# Individual Project - Henry Countries


## Objetivos del Proyecto

- Construir una App utlizando React, Redux, Node y Sequelize.
- Afirmar y conectar los conceptos aprendidos en la carrera.
- Aprender mejores prácticas.
- Aprender y practicar el workflow de GIT.
- Usar y practicar testing.

## Horarios y Fechas

El proyecto tendrá una duración máxima de tres semanas. En el caso de que completan todas las tareas antes de dicho lapso podrán avisar a su Instructor para coordinar una fecha de presentación del trabajo (DEMO).

## Comenzando

1. Forkear el repositorio para tener una copia del mismo en sus cuentas
2. Clonar el repositorio en sus computadoras para comenzar a trabajar

Tendrán un `boilerplate` con la estructura general tanto del servidor como de cliente.

**IMPORTANTE:** Es necesario contar minimamente con la última versión estable de Node y NPM. Asegurarse 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 tienen instalada:

```
node -v

npm -v
```

# BoilerPlate

El boilerplate cuenta con dos carpetas: `api` y `client`. En estas carpetas estará el código del back-end y el front-end respectivamente.

En `api` crear un archivo llamado: `.env` que tenga la siguiente forma:

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

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).

Adicionalmente será necesario que creen desde psql una base de datos llamada `countries`

El contenido de `client` fue creado usando: Create React App.

# Enunciado

La idea general es crear una aplicación en la cual se pueda ver información de distintos paises utilizando la api externa restcountries y a partir de ella poder, entre otras cosas:

- Buscar paises
- Filtrarlos / Ordenarlos
- Crear actividades turísticas

**IMPORTANTE**: Para las funcionalidades de filtrado y ordenamiento NO pueden utilizar los endpoints de la API externa que ya devuelven los resultados filtrados u ordenados sino que deben realizarlo ustedes mismos. En particular alguno de los ordenamientos o filtrados debe si o si realizarse desde el frontend.

## Únicos Endpoints/Flags que pueden utilizar

- GET https://restcountries.com/v3/all
- GET https://restcountries.com/v3/name/{name}
- GET https://restcountries.com/v3/alpha/{code}

## Requerimientos mínimos:

A continuación se detallaran los requerimientos mínimos para la aprobación del proyecto individial. Aquellos que deseen agregar más funcionalidades podrán hacerlo. En cuanto al diseño visual no va a haber wireframes ni prototipos prefijados sino que tendrán libertad de hacerlo a su gusto pero tienen que aplicar los conocimientos de estilos vistos en el curso para que quede agradable a la vista.

**IMPORTANTE**: No se permitirá utilizar librerías externas para aplicar estilos a la aplicación. Tendrán que utilizar CSS con algunas de las opciones que vimos en dicha clase (CSS puro, CSS Modules o Styled Components)

**Tecnologías necesarias:**

- [ ] React
- [ ] Redux
- [ ] Express
- [ ] Sequelize - Postgres

**Frontend**

Se debe desarrollar una aplicación de React/Redux que contenga las siguientes pantallas/rutas.

**Pagina inicial**: deben armar una landing page con

- [ ] Alguna imagen de fondo representativa al proyecto
- [ ] Botón para ingresar al home (`Ruta principal`)

**Ruta principal**: debe contener

- [ ] Input de búsqueda para encontrar países por nombre
- [ ] Área donde se verá el listado de países. Al iniciar deberá cargar los primeros resultados obtenidos desde la ruta `GET /countries` y deberá mostrar su:
  - Imagen de la bandera
  - Nombre
  - Continente
- [ ] Botones/Opciones para filtrar por continente y por tipo de actividad turística
- [ ] Botones/Opciones para ordenar tanto ascendentemente como descendentemente los países por orden alfabético y por cantidad de población
- [ ] Paginado para ir buscando y mostrando los siguientes paises, 10 paises por pagina, mostrando los primeros 9 en la primer pagina.

**Ruta de detalle de país**: debe contener

- [ ] Los campos mostrados en la ruta principal para cada país (imagen de la bandera, nombre, código de país de 3 letras y continente)
- [ ] Código de país de 3 letras (id)
- [ ] Capital
- [ ] Subregión
- [ ] Área (Mostrarla en km2 o millones de km2)
- [ ] Población
- [ ] Actividades turísticas con toda su información asociada

**Ruta de creación de actividad turística**: debe contener

- [ ] Un formulario **controlado** con los siguientes campos
  - Nombre
  - Dificultad
  - Duración
  - Temporada
- [ ] Posibilidad de seleccionar/agregar varios países en simultaneo
- [ ] Botón/Opción para crear una nueva actividad turística

**Base de datos**

El modelo de la base de datos deberá tener las siguientes entidades (Aquellas propiedades marcadas con asterísco deben ser obligatorias):

- ☐ País con las siguientes propiedades:
  - ID (Código de 3 letras) *
  - Nombre *
  - Imagen de la bandera *
  - Continente *
  - Capital *
  - Subregión
  - Área
  - Población
- ☐ Actividad Turística con las siguientes propiedades:
  - ID
  - Nombre
  - Dificultad (Entre 1 y 5)
  - Duración
  - Temporada (Verano, Otoño, Invierno o Primavera)

La relación entre ambas entidades debe ser de muchos a muchos ya que un país puede contener varias actividades turísticas y, a su vez, una actividad turística puede darse en múltiples países. Por ejemplo una actividad podría ser "Ski" que podría ocurrir en Argentina y también en Estados Unidos, pero a su vez Argentina podría también incluir "Rafting".

**Backend**

Se debe desarrollar un servidor en Node/Express con las siguientes rutas:

**IMPORTANTE**: No está permitido utilizar los filtrados, ordenamientos y paginados brindados por la API externa, todas estas funcionalidades tienen que implementarlas ustedes.

- ☐ **GET /countries**:
  - En una primera instancia deberán traer todos los países desde restcountries y guardarlos en su propia base de datos y luego ya utilizarlos desde allí (Debe almacenar solo los datos necesarios para la ruta principal)
  - Obtener un listado de los paises.
- ☐ **GET /countries/{idPais}**:
  - Obtener el detalle de un país en particular
  - Debe traer solo los datos pedidos en la ruta de detalle de país
  - Incluir los datos de las actividades turísticas correspondientes
- ☐ **GET /countries?name="..."**:
  - Obtener los países que coincidan con el nombre pasado como query parameter (No necesariamente tiene que ser una matcheo exacto)
  - Si no existe ningún país mostrar un mensaje adecuado
- ☐ **POST /activity**:

    - Recibe los datos recolectados desde el formulario controlado de la ruta de creación de actividad turística por body
    - Crea una actividad turística en la base de datos

**Testing**

- [ ] Al menos tener un componente del frontend con sus tests respectivos
- [ ] Al menos tener una ruta del backend con sus tests respectivos
- [ ] Al menos tener un modelo de la base de datos con sus tests respectivos

# Countries