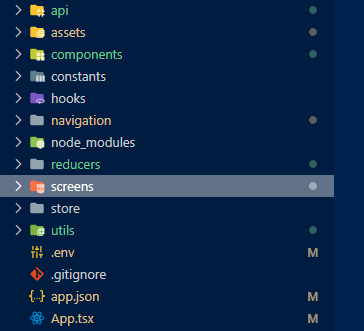
**Prueba 2 (React Native)**: Analiza las dos siguientes aplicaciones de geolocalización: [Find my Friends](https://apps.apple.com/us/app/find-my-friends/id466122094) y [F Circle](https://apps.apple.com/us/app/f-circle-location-finder/id1395341331) .  Por favor, haz un resumen detallado de cómo estructurarías una aplicación similar a nivel de programación (carpetas) y fija cuanto tiempo tardarías en desarrollar cada una de las partes. Cuanto más detallado sea, mejor.

Principalmente este sería el modelo estándar para cualquier frontal.

Determinar los tiempos de desarrollo de cada “carpeta” es muy relativo.

Lo cierto es que, por lo general, la carpeta ***components*** es la que mas tiempo lleva realizar pues es en dicha carpeta donde se encuentra la mayoría de las funcionalidades “features” que tiene la App.

**Imagen de carpetas de las Apps:**



**Detalle de carpetas:**

**Api:** Esta carpera contiene todas las Acciones de Redux, es decir, todas las peticiones al servidor backend oportuno.

**Assets:** En esta carpera se encuentra toda la estructura de archivos y recursos como logos, iconos, splash, también las fuentes, y además los ficheros de traducción de los distintos lenguages.

**Components:** Esta carpera contiene todos los elementos usados en nuestra App, es decir, aquellos elementos con los que formamos una Screen.

**Constans:** En esta carpeta se almacena todos los recursos que son invariantes en la app, colores, layouts, tamaños de fuentes, inclusive la configuración de los distintos idomas.

**Hooks**: En esta carpeta se almacenan los distintos Hooks que aporta funcionalidad a nuestra app, por ejemplo, hooks para formularios, para carga inicial de recursos, para el theme entre otros.

**Navigation:** Esta carpeta va a determinar el sistema de rutas de nuestra app, para mi gusto seguir el sistema de un MainNavigator que contenga todas las subrutas, suele ser clave para no perderse en una aplicación grande.

**Reducers:** En esta carpera están los Reducers, atrapan las acciones lanzadas por Redux para finalmente asignar el valor que viene en la respuesta del servidor.

**Screens:** En esta carpeta están las ventanas que tendrá nuestra App y estas a su vez se componen de los distintos componentes.

**Store:** Aquí va la configuración de Redux.

**Utils:** Normalmente todos los proyectos necesitan esta carpeta porque son necesarias tranformaciones, transcodificaciones, validaciones entre otras cosas, es el lugar ideal e indicado para toda esta funcionalidad necesaria para el correcto funcionamiento de nuestra app.

**.env:** Aquí van todas aquellas variables de entono que sabemos que no van a cambiar y que podemos parametrizar por nuestra app.

**.gitignore:** Aquello que no queremos que se suba a nuestro repositorio por ejemplo /node\_modules, todos sabemos lo que puede pesar…

**App.json:** Configuración de nuestra para para las distintas stores: Google play y Apple store.

**App.js/tsx:** Entrada / Main de nuestra aplicación.

Los periodos de realización son muy relativos, pero para una aplicación de este tipo el principal reto es encontrar un sistema de mapas económico cosa que Google maps no lo es, y utilizar el geolocalizador del dispositivo para el posicionamiento en el mapa de los distintos individuos.

Por lo demás podría llevarse a cabo en un periodo de 1 mes, mes y medio aproximadamente.

**Prueba 3 (React Native):** Conéctate a la API de Rick y Morty: <https://rickandmortyapi.com/>

Haz una app de React Native que tenga dos pantallas:

1. Inicio: listado de personajes de Rick y Morty, con resultados paginados
2. Detalle: detalle de un personaje al hacer click sobre él en el listado. La información debe ser: Nombre, Especie, Género, Estado (muerto/vivo), Imagen de perfil.

Deberás utilizar REDUX en la app, así como React Navigation. También, deberás compilar .apk en modo release para Android.

Como extra:

* Listar episodios en el detalle de un carácter y al hacer click sobre el episodio despliega en otra pantalla datos sobre ese episodio.
* Incluir buscador en el listado y que filtre resultados por palabras.

Enlace del código: <https://github.com/p4pupro/CarrotLab-Task3>