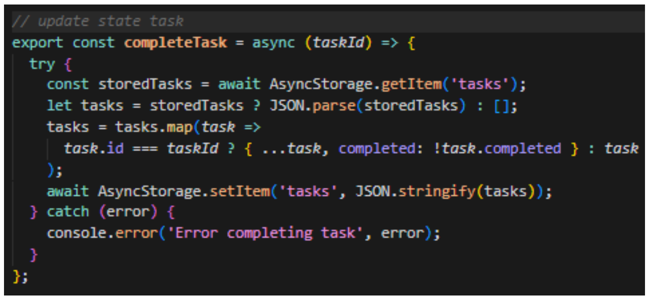


```
>npm install @react-native-async-storage/async-storage
```

15. Create the task actions in a separate file for better control of them

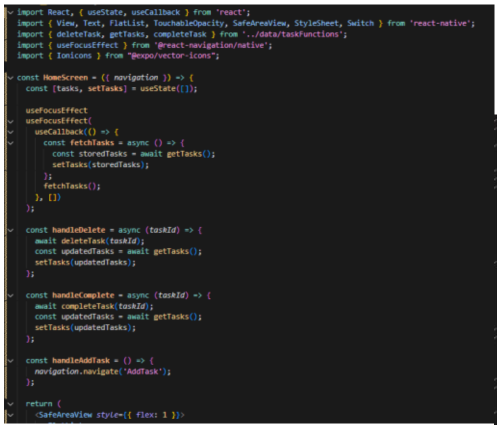


```
import AsyncStorage from '@react-native-async-storage/async-storage';

// create task
export const addTask = async (task) => {
  try {
    const storedTasks = await AsyncStorage.getItem('tasks');
    let tasks = storedTasks ? JSON.parse(storedTasks) : [];
    tasks.push(task);
    await AsyncStorage.setItem('tasks', JSON.stringify(tasks));
  } catch (error) {
    console.error('Error adding task', error);
  }
};

// get all task
export const getTasks = async () => {
  try {
    const storedTasks = await AsyncStorage.getItem('tasks');
    return storedTasks ? JSON.parse(storedTasks) : [];
  } catch (error) {
    console.error('Error getting tasks', error);
    return [];
  }
};

// remove one task
export const deleteTask = async (taskId) => {
  try {
    const storedTasks = await AsyncStorage.getItem('tasks');
    let tasks = storedTasks ? JSON.parse(storedTasks) :
    tasks = tasks.filter(task => task.id !== taskId);
    await AsyncStorage.setItem('tasks', JSON.stringify(tasks));
  } catch (error) {
    console.error('Error deleting task', error);
  }
};
```

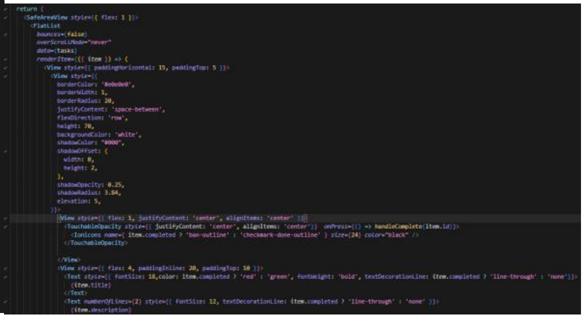


```
// update state task
export const completeTask = async (taskId) => {
  try {
    const storedTasks = await AsyncStorage.getItem('tasks');
    let tasks = storedTasks ? JSON.parse(storedTasks) : [];
    tasks = tasks.map(task =>
      task.id === taskId ? { ...task, completed: !task.completed } : task
    );
    await AsyncStorage.setItem('tasks', JSON.stringify(tasks));
  } catch (error) {
    console.error('Error completing task', error);
  }
};
```

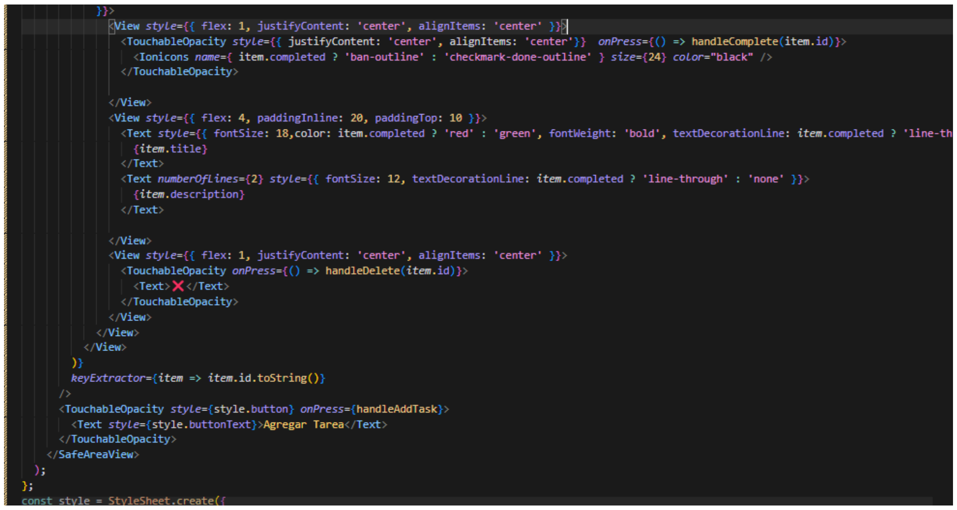
16. Create the design of the HomeScreen view with its navigation actions, and actions for completing and deleting tasks



