# Taller Repaso: Recursos anidados

El objetivo de esta actividad es la implementación de relaciones 1 a N en un proyecto utilizando Ruby on Rails y el manejo de lectura (consulta), creación, edición y eliminación de estas.

Una vez finalizada la actividad, Sube tu proyecto a Github, y luego, añade el link en el desafío correspondiente en la plataforma.

# Requerimientos:

- Una empresa llamada "Musify" requiere la creación de su plataforma donde poder gestionar usuarios, playlists y canciones.
    - El cliente especifica que:
        - Un usuario puede tener varios playlists.
        - Un playlist pertenece a un usuario.
        - Un playlist puede tener varias canciones.
        - Una canción pertenece sólo a un playlist.
    - Se debe poder crear, leer, actualizar y eliminar usuarios, playlists y canciones.

# Actividad:

- Crear un nuevo proyecto en Rails 5.1 utilizando **PSQL** como motor de base de datos.
- Crear un scaffold de **User** con los campos name(string) y email(string).
- Revisar y correr la migración correspondiente.
- Crear un scaffold de **Playlist** con los campos name(string) y user(references).
- Revisar y correr la migración correspondiente.
- Validar en el modelo **User** que el nombre debe estar presente y el correo debe estar presente y ser único.
- Validar en el modelo **Playlist** que el campo name debe estar presente y ser único.
- **Primer Commit**.

- Crear 10 usuarios de prueba a través del **Seed** utilizando la gema **Faker** y sus módulos para nombres y correos.
- En el formulario de la vista new de playlists, debo poder **seleccionar un usuario existente a través de una lista desplegable** (utilizar el helper de Rails correspondiente para generar la colección desplegable).

  > Formulario con asociación.

- En la vista **index** de playlists debo incluir, en la tabla, el **nombre del usuario** al que pertenece el playlist.
- En la vista **show** de playlists debo poder ver el nombre del playlist y el nombre del usuario al que pertenece.
- Crear el modelo **Song** con los campos artist(string), name(string) y playlist(references).
- Revisar y correr la migración correspondiente.
- Validar en el modelo Song que los campos artist y name deben estar persentes.
- **Segundo Commit**.
- Crear el controller **songs**.
- En la vista show de playlist debo **incluir un formulario de ingreso de una nueva canción**.
  - Crear la ruta para el método 'songs#create' **anidada** a playlist utilizando arquitectura **rest**.
  - Crear el **formulario anidado** correspondiente utlizando el helper **form_with**:
    - En el atributo **model** el formulario debe recibir ambos objetos: **playlist** y **song**.
    - Donde **playlist** es el objeto correspondiente y **song** es una instancia no persistente de Song.
  - Crear el método 'songs#create' donde **almacenaremos una nueva canción asociada al playlist** y luego debe redirigir al playlist correspondiente con un mensaje: 'La canción se ha creado con éxito!'

    > Utilizar strong params y asignación masiva.

- En la vista **show** de playlist debemos listar, bajo el formulario, **las canciones que posee ese playlist**.
- Crear la ruta para el método 'songs#destroy' **anidada** a playlist utilizando arquitectura **rest**.
- Cada canción listada en 'playlists#show' debe tener un **link al lado para ser eliminada**.
- Crear el método 'songs#destroy' que debe **eliminar la canción y luego redirigir** a la vista show del

playlist correspondiente con un mensaje: 'La canción se ha eliminado con éxito!'

- **Tercer Commit**.
- En la vista Index de users debo poder ver, en la tabla, la cuenta de playlists que posee cada usuario.
- En la vista Index de playlists debo poder ver, en la tabla, la cuenta de canciones que posee cada playlist.
- Agregar los CDN de Bootstrap 4 y sus dependencias.
- Agregar un navbar de Bootstrap al layout (application.html)
- El navbar debe contener links a los Index de users y playlists.
- **Cuarto Commit**.
- Push del proyecto a repositorio GitHub.