Step by step, practical point

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

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

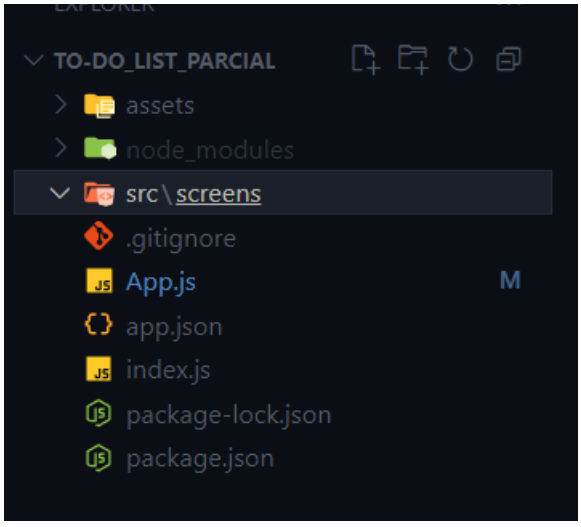
- Template: Blank is selected and the project name App expo is assigned.

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

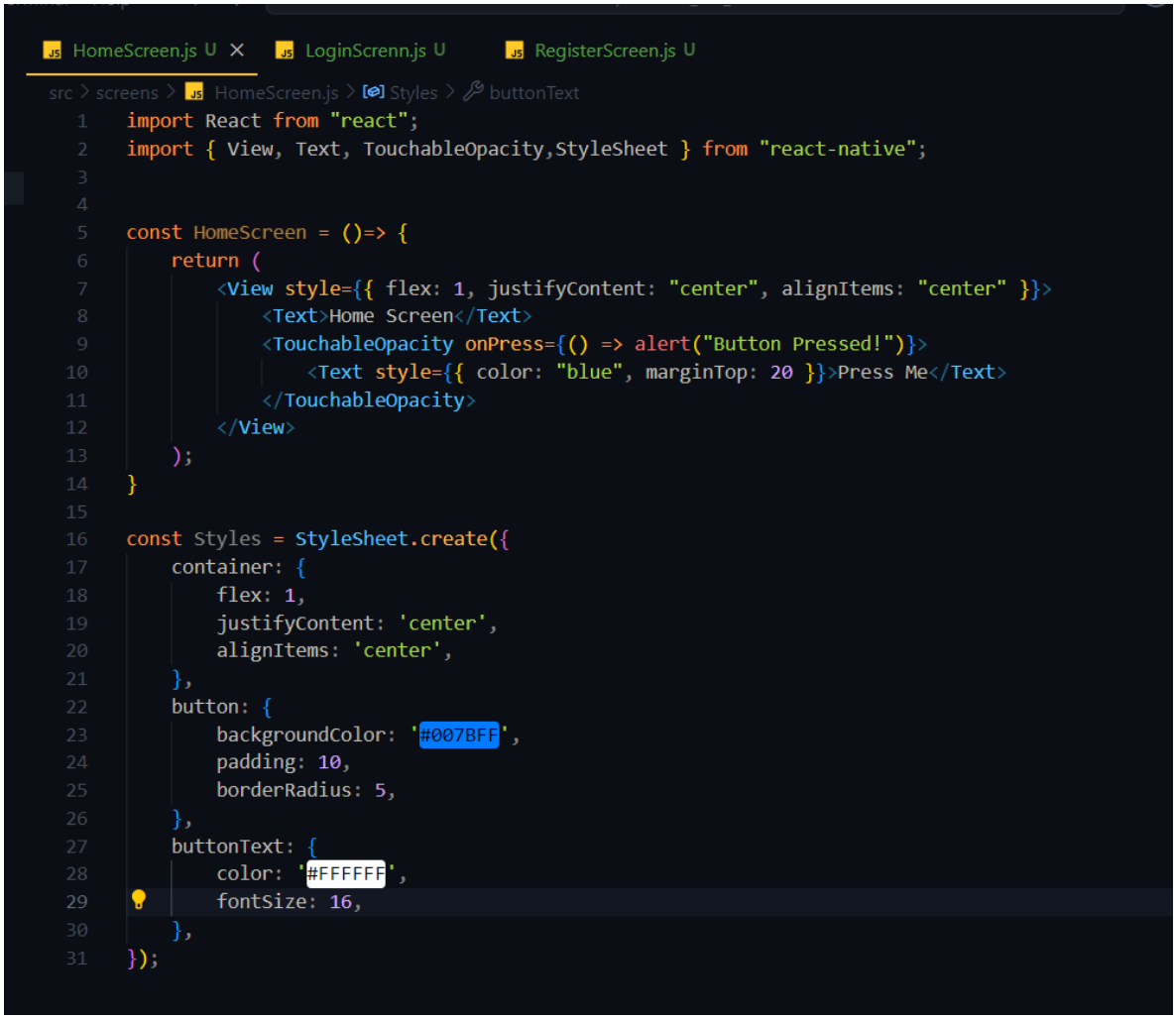
- We access our created project with “cd project_name” and execute the line “code .” to open the program