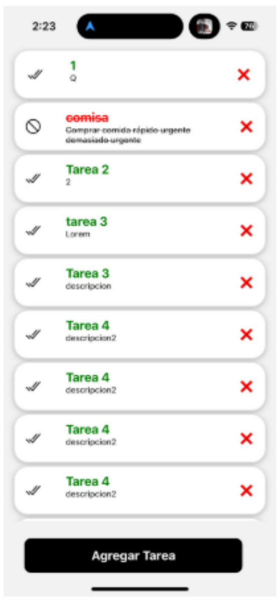
```
import React, { useState, useCallback } from 'react';
import { View, Text, FlatList, TouchableOpacity, SafeAreaView, StyleSheet, Switch } from 'react-native';
import { deleteTask, getTasks, completeTask } from '../data/taskFunctions';
import { useFocusEffect } from '@react-navigation/native';
import { Ionicons } from "@expo/vector-icons";

const HomeScreen = ({ navigation }) => {
  const [tasks, setTasks] = useState([]);

  useFocusEffect
  useFocusEffect(
    useCallback(() => {
      const fetchTasks = async () => {
        const storedTasks = await getTasks();
        setTasks(storedTasks);
      };
      fetchTasks();
    }, [])
  );

  const handleDelete = async (taskId) => {
    await deleteTask(taskId);
    const updatedTasks = await getTasks();
    setTasks(updatedTasks);
  };

  const handleComplete = async (taskId) => {
    await completeTask(taskId);
    const updatedTasks = await getTasks();
    setTasks(updatedTasks);
  };

  const handleAddTask = () => {
    navigation.navigate('AddTask');
  };

  return (
    <SafeAreaView style={{ flex: 1 }}>
```



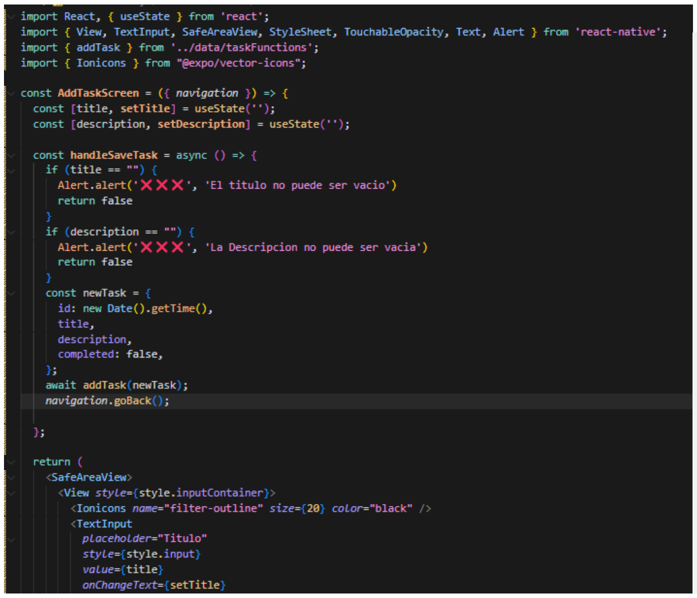
```
return (
  <SafeAreaView style={{ flex: 1 }}>
    <FlatList
      bounces={false}
      overScrollMode="never"
      data={tasks}
      renderItem={({ item }) => (
        <View style={{ paddingHorizontal: 15, paddingTop: 5 }}>
          <View style={{
            borderColor: '#e0e0e0',
            borderWidth: 1,
            borderRadius: 20,
            justifyContent: 'space-between',
            flexDirection: 'row',
            height: 70,
            backgroundColor: 'white',
            shadowColor: "#000",
            shadowOffset: {
              width: 0,
              height: 2,
            },
            shadowOpacity: 0.25,
            shadowRadius: 3.84,
            elevation: 5,
          }}>
            <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
              <TouchableOpacity style={{ justifyContent: 'center', alignItems: 'center'}}  onPress={() => handleComplete(item.id)}>
                <Ionicons name={ item.completed ? 'ban-outline' : 'checkmark-done-outline' } size={24} color="black" />
              </TouchableOpacity>

            </View>
            <View style={{ flex: 4, paddingInline: 20, paddingTop: 10 }}>
              <Text style={{ fontSize: 18,color: item.completed ? 'red' : 'green', fontWeight: 'bold', textDecorationLine: item.completed ? 'line-through' : 'none' }}>
                {item.title}
              </Text>
              <Text numberOfLines={2} style={{ fontSize: 12, textDecorationLine: item.completed ? 'line-through' : 'none' }}>
                {item.description}
```



