



► 0. Nx, grandes proyectos requieren mejores herramientas


Objetivos

- Las grandes aplicaciones necesitan herramientas adecuadas
- Nx es la caja de herramientas ideal para desarrollar grandes aplicaciones
- Aprende a Instalar, configurar y usar las extensiones de Narwhal

Referencia

Nx is Modern Angular


In addition to using modern tools, developers want to work with a CLI that reflects how their needs and expectations have evolved. Angular CLI is an impressive piece of technology. It has robust code-generation and execution capabilities that facilitate onboarding and developer mobility, but it has a few

 <https://blