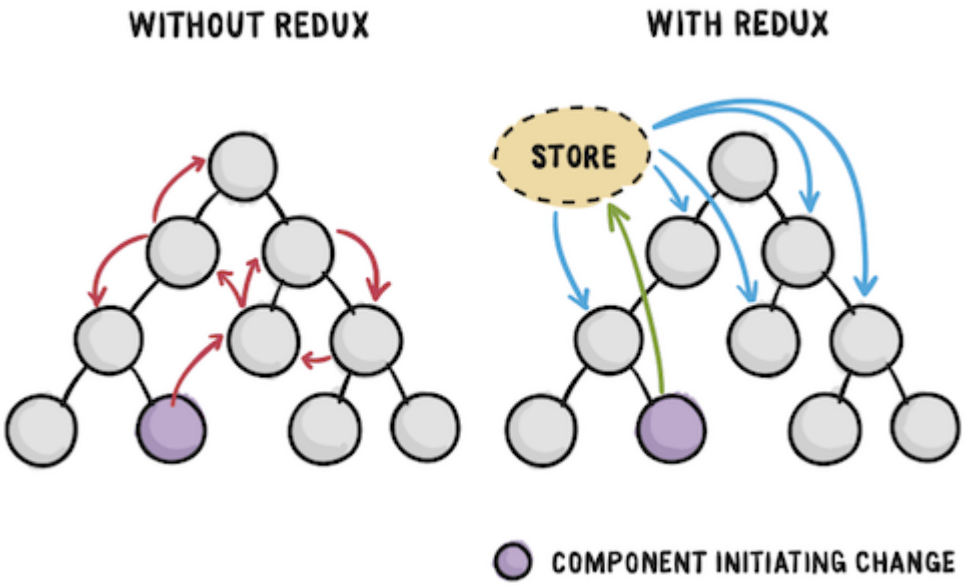
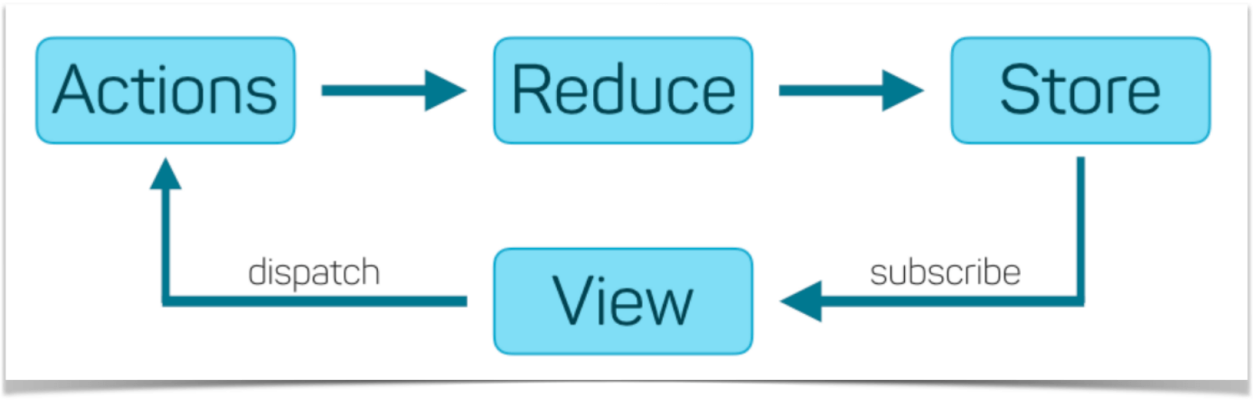
Redux es un patrón de arquitectura de datos que permite manejar el estado de la aplicación de una manera predecible.