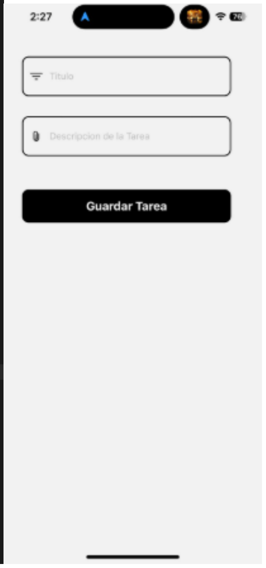
```
            }}>
              <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
                <TouchableOpacity style={{ justifyContent: 'center', alignItems: 'center'}}  onPress={() => handleComplete(item.id)}>
                  <Ionicons name={ item.completed ? 'ban-outline' : 'checkmark-done-outline' } size={24} color="black" />
                </TouchableOpacity>

              </View>
              <View style={{ flex: 4, paddingInline: 20, paddingTop: 10 }}>
                <Text style={{ fontSize: 18,color: item.completed ? 'red' : 'green', fontWeight: 'bold', textDecorationLine: item.completed ? 'line-th
                  {item.title}
                </Text>
                <Text numberOfLines={2} style={{ fontSize: 12, textDecorationLine: item.completed ? 'line-through' : 'none' }}>
                  {item.description}
                </Text>

              </View>
              <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
                <TouchableOpacity onPress={() => handleDelete(item.id)}>
                  <Text>❌</Text>
                </TouchableOpacity>
              </View>
            </View>
          </View>
        )}
        keyExtractor={item => item.id.toString()}
      />
      <TouchableOpacity style={style.button} onPress={handleAddTask}>
        <Text style={style.buttonText}>Agregar Tarea</Text>
      </TouchableOpacity>
    </SafeAreaView>
  );
};
const style = StyleSheet.create({
```

