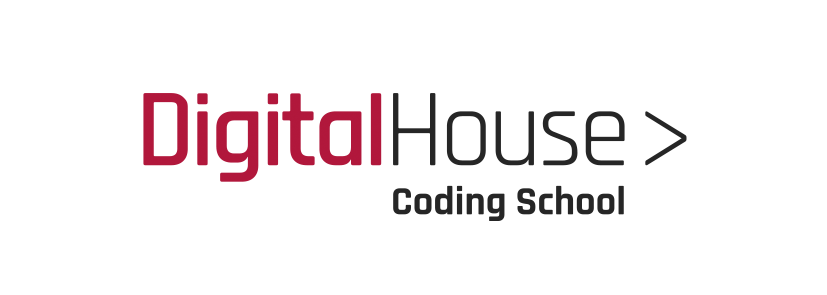
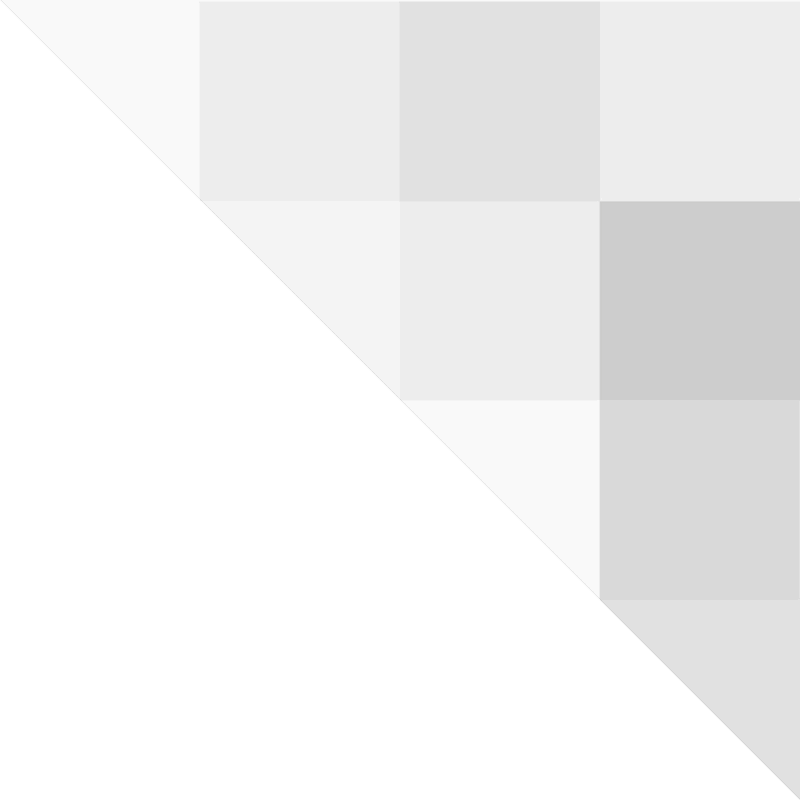
****

**Programación III**

Proyecto Integrador

**Primera entrega**

**Fecha de entrega : ~~27/09~~ 20/9**

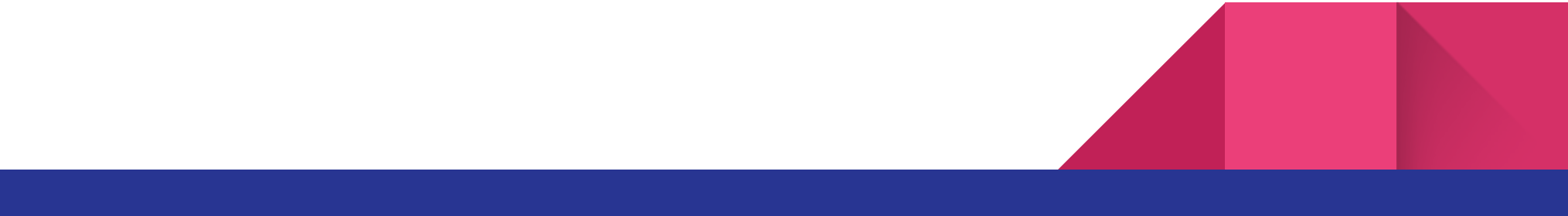
# Condiciones generales

El **proyecto integrador** es grupal y se realiza de a grupos **de 2 o 3 personas**. Los alumnos pueden elegir sus compañeros de trabajo libremente. Deben confirmar los grupos de trabajo vía Slack hacia la **cátedra para el día 17/8** En caso de tener algún conflicto es fundamental que lo avisen previamente.

**La fecha**  de entrega del proyecto integrador será el  **~~27 de septiembre~~** **20 de septiembre**. Las presentaciones se harán en forma grupal vía zoom.

El equipo deberá organizarse para que cada integrante de manera equilibrada pueda sin excepción:

* Contar funcionalidades incluidas en una o varias secciones de la aplicación.
* Explicar técnicamente **en detalle** una sección del código hecha por quien expone. La exposición implica contar **que va haciendo el código** **línea por línea incluyendo** los **conceptos teóricos** que lo respaldan. **Serán inválidas las exposiciones que carezcan de sustento teórico y/o no puedan detallar paso a paso lo que sucede.** Durante la presentación los profesores podrán hacer preguntas **individuales** a los alumnos para precisar conceptos o validar información que hayan expresado.

