Ver; https://ngrx.io/guide/store/install

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

Instalando ngrx

npm install @ngrx/store --save

Creando reducer en “contador”

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

export function contadorReducer(state: number = 10, action: Action) {
  switch (action.type) {
    case "INCREMENTAR":
      return (state += 1);
    case "DECREMENTAR":
      return (state -= 1);

    default:
      return state;
  }
}
```

Importar en el appModule

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

/* ngrx */
import { StoreModule } from "@ngrx/store";
import { contadorReducer } from "./contador/contador.reducer";

import { AppRoutingModule } from "./app-routing.module";
import { AppComponent } from "./app.component";
import { HijoComponent } from "./contador/hijo/hijo.component";
import { NietoComponent } from "./contador/nieto/nieto.component";

@NgModule({
  declarations: [AppComponent, HijoComponent, NietoComponent],
  imports: [
    BrowserModule,
    AppRoutingModule,
    StoreModule.forRoot({ contador: contadorReducer }),
  ],
  providers: [],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

Usando el reducer en app-componet

```
import { Component } from "@angular/core";
import { Store, Action } from "@ngrx/store";

interface AppState {
  contador: number;
}

@Component({
  selector: "app-root",
  templateUrl: "./app.component.html",
  styleUrls: ["./app.component.css"],
})
export class AppComponent {
  contador: number;
```

```
  constructor(private store: Store<AppState>) {
    /* this.contador = 10; */
    this.store.subscribe((state) => {
      this.contador = state.contador
    });
  }

  incrementar() {
    // Acción
    const accion: Action = {
      type: "INCREMENTAR",
    };

    this.store.dispatch(accion);
  }

  decrementar() {
    // Acción
    const accion: Action = {
      type: "DECREMENTAR",
    };

    this.store.dispatch(accion);
  }
}
```

Action Creator - Creador de acciones

Creando en la carpeta “contador” contador.actions.ts:

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

export const INCREMENTAR = "[Contador] Incrementar";
export const DECREMENTAR = "[Contador] Decrementar";

export class IncrementarAction implements Action {
  readonly type = INCREMENTAR;
}

export class DecrementarAction implements Action {
  readonly type = DECREMENTAR;
}
```

Modificar el reducer:

```
import { Action } from "@ngrx/store";
import { INCREMENTAR, DECREMENTAR } from "./contador.actions";

export function contadorReducer(state: number = 10, action: Action) {
  switch (action.type) {
    case INCREMENTAR:
      return (state += 1);
    case DECREMENTAR:
      return (state -= 1);

    default:
      return state;
  }
}
```

En el app-component:

```
import { Component } from "@angular/core";
import { Store } from "@ngrx/store";
```

```
import {
  IncrementarAction,
  DecrementarAction,
} from "./contador/contador.actions";

interface AppState {
  contador: number;
}

@Component({
  selector: "app-root",
  templateUrl: "./app.component.html",
  styleUrls: ["./app.component.css"],
})
export class AppComponent {
  contador: number;

  constructor(private store: Store<AppState>) {
    /* this.contador = 10; */
    this.store.subscribe((state) => {
      this.contador = state.contador;
    });
  }

  incrementar() {
    // Acción
    const accion = new IncrementarAction();
    this.store.dispatch(accion);
  }

  decrementar() {
    // Acción
    const accion = new DecrementarAction();
    this.store.dispatch(accion);
  }
}
```

Store DevTools

Ver: https://github.com/ngrx/platform/tree/master/docs/store-devtools

Herramienta de crhome para ver los estados ➔ Al final se puede quitar

npm install @ngrx/store-devtools --save

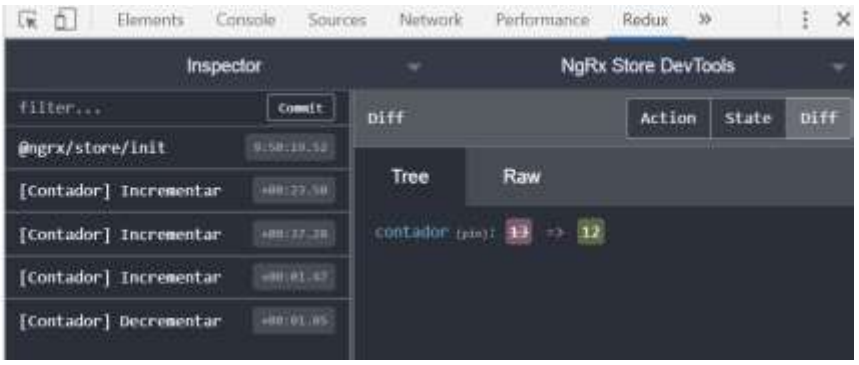
```
F:\proyectos angular\redux-app>npm install @ngrx/store-devtools --save
```

Importar en el appModule:

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

/* ngrx */
import { StoreModule } from "@ngrx/store";
import { StoreDevtoolsModule } from "@ngrx/store-devtools";
import { contadorReducer } from "./contador/contador.reducer";

import { AppRoutingModule } from "./app-routing.module";
import { AppComponent } from "./app.component";
import { HijoComponent } from "./contador/hijo/hijo.component";
import { NietoComponent } from "./contador/nieto/nieto.component";
import { environment } from "src/environments/environment";

