

Implementacion de React Redux

Partes que conforman el Slice:

Importaciones: `import { PayloadAction, createAsyncThunk, createSlice } from "@reduxjs/toolkit";`

Estado Inicial: como su nombre lo describe es el estado inicial del slice. Aqui podemos declarar valores iniciales y campos que maneja el slice. (ver imagen 1)

```
const initialState: AroSliceState = {
  items: [],
  isReloadNeeded: true,
  isFetching: false,
  isError: false,
};
```

Figure 1: imagen 1

La creacion del slice, contiene los campos name: “Nombre del Slice que no puede ser repetido en el caso de tener varios slice”, estado inicial, reducers: { acciones : (state)=> “ Acciones del slices en el cual podemos realizar cambios a los estados iniciales deben ser utilizados con el dispatch” }, extraReducers: (builder) =>{ “Acciones que pueden realizarse como observadores de otras acciones o de funciones async Thunk” builder()....} (ver imagen 2). En la imagen podemos ver como mantenemos control de una funcion async Thunk que hace una peticion para obtener datos y podemos manejar sus estados en cada estado de la peticion. El extraReducer no es requerido para su funcionalidad si no que es una herramienta mas.

```
export const AroSlice = createSlice({
  name: "AroHookSlice",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder.addCase(getAroHook.pending, (state): void => {
      state.isFetching = true;
      state.isError = false;
      state.isReloadNeeded = false;
    });
    builder.addCase(getAroHook.fulfilled, (state, action): void => {
      state.isFetching = false;
      state.isError = false;
      state.isReloadNeeded = false;
      state.items = action.payload;
    });
    builder.addCase(getAroHook.rejected, (state): void => {
      state.isFetching = false;
      state.isError = true;
      state.isReloadNeeded = false;
    });
  },
});
```

Figure 2: Imagen 2

Funcion AsyncThunk: Funcion usualmente generada para el uso de peticiones rest api

el createAsyncThunk genera la funcion y su orden seria el siguiente <{modelo de datos de lo que se recibira},{modelos de datos que se pasan a la funcion, no son requeridos}>("nombre unico de la accion no se puede repetir", async ("aqui iria los datos que se pasarían en la funcion en el caso de haber sidos pasados") =>{return "peticion api"}). (ver Imagen 3)

```
export const getPermisoStatusSlice = createAsyncThunk<
  { data: EstadosPermiso[]; estadoId: number },
  { estadoId: number }
>("getStatusPermiso",
  async ({ estadoId }) => {
    return await api.getEstadosPermiso(estadoId);
  },
```

Figure 3: image 3

Reducers: Acciones del slice para la manipulacion del estado, (state:"datos del slice", action:"no requerido, pero al usarse indica que la funcion reducer va a recibir algo en el caso de la imagen un dato booleano" (ver Imagen 4)

```
setShowDrawer: (state, action: PayloadAction<boolean>): void => {
  state.showDrawer = action.payload;
},
```

Figure 4: Imagen 4

Exportacion de funciones reducers.

A diferencia de la funcion async thunk que puede ser exportada directamente para exportar la funcion reducer se debe de exportar con el [slice.variable.name].action. (Ver Imagen 5)

```
export const {
  setShowDrawer,
  setShowDrawerEdit,
  setShowDrawerDelete,
  setTab,
  setSelectedDate,
  reset,
} = CierresUI.actions;
```

Figure 5: Imagen 5

Para el uso del estado de un slice se utiliza el useAppSelector() y para el uso de las acciones Thunk o reducer se utiliza el useAppDispatch().

Estas 2 funciones son importadas de "src/store/hooks" que es donde se generan los hooks

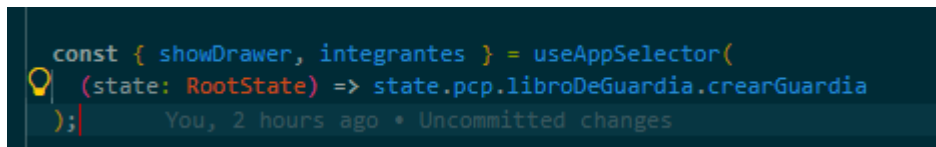
para el uso del AppDispatch hacemos una instancia del hook en una variable llamada dispatch. (ver Imagen 6)

```
const dispatch = useAppDispatch();

const handleClose = useCallback(() => {
  dispatch(setShowDrawer(false));
}, [dispatch]);
```

Figure 6: Imagen 6

con el uso del AppSelector podemos llamar desde cualquier lado un slice o varios para consumir o ver datos. (ver Imagen 7)

A screenshot of a code editor showing a TypeScript function call. The code is:

```
const { showDrawer, integrantes } = useAppSelector(  
  (state: RootState) => state.pcp.libroDeGuardia.crearGuardia  
);
```

 A lightbulb icon is visible on the left side of the code block.

Figure 7: Imagen 7

Todo reducer o AsyncThunk debe ser ejecutado con un `dispatch(reducer() || asyncThunk());`

Para agregar tu slice al store debes exportarlo primero (`export default [slice.variable.name].reducer;`) y sobre el root-reducer (archivo usualmente ubicado en “src/store/root-reducer.ts”) importarlo. (ver Imagen 8) En el caso de que no quieras tener un archivo muy grande en tu root-reducers puedes separar por modulos los reducer y como muestra abajo hacer un nuevo `combineReducers` que puedes exportar. Pero debe estar en tu reducer principal para poder ser utilizado por el store.

A screenshot of a code editor showing the root-reducer.ts file. The code is:

```
import { combineReducers } from "redux";  
import CommonReducer from "common/store";  
import PCPReducer from "pcp/store";  
import AuthSliceReducer from "common/auth/store/index";  
import CommunicationReducer from "common/communication/store";  
  
export default combineReducers({  
  auth: AuthSliceReducer,  
  communication: CommunicationReducer,  
  common: CommonReducer,  
  pcp: PCPReducer,  
});
```

Figure 8: Imagen 8