

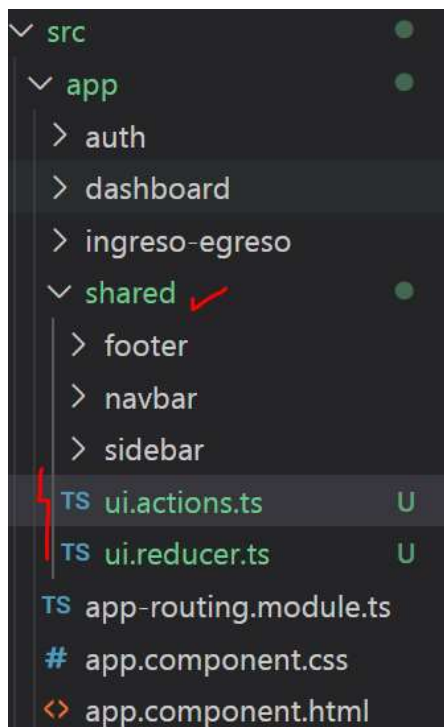
Documentación: <https://github.com/ngrx/platform/tree/master/docs/store>

### Instalando ngrx:

```
npm install @ngrx/store
```

```
F:\proyectos angular\ingresos-egresos>npm install @ngrx/store
```

### Creando las acciones y reducers relacionadas a la interfaz de usuario (shared):



Action y reducer para el “Loading”

Actions:

```
import { Action } from "@ngrx/store";

export const ACTIVAR_LOADING = "ACTIVAR_LOADING";
export const DESACTIVAR_LOADING = "DESACTIVAR_LOADING";

export class DesactivarLoadingAction implements Action {
  readonly type = DESACTIVAR_LOADING;
  constructor() {}
}

export class ActivarLoadingAction implements Action {
  readonly type = ACTIVAR_LOADING;
  constructor() {}
}

export type acciones = DesactivarLoadingAction | ActivarLoadingAction;
```

Reducer:

```
import * as fromUI from "../ui.actions";

export interface State {
  isLoading: boolean;
}

export const initialState: State = {
  isLoading: false,
};

export function uiReducer(
```

```

    state = initialState,
    action: fromUI.acciones
  ): State {
    switch (action.type) {
      case fromUI.ACTIVAR_LOADING: {
        return {
          isLoading: true,
        };
      }

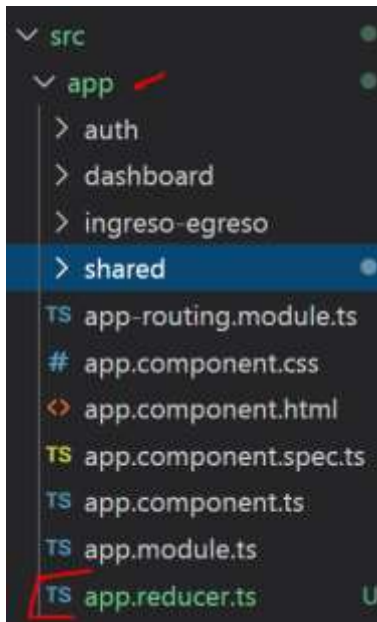
      case fromUI.DESACTIVAR_LOADING: {
        return {
          isLoading: false,
        };
      }

      default:
        return state;
    }
  }
}

```

## Estado global de la aplicación - ActionReducerMap

En el app, creando el archivo “app.reducer.ts” que tendrá toda la definición de los estados de la aplicación:



```

import { ActionReducerMap } from "@ngrx/store";

import * as fromUI from "../shared/ui.reducer";

export interface AppState {
  ui: fromUI.State;
}

// Configuración global
export const appReducers: ActionReducerMap<AppState> = {
  ui: fromUI.uiReducer,
};

```

Configurando en el módulo app:

```

import { BrowserModule } from "@angular/platform-browser";
import { NgModule } from "@angular/core";

// Módulos
import { FormsModule } from "@angular/forms";
import { AppRoutingModuleModule } from "../app-routing.module";

