

# CHALLENGE BACKEND - NodeJs



## Objetivo

Desarrollar una API para explorar el mundo de Disney, la cual permitirá conocer y modificar los personajes que lo componen y entender en qué películas estos participaron. Por otro lado, deberá exponer la información para que cualquier frontend pueda consumirla.

- 👉 Utilizar NodeJs y Express. ✓
- 👉 No es necesario armar el Frontend. ✓
- 👉 Las rutas deberán seguir el patrón REST. ✓
- 👉 Utilizar la librería Sequelize. ✓

⚠️ ¡No es indispensable hacer todo!

Mientras más completes, mayor puntaje obtendrás, pero puedes enviar la app hasta el estadio que la tengas en base a tu conocimiento actual. Recuerda que el objetivo del challenge es entender tu nivel de conocimiento actual.

## Requerimientos técnicos

### 1. Modelado de Base de Datos

- ✓ **Personaje:** deberá tener,
  - Imagen.
  - Nombre.
  - Edad.
  - Peso.
  - Historia.
  - Películas o series asociadas.
- ✓ **Película o Serie:** deberá tener,
  - Imagen.
  - Título.
  - Fecha de creación.
  - Calificación (del 1 al 5).

- ✓
  - Personajes asociados.
- ✓ **Género:** deberá tener,
  - Nombre.
  - Imagen.
  - Películas o series asociadas.

## 2. Autenticación de Usuarios

Para realizar peticiones a los endpoints subsiguientes el usuario deberá contar con un token que obtendrá al autenticarse. Para ello, deberán desarrollarse los endpoints de registro y login, que permitan obtener el token.

Los endpoints encargados de la autenticación deberán ser:

- ✓✓ /auth/login
- ✓✓ /auth/register

## 3. Listado de Personajes

- ✓ El listado deberá mostrar:
  - ✓ Imagen.
  - ✓ Nombre.

- ✓ El endpoint deberá ser:
  - ✓ /characters

## 4. Creación, Edición y Eliminación de Personajes (CRUD) ✓

Deberán existir las operaciones básicas de creación, edición y eliminación de personajes.

## 5. Detalle de Personaje ✓

En el detalle deberán listarse todos los atributos del personaje, como así también sus películas o series relacionadas.

## 6. Búsqueda de Personajes —

Deberá permitir buscar por nombre, y filtrar por edad, peso o películas/series en las que participó. Para especificar el término de búsqueda o filtros se deberán enviar como parámetros de query:

- GET /characters?name=nombre
- GET /characters?age=edad
- GET /characters?movies=idMovie

Pendiente: la búsqueda por el momento se realiza por ID

## 7. Listado de Películas ✓

Deberá mostrar solamente los campos imagen, título y fecha de creación.

- ✓ El endpoint deberá ser:
- GET /movies

## 8. Detalle de Película / Serie con sus personajes ✓

Devolverá todos los campos de la película o serie junto a los personajes asociados a la misma

## 9. Creación, Edición y Eliminación de Película / Serie ✓

Deberán existir las operaciones básicas de creación, edición y eliminación de películas o series.

## 10. Búsqueda de Películas o Series —

Deberá permitir buscar por título, y filtrar por género. Además, permitir ordenar los resultados por fecha de creación de forma ascendiente o descendiente.

El término de búsqueda, filtro u ordenación se deberán especificar como parámetros de query:

- GET /movies?name=nombre
- GET /movies?genre=idGenero
- GET /movies?order=ASC | DESC

Pendiente: la búsqueda por el momento se realiza por ID

## 11. Envío de emails [Está en desarrollo](#)

Al registrarse en el sitio, el usuario deberá recibir un email de bienvenida. Es recomendable, la utilización de algún servicio de terceros como [SendGrid](#).

## Documentación

Es deseable documentar los endpoints utilizando alguna herramienta como Postman o Swagger.

## Tests

De forma opcional, se podrán agregar tests de los diferentes endpoints de la APP, verificando posibles escenarios de error:

- Campos faltantes o con un formato inválido en BODY de las peticiones
- Acceso a recursos inexistentes en endpoints de detalle

Los tests pueden realizarse utilizando Mocha + Chai.