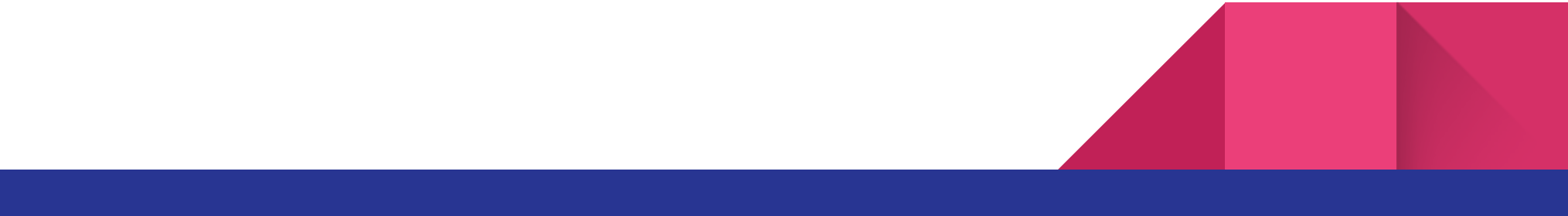
La nota de la presentación oral es individual\* mientras que la del trabajo en si es grupal. La entrega se realizará mediante repositorio del proyecto en Github.

Es importante que utilicen GIT durante el proyecto y no únicamente para la entrega. Estaremos observando el commit log y quienes realizan dichos commits. Esto **afecta tanto a la nota del trabajo como a la nota de concepto**.

(\*)Nota: La **aprobación de la presentación individual** es **condición excluyente** para la aprobación de la entrega. La desaprobación de la presentación individual inhabilita el cómputo de los puntos obtenidos en la realización del proyecto.

Para el desarrollo del proyecto contarán con un [***proyecto base que pueden descargar aquí***](https://github.com/alejandrodh/maquetaReact.git). En el repositorio encontrarán una carpeta wireframe guía y la estructura HTML. No se proveen archivos CSS. Cada grupo deberá aplicar el estilo css necesario para organizar las tarjetas de acuerdo al wireframe. La estética general de la app queda a criterio de cada grupo.

# **Consignas**

El proyecto consiste en realizar una aplicación en REACT que deberá:

* Obtener información de una API y mostrarla en forma de tarjetas.
* Sumar funcionalidades globales y funcionalidades en cada tarjeta.

A su vez:

* La estructura de archivos deberá respetar el formato y orden vistos en clase.
* La estética del sitio debe respetarse en todos los componentes creados. (mantener coherencia estética) y deberá respetar diseño responsive.
* El footer deberá ser actualizado con la información correspondiente de cada grupo.
* El proyecto debe realizarse **SIN EXCEPCIÓN** con las estructuras de contenido vistas durante el presente curso.

Punto 1 - API

La información a mostrar se obtendrá de las APIs vistas en Programación 1. Las APIs a utilizar son las de **TMDB** y **Deezer**. Cada grupo elegirá **UNA** de las 2 para trabajar y **UN** endpoint del cual obtendrán datos. El endpoint elegido deberá ser uno de los siguientes:

**TMBD API**

Para trabajar con la API deberán crear una cuenta en <https://www.themoviedb.org/login> y buscar la API Key en <https://www.themoviedb.org/settings/api>

Recuerden que la API Key debe colocarse como parte de la url del endpoint consultado. En cada endpoint tiene la opción ***“send request”*** con el modelo de url. Ante cualquier duda, consulten con sus profesores.

**Endpoints de películas**

**Movies Popular**

<https://developers.themoviedb.org/3/movies/get-popular-movies>

**Movies Top rated**

<https://developers.themoviedb.org/3/movies/get-top-rated-movies>

**Movies latest**

<https://developers.themoviedb.org/3/movies/get-latest-movie>

**Movies Now Playing**

<https://developers.themoviedb.org/3/movies/get-now-playing>

**Movies upcoming**

<https://developers.themoviedb.org/3/movies/get-upcoming>

**Endpoints de series**

**Series Popular**

<https://developers.themoviedb.org/3/tv/get-popular-tv-shows>

**Series Top rated**

<https://developers.themoviedb.org/3/tv/get-top-rated-tv>

**Series latest**

<https://developers.themoviedb.org/3/tv/get-latest-tv>

**Series airing today**

https://developers.themoviedb.org/3/tv/get-tv-airing-today

**Series on the air**

<https://developers.themoviedb.org/3/tv/get-tv-on-the-air>

**DEEZER API**

Para trabajar con la API deberán crear una cuenta en https://developers.deezer.com/