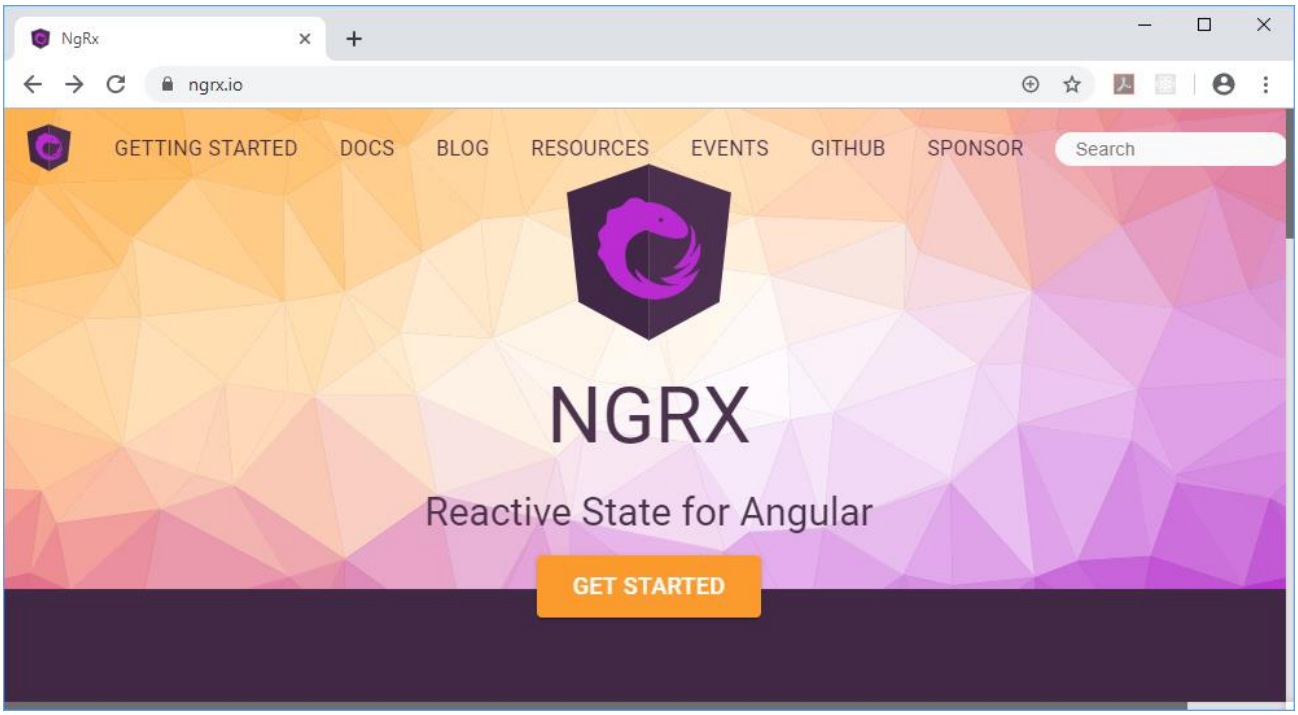
Elementos:

Store: Mantiene el estado, despacha acciones y comunica cambios.

State: Árbol de objetos que contienen la única copia válida de la información.

Actions: Objetos identificados por un tipo.

Reducers : Son funciones puras que reciben dos argumentos: el estado actual y una acción con su tipo y su carga, Clonan el estado, realizan los cambios oportunos y devuelven el estado mutado.



NgRx es el estándar de facto para implementar Redux en Angular,es una librería modular con todo lo necesario para crear grandes aplicaciones.

Componentes:

store: Módulo principal administrador del estado centralizado y reactivo.

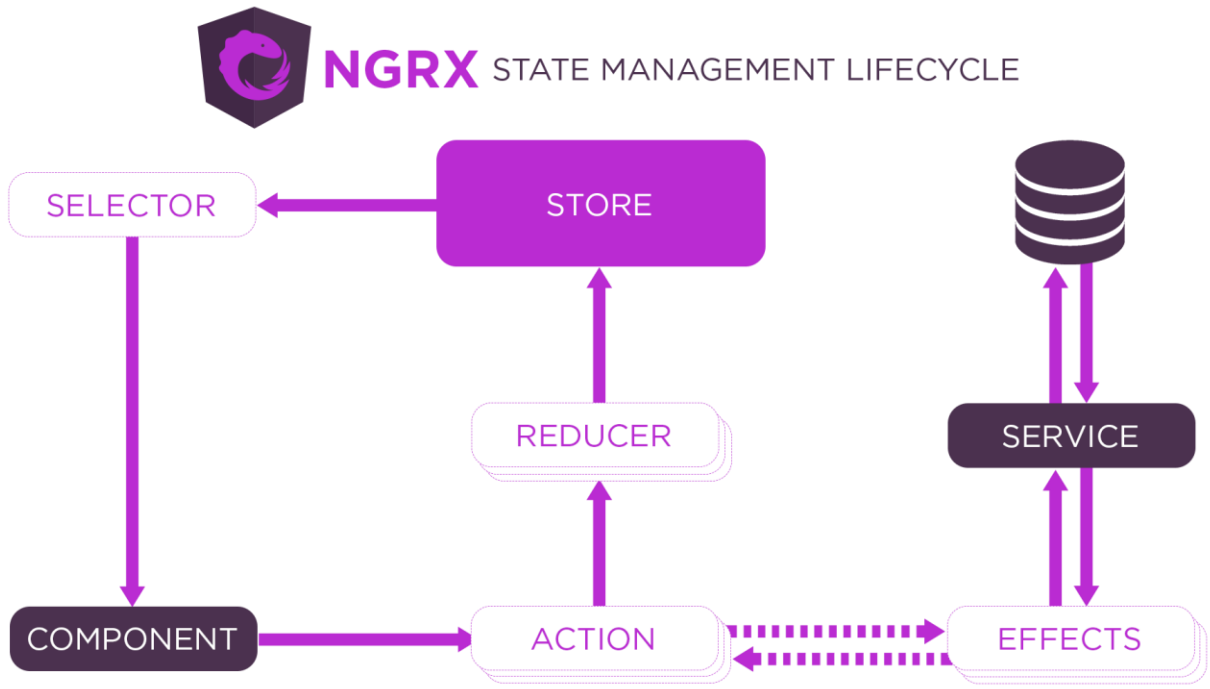
store-devtools: Instrumentación para depurar desde el navegador.