@NgModule({
  declarations: [AppComponent, HijoComponent, NietoComponent],
  imports: [
    BrowserModule,
```

```
    AppRoutingModule,
    StoreModule.forRoot({ contador: contadorReducer }),
    StoreDevtoolsModule.instrument({
      maxAge: 25, // Mantener las últimas 25 aciones
      logOnly: environment.production, // Restrict extension to log-only mode
    }),
  ],
  providers: [],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

En crhome:



Escuchar cambios especificos de un elemento del State

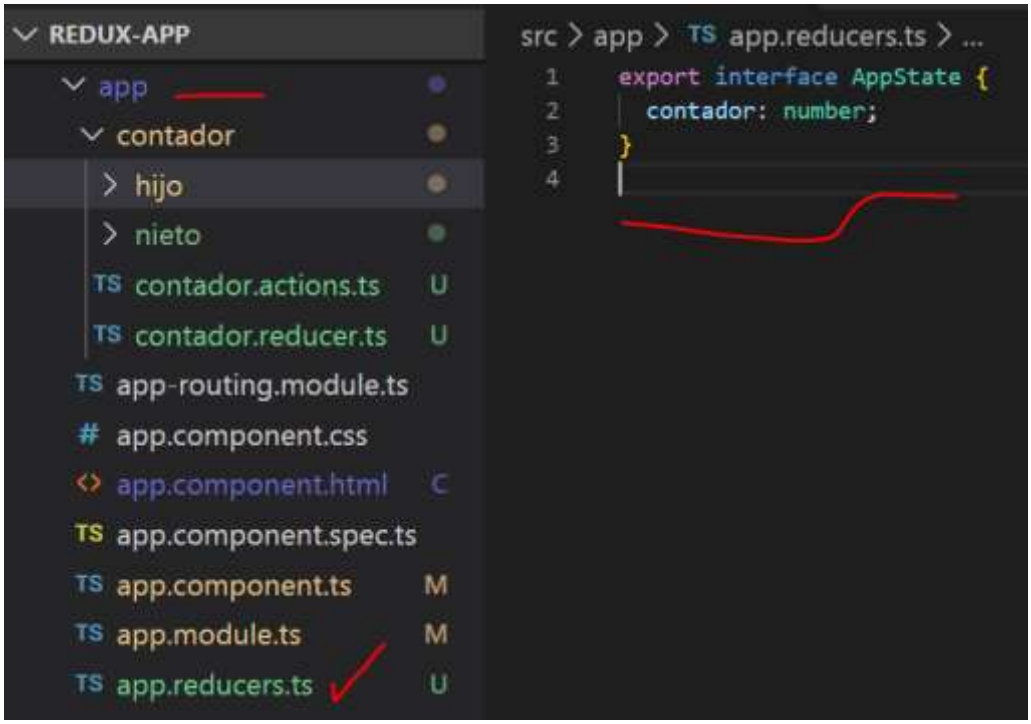
TS app.component.ts

src > app > TS app.component.ts > AppComponent

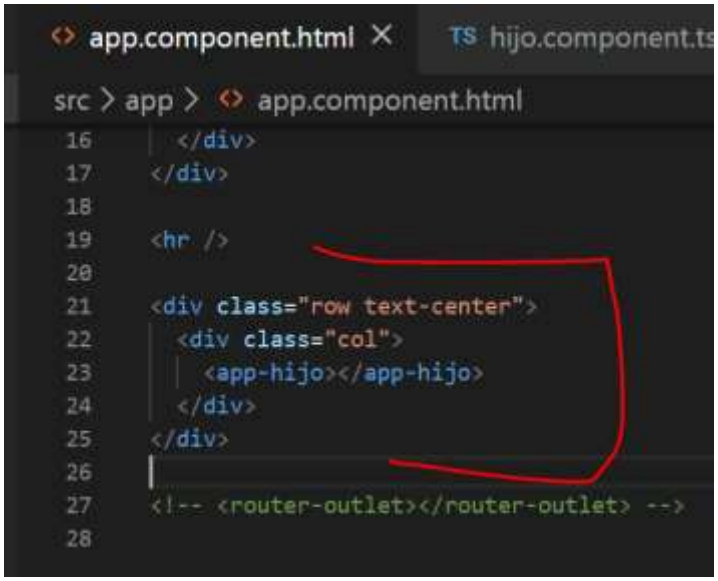
```
16  })
17  export class AppComponent {
18    contador: number;
19
20    constructor(private store: Store<AppState>) {
21      /* this.contador = 10; */
22      this.store.select("contador").subscribe((contador) => {
23        this.contador = contador;
24      });
25    }
26
27    incrementar() {
28      // Acción
29      const accion = new IncrementarAction();
30      this.store.dispatch(accion);
31    }
32
33    decrementar() {
34      // Acción
35      const accion = new DecrementarAction();
36      this.store.dispatch(accion);
37    }
38  }
```

