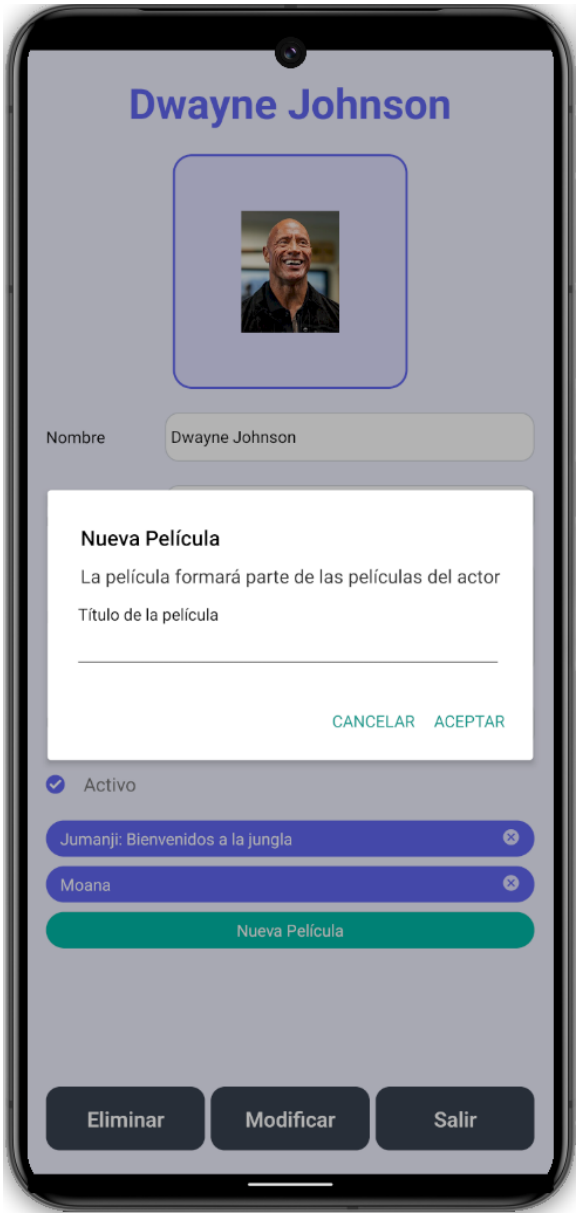
TUTORIAL 21

DIÁLOGOS MODALES



En este tutorial vamos a terminar la app de actores incluyendo las operaciones de añadir y eliminar película. Para ello, mostraremos una diálogo modal personalizado con un cuadro de texto para añadir el título de la película y botones para aceptar y cancelar.

1.- Inicio del proyecto

*El punto de partida de este tutorial es el final del tutorial anterior. Allí dejamos vacías dos funciones en **EditorActor** llamadas **accionNuevaPelicula** y **accionBorrarPelicula** y en este tutorial vamos a programarlas, de manera que la app de actores se quede completamente terminada.*

1. Abre el proyecto **rn_actores**
2. Crea una rama llamada **tutorial21**
3. Instala la siguiente librería
    - **React-native-dialog →npm install react-native-dialog**

> ☞ ***¿Para qué sirve esta librería?*** *Sirve para construir diálogos modales, que son ventanas modales personalizadas con el contenido que queramos darle (cuadros de texto, botones, etc), más allá de las escasas opciones que nos ofrece el **Alert**.*

2.- Preparación del componente EditorActor

*Vamos a preparar en **EditorActor** todo lo necesario para poder añadir y borrar películas.*

1. Abre **EditorActor** y añade una variable de estado (con su setter) llamada **dialogoVisible** que controlará si se muestra o no el diálogo

```
export default function EditorActor({actorSeleccionado,accionCrearActor,accionModificarActor,
  accionBorrarActor,setModalVisible}: EditorActorProps) {
    const [dialogoVisible, setDialogoVisible] = useState(false)
    // resto omitido
}
```

2. Haz que la función **accionNuevaPelicula** ponga a **true** esa variable

```
function accionNuevaPelicula() {
  setDialogoVisible(true)
}
```

3. Añade también a **EditorActor** una función llamada **nuevaPelicula**, que reciba un objeto **Pelicula** (que recordemos, es un alias para el tipo **string**) y lo añada a **peliculas** de manera inmutable (o sea, creando una nueva lista)

```
function nuevaPelicula(pelicula:Pelicula){
  const nuevaListaPeliculas = [...peliculas,pelicula]
  setPeliculas(nuevaListaPeliculas)
}
```

4. Por último, programa la función **accionBorrarPelicula** de manera que elimine de manera inmutable de **peliculas** la película que recibe como parámetro.

```
function accionBorrarPelicula(pelicula: Pelicula) {
  const nuevaListaPeliculas = peliculas.filter( p => p!== pelicula)
  setPeliculas(nuevaListaPeliculas)
}
```

5. Ejecuta la app y comprueba que se pueden borrar películas de los actores, aunque todavía no se pueden añadir nuevas.

2.- Creación de un diálogo modal

Vamos a hacer que al pulsar el botón para añadir una nueva película, se abra un diálogo modal personalizado con un cuadro de texto para escribir el título de la nueva película. Comenzaremos diseñando el componente que da soporte al diálogo.

1. En **components** crea un archivo llamado **DialogoNuevaPelicula.tsx**
2. Vamos a añadir a **DialogoNuevaPelicula** estos props:
   - **dialogoVisible** → boolean que indica si se debe mostrar o no el diálogo
   - **setDialogoVisible** → setter que sirve para mostrar u ocultar el diálogo
   - **setTitulo** → Setter que sirve para poner el título de la película
3. Abre **DialogoNuevaPelicula.tsx** y teclea **rnfs+intro**
4. Añade a la función **DialogoNuevaPelicula** estos props:
   - **setDialogoVisible:** Setter que oculta el diálogo
   - **nuevaPelicula:** Función que recibe una película y la añade al actor

```
type DialogoNuevaPeliculaProps = {
    dialogoVisible: boolean
    setDialogoVisible: React.Dispatch<React.SetStateAction<boolean>>
    nuevaPelicula: (pelicula:Pelicula) => void
}
export default function DialogoNuevaPelicula(
 {dialogoVisible, setDialogoVisible, nuevaPelicula}:DialogoNuevaPeliculaProps
) {
  // resto omitido
}
```

5. Añade a **DialogoNuevaPelicula** una variable de estado llamada **titulo**

```
export default function DialogoNuevaPelicula(
  {dialogoVisible, setDialogoVisible, nuevaPelicula}:DialogoNuevaPeliculaProps
) {
    const [titulo, setTitulo] = useState("")
    // resto omitido
}
```

6. Añade en **DialogoNuevaPelicula** una función llamada **cerrarDialogo** que simplemente, cierre el diálogo

```
function cerrarDialogo(){
  setDialogoVisible(false)
}
```

7. Haz que la función **DialogoNuevaPelicula** devuelva un componente de tipo **Dialog.Container,** con el siguiente valor para sus props
   - **visible:** Es un boolean que vale true si hay que mostrar el diálogo. Pasaremos aquí el prop **dialogoVisible**
   - **onBackdropPress:** Es la función que se ejecuta al pulsar fuera de la ventana. Pasaremos un lambda que cierre el modal
   - **onRequestClose:** Es la función que se ejecuta al pulsar el botón de atrás del dispositivo. Pasaremos un lambda que cierre el modal

```
export default function DialogoNuevaPelicula(
    { dialogoVisible ,setDialogoVisible, nuevaPelicula}:DialogoNuevaPeliculaProps) {
  // inicio omitido
  return (
   <Dialog.Container
        onBackdropPress={ cerrarDialogo }
        onRequestClose={ cerrarDialogo }
    >
   </Dialog.Container>
  )
}
```

*El **DialogContainer** es una ventana modal de diálogo vacía, y en su interior pondremos componentes para personalizar el diálogo como:*

- ***Dialog.Title** → Título principal del diálogo*
- ***Dialog.Description** → Texto descriptivo con información sobre el diálogo*
- ***Dialog.Input** → Un cuadro de texto para introducir información*
- ***Dialog.Button** → Un botón que forma parte del diálogo*
- ***Dialog.Switch** → Es un "interruptor" (botón que se puede dejar presionado)*
- ***Dialog.CodeInput** → Para introducir contraseñas o números de pin*

8. Encierra en el **Dialog.Container** estos componentes:
    - **Dialog.Title** con el mensaje "Nueva Película"
    - **Dialog.Description** con un mensaje de información
    - **Dialog.Input** para indicar que la operación creará una nueva película para el actor. Su contenido estará vinculado a la variable de estado **titulo** (los props que hay que usar son los mismos que el **TextInput**)
    - **Dialog.Button** para cancelar. Al pulsarlo, se cerrará el diálogo
    - **Dialog.Button** para aceptar. Al pulsarlo, se llamará a **nuevaPelicula** pasando la variable de estado **titulo**

```
export default function DialogoNuevaPelicula(
  { dialogoVisible ,setDialogoVisible, nuevaPelicula}:DialogoNuevaPeliculaProps
) {
  // inicio omitido
  return (
   <Dialog.Container
        visible={dialogoVisible}
        onBackdropPress={ cerrarDialogo() }
        onRequestClose={ cerrarDialogo() }
    >
        <Dialog.Title>Nueva Película</Dialog.Title>
        <Dialog.Description>La película se añadirá al actor</Dialog.Description>
        <Dialog.Input label={"Título de la película"} value={titulo} onChangeText={setTitulo}/>
        <Dialog.Button label={"cancelar"} onPress={cerrarDialogo}/>
        <Dialog.Button label={"aceptar"} onPress={() => nuevaPelicula(titulo)}/>
   </Dialog.Container>
  )
}
```

9. Por último, abre **EditorActor** y coloca al fondo una etiqueta **DialogoNuevaPelicula**

```
export default function EditorActor({actorSeleccionado,accionCrearActor,accionModificarActor,
  accionBorrarActor,setModalVisible,
}: EditorActorProps) {
  // inicio omitido
  return (
    <View style={styles.contenedor}>
      // resto omitido
    <DialogoNuevaPelicula
         dialogoVisible={dialogoVisible}
         setDialogoVisible={setDialogoVisible}
         nuevaPelicula={nuevaPelicula}/>
     )
   }
  </View>
 );
}
```

10. Ejecuta la app y comprueba que se pueden añadir y borrar películas a los actores
11. Haz un commit titulado “finalizado tutorial 21”
12. Sitúate en la rama **main** y mezcla en ella la rama **tutorial21**
13. Haz **push**