```

```
// NGRX
import { StoreModule } from "@ngrx/store";
import { appReducers } from "../app.reducer";

// firebase
import { AngularFireModule } from "angularfire2";
import { AngularFirestoreModule } from "angularfire2/firestore";
import { AngularFireAuthModule } from "angularfire2/auth";
// Enviroments
import { environment } from "src/environments/environment";

import { AppComponent } from "../app.component";
import { LoginComponent } from "../auth/login/login.component";
import { DashboardComponent } from "../dashboard/dashboard.component";
import { IngresoEgresoComponent } from "../ingreso-egreso/ingreso-egreso.component";
import { EstadisticaComponent } from "../ingreso-egreso/estadistica/estadistica.component";
import { DetalleComponent } from "../ingreso-egreso/detalle/detalle.component";
import { FooterComponent } from "../shared/footer/footer.component";
import { NavbarComponent } from "../shared/navbar/navbar.component";
import { SidebarComponent } from "../shared/sidebar/sidebar.component";
import { RegisterComponent } from "../auth/register/register.component";


@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    DashboardComponent,
    IngresoEgresoComponent,
    EstadisticaComponent,
    DetalleComponent,
    FooterComponent,
    NavbarComponent,
    SidebarComponent,
    RegisterComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    AngularFireModule.initializeApp(environment.firebase),
    AngularFirestoreModule,
    AngularFireAuthModule,
    StoreModule.forRoot(appReducers),
  ],
  providers: [],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

## Implementando el devTools

`npm install @ngrx/store-devtools --save`

### En el app.module:

```
// NGRX
import { StoreModule } from "@ngrx/store";
import { appReducers } from "../app.reducer";
import { StoreDevtoolsModule } from "@ngrx/store-devtools";
```



```
imports: [
  BrowserModule,
  AppRoutingModule,
  FormsModule,
  AngularFireModule.initializeApp(environment.firebase),
  AngularFirestoreModule,
  AngularFireAuthModule,
  StoreModule.forRoot(appReducers),
  StoreDevtoolsModule.instrument({
    maxAge: 25, // Retains last 25 states
    logOnly: environment.production, // Restrict extension to log-only mode
  })),
]
```

### Dispatch - Activar y Desactivar loading

Cuando se registre un usuario, cambiar el estado "isLoading" a true y cuando se tenga información de regreso (del usuario firebase) cambiarlo a false:

#### En el servicio "auth.service":

En crear usuario:

```
import { Store } from "@ngrx/store";
import {
  ActivarLoadingAction,
  DesactivarLoadingAction,
} from "../shared/ui.actions";

crearUsuario(nombre: string, email: string, password: string) {
  // dispatch de la acción
  this.store.dispatch(new ActivarLoadingAction());

  this.afAuth.auth
    .createUserWithEmailAndPassword(email, password)
    .then((resp) => {
      // Creando usuario en la BD
      const user: User = {
        uid: resp.user.uid,
        nombre: nombre,
        email: resp.user.email,
      };
      /* Mandando a firebase */
      // primer Nivel/Segundo Nivel
      this.afDB
        .doc(`${user.uid}/usuario`)
        .set(user)
        .then(() => {
          this.router.navigate(["/"]);
          this.store.dispatch(new DesactivarLoadingAction());
        });
    })
    .catch((error) => {
      this.store.dispatch(new DesactivarLoadingAction());
      Swal.fire("Error en el registro", error.message, "error");
    });
}
```

En el login

```
login(email: string, password: string) {
  this.store.dispatch(new ActivarLoadingAction());

  this.afAuth.auth
    .signInWithEmailAndPassword(email, password)
    .then((resp) => {
      // console.log(resp);
      this.router.navigate(["/"]);
      this.store.dispatch(new DesactivarLoadingAction());
    })
    .catch((error) => {
      this.store.dispatch(new DesactivarLoadingAction());
      Swal.fire("Error en el login", error.message, "error");
      /* console.error(error); */
    });
}
```

## Colocando un cargador en el register – lo mismo para el login:

Register.ts

