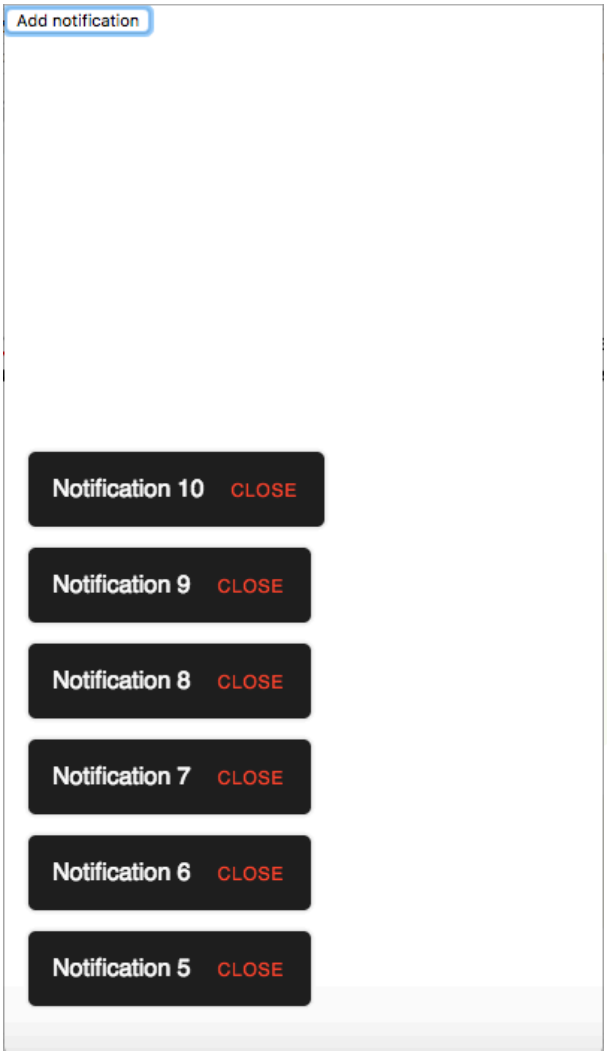
Ejercicio del tema 9

Ejercicico1 React-notifications: En este ejercicio aprenderemos ha realizar una pila de mensajes que se acumulan y van desapareciendo uno a uno después de un determinado tiempo.



Lo primero que necesitaremos será instalar la librería **immutable** que proporciona estructuras de datos persistentes y ordenados como **orderedSet.**

Para ello en el directorio de nuestra aplicación react ejecutaremos en la consola de comandos el siguiente comando:

npm install immutable

Realizaremos los siguientes imports en el archivo js de nuestra aplicación:

import { OrderedSet } from 'immutable';

import { NotificationStack } from 'react-notification';

Crearemos el constructor inicializando las variables.

```
class App extends Component {
  constructor(props) {
    super(props);
   this.state = {
      notifications: OrderedSet(),
      count: 0
    };
this.removeNotification=this.removeNotification.bind(this);

  }
```

Con el metodo siguiente añadiremos una notificación a la lista de notificaciones orderedSet:

```
addNotification() {
    const { notifications, count } = this.state;
    const id = notifications.size + 1;
    const newCount = count + 1;
    return this.setState({
      count: newCount,
      notifications: notifications.add({
        message: `Notification ${id}`,
        key: newCount,
        action: 'Close',
        dismissAfter: 4000,
        onClick: () => this.removeNotification(newCount),
       })
    });
  }
```

DismissAfter permite que desaparezcan las notificaciones en un tiempo determinado.

el evento onClick llamara a la función siguiente definida justo debajo:

```
removeNotification (count) {
    const { notifications } = this.state;
    this.setState({
      notifications: notifications.filter(n => n.key !== count)
    })
  }
```

De manera que cuando se haga clic en el action “close” se borrara la notificación de la lista.

En la renderización devolveremos un botón que llamara a la función de añadir notificación, y el componente **<NotificationStack/>** en el que se basa la pila de notificaciones.

```
render () {
    return (
      <div>
        <button onClick={this.addNotification.bind(this)}>
          Add notification
        </button>
        <NotificationStack
          notifications={this.state.notifications.toArray()}
          onDismiss={notification => this.setState({
         notifications:
this.state.notifications.delete(notification)
          })}
        />
      </div>
    );
  }
```

Convertiremos **notifications** en un array generico para después utilizar el método **delete( )** cuando ocurra el evento **onDismiss** que se producirá con el **dismissAfter** de esta manera también quitaremos de la lista la notificación cuando se produzca dicho evento.