Store en el componente hijo

Creando un archivo en la carpeta “app” llamado “app.reducers.ts” dondes irán todas las interfaces que se usarán:



En el componente padre (app-component) no será necesario enviar ni recibir del hijo:



En el componente hijo:

Creando accion de multiplicar y dividir

Creando nueva acción constante: Necesita un argumento (payload ➔ en el constructor)

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

export const INCREMENTAR = "[Contador] Incrementar";
export const DECREMENTAR = "[Contador] Decrementar";
export const MULTIPLICAR = "[Contador] Multiplicar";

export class IncrementarAction implements Action {
  readonly type = INCREMENTAR;
}

export class DecrementarAction implements Action {
  readonly type = DECREMENTAR;
}

export class MultiplicarAction implements Action {
  readonly type = MULTIPLICAR;
  constructor(public payload: number) {}
}

// Como Action no tiene el payload, se hará lo siguiente:
export type actions = IncrementarAction | DecrementarAction | MultiplicarAction;
```

En el reducers:

```
import {
  INCREMENTAR,
  DECREMENTAR,
  MULTIPLICAR,
  actions,
} from "./contador.actions";

export function contadorReducer(state: number = 10, action: actions) {
  switch (action.type) {
    case INCREMENTAR:
      return (state += 1);
    case DECREMENTAR:
      return (state -= 1);
    case MULTIPLICAR:
      return state * action.payload;

    default:
      return state;
  }
}
```

Enviando la acción a realizar en el hijo (dispatch):

Haciendo lo mismo con la acción para dividir:

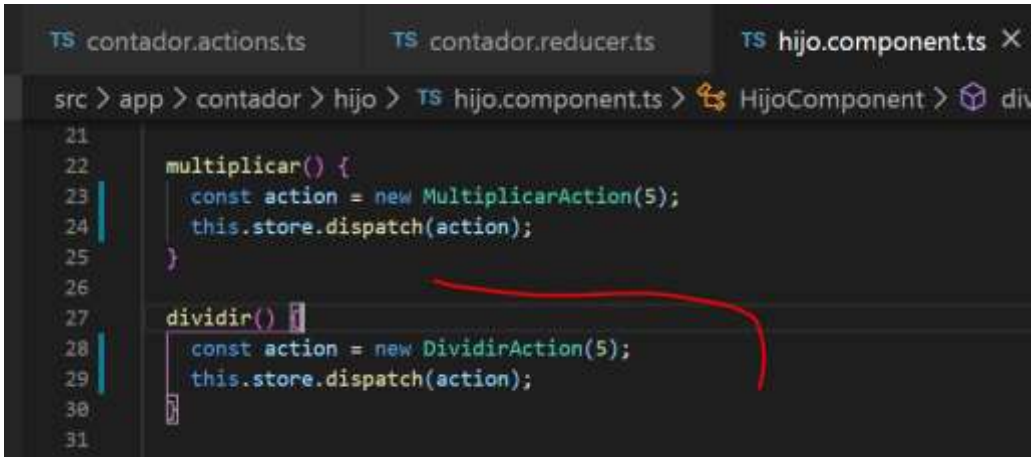
En actions:

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

export const INCREMENTAR = "[Contador] Incrementar";
export const DECREMENTAR = "[Contador] Decrementar";
export const MULTIPLICAR = "[Contador] Multiplicar";
export const DIVIDIR = "[Contador] Dividir";

export class IncrementarAction implements Action {
  readonly type = INCREMENTAR;
}

export class DecrementarAction implements Action {
  readonly type = DECREMENTAR;
}

export class MultiplicarAction implements Action {
  readonly type = MULTIPLICAR;
  constructor(public payload: number) {}
}

export class DividirAction implements Action {
  readonly type = DIVIDIR;
  constructor(public payload: number) {}
}

// Como Action no tiene el payload, se hará lo siguiente:
export type actions =
  | IncrementarAction
  | DecrementarAction
  | MultiplicarAction
  | DividirAction;
```

En reducers:

```
import {
  INCREMENTAR,
  DECREMENTAR,
  MULTIPLICAR,
  DIVIDIR,
  actions,
} from "./contador.actions";

export function contadorReducer(state: number = 10, action: actions) {
  switch (action.type) {
    case INCREMENTAR:
      return (state += 1);
    case DECREMENTAR:
      return (state -= 1);
    case MULTIPLICAR:
      return state * action.payload;

    case DIVIDIR:
      return state / action.payload;

    default:
      return state;
  }
}
```

Dispatch en el componente hijo:



Accion Reset del nieto

En el template del hijo:

```
<h3>Contador</h3>
<h4>{{ contador }}</h4>

<div class="row text-center">
  <div class="col">
    <button (click)="multiplicar()" class="btn btn-primary">
      Multiplicar
    </button>
    <button (click)="dividir()" class="btn btn-info">
      Dividir
    </button>
  </div>
</div>

<hr />

<div class="row text-center">
  <div class="col">
    <app-nieto></app-nieto>
  </div>
</div>
```