# Workshop NodeJs + AngularJs:

Desarrollar una SPA que permite explorar contenido musical via Banda, Artista y Álbum, con posibilidad de comentar cada Track musical (Alta, baja y modificación de tracks).

## Repositorios de backend y frontend:

NodeJs: https://github.com/globant-ui/workshop2016-nodeconfar-express

AngularJs: https://github.com/globant-ui/workshop2016-nodeconfar-angular

`git clone ${repo} ${folderName}`

# Producto:

## Definición técnica:

- Esta aplicacion constara back-end (NodeJs) y front-end (AngularJs)
- Será una demostración de integración entre cliente y servidor.
- La app debe ser solo version desktop (1280x768px)
- El usuario no necesita hacer login
- La interfaz debe ser fluida en cuanto a UX
- Se utilizará bootstrap como framework UI
- El usuario puede comentar un track y puede ver los comentarios dados. También puede modificar y borrar los comentarios.
- El tiempo es limitado, debes gestionarlo para alcanzar el deadline (priorizando features segun tu criterio)
- Usaremos un flujo “agile” básico de trabajo para tener conciencia del estado del desarrollo

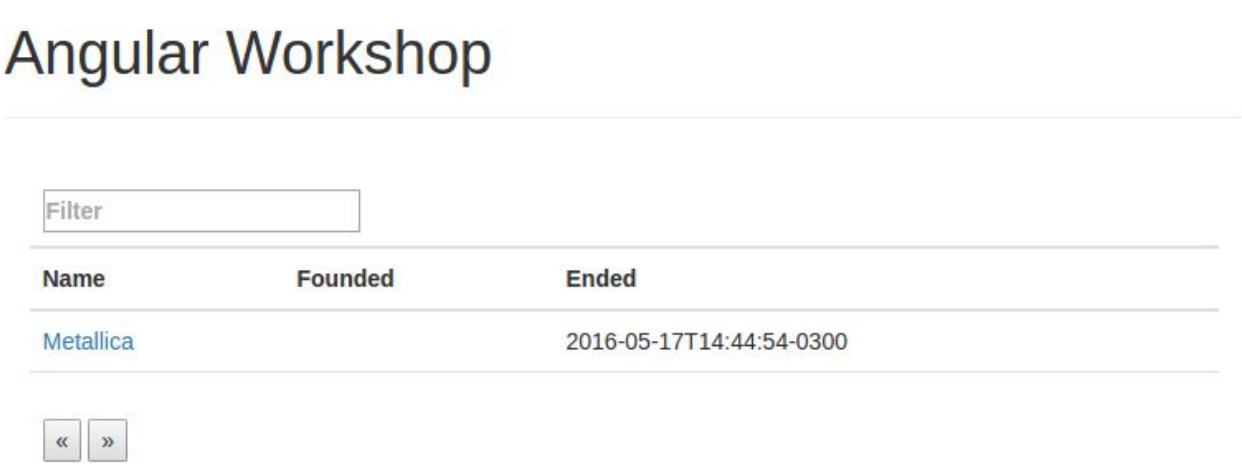
**Features tecnicas:**

- ES6
- RESTful
- Express
- NEDB (memory database for Node.js)
- Deep-linking / Routing
- Angular 1.5 (Utilización de componentes)
- Webpack

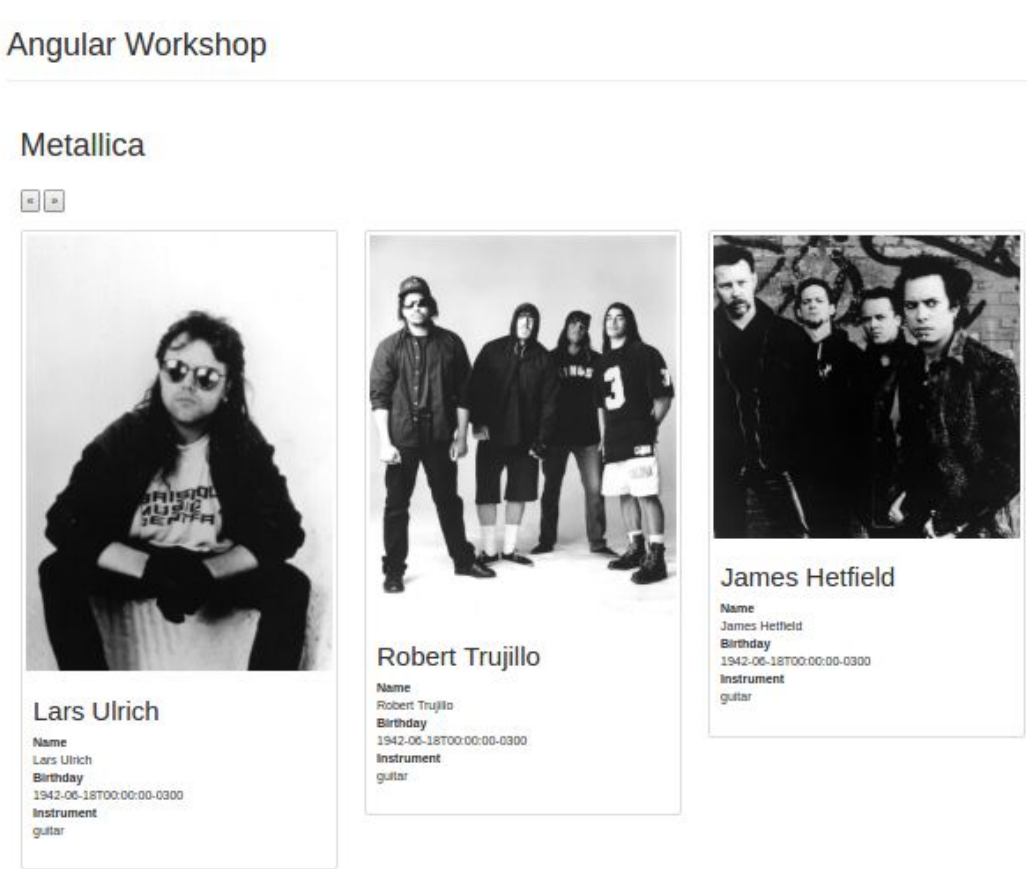
App:

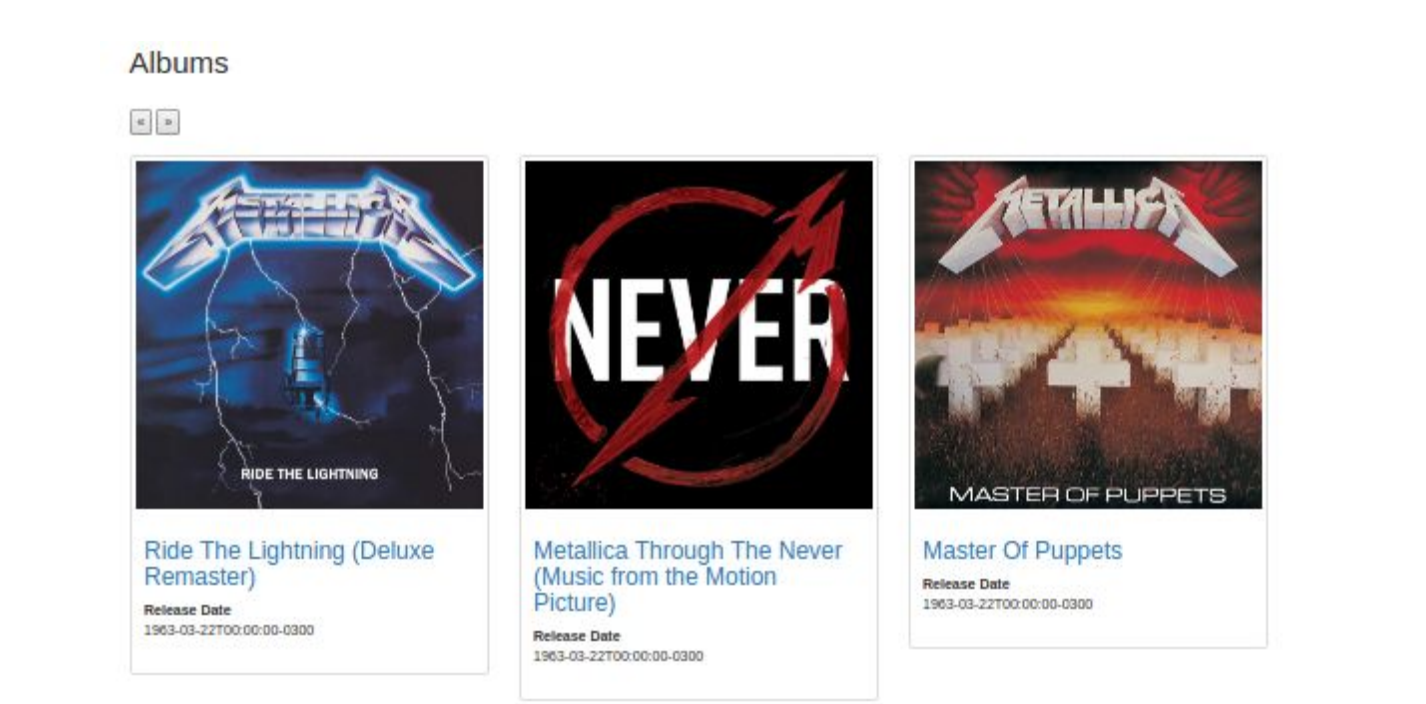
Wires:

