0-Nx

Repositorios profesionales con Nx

```
1. Nx y el CLI
  1.1 Instalación de Nx y CLI
  1.2 Crear y configurar un workspace

2. Estructura de un workspace
  2.1 Apps
  2.2 Libs

3 Aplicaciones
  3.1 Frontend webs
  3.2 Backend Apis

4. Librerías
  4.1 Librerías en TypeScript
  4.2 Librerías de Angular
```

class: impact

1 Nx y el CLI

Instalación de Nx y CLI

Crear y configurar un workspace

1.1 Instalación de Nx y CLI

> Angular es una plataforma de desarrollo dogmática y llave en mano.

> Nrwl eXtensions es un conjunto de mejoras para el desarrollo empresarial moderno.

```
yarn add global @angular/cli
yarn add global @nrwl/schematics
```

Nx.dev

1.2 Crear y configurar un workspace

```
yarn create nx-workspace angular-boss

empty # para no crear aplicaciones previas

angular CLI # pero listo para trabajar con angular
```

Recomendaciones

Extensiones de Visual Studio Code

```
{
  "recommendations": [
    "nrwl.angular-console",
    "angular.ng-template",
    "ms-vscode.vscode-typescript-tslint-plugin",
    "esbenp.prettier-vscode",
    "pkief.material-icon-theme",
    "christian-kohler.path-intellisense",
    "ban.spellright",
    "johnpapa.angular-essentials"
  ]
}
```

> Recap:

1 Nx y el CLI

Instalación de Nx y CLI

Crear y configurar un workspace

---

class: impact

2 Estructura de un workspace

Apps

Libs

Cosas comunes

- angular.json
- package.json
- ts...

Cosas distintas

- nx.json
    - /tools
    - /apps
    - /libs

2.1 Apps

```
As a: customer,
  I want: to see a shop
  so that: I can buy products

As a: seller,
  I want: to see a warehouse
  so that: I can take control
```

Aplicaciones

- shop
- warehouse

2.2 Libs

```
As a: customer,
  I want: to be greeted
  so that: I feel at home

As a: seller,
  I want: to be greeted
  so that: I feel at home
```

Librerías

- models
- views

> Recap:

2 Estructura de un workspace

Apps

Libs

---

class: impact

3 Aplicaciones

Frontend webs

Backend Apis

3.1 Frontend webs

```
yarn add --dev @nrwl/angular
```

```
ng generate @nrwl/schematics:application shop --inlineStyle --routing --directory=
-p ab-shop  --no-interactive
./apps/shop
./apps/shop-e2e
yarn start
```

```
ng generate @nrwl/schematics:application warehouse --inlineStyle --routing --
directory= -p ab-warehouse  --no-interactive
./apps/warehouse
./apps/warehouse-e2e
ng serve warehouse --port=4202 -o
```

```
  "start:shop": "ng serve shop --port=4201 -o",
  "start:warehouse": "ng serve warehouse --port=4202 -o",
```

```
yarn start:store
yarn start:warehouse
```

3.2 Backend Apis

```
yarn add --dev @nrwl/nest
ng generate @nrwl/nest:application api --no-interactive
ng serve api
"start:api": "ng serve api",
```

```
{
  "start:shop": "ng serve shop --port=4201 -o",
  "build:shop": "ng build shop --prod",
  "test:shop": "ng test shop",
  "start:warehouse": "ng serve warehouse --port=4202 -o",
  "build:warehouse": "ng build warehouse --prod",
  "test:warehouse": "ng test warehouse",
  "start:api": "ng serve api",
  "build:api": "ng build api --prod",
  "test:api": "ng test api",
}
```

> Recap:

3 Aplicaciones

Frontend webs

Backend Apis

---

class: impact

4 Librerías

4.1 Librerías en TypeScript

4.2 Librerías de Angular

4.1 Librerías en TypeScript

```
ng generate @nrwl/workspace:library domain --directory=shared --no-interactive
```

libs\shared\domain\src\lib\models\greetings.interface.ts

```
export interface Greetings {
  message: string;
}
```

tsconfig.json

```
"paths": {
  "@a-boss/domain": ["libs/shared/domain/src/index.ts"]
}
```

