Sprint 4 – Ejercicios

Índice

Objetivo

El objetivo de este Sprint es aprender sobre:

- React (JSX, Props, State, Componentes).
- Ciclos de Vida.
- Hooks.
- Event Listeners, Forms, HTTP Requests, PropTypes.
- React Router.
- Redux.

Comentarios generales

- Leer en detalle la pauta de cada ejercicio.
- Notar que algunos ejercicios requieren que sea haya dictado una clase previa (teórico) antes de poder resolverlos. Estos ejercicios estarán debidamente señalizados.
- En caso de dudas, pueden recurrir a sus compañeros, docentes (por Slack) y/o sitios en Internet (ej: Stack Overflow). Recuerden la importancia de apoyarse entre ustedes ya que una gran forma de aprender y reforzar conocimientos es explicarle a otro.
- También recomendamos tener una carpeta llamada `ha_bootcamp_sprint4` (o similar) para tener todos los ejercicios de este sprint juntos.

Ejercicio 1

Clase previa: “Introducción a React”.

Pauta:

1. Crear una app utilizando `create-react-app`.
2. Cambiar el título de la aplicación (pestaña en el navegador) a “Mi primer aplicación React”.
3. Borrar todo el contenido que está dentro del primer `div` del componente `App` en `src/App.js`.
4. Crear una carpeta llamada `components` dentro de `src/` y crear un archivo llamado `Welcome.js` dentro de ella.
5. Dentro de este archivo declarar un componente que reciba una *prop* llamada `name` y un mensaje de bienvenida dentro de un `<h1>`. Por ejemplo: si el valor de la *prop* `name` es “María Pérez”, el componente tiene que mostrar una etiqueta `h1` con el texto “¡Bienvenido/a a React, María Pérez!”.
6. Exportar como `default` el componente `Welcome`.
7. Importar el componente `Welcome` desde el componente `App`, y utilizarlo dentro del `div`, ahora vacío, para mostrar un mensaje de bienvenida.
8. Probar distintos valores en la variable `name` y ver como al guardar se actualizan los cambios sin necesidad de refrescar la página.

Extra: Analizar la estructura de archivos y carpetas generadas automáticamente. Identificar archivos desconocidos y discutirlos en clase.

Ejercicio 2

Clase previa: “Introducción a React”.

Pauta:

1. Dentro de una app generada con `create-react-app`, crear cuatro componentes separados:
    a. `Title`, el cual debe renderizar un elemento `h1` con un título.
    b. `Subtitle`, el cual debe renderizar un elemento `h2` con un subtítulo.
    c. `Description`, el cual debe renderizar un elemento `p` que muestre “Mi nombre es ” + `nombre`, siendo `nombre` una variable definida externamente.
2. Borrar el contenido dentro del primer `div` en `App.js`, y en sustituirlo por los tres componentes anteriores, en el orden indicado.

El HTML resultante en el navegador debería ser algo similar a:

```
<div id="root">
  <div class="App">
    <h1>Hola</h1>
    <h2>Subtitulo</h2>
    <p>Mi nombre es María</p>
  </div>
</div>
```

Extra: Investigar sobre buenas prácticas de estructuración de componentes en React, y crear estos archivos de la forma más escalable (por ejemplo, pueden empezar con esta guía).

Ejercicio 3

Clase previa: “Introducción a React”.

Pauta:

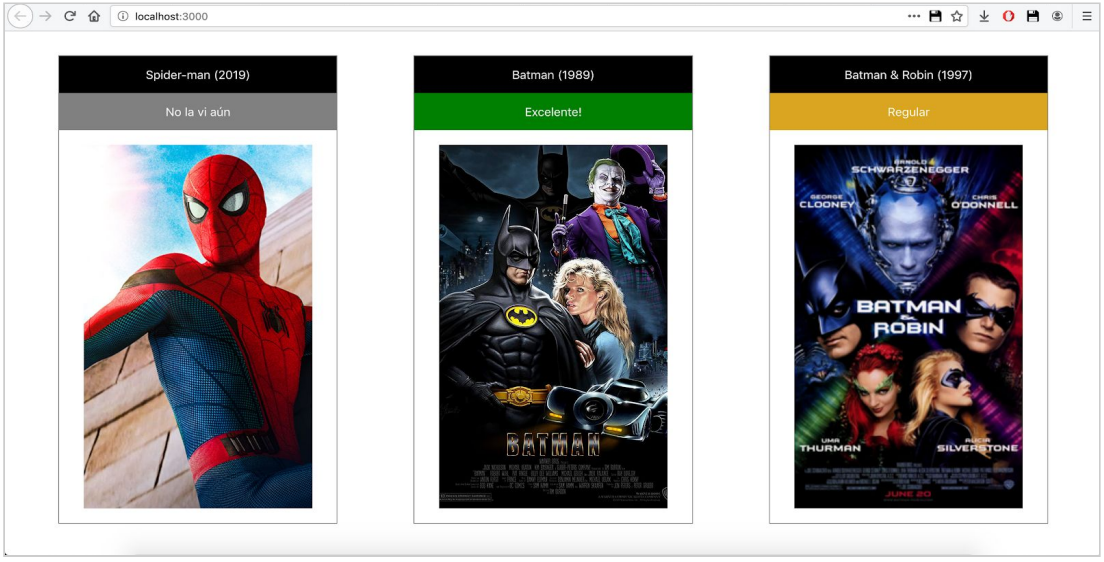
1. Crear un componente `Persona`.
2. Crear las variables `nombre` y `edad` dentro del componente.
3. Renderizar `nombre` dentro de un elemento `<p>`.
4. Agregar otro párrafo `<p>` que muestre el mensaje *“Lo sentimos, no tiene edad legal para beber alcohol”* si la `edad` es menor a 18, o que muestre *“Bienvenido. Lo invitamos a tomar una cerveza"* si la edad es mayor o igual que 18.
5. Modificar el valor de la variable `edad` y ver cómo se actualiza el texto mostrado en la *app*.

Ejercicio 4

Clase previa: “Introducción a React”.

Pauta:

Crear un sitio web usando React, cuyo aspecto final sea como el siguiente:



De cada película se debe conocer: nombre, año, imagen y puntaje (“bueno”, “regular”, “malo”, o que la película no se haya visto aún).

El alumno deberá inventar los datos de las 3 películas. No es necesario utilizar exactamente las mismas imágenes del diagrama, ni siquiera es necesario crear las mismas tres películas, pero en caso de querer hacerlo, pueden usar los siguientes links: Spider-man, Batman y Batman & Robin.

Intentar deducir los requerimientos y diseñar una solución a partir de lo que ve en el diagrama anterior. Por ejemplo, hacerse preguntas como:

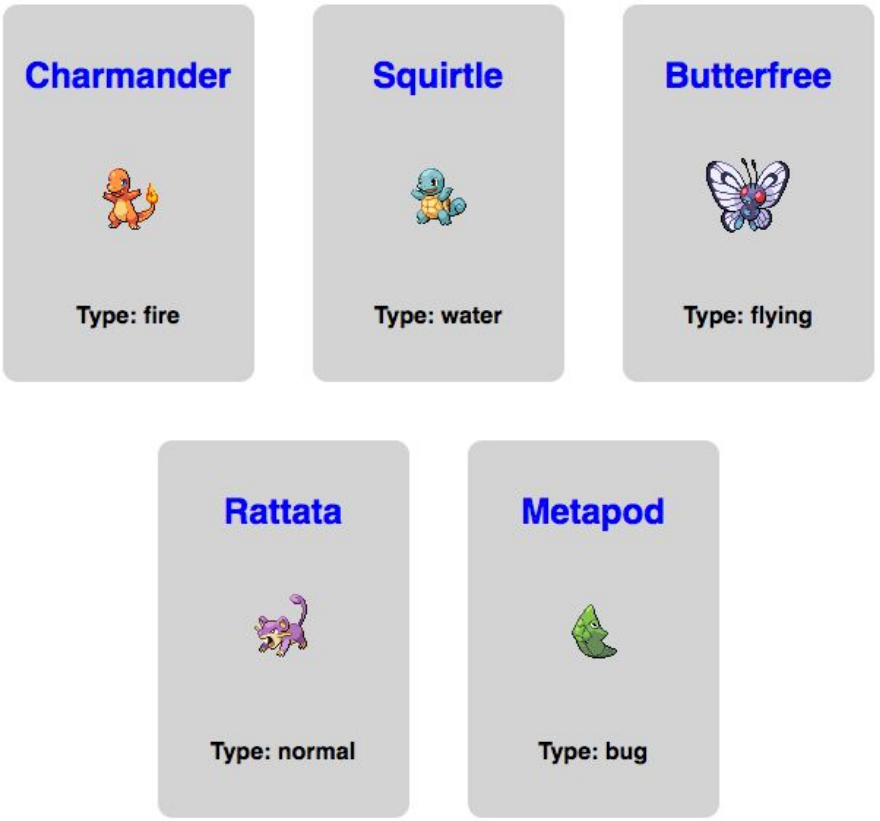
- ¿Qué componentes se deberían crear?
- ¿Qué *props* (datos) hay que pasarle a dichos componentes?

Ejercicio 5

Clase previa: “List Rendering”.

Pauta:

1. Crear una aplicación web que muestre una lista con los **Pokemons** presentes en este archivo.
2. Se deberá crear un componente llamado `Pokemon`. La información deberá ser pasada por *props*.
3. El resultado final debería ser algo así:



Ejercicio 6

Clase previa: “Formularios”.

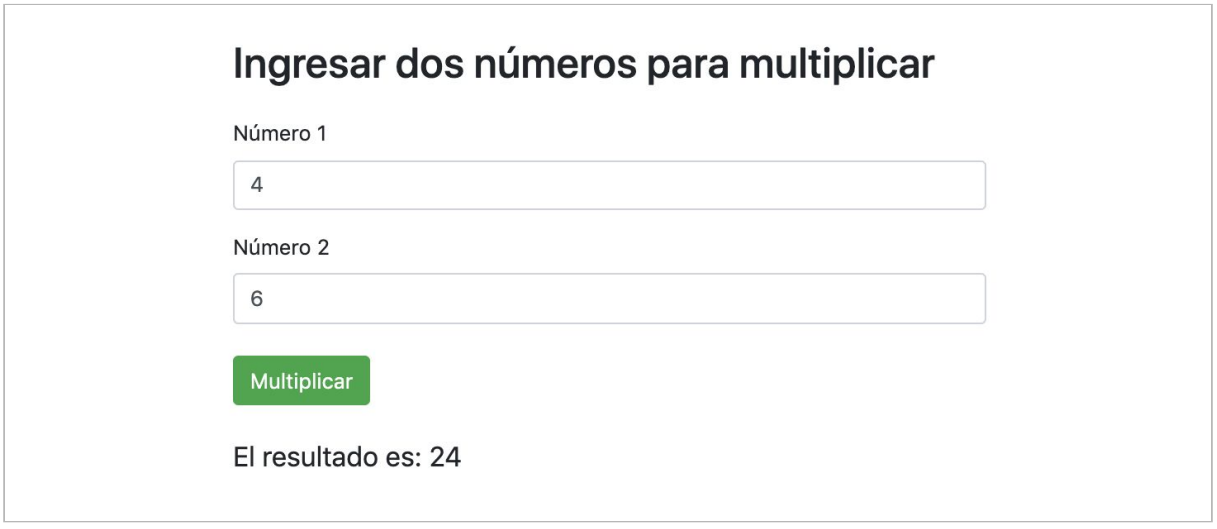
Pauta:

Crear un formulario HTML que contenga dos campos:

- Número 1.
- Número 2.

El formulario debe tener un botón llamado “**Multiplicar**”. Al hacer click sobre el mismo se debe calcular la multiplicación de ambos números y mostrar el resultado debajo del botón.