**Home:**




**Artists**:






**Tracks and comments**

## Tracks in Ride The Lightning (Deluxe Remaster)

| Name | Track # | Length | Comments |
|---|---|---|---|
| For Whom The Bell Tolls (Remastered) | 3 | 309973 | 0 |
| Ride The Lightning (Remastered) | 2 | 396986 | 2 |
| Fade To Black (Remastered) | 4 | 417226 | 2 |

- 11/8/2016 My favorite song of all time :D
- 11/8/2016 Awesome Track!

Submit

## Flow:

### Home:

- El usuario inicia su navegación desde el listado de bandas
- El usuario puede filtrar por nombre (filtro local)
- Al acceder a una banda
    - mostrar artistas y álbumes (detail)
    - ocultar listado de bandas (home)

### Detail:

- Al cliquear sobre un álbum mostrar tracks (sin salir de "detail", todo en la misma pantalla)
- Mostrar una imagen representativa de: Artistas y Álbum (utilizar un dummy).
- Paginado horizontal para Artistas y Albumes.
- Utilizar el comportamiento “hover” del mouse para mostrar mas info
    - Artista: Name, Birthday, Instrument
    - Album: Name, Release
- Los tracks deben tener paginado.
- Se debe resaltar el álbum seleccionado
- Se debe mostrar el número de comentarios.

- Mostrar paginado solo si supera los 10 tracks

Comments:

- Al cliquear sobre un track debe:
    - Marcar el track seleccionado.
    - Mostrar el form de comentarios.
- Al “submit” del comentario
    - Debe actualizarse en la base de datos
    - Debe actualizarse la lista de tracks (al menos el icono de comentario)
- Mostrar el form de comentario solo si esta seleccionado un track.

Generales:

- Hacer que el paginado y filtros trabajen de manera local (sin request al servidor por cambios)

# Tasks:

## FRONT-END:

### Home:

FE01) Mostrar Lista de bandas
    FE01.1) Crear llamadas API al servidor, usar data mock.
    FE01.2) Pasar data al componente tabla.
    FE01.3) Al clickear sobre una banda pasar a la pagina de detalles.

### Detail:

FE02) Mostrar Artistas de la banda
    FE02.1) Crear ruta para los detalles de la banda
    FE02.2) Crear llamadas API al servidor para conseguir los artistas (data mock).
    FE02.3) Usar las llamadas API para recibir data en la ruta de la pagina de detalles.
    FE02.4) Iterar sobre los resultados para mostrar la lista de artistas con un markup HTML acorde.

FE03) Mostrar Albums
    FE03.1) Crear ruta para los albums
    FE03.2) Crear llamadas API al servidor para conseguir los albums (data mock).
    FE03.3) Usar las llamadas API para recibir data en la ruta de la pagina de detalles (router).
    FE03.4) Iterar sobre los resultados para mostrar la lista de álbumes con un markup HTML acorde.

FE04) Mostrar Tracks
    FE04.1) Crear ruta para los tracks
    FE04.2) Crear llamadas API al servidor para conseguir los tracks (data mock).
    FE04.3) Usar las llamadas API para recibir data en la ruta de la pagina de detalles (router).
    FE04.4) Iterar sobre los resultados para mostrar la lista de tracks con un markup HTML acorde.

FE05) Mostrar Comentarios (mensaje y fecha)
    FE05.1) Crear ruta para los comentarios
    FE05.2) Crear llamadas API al servidor para conseguir los comentarios (data mock).
    FE05.3) Usar las llamadas API para recibir data en la ruta de la pagina de detalles (router).
    FE05.4) Iterar sobre los resultados para mostrar la lista de tracks con un markup HTML acorde.

FE06) Conectar con servicios reales
    FE06.1) Remover mocks
    FE06.2) Llamar a los servicios creados

FE07) Enhancements
    FE07.1) Implementar filtrado de resultados.
    FE07.2) Crear un componente común para el listado de artistas, albums y tracks.
    FE07.3) Implementar paginador en listado de artistas y albums.
    FE07.4) Implementar borrado de comentarios

## BACK-END:

BE01) Bases de datos y configuración.