Ejercicio 2 React-Slick: Crea un slider de imágenes distintas con las siguientes condiciones:

- Que sea infinito, es decir que cuando llegue al ultimo se pueda pasar al primero arrastrando hacia la izquierda.
- Que tenga una intervalo de transición de 5 segundos
- Que muestre un elemento en cada slide
- Que la transición sea automatica
- Que tenga una transición de desvanecimiento

Puedes empezar con el siguiente código como base:

```
render: function () {
    var settings = {
      dots: true,
      speed: 500,
      slidesToScroll: 1
    };
    return (
      <Slider {...settings}>
        <div><img></img></div>
        <div><img src="" alt=""></img></div>
        <div><img src="" alt=""></img></div>
        <div><img src="" alt=""></img></div>
        <div><img src="" alt=""></img></div>
        <div><img src="" alt=""></img></div>
      </Slider>
    );
  }
```


Ejercicio 3 - *Librerías*

En el temario hemos visto cómo añadir vídeos a nuestros proyectos de una manera sencilla, en este ejercicio vamos a usar lo aprendido y crear una aplicación que nos muestre un vídeo y además, esta aplicación tendrá un botón que nos permitirá pasar entre distintos vídeos que tendremos en una lista.

Empezamos creando un nuevo proyecto vacío y en la clase principal añadimos los dos imports necesarios, uno para usar el componente Player y otro para cargar su correspondiente .css.

```
import { Player, ControlBar } from 'video-react';
import '../node_modules/video-react/dist/video-react.css';
```

Lo siguiente será crear una lista con los vídeos, se hará de la misma forma que hemos hecho en teoría. Para este ejercicio hemos usado los que aparecen de ejemplo en la página de la librería pero si queréis podéis poner otros. Se puede ver una diferencia en este código, y es que a cada vídeo le hemos asignado un número en vez de un nombre, de esta forma nos será más fácil hacer una transición entre los distintos vídeos.

También creamos una variable contador que va a ser la encargada de ir actualizándose con el valor de cada vídeo cada vez que pulsemos el botón.

```
const sources = {
  0: 'http://media.w3.org/2010/05/sintel/trailer.mp4',
  1: 'http://media.w3.org/2010/05/bunny/trailer.mp4',
  2: 'http://media.w3.org/2010/05/bunny/movie.mp4',
  3: 'http://media.w3.org/2010/05/video/movie_300.webm',
};

var contador = 0;
```

Pasamos ahora a crear el constructor de la clase, el state y hacer bind del método encargado de cambiar de vídeo para poder usar la directiva **this**. Al state source le pasamos como valor inicial el vídeo de la posición 0.


```
constructor() {
    super();
    this.state = {
      source: sources[contador],
    };
    this.cambiarVideo = this.cambiarVideo.bind(this);
  }
```

Lo siguiente es renderizar el objeto Player y el botón encargado de cambiar de vídeo. Cuando creamos el Player vamos a asignarle una referencia llamada **"player"**, esta referencia nos servirá para poder acceder a este objeto desde el método **cambiarVideo**, también añadimos autoPlay para que los vídeos comiencen a reproducirse automáticamente. Cómo hemos visto en teoría, el source de este objeto será el valor del state.

Para el botón simplemente tenemos que añadirle un método llamado cambiarVideo y Siguiente como texto a mostrar.

```
render() {
    return (
      <div>
        <Player
          ref="player"
          autoPlay >
          <source src={this.state.source} />
        </Player>
        <button onClick={this.cambiarVideo}>Siguiente</button>
      </div>
    );
  }
```

Por último nos falta crear el método **cambiarVideo**, este método comprueba el valor de la variable contador y si es menor que tres incrementa su valor en uno. Si es igual a tres vuelve a poner el valor a cero, con esto conseguimos que el valor de la variable contador no se salga del número de elementos de la lista de vídeos y que cuando llegamos al último vídeo y pulsemos en el siguiente se vuelva a poner el primero. Después de actualizar el valor del contador actualizamos el valor del state y usando la referencia player recargamos el vídeo.


```
cambiarVideo() {
    if (contador <3) {
      contador++;
    } else {
      contador = 0;
    }
    this.setState({source: sources[contador]})
    this.refs.player.load();
  }
}

export default App;
```

Ampliación 1 - *Librerías*

Añade un botón que permita silenciar/no silenciar el vídeo cada vez que le pulsemos, para ello deberás usar el método **muted( )**.