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

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

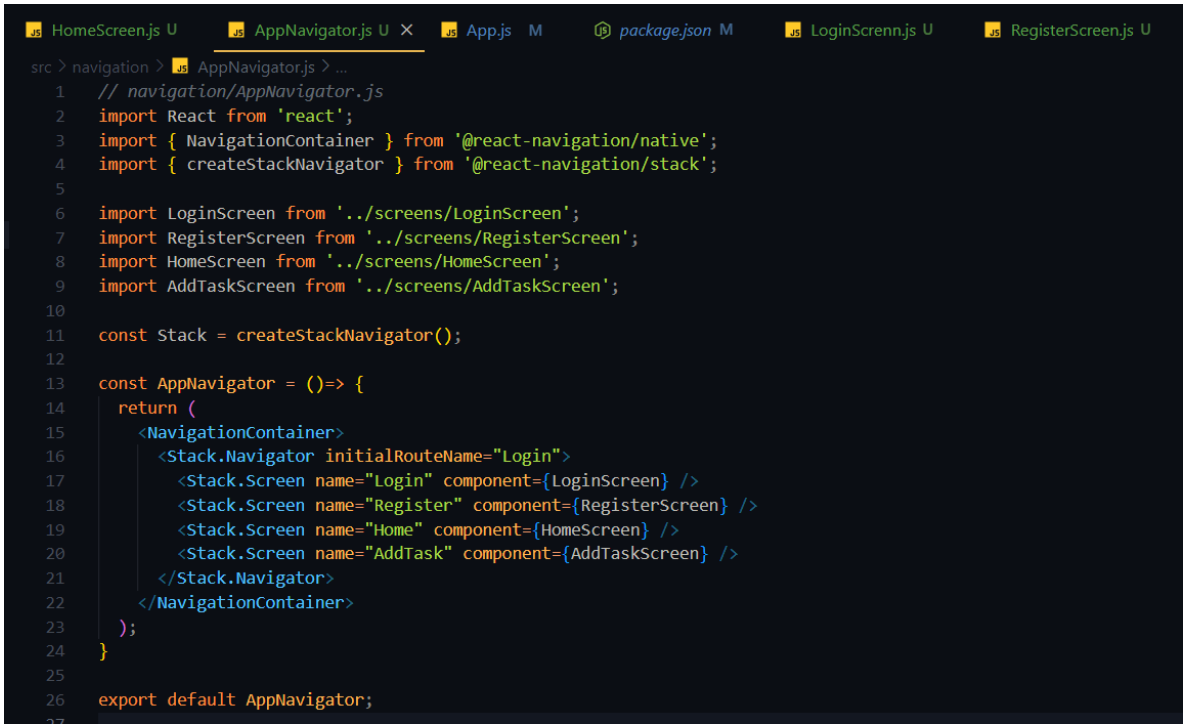
- We create the SRC folder to store the elements of our app Project



