CLASE 5 - Práctica -

1 - Children props

- Insertar en nuestro index.html el css de bootstrap

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
```

- Generar un componente stateless llamado <Alert>, que retorne el HTML y las clases CSS necesarias de bootstrap para mostrar un alert (*https://getbootstrap.com/docs/4.0/components/alerts/*). Este componente, deberá recibir en la prop children, el contenido que se mostrará como cuerpo del alert.

- A nuestro componente <Alert>, agregarle una prop llamada “type”, que pueda recibir un string de las siguientes opciones: primary, secondary, success, danger, warning, info, light, dark. Esta prop la utilizaremos para agregarle al alert la clase CSS del tipo que queremos que sea. Validar las props del Alert con PropTypes y setear como default para la prop type el valor “primary”.

- Generar un componente <MyAlerts> que renderee 8 componentes Alert, uno con cada type y contenido distinto.

- Agregar al componente <MyAlerts> un state inicial con la siguiente estructura:

```
this.state = {
      alerts: [
        {type: 'danger', content: 'Este es un alert de tipo danger'},
        {type: 'secondary', content: 'Este es un alert de tipo secondary'},
        {type: 'success', content: 'Este es un alert de tipo success'},
      ]
    }
```

  Configurar el render para que renderee los alerts que están en this.state.alerts con su type y contenido correspondiente. Agregar a cada alert un evento onClick para que al clickear en uno, el mismo sea quitado del array this.state.alerts. Mostrar al usuario un mensaje de “No hay Mensajes para mostrar” que se muestre cuando el array de this.state.alerts esté vacío.

- Componentizar *https://getbootstrap.com/docs/4.0/components/card/* para que reciba por parámetro el source de la imagen en la prop imgSource, y también children, que lo imprimiremos dentro del body de la card.

  Crear Componentes stateless separados para `card-title` y `card-text` y utilizarlos a la hora de generar los hijos que pasaremos dentro de <Card>.

  Generar las PropTypes para todas las props de todos los componentes y validar que la imagen de la Card solo se muestre si se setea la prop.

El uso de nuestro componente <Card> debería verse de la siguiente forma:

```
<Card imgSource="url-de-la-imagen">
   <CardTitle> Titulo de la card </CardTitle>
   <CardText> Texto para el cuerpo de la card </CardText>
</Card>
```

- Componentizar *https://getbootstrap.com/docs/4.0/components/modal/* de forma tal que muestre dentro de `modal-content` lo que venga en la prop children. Crear componentes para `modal-header, modal-body y modal-footer` para poder utilizarlos a la hora de generar un modal sin necesidad de escribir HTML como children para estas estructuras.

  El uso de nuestro componente Modal debería quedar así:

```
<Modal>
   <ModalHeader> Titulo de mi modal </ModalHeader>
   <ModalBody> Cuerpo de mi modal </ModalBody>
   <ModalFooter> Footer de mi modal </ModalFooter>
</Modal>
```

- Agregar a <Modal> la prop “visible”, que sea un booleano utilizado para agregar la clase css “show” y quitarla cuando sea false.

- Crear un componente <MyModal> que renderee un modal y utilice su state para mostrar y ocultar el modal.

2 - React.Children.map & React.cloneElement

- Continuando con la lógica de componentizar en React y los componentes de bootstrap para reutilizar, crear un componente para *https://getbootstrap.com/docs/4.0/components/list-group/* que pueda recibir como children múltiples items, y que a cada uno utilizando lo visto de React.Children.map le agregue un wrapper con la clase CSS `list-group-item`.

- Crear un componente de clase que tenga un input y un botón “Agregar“, que utilice su state para generar una lista (como en el ejercicio de la clase anterior), pero al renderear utilizar nuestro nuevo componente ListGroup.

- Configurar para que el onClick de cada ítem lo elimine de la lista.

3 - API

- Crear nuestra clase de API y guardar en una constante API_KEY la key que nos provee TheMovieDB

- Generar los métodos para obtener:
  - listado de géneros de películas
  - listado de géneros de series (tv)
  - listado de películas por género

    - listado de series por género
    - info de una película
    - info de una serie

- Crear un componente de clase con el que podamos:

    - Tener un select con las opciones “Películas” y “Series”, que al cambiar genere un segundo select con los Géneros de la opción seleccionada.
    - Al cambiar el select de géneros, generar un tercer select que se llene con las películas o series de ese género.
    - Al seleccionar una película o serie del tercer select, ir a buscar la info de la misma e imprimir en el render el título, la sinopsis y la imagen.

- Cada llamado asincrónico de nuestro componente deberá mostrar un loading para informar al usuario que se está realizando algo por AJAX.

- En caso de error, mostrar un Alert de tipo danger con el error ocurrido.

Ejercicio Integrador:

1. Modificar Button.js para que el contenido a mostrar del botón se pase por la prop chldren. Agregar dicha prop en la definición de PropTypes del componente.

2. Crear un componente stateless llamado “PrimaryTitle”, que imprima un h1 que como contenido mostrará lo que venga en la prop children.

3. Modificar nuestro componente “ItemsSection” para que tengamos la posibilidad de utilizarlo con las siguientes props:

```
<ItemsSection
   title="Películas más Populares"
   items={arrayDeItems}
   loading={loading}
   viewAllLink={'http://google.com'}
/>
```

   Basarnos en el html del maquetado para utilizar las props donde correspondan, y que dependiendo de su valor, muestren o no bloques de código (por ejemplo loading).

4. Crear componentes de clase “PopularMoviesItemsSection” y “PopularSeriesItemsSection”, que contengan la lógica y el state para en cada uno llamar a la método de API correspondiente y renderear un “ItemsSection” con la data necesaria.

5. Crear un componente Dimmer para que pueda reciba en la prop children el contenido a renderear. Crear un componente Loading y utilizar ambos componentes dentro de ItemsSection para mostrar y ocultar el loading y para mostrar y ocultar el mensaje de que no hay ítems.

6. Crear un componente para el Header de nuestra aplicación (html de <header>...</header>).

7. Crear un componente para la Home de nuestro sitio.