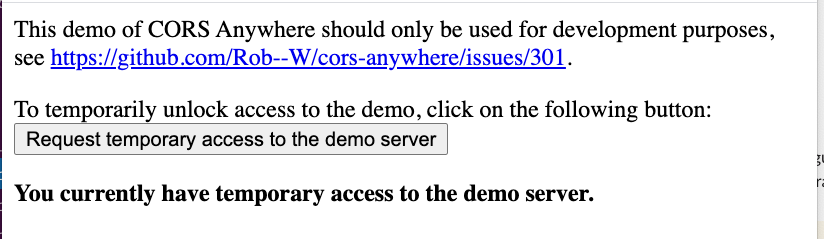
La documentación de los endpoints se encuentra en: <https://developers.deezer.com/api>

Para poder utilizar la API sin problemas deben agregar este proxy (<https://cors-anywhere.herokuapp.com/> ) previo a la url del endpoint. Por ejemplo para consultar por la lista de **top albums** el end point debe quedar así:

*https://cors-anywhere.herokuapp.com/https://api.deezer.com/chart/0/albums*

**Importante:** si reciben un **error 403** por consola al usar el proxy ingresen en el siguiente link y denle **click al botón** *“request temporary access to the demo server”.*

[*https://cors-anywhere.herokuapp.com/corsdemo*](https://cors-anywhere.herokuapp.com/corsdemo)



Dejamos los endpoints que pueden utilizar. Recordá que todos los detalles están en la documentación de la API.

1. **Tracks**

API docs para charts: <https://developers.deezer.com/api/explorer?url=chart/0/tracks>

Endpoint: <https://api.deezer.com/chart/0/tracks>

1. **Albums**API docs para albums: <https://developers.deezer.com/api/explorer?url=chart/0/albums>

Endpoint: <https://api.deezer.com/chart/0/albums>

1. **Artists**

API docs para artists: <https://developers.deezer.com/api/explorer?url=chart/0/artists>

Endpoint: <https://api.deezer.com/chart/0/artists>

Punto 2 - Visualización de tarjetas

Utilizando la API elegida en el punto 1 debemos visualizar una página que muestre 10 tarjetas de presentación con los siguientes elementos:

1. Foto.
2. Un nombre o título.
3. Una descripción.
4. Link o botón “ver más”.
5. Zona de Información adicional que se mostrará al hacer click en “ver más”.
6. Link o botón eliminar.
7. Flechas o menú izq/ der para mover las tarjetas.

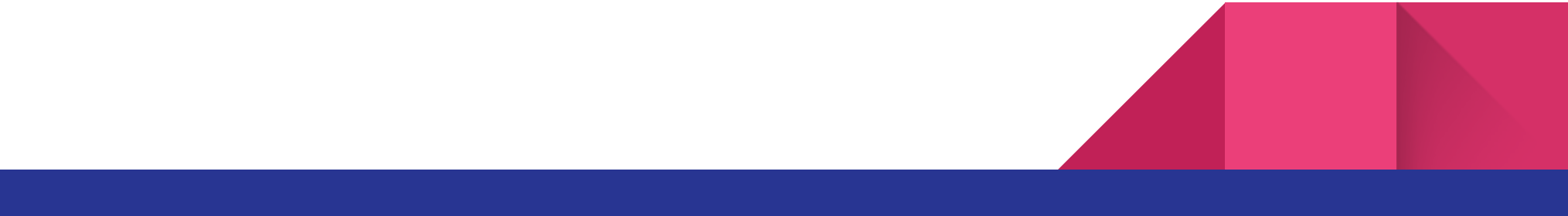
Punto 3 - Funcionalidades a desarrollar

1. **“Ver más”** en tarjetaLos datos adicionales de la tarjeta definidos en el punto anterior se mostrarán únicamente si el usuario clickea en la opción “ver más”.  
   Si los datos adicionales se están mostrando hacer click sobre el menú/ botón deberá ocultar los datos adicionales.  
   Cada equipo deberá elegir 3 datos adicionales para mostrar.
2. **Eliminar tarjetas**Clickear en el botón/ link “eliminar” deberá eliminar la tarjeta de la lista.
3. **Agregar tarjetas**

La app debe contar con la posibilidad de agregar un nuevo set de tarjetas mediante una nueva consulta a la API. La funcionalidad permitirá adicionar 10 nuevas tarjetas.

1. **Loader**Previo a la carga de tarjetas debe verse un gif animado, spiner de carga o una leyenda “Cargando…”
2. **Filtro de tarjetas/ búsquedaLa app debe permitir** filtrar las tarjetas en base a UNO SOLO de los criterios propuestos:
3. Título/ nombre
4. descripción

Si no hay resultados debe mostrarse un aviso al usuario: ***“No hay datos que coincidan con su búsqueda"***.

1. **~~SORT/ Ordenar~~** (No aplica para la entrega)~~Ordenar las tarjetas de forma ascendente o descendente tomando el campo título/ nombre.  
   La funcionalidad debe aplicar también sobre las tarjetas filtradas/ los resultados de búsqueda.~~
2. **~~Cambiar la ubicación de las tarjetas.~~** (No aplica para la entrega).~~Mediante el uso de flechas/ menú izq/ der permitir que las tarjetas cambien su posición en la lista.~~
3. **Orientación interna de la tarjeta**Permitir cambiar la visualización de las tarjetas: de columnas a filas. (Ejemplo: la disposición de los productos en mercadolibre)  
   