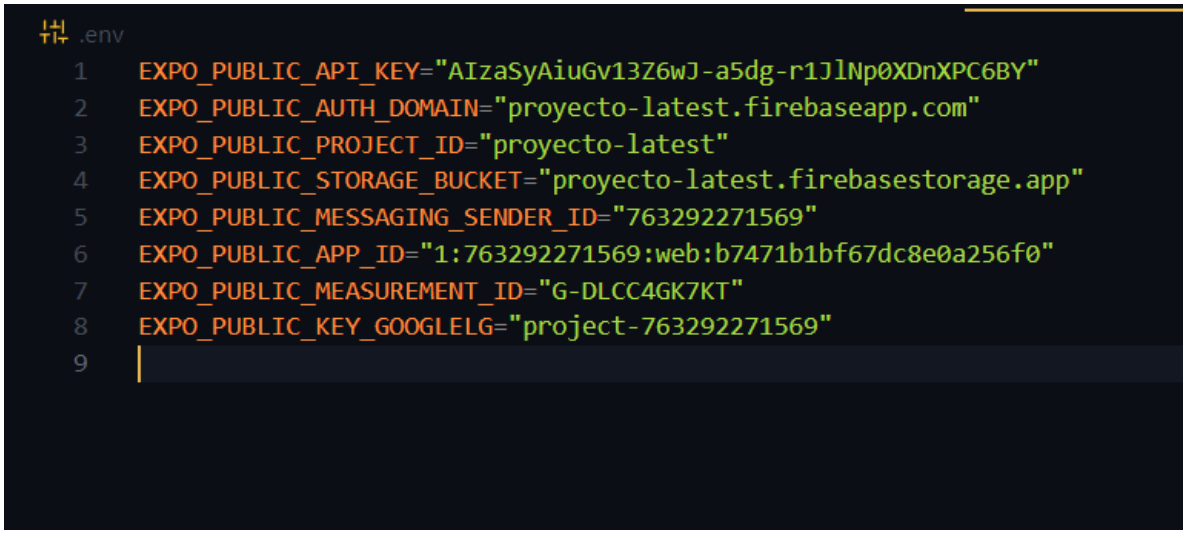
- We generate the structure of each of the requested screens and your styles



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



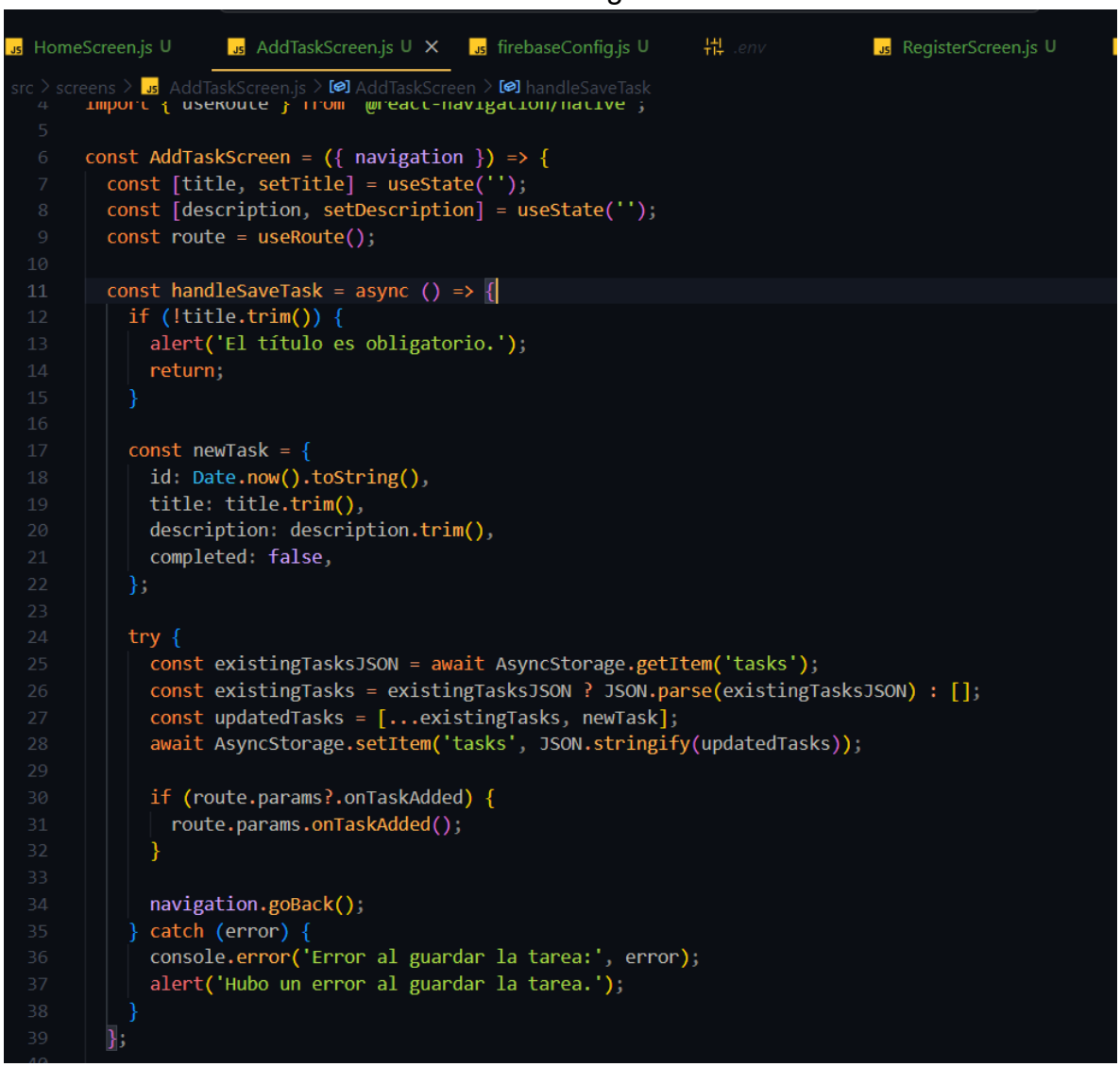
- The ".env" file is created, which contains the information to consume the Firebase API.