Si alguno de los campos está vacío, mostrar el mensaje *“Por favor complete todos los campos”*.



Ejercicio 7

Clase previa: “Event listeners”.

Pauta:

Crear una aplicación que consista en dos columnas. La de la izquierda deberá mostrar un listado con todos los productos que comercializa una tienda (disponible en este archivo). La de la derecha deberá mostrar un “**carrito de compras**”, el cual se compone de los productos seleccionados por el usuario.

Cada vez que se hace click sobre un elemento de la lista izquierda, el mismo se agrega a la lista de la derecha (carrito) y se coloca su cantidad en 1. Si se hace click sobre un elemento que ya había sido agregado al carrito, su cantidad se deberá incrementar en una unidad.

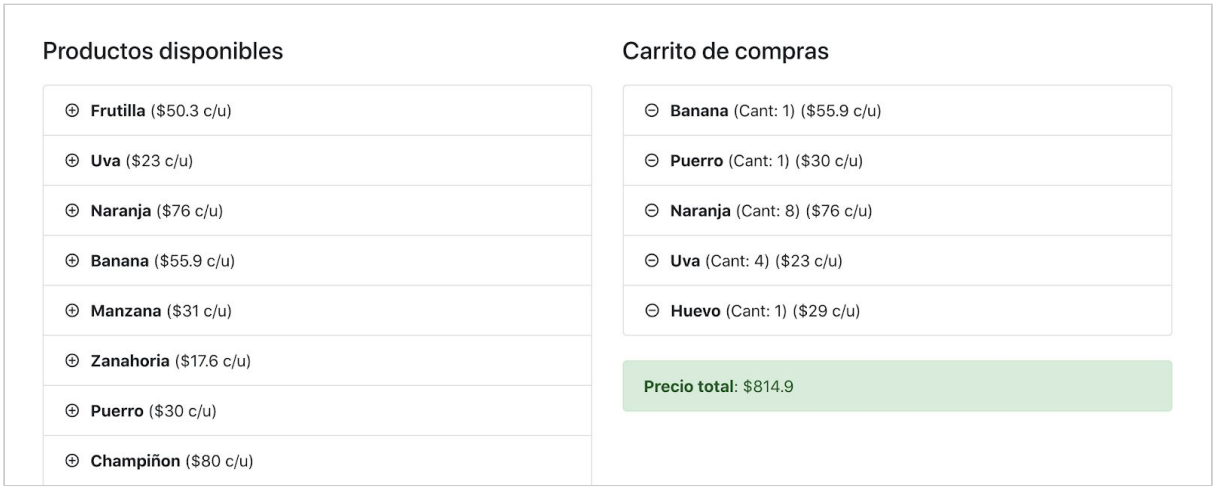
Al hacer click sobre un elemento del carrito, su cantidad se deberá decrementar en una unidad. Al llegar a cero, el elemento se remueve completamente de la lista.

Nota: Pensar bien los componentes que se deberán crear.

Extra 1: Agregar la funcionalidad de que si se agregan más de 5 unidades de Papel Higiénico o Alcohol en Gel, se muestre un mensaje abajo del carrito que diga: *"Lo sentimos. No es posible comprar más unidades. Otras familias también necesitan abastecerse"*.

Extra 2: Agregar la funcionalidad de mostrar el precio de cada producto y el costo total de los productos del carrito.

1. En ambas listas (izquierda y derecha), agregar el precio unitario de cada producto.
2. Agregar el precio total de los productos del carrito.



Ejercicio 8

Clase previa: “External Data Access”.

