VIDEOGAMES | Proyecto Individual

□ OBJETIVOS

- Construir una Single Page Application utlizando las tecnologías: **React**, **Redux**, **Node**, **Express** y **Sequelize**.
- Poner en práctica recursos básicos de estilos y diseño (UX : UI).
- Afirmar y conectar los conceptos aprendidos en la carrera.
- Aprender mejores prácticas.
- Aprender y practicar el workflow de GIT.
- Utilizar y practicar testing.

□ HORARIOS Y FECHAS

El proyecto individual tiene una duración máxima de tres semanas. Se inicia la primera semana con un Kick-Off, y se agendará una corrección personalizada la última semana.

En el caso de completar todas las tareas antes de dicho lapso se podrá avisar a su instructor para coordinar una fecha de presentación del trabajo (DEMO).

□□ IMPORTANTE

Es necesario contar minimamente con la última versión estable de NodeJS y NPM. Asegúrate de contar con ella para poder instalar correctamente las dependecias necesarias para correr el proyecto. Actualmente las versiónes necesarias son:

- **Node**: 12.18.3 o mayor
- **NPM**: 6.14.16 o mayor

Para verificar que versión tienes instalada:

```
node -v
npm -v
```

ACLARACIÓN: las dependencias actuales se encuentran en las versiones que venimos trabajando durante el bootcamp.

- **react**: 17.0.1
- **react-dom**: 17.0.1
- **react-router-dom**: 5.2.0
- **redux**: 4.0.5
- **react-redux**: 7.2.3

Está permitido, **bajo tu responsabilidad**, actualizar las dependencias a versiones más actuales si lo deseas. Versiones mas actuales podrían presentar configuraciones diferentes respecto a las versiones en las que venimos trabajando durante el bootcamp.

□□ Está rotundamente prohibido utilizar librerías externas para aplicar estilos a la SPA. Tendrás que utilizar CSS mediante algunas de las opciones vistas en el bootcamp (CSS puro, CSS Modules o Styled Components).

□ PARA COMENZAR...

1. Deberás forkear este repositorio para tener una copia del mismo en tu cuenta personal de GitHub.
2. Clona el repositorio en tu computadora para comenzar a trabajar. Este repositorio contiene un `BoilerPlate` con la estructura general del proyecto, tanto del servidor como del cliente. El boilerplate cuenta con dos carpetas: `api` y `client`. En estas carpetas estará el código del back-end y el front-end respectivamente.
3. En la carpeta `api` deberás crear un archivo llamado: `.env` que tenga la siguiente forma:

```
DB_USER=usuariodepostgres
DB_PASSWORD=passwordDePostgres
DB_HOST=localhost
```

4. Reemplazar `usuariodepostgres` y `passwordDePostgres` con tus propias credenciales para conectarte a postgres. Este archivo va ser ignorado en la subida a github, ya que contiene información sensible (las credenciales).
5. Adicionalmente será necesario que crees, **desde psql (shell o PGAdmin)**, una base de datos llamada `videogames`. Si no realizas este paso de manera manual no podrás avanzar con el proyecto.

ENUNCIADO GENERAL

La idea de este proyecto es construir una aplicación web a partir de la API **rawg** en la que se pueda:

- Buscar videojuegos.
- Visualizar la información de los videojuegos.
- Filtrarlos.
- Ordenarlos.
- Crear nuevos videojuegos.

Para las funcionalidades de filtrado y ordenamiento NO se puede utilizar los endpoints de la API externa que ya devuelven los resultados filtrados u ordenados.

IMPORTANTE: para poder utilizar la API es necesario crear una cuenta y obtener una ApiKey que luego deberá ser incluida en todos los request que hagamos. Esto se logra simplemente agregando `?api_key={YOUR_API_KEY}` al final de cada end-point. Agregar la clave en el archivo `.env` para que la misma no se suba al repositorio por cuestiones de seguridad.

Únicos end-points que se pueden utilizar

- **Rawg**
- **Por id**: *"https://api.rawg.io/api/games/{id}"*
- **Por nombre**: *"https://api.rawg.io/api/games?search={game}"*
- **Por genero**: *"https://api.rawg.io/api/genres"*

INSTRUCCIONES

BASE DE DATOS

Deberás crear dos modelos para tu base de datos. Una será para los videojuegos y la otra será para los géneros (pueden llevar el nombre que tu quieras). La relación entre ambos modelos debe ser de muchos a muchos. A continuación te dejamos las propiedades que debe tener cada modelo.

MODELO 1 | Videogames

- ID (deben ser distintos a los que vienen de la API). *
- Nombre. *
- Descripción. *
- Plataformas. *
- Imagen. *

- Fecha de lanzamiento. *
- Rating. *

MODELO 2 | Genres

- ID. *
- Nombre. *

BACK-END

Para esta parte deberás construir un servidor utilizando **NodeJS** y **Express**. Tendrás que conectarlo con tu base de datos mediante **Sequelize**.