- ~~BE01.01)~~ Dar de alta a la base de datos por medio del script **dbseed - DONE**
  *Previamente se va a explicar como funciona neDB, sus similitudes con mongoDB y como utilizarla a base de Promises en el desarrollo de esta aplicación*
- BE01.02) Crear un archivo de texto .md donde esté expresado cual es el contrato que se establecerá con el FrontEnd (utilizar la base que se encuentra en el README.md)
  *El archivo debe ser explícito en cuanto a qué rutas se van a crear, qué payload se va a recibir y qué datos se van a retornar.*

BE02) Endpoints. Retornar datos relacionados a las bandas.

*(A modo de ejemplo se va a dejar un controller y un model, asi como tambien el archivo del router con dos rutas habilitadas)*

- BE02.01) Agregar al archivo del router las rutas necesarias para:
  - *Retornar el listado de bandas*
  - *Retornar una banda en particular, recibirá como parámetro un "id"*
  - *Retornar los albumnes de una banda en particular, recibirá como parámetro un "id"*
  - *Retornar los artistas que pertenecen a una banda en particular, recibirá como parámetro un "id"*

- BE02.02) Poner en funcionamiento la ruta para retornar el listado de bandas:
  - *Crear un nuevo model "BandModel" con un método que retorne el listado completo de bandas de la base de datos*
  - *Crear un nuevo controller "BandController" con un método que llame a "BandModel" para devolver el listado completo de bandas.*
  - *Asignar dicho método a la ruta correspondiente*

- BE02.03) Poner en funcionamiento la ruta para retornar una banda en particular:
  - *Agregar un método a "BandModel" que permita retornar los datos de una sóla banda en particular a partir de su "id"*
  - *Agregar un método a "BandController" que reciba el "id" de una banda desde el router. Con ese "id" llamar a "BandModel" para devolver los datos de la banda.*
  - *Asignar dicho metodo a la ruta correspondiente*

- BE02.04) Poner en funcionamiento la ruta para retornar los albumnes de una banda en particular:
  - *Agregar un método a "BandModel" que permita obtener un listado de los albumnes pertenecientes a una banda en particular a partir del "id" de la banda.*
  - *Agregar un método a "BandController" que reciba el "id" de una banda desde el router. Ese "id" debe ser usado para llamar a "BandModel" y devolver el listado de albumnes.*
  - *Asignar dicho método a la ruta correspondiente*

- BE02.05) Poner en funcionamiento la ruta para retornar los artistas de una banda en particular:
    - *Agregar un método a "BandModel" que permita obtener un listado de los artistas pertenecientes a una banda en particular a partir del "id" de la banda.*
    - *Agregar un método a "BandController" que reciba el "id" de una banda desde el router. Ese "id" debe ser usado para llamar a "BandModel" y devolver el listado de artistas.*
    - *Asignar dicho método a la ruta correspondiente*

BE03) Endpoints. Retornar los datos relacionados a los albumnes.

- BE03.01) Dar de alta a la ruta
    - *Agregar una nueva ruta al archivo del router que permita retornar el listado de los tracks pertenecientes a un album en particular. Recibirá como parametro un "id"*
- BE03.02) Poner en funcionamiento la ruta
    - *Crear un model "AlbumModel" con un método que permita retornar la lista de tracks de un album en particular a partir de su "id"*
    - *Crear un controller "AlbumController" con un método que reciba el "id" de una banda desde el router. Con ese "id" llamar a "AlbumModel" para devolver la lista de tracks del album.*
    - *Asignar dicho método a la ruta correspondiente*

BE04) Endpoints. Agregar comentarios en los tracks.

- BE04.01) Dar de alta a la ruta
    - *Agregar una nueva ruta al archivo del router que permita al usuario escribir un comentario en un track determinado.*
    - *Crear un model "CommentModel" con un método que permita agregar un comentario a la base de datos. Recibirá como parámetros el id del track, un nombre y un mensaje.*
- BE04.02) Poner en funcionamiento la ruta
    - *Crear un controller "CommentController" con un método que reciba el "id" del track, el "nombre" y el "mensaje" dentro del body de la request. Con estos datos llamar a "CommentModel" para dar de alta al comentario*
    - *Asignar dicho método a la ruta correspondiente*

### Extras:

BE05) Endpoints. Eliminar comentarios.
BE06) Registro de usuarios.
BE07) Login de usuarios.
BE08) Proteger las rutas con un JWT.

## Comandos útiles de git:

- Descargar y actualizar desde el repositorio

```
git pull
```

- Crear commit para subir cambios al repositorio

```
git add .
git commit -m "Mis cambios"
```

- Subir los cambios

```
git push origin <nombre-branch>
```