Guía de Cómo empezar un Proyecto Individual

- [ ] Antes de comenzar, te recomiendo **leer el README y la Documentación de la API Externa,** para comprender la temática que debes desarrollar y lo que se considera como requisito Obligatorio para estar en condiciones de presentar un PI:

Listado de Requisitos Indispensables:

- [ ] **Estilos**:
  - [ ] No se permitirá utilizar librerías externas para aplicar estilos a la aplicación
  - [ ] La *landing Page* debe tener alguna *imagen de fondo* representativa al proyecto
  - [ ] Elementos centrados y estilizados
  - [ ] Es requisito que el *formulario* de creación esté *validado con JavaScript* y no sólo con validaciones HTML
- [ ] **Filtrados y Ordenamientos:**
  - [ ] Para las funcionalidades de filtrado y ordenamiento NO pueden utilizar los endpoints de la API externa que ya devuelven los resultados filtrados u ordenados sino que debes realizarlo tu mismo
- [ ] **Enpoints de la API Externa**
  - [ ] Utilizar únicamente los Endpoints/Flags que están indicados en el Readme

- [ ] **Comencemos con el Backend**. Los pasos a seguir son:
  - [ ] Configurar el archivo .env
    - DB_USER= usuariodepostgres
    - DB_PASSWORD= passwordDePostgres
    - DB_HOST= localhost
    - API_KEY si tu proyecto lo necesita

  - [ ] Generar la base de datos
    - utiliza el comando para crearla o creala por medio de una interfaz gráfica: Ej, pgAdmin

- colocale el mísmo nombre que aparece en el archivo db.js
  En el siguiente ejemplo, pi sería el nombre de la BD

```
new Sequelize(`postgres://${DB_USER}:${DB_PASSWORD}@${DB_HOST}/pi`
```

- [ ] Realizados estos pasos, en la línea de comandos, debes posicionarte en la carpeta /api. Una vez allí, ya puedes ejecutar el comando *npm start*. Si el servidor está corriendo, deberías ver algo así:

```
[nodemon] starting `node index.js`
%s listening at 3001
```

- [ ] **Base de Datos**
  - [ ] Sigamos con la generación de Modelos de la Base de datos
    - [ ] Debemos generar el código para ambos modelos y tener en cuenta que en el README nos especifica cuáles campos son obligatorios (lo que te ayudará a utilizar validaciones y restricciones en cada campo, de ser necesario)
    - [ ] **IMPORTANTE:** *Busca la forma de generar un ID que no te traiga conflictos con los IDs que tienen los elementos traídos de la API Externa, existe por ejemplo, el identificador único universal o UUID : Investiga sobre esto ;)*
  - [ ] Luego del paso anterior, debemos aplicar destructuring de los modelos , en el archivo db.js. En este archivo, encontrarás comentarios que te indican dónde hacerlo y un ejemplo de cómo hacerlo.

```
// En sequelize.models están todos los modelos importados como propiedades
// Para relacionarlos hacemos un destructuring
const { Dog } = sequelize.models;
```

Seguido a esas líneas, encontrarás un comentario con un ejemplo para definir las relaciones entre modelos

```
// Aca vendrían las relaciones
// Product.hasMany(Reviews);
```

En todos los Proyectos Individuales, se plantea la necesidad de generar una relación de tipo N a N. Investiga en la documentación de sequelize sobre cómo definirla en forma correcta.

- [ ] Una vez realizados los modelos y las relaciones, podemos pensar en las **Rutas**: Recuerda leer el Readme, donde te indican cuáles son las rutas necesarias, además de si son de tipo GET o POST y si necesitan params o query params.

a) La ruta get que retorna todos los resultados, debe devolver solo los datos necesarios para la ruta principal (tanto los mostrados en cada card, como los necesarios para realizar los filtros y ordenamientos)
b) La ruta get por id utilizada para mostrar el detalle de cada elemento, debe traer solo los datos pedidos en la ruta de detalle (Según lo indicado en el Readme)

Recuerda que para usar librerías como, por ejemplo axios: deberás instalarla previamente

☐ Luego de hacer cada ruta, te conviene testearlas:
- Puedes usar algún cliente HTTP para realizar solicitudes a las API RESTful cliente HTTP como ser Postman o Insomnia para realizar solicitudes a las API RESTful

☐ **Front end**

Estilos: Intenta usar estilos uniformes en todo el sitio. Puedes buscar una paleta de colores y mantenerla, es recomendable usar la misma fuente y el mismo tamaño de letra, botones con el mismo estilo en todo el sitio y con el mismo color, para los que realizan la misma acción: Ej: borrar.
Consejo: Mira varios sitios web para ver la uniformidad en sus estilos

Landing Page:

☐ Mínimamente una imagen de fondo y un botón que redirija a la página principal

Rutas del Front:

☐ Rutas para cada una de las páginas que necesites (Landing, Página Principal, Detalle,etc)

Store de Redux:

☐ Configura el store para tener tu fuente de verdad y poder usarla donde la necesites

Página principal:

Aquí vas a renderizar los resultados obtenidos, cada uno en una card.
Además, existen otros elementos necesarios, enumerados a continuación:

☐ Paginado: Con la cantidad de elementos mencionados en el Readme (paginación)
☐ Search: Buscar por algún criterio. Lee en el Readme si la búsqueda debe ser exacta o que contenga la palabra ingresada
☐ Filtros: los resultados deben estar paginados
☐ Ordenamiento: Debe funcionar combinado con el/los filtro/s

Detalle de la Card:

- ☐ Debe tener los mismos campos de las cards de la página principal, más datos adicionales , según el Readme

Formulario Controlado

- ☐ Utilizar validaciones javascript
- ☐ Utiliza las validaciones para que tu formulario sea reactivo y valide datos a medida que completas cada campo
- ☐ Confirma si el elemento se ha creado correctamente
- ☐ Si ocurre algún error en el backend: comunicarlo a los usuarios de tu página
- ☐ Al finalizar la creación: limpia los campos de tu formulario

☐ **Puntos Extras**

TEST

- ☐ Al menos tener un componente del frontend con sus tests respectivos
- ☐ Al menos tener una ruta del backend con sus tests respectivos
- ☐ Al menos tener un modelo de la base de datos con sus tests respectivos

BUENAS PRÁCTICAS

- ☐ Utilizar código modularizado. Reutilizar componentes en el front end. Usar helpers en el Back end

EXTRA FEATURES

- ☐ Agregar funcionalidades extras, que no fueron solicitadas en el Readme