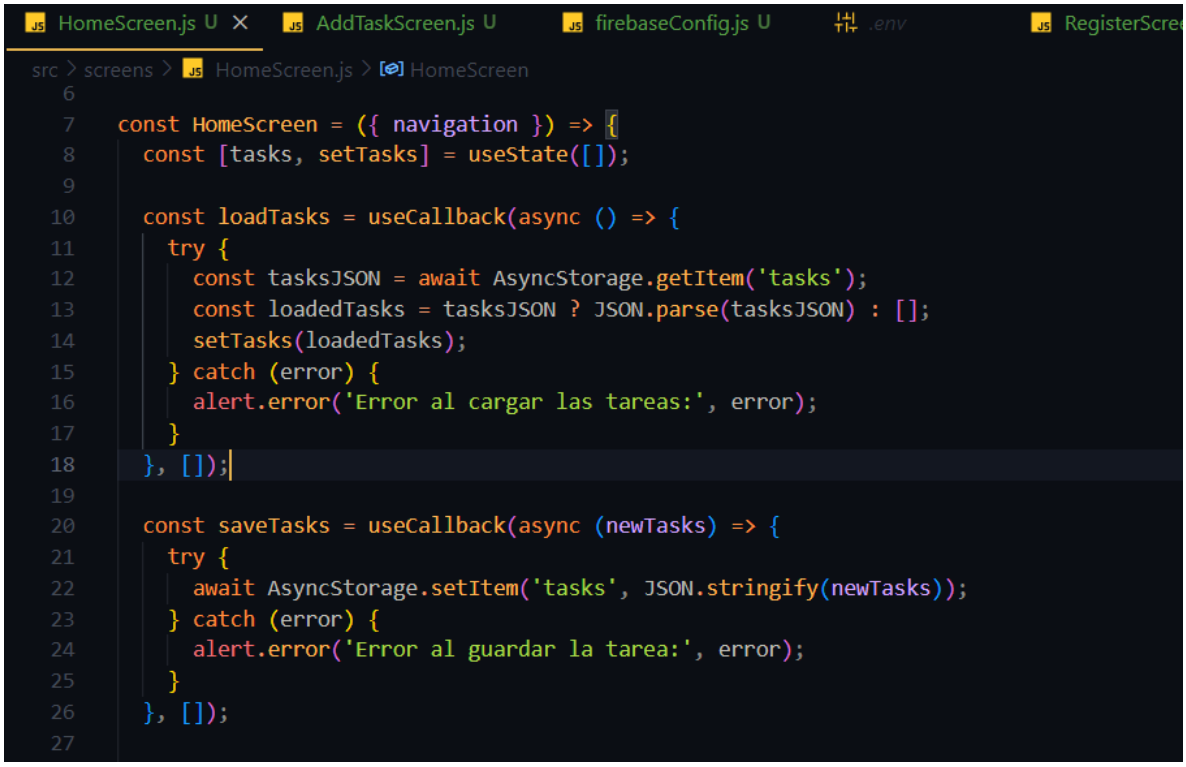
- The "firebaseConfig.js" file is created to consume the firebase service in our project

```
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



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



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