router-Store : Almacena el estado del router de Angular en el store.

effects: Reductores son funciones puras, este módulo es la solución para comandos asíncronos.

schematics, entity, ngrx-data: Son otros módulos opcionales con ayudas y plantillas de NgRX.

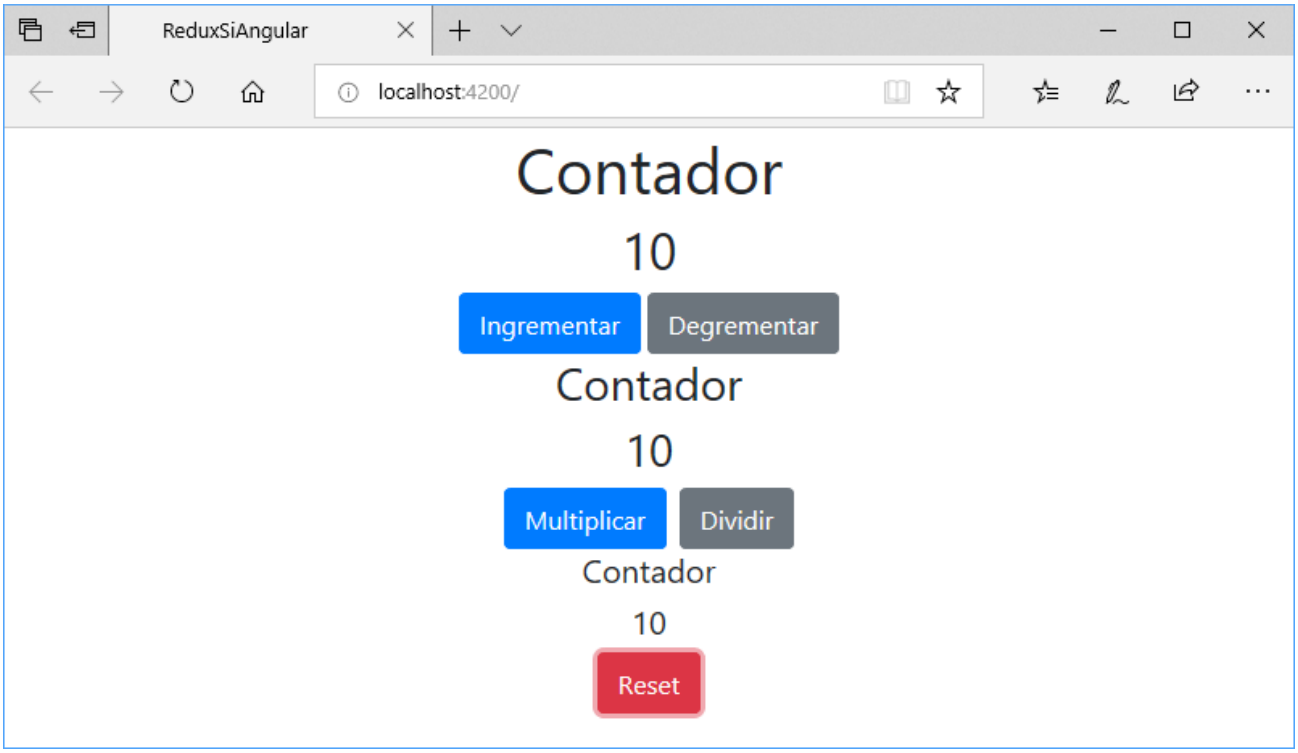
https://ngrx.io/guide/store/install



https://ngrx.io/guide/store

<laboratorio/>

npm install @ngrx/store --save



ng new redux-angular-lab

ng generate component hijo

ng generate component nieto

/src/app/contador.reducer.ts

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

export const contadorReducer = (state: number = 10, action: Action) => {

    switch( action.type ){
        case 'INCREMENTAR':
            return state + 1;
        case 'DECREMENTAR':
            return state - 1;
        case 'MULTIPLICAR':
            return state * 2;
        case 'DIVIDIR':
            return state / 2;
        case 'RESET':
            return 0;
        default:
            return state;
    }

}
```

/src/app/app.module.ts

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

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

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

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

/src/app/app.reducer.ts

```
export interface AppState {
    contador: number;
}
```

/src/app/app.component.ts

```
import { Component } from '@angular/core';
import { Store,Action } from '@ngrx/store';
import { AppState } from './app.reducers';

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

  contador: number;

  constructor(private store: Store<AppState>){
    this.store.select('contador').subscribe( contador =>{
      this.contador = contador;
    });
  }

  incrementar(){
    const accion: Action = {
      type: 'INCREMENTAR'
    }
    this.store.dispatch(accion);
  }

  decrementar(){
    const accion: Action = {
      type: 'DECREMENTAR'
    }
    this.store.dispatch(accion);
  }

}
```

/src/app/app.component.html

```
<div class="container">

  <div class="row">
    <div class="col-12 text-center">
      <h1>Contador</h1>
      <h2>{{ contador }}</h2>
    </div>
  </div>
  <div class="row">
    <div class="col text-center">
      <button class="btn btn-primary" (click)="incrementar()">Ingrementar</button>