Tu servidor deberá contar con las siguientes rutas:

GET | /videogames

- Obtiene un arreglo de objetos, donde cada objeto es un videojuego con su información.

GET | /videogames/:idVideogame

- Esta ruta obtiene el detalle de un videojuego específico. Es decir que devuelve un objeto con la información pedida en el detalle de un videojuego.
- El videojuego es recibido por parámetro (ID).
- Tiene que incluir los datos del género del videojuego al que está asociado.
- Debe funcionar tanto para los videojuegos de la API como para los de la base de datos.

GET | /videogames/name?="..."

- Esta ruta debe obtener los primeros 15 videojuegos que se encuentren con la palabra recibida por query.
- Debe poder buscarlo independientemente de mayúsculas o minúsculas.
- Si no existe el videojuego, debe mostrar un mensaje adecuado.
- Debe buscar tanto los de la API como los de la base de datos.

POST | /videogames

- Esta ruta recibirá todos los datos necesarios para crear un videojuego y relacionarlo con sus géneros solicitados.
- Toda la información debe ser recibida por body.
- Debe crear un videojuego en la base de datos, y este debe estar relacionado con sus géneros indicados (al menos uno).

GET | /genres

- Obtiene un arreglo con todos los géneros existentes de la API.
- En una primera instancia, cuando la base de datos este vacía, deberás guardar todos los géneros que encuentres en la API.
- Estos deben ser obtenidos de la API (se evaluará que no haya hardcodeo). Luego de obtenerlos de la API, deben ser guardados en la base de datos para su posterior consumo desde allí.

FRONT-END

Se debe desarrollar una aplicación utilizando **React** y **Redux** que contenga las siguientes vistas:

LANDING PAGE | deberás crear una página de inicio o bienvenida con:

- Alguna imagen de fondo representativa al proyecto.
- Botón para ingresar a la `home page`.

HOME PAGE | la página principal de tu SPA debe contener:

- SearchBar: un input de búsqueda para encontrar videojuegos por nombre.
- Sector en el que se vea un listado de cards con los videojuegos. Al iniciar deberá cargar los primeros resultados obtenidos desde la ruta `GET /videogames` y deberá mostrar su:
  - Imagen.
  - Nombre.
  - Géneros.
- Cuando se le hace click a una Card deberá redirigir al detalle de ese videojuego específico.
- Botones/Opciones para **filtrar** por género, y por si su origen es de la API o de la base de datos (creados por nosotros desde el formulario).
- Botones/Opciones para **ordenar** tanto ascendentemente como descendentemente los videojuegos por orden alfabético y por rating.
- Paginado: el listado de videojuegos se hará por partes. Tu SPA debe contar con un paginado que muestre un total de 15 videojuegos por página.

IMPORTANTE: se deben mostrar tanto los videojuegos traidos desde la API como así también los de la base de datos, pero **NO** está permitido almacenar en la base de datos los videojuegos de la API. **Solamente se pueden guardar aquellos creados desde el form**.

 IMPORTANTE: debido a que en la API existen alrededor de 500.000 videojuegos, por cuestiones de performance puedes tomar la simplificación de obtener y **paginar** los primeros 100 videojuegos.

 DETAIL PAGE | en esta vista se deberá mostrar toda la información específica de un videojuego:

- ID.
- Nombre.
- Imagen.
- Plataformas.
- Descripción.
- Fecha de lanzamiento.
- Rating.
- Géneros.

 FORM PAGE |: en esta vista se encontrará el formulario para crear un nuevo videojuego.

Este formulario debe ser **controlado completamente con JavaScritp**. No se pueden utilizar validaciones HTML, ni utilizar librerías especiales para esto. Debe contar con los siguientes campos:

- Nombre.
- Imagen.
- Descripción.
- Plataformas.
- Fecha de lanzamiento.
- Rating.
- Posibilidad de seleccionar/agregar varios géneros en simultáneo.
- Botón para crear el nuevo videojuego.

> [**IMPORANTE**]: es requisito que el formulario de creación esté validado sólo con JavaScript. Puedes agregar las validaciones que consideres. Por ejemplo: que el nombre del videojuego no pueda contener símbolos, o que el rating no pueda exceder determinado valor, etc.

□ TESTING

Ten en cuenta que en esta instancia no es obligatorio el desarrollo de testing para tu aplicación. De igual manera, te desafiamos a que los hagas, ¡ya que suman puntos!

- Al menos tener un componente del frontend con sus tests respectivos.
- Al menos tener dos rutas del backend con sus tests respectivos.
- Al menos tener un modelo de la base de datos con sus tests respectivos.