4.2 Librerías de Angular

4.2.1 Componentes

```
ng g @nrwl/angular:library ui --directory=shared --prefix=ab-ui --simpleModuleName
--no-interactive
ng g @schematics/angular:component greetings --project=shared-ui --
module=ui.module.ts --export --inlineStyle --inlineTemplate
```

---

libs\shared\ui\src\lib\greetings\greetings.component.ts

```
import { Greetings } from '@a-boss/domain';
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'ab-ui-greetings',
  template: `
    <p>
      {{ theGreeting.message }}
    </p>
  `,
  styles: []
})
export class GreetingsComponent implements OnInit {
  public theGreeting: Greetings = { message: 'Hello world' };
  constructor() {}

  ngOnInit() {}
}
```

---

apps\shop\src\app\app.module.ts

```
import { UiModule } from '@a-boss/ui';
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouterModule } from '@angular/router';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    RouterModule.forRoot([], { initialNavigation: 'enabled' }),
    UiModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

---

apps\shop\src\app\app.component.html

```
<ab-ui-greetings></ab-ui-greetings>
<router-outlet></router-outlet>
```

apps\shop\src\app\app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'ab-shop-root',
  templateUrl: './app.component.html',
  styles: []
})
export class AppComponent {
  title = 'shop';
}
```

4.2.2 Servicios

```
ng g @nrwl/angular:library data --directory=shared --prefix=ab-data --
simpleModuleName
ng g service greetings --project=shared-data --no-flat
```

---

libs\shared\data\src\lib\greetings\greetings.service.ts

```
import { Greetings } from '@a-boss/domain';
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
@Injectable({
  providedIn: 'root'
})
export class GreetingsService {
  private apiUrl = 'http://localhost:3333/api';
  constructor(private httpClient: HttpClient) {}
  public getGrettings$(): Observable<Greetings> {
    return this.httpClient.get<Greetings>(this.apiUrl);
  }
}
```

---

libs\shared\data\src\index.ts

```
export * from './lib/data.module';
export * from './lib/greetings/greetings.service';
```

tsconfig.json

```
"paths": {
  "@a-boss/domain": ["libs/shared/domain/src/index.ts"]
}
```

Y cambiamos libs\shared\ui\src\lib\greetings\greetings.component.ts

```
import { GreetingsService } from '@a-boss/data';
import { Greetings } from '@a-boss/domain';
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'ab-ui-greetings',
  template: `
    <p>
      {{ theGreeting.message }}
    </p>
  `,
  styles: []
})
export class GreetingsComponent implements OnInit {
```

```
  public theGreeting: Greetings = { message: 'Hello world' };
  constructor(private greetingsService: GreetingsService) {}
  public ngOnInit() {
    this.greetingsService.getGrettings$().subscribe(this.appendApiMessage);
  }
  private appendApiMessage = (apiGreetings: Greetings) =>
    (this.theGreeting.message += ' and ' + apiGreetings.message);
}
```

Importando el `DataMdule` en

`libs\shared\ui\src\lib\ui.module.ts`

```
import { DataModule } from '@a-boss/data';
import { CommonModule } from '@angular/common';
import { NgModule } from '@angular/core';
import { GreetingsComponent } from './greetings/greetings.component';

@NgModule({
  imports: [CommonModule, DataModule],
  declarations: [GreetingsComponent],
  exports: [GreetingsComponent]
})
export class UiModule {}
```

Ya puestos podemos adecentar el API

`apps\api\src\app\app.controller.ts`

```
import { Greetings } from '@a-boss/domain';
import { Controller, Get } from '@nestjs/common';
import { AppService } from './app.service';

@Controller()
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get()
  getData(): Greetings {
    return this.appService.getData();
  }
}
```

> Recap:

4 Librerías

Librerías en TypeScript

Librerías de Angular

> Next:

Testing unitario y de integración

Jest para tests unitarios

Cypress para test de integración

> **Blog de apoyo:** Repositorios profesionales con Nx
>
> > By Alberto Basalo

yarn add cypress --dev