Pauta:

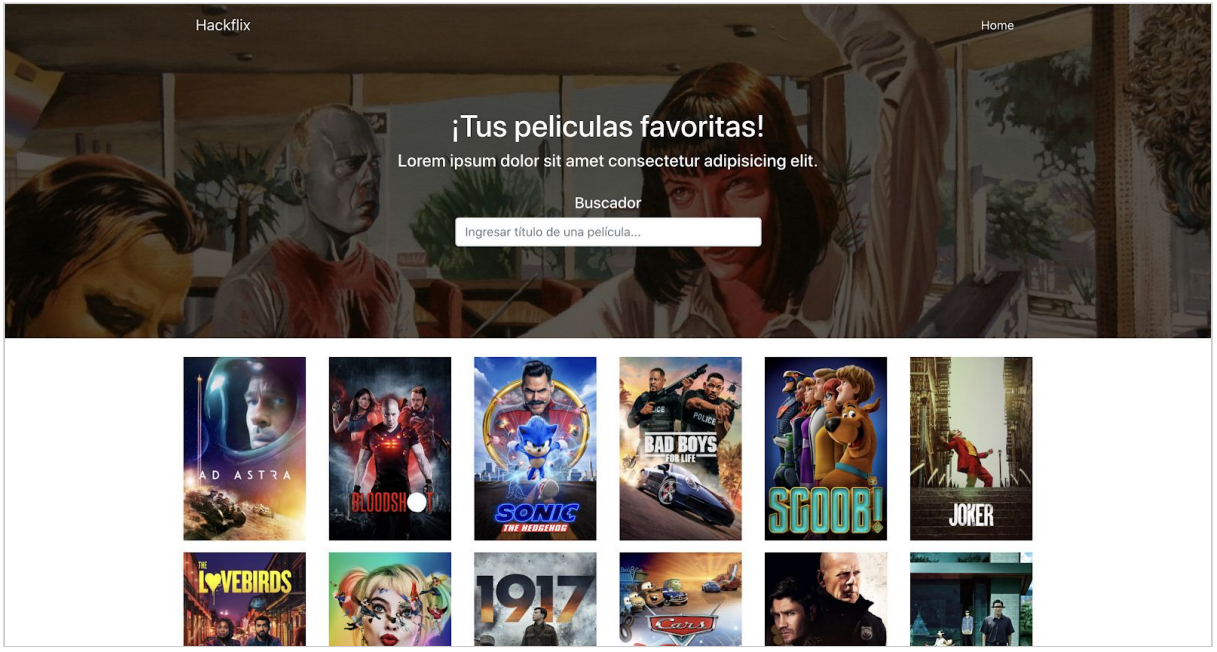
1. Duplicar el proyecto de las **tres películas** y usar dicho proyecto como base para este ejercicio.
2. En lugar de “inventar” los datos de las tres películas, se deberán obtener del servicio The Movie Database (TMDb). Para eso, cada alumno deberá crearse una cuenta gratuita en el sitio y obtener una API Key.
3. La idea es tener un componente llamado `Movie` al cual se le pasará (como *prop*) únicamente el `id` de la película.
4. Por más de que los datos de la película provengan de un servicio externo, el alumno igual deberá elegir las tres películas a mostrar.

Ejercicio 9

Clase previa: “Event listeners”.

Pauta:

Crear la aplicación **Hackflix**, que consiste de un listado de películas como se ve en el siguiente diagrama.



Las películas se deberán obtener de este archivo JSON (el cual se deberá descargar y colocar dentro del proyecto).

A medida que se escribe en el campo de texto, se deberán filtrar las películas del listado. Es decir, sólo se deberán mostrar las películas cuyo título contiene el texto ingresado.

En caso de que no haya resultados, mostrar un mensaje de error acorde.

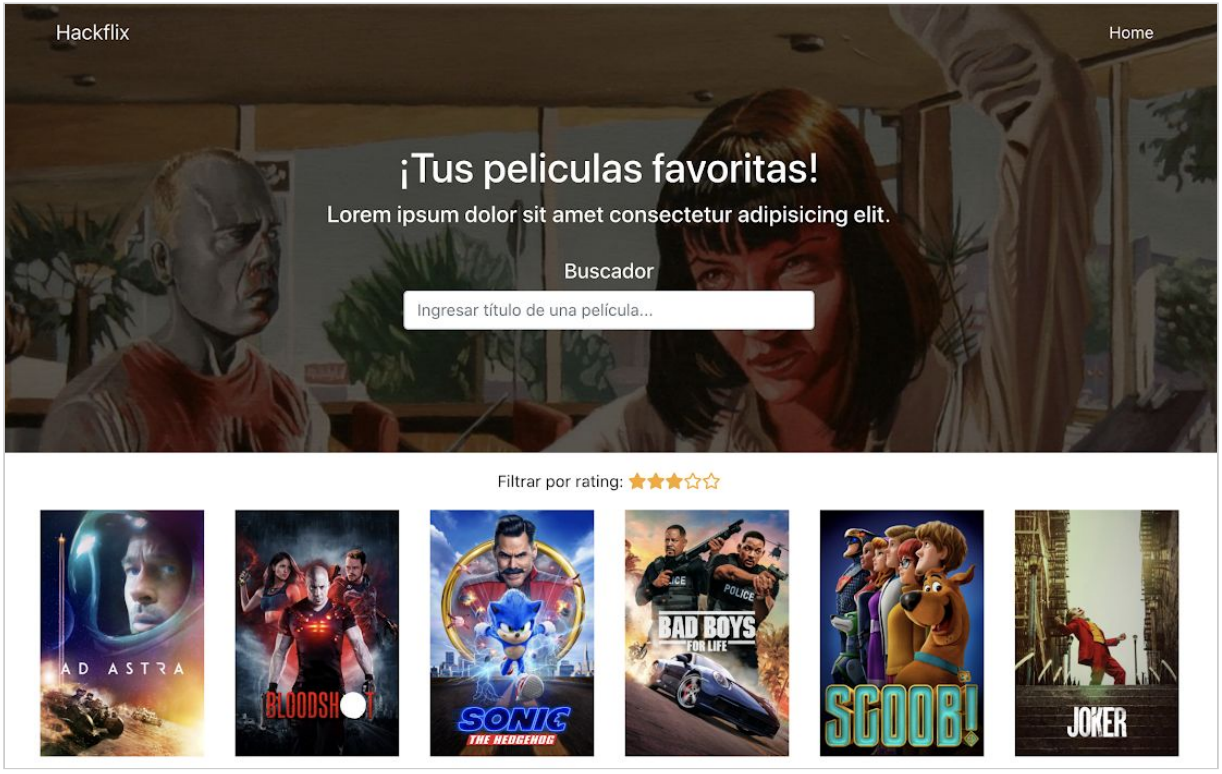
Nota: Pensar bien los componentes que se deberán crear.

Ejercicio 10

Clase previa: “Event listeners”.

Pauta:

Continuar con el ejercicio de Hackflix y agregar la funcionalidad de filtrar por calificación de las películas.



Notar que las películas tienen un atributo llamado `vote_average` cuyo valor oscila entre 1 y 10. La consigna es que cada estrella tenga un valor de dos puntos de rating. De esta manera, si se seleccionaron 4 estrellas, se deben filtrar las películas con 8 o más rating.

Ejercicio 11

Clase previa: “Event listeners”.

Pauta:

Continuar con el ejercicio de Hackflix y agregar la funcionalidad de que al hacer click sobre una película, se abra un modal mostrando información detallada de la misma (título, rating, descripción, etc).

Ejercicio 12

Clase previa: “Event listeners”.

Pauta:

Continuar con el ejercicio de Hackflix, pero ahora, en lugar de obtener las películas desde el archivo `movies.json`, se deberán obtener desde la API de The Movie Database (https://developers.themoviedb.org/3/discover/movie-discover) vía AJAX.

Ejercicio 13

Clase previa: “Event listeners”.

Pauta:

Continuar con el ejercicio de Hackflix y agregar la funcionalidad de paginación. La idea es que se muestren sólo 10 películas a la vez. Para ver películas adicionales se deberá pasar a la siguiente página.

Ejercicio 14

Clase previa: “Event listeners”.

Pauta:

Continuar con el ejercicio de Hackflix y agregar la funcionalidad de scroll infinito. Es decir, en lugar de tener un paginador, las películas adicionales se mostrarán cuando el navegante desciende con el scroll y llega al final de la página.