      <button class="btn btn-secondary ml-1" (click)="decrementar()">Degrementar</button>
    </div>
  </div>
  <app-hijo></app-hijo>

</div>
```

/src/app/hijo/hijo.component.ts

```
import { Component, OnInit } from '@angular/core';
import { AppState } from '../app.reducers';
import { Store,Action } from '@ngrx/store';

@Component({
  selector: 'app-hijo',
  templateUrl: './hijo.component.html',
  styleUrls: ['./hijo.component.css']
})
export class HijoComponent implements OnInit {

  contador : number;

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

  ngOnInit() {
    this.store.select('contador').subscribe(contador => {
      this.contador = contador
    });
  }

  multiplicar(){
    const accion: Action = {
      type: 'MULTIPLICAR'
    }
    this.store.dispatch(accion);
  }

  dividir(){
    const accion: Action = {
      type: 'DIVIDIR'
    }
    this.store.dispatch(accion);
  }

}
```

/src/app/hijo/hijo.component.html

```
<div class="row">
  <div class="col text-center">
    <h3>Contador</h3>
    <h3>{{contador}}</h3>
  </div>
</div>
<div class="row">
  <div class="col text-center">
    <button class="btn btn-primary" (click)="multiplicar()">Multiplicar</button>
    <button class="btn btn-secondary ml-2" (click)="dividir()">Dividir</button>
  </div>
</div>

<app-nieto></app-nieto>
```

/src/app/nieto/nieto.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Store,Action } from '@ngrx/store';
import { AppState } from '../app.reducers';

@Component({
  selector: 'app-nieto',
  templateUrl: './nieto.component.html',
  styleUrls: ['./nieto.component.css']
})
export class NietoComponent implements OnInit {

  contador: number;

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

  ngOnInit() {
    this.store.select('contador').subscribe(contador => {
      this.contador = contador
    });
  }

  reset(){
    const accion: Action = {
      type: 'RESET'
    }
    this.store.dispatch(accion);
  }

}
```

/src/app/nieto/nieto.component.html

```
<div class="row">
  <div class="col text-center">
    <h5>Contador</h5>
    <h5>{{contador}}</h5>
  </div>
</div>
<div class="row">
  <div class="col text-center">
    <button class="btn btn-danger" (click)="reset()">Reset</button>
  </div>
</div>
```