

El patrón Redux – Reducers

Los Reducers son funciones que especifican cómo cambia el estado en respuesta a los Actions.

Algunos ejemplos de Reducers que podríamos tener son:

- Establecer una propiedad “userDetails” con los datos del usuario después de iniciar sesión.
- Cambiar a true una propiedad “sideMenuVisible” para mostrar el menú lateral de nuestra aplicación.
- Cambiar a true una propiedad “globalSpinnerVisible” al guardar datos, para bloquear la pantalla con una animación de espera.

No todos los Actions emitidos pueden actualizar directamente el Store, ya que algunos tienen Effects. Para gestionar los Effects, utilizamos la librería NgRx Effects.

Los Reducers se encargan de gestionar las transiciones de un estado al siguiente en tus aplicaciones:

```
export const userReducer = createReducer (
  initialState,
  on(UserActions.loadUser, (state, action) => {
    return {
      ...state,
      user: action.user
    };
  })
);
```

Las funciones Reducer manejan estas transiciones determinando qué Action manejar basándose en el “type” del Action que se emite:


```
export const userReducer = createReducer (
  initialState,
  on(UserActions.loadUser, (state, action) => {
    return {
      ...state,
      user: action.user
    };
  })
);
```

Los Reducer son *funciones puras* y manejan cada transición de estado de forma sincrónica. Cada función Reducer toma el estado inicial y una selección de funciones que manejan los cambios de estado para sus Actions asociados:

```
export const userReducer = createReducer (
  initialState,
  on(UserActions.loadUser, (state, action) => {
    return {
      ...state,
      user: action.user
    };
  })
);
```

Estas funciones puras toman el estado actual y un Action y devuelven el nuevo estado:

```
export const userReducer = createReducer (
  initialState,
  on(UserActions.loadUser, (state, action) => {
    return {
      ...state,
      user: action.user
    };
  })
);
```



¿Qué es una función pura?

Una función pura es una función que, dados los mismos argumentos, siempre devolverá el mismo valor sin efectos secundarios observables o “Side Effects”. Por ejemplo, la siguiente función de suma:

```
function sum(a, b) {
  const result = a + b;
  return result;
}
```

Esta función toma dos argumentos (a y b) y simplemente los suma, devolviendo siempre el mismo valor cuando se llame con los mismos dos argumentos. Así, las funciones puras siempre devolverán resultados coherentes, pero además las funciones puras no mutarán ni accederán a propiedades fuera del ámbito de su función.

Esta es una función impura porque depende de variables externas:

```
let c = 1;

function sum(a, b) {
  result = a + b + c;
  return result;
}
```