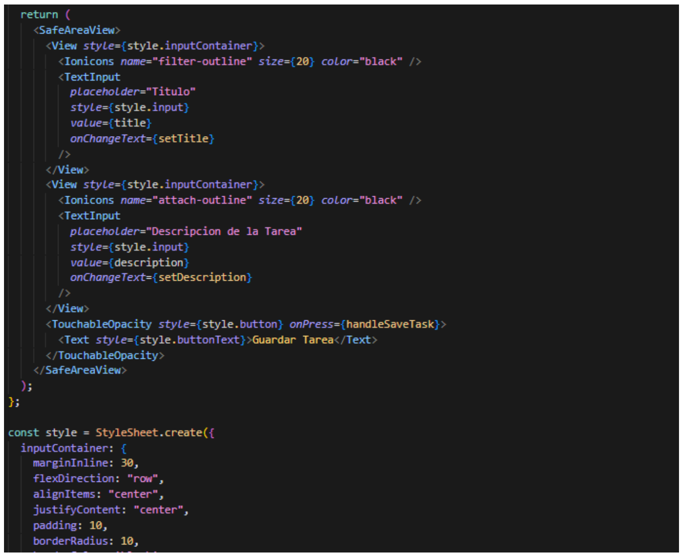


17. Create the design of the AddTaskScreen view with its navigation actions and task creation actions



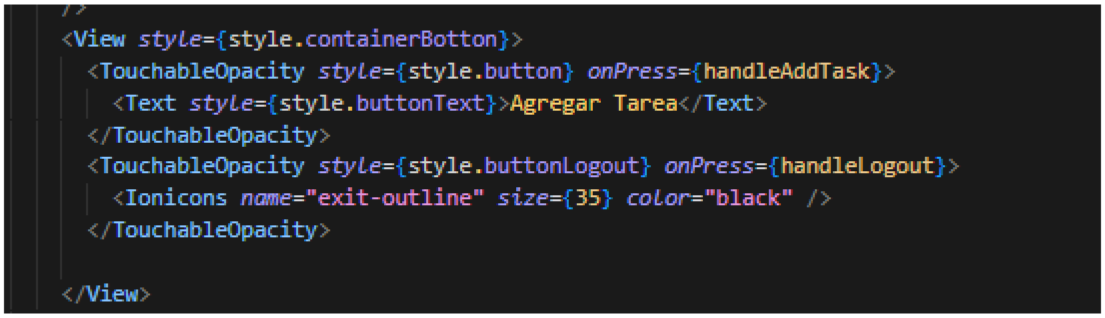
```
import React, { useState } from 'react';
import { View, TextInput, SafeAreaView, StyleSheet, TouchableOpacity, Text, Alert } from 'react-native';
import { addTask } from '../data/taskFunctions';
import { Ionicons } from "@expo/vector-icons";

const AddTaskScreen = ({ navigation }) => {
  const [title, setTitle] = useState('');
  const [description, setDescription] = useState('');

  const handleSaveTask = async () => {
    if (title == "") {
      Alert.alert('❌❌❌', 'El titulo no puede ser vacio')
      return false
    }
    if (description == "") {
      Alert.alert('❌❌❌', 'La Descripcion no puede ser vacia')
      return false
    }
    const newTask = {
      id: new Date().getTime(),
      title,
      description,
      completed: false,
    };
    await addTask(newTask);
    navigation.goBack();

  };

  return (
    <SafeAreaView>
      <View style={style.inputContainer}>
        <Ionicons name="filter-outline" size={20} color="black" />
        <TextInput
          placeholder="Titulo"
          style={style.input}
          value={title}
          onChangeText={setTitle}
```



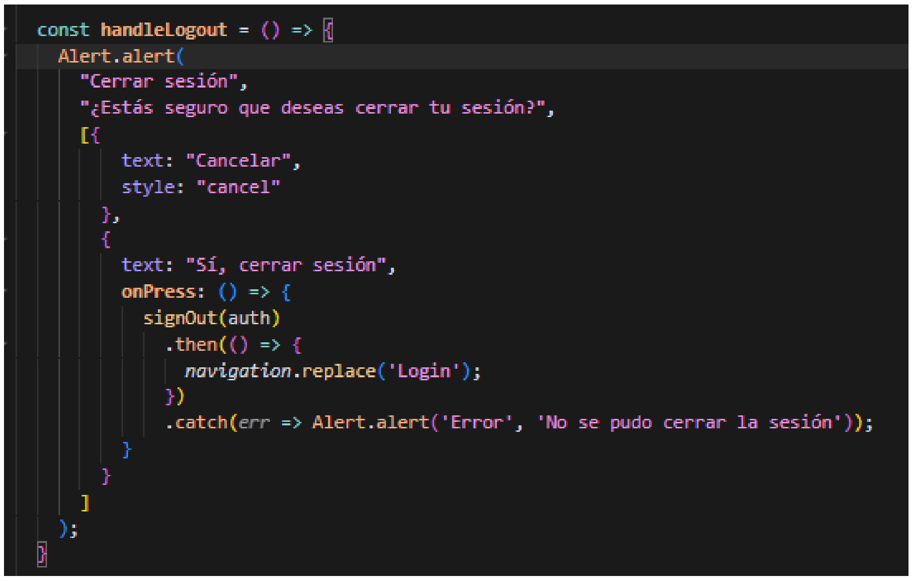


```
  return (
    <SafeAreaView>
      <View style={style.inputContainer}>
        <Ionicons name="filter-outline" size={20} color="black" />
        <TextInput
          placeholder="Titulo"
          style={style.input}
          value={title}
          onChangeText={setTitle}
        />
      </View>
      <View style={style.inputContainer}>
        <Ionicons name="attach-outline" size={20} color="black" />
        <TextInput
          placeholder="Descripcion de la Tarea"
          style={style.input}
          value={description}
          onChangeText={setDescription}
        />
      </View>
      <TouchableOpacity style={style.button} onPress={handleSaveTask}>
        <Text style={style.buttonText}>Guardar Tarea</Text>
      </TouchableOpacity>
    </SafeAreaView>
  );
};

const style = StyleSheet.create({
  inputContainer: {
    marginInline: 30,
    flexDirection: "row",
    alignItems: "center",
    justifyContent: "center",
    padding: 10,
    borderRadius: 10,
```

18. Implement button logout



```
/>
<View style={style.containerBotton}>
  <TouchableOpacity style={style.button} onPress={handleAddTask}>
    <Text style={style.buttonText}>Agregar Tarea</Text>
  </TouchableOpacity>
  <TouchableOpacity style={style.buttonLogout} onPress={handleLogout}>
    <Ionicons name="exit-outline" size={35} color="black" />
  </TouchableOpacity>

</View>
```



```
const handleLogout = () => {
  Alert.alert(
    "Cerrar sesión",
    "¿Estás seguro que deseas cerrar tu sesión?",
    [{
        text: "Cancelar",
        style: "cancel"
      },
      {
        text: "Sí, cerrar sesión",
        onPress: () => {
          signOut(auth)
            .then(() => {
              navigation.replace('Login');
            })
            .catch(err => Alert.alert('Error', 'No se pudo cerrar la sesión'));
        }
      }
    ]
  );
}
```