```
import { Component, OnInit, OnDestroy } from "@angular/core";
import { AuthService } from "../auth.service";
import { Store } from "@ngrx/store";
import { AppState } from "src/app/app.reducer";
import { Subscription } from "rxjs";

@Component({
  selector: "app-register",
  templateUrl: "../register.component.html",
  styleUrls: ["../register.component.css"],
})
export class RegisterComponent implements OnInit, OnDestroy {
  cargando: boolean;
  subscription: Subscription;

  constructor(
    private authService: AuthService,
    private store: Store<AppState>
  ) {}

  ngOnInit() {
    this.store.select("ui").subscribe((ui) => (this.cargando = ui.isLoading));
  }

  ngOnDestroy(): void {
    this.subscription.unsubscribe();
  }

  onSubmit(data: any) {
    this.authService.crearUsuario(data.nombre, data.email, data.password);
  }
}
```

En el template:

```
<div class="form-group">
  <button
    *ngIf="!cargando"
    [disabled]="form.invalid"
    class="btn btn-primary submit-btn btn-block"
  >
    Crear cuenta
  </button>
  <button
    *ngIf="cargando"
    [disabled]="true"
    class="btn btn-primary submit-btn btn-block"
  >
    <i class="fa fa-spin fa-sync"></i> Creando usuario ...
  </button>
</div>
```

## Auth Actions y Reducer

Creando las acciones y el reducer auth:

Action:

```
import { Action } from "@ngrx/store";
import { User } from "../user.model";

export const SET_USER = "[Auth] SET_USER";

export class SetUserAction implements Action {
  readonly type = SET_USER;
  constructor(public user: User) {}
}

export type acciones = SetUserAction;
```

Reducer:

```
import * as fromAuth from "../auth.actions";
import { User } from "../user.model";

export interface AuthState {
  user: User;
}

export const initialState: AuthState = {
  user: null,
};

export function authReducer(
  state = initialState,
  action: fromAuth.acciones
): AuthState {
  switch (action.type) {
    case fromAuth.SET_USER: {
      return {
        user: {
          ...action.user,
        },
      };
    }
  }

  default:
```

```

        return state;
    }
}

```

En el app.reducer:

```

import { ActionReducerMap } from "@ngrx/store";

import * as fromUI from "../shared/ui.reducer";
import * as fromAuth from "../auth/auth.reducer";

export interface AppState {
  ui: fromUI.State;
  auth: fromAuth.AuthState;
}

// Configuración global
export const appReducers: ActionReducerMap<AppState> = {
  ui: fromUI.uiReducer,
  auth: fromAuth.authReducer,
};

```

Cambiando el modelo:

```

export class User {
  public nombre: string;
  public email: string;
  public uid: string;

  constructor(obj: DataObj) {
    this.nombre = (obj && obj.nombre) || null;
    this.email = (obj && obj.email) || null;
    this.uid = (obj && obj.uid) || null;
  }
}

interface DataObj {
  uid: string;
  email: string;
  nombre: string;
}

```

Llamando la acción en el service:

```

// Escuchar el estado del usuario
initAuthListener() {
  this.afAuth.authState.subscribe((fbUser: firebase.User) => {
    if (fbUser) {
      this.userSubscription = this.afDB
        .doc(`${fbUser.uid}/usuario`)
        .valueChanges()
        .subscribe((usuarioObj: any) => {
          // console.log(usuarioObj);
          const newUser = new User(usuarioObj);
          this.store.dispatch(new SetUserAction(newUser));
        });
    } else {
      // Si ya no se tiene el usuario firebase, desuscribirse
      this.userSubscription.unsubscribe();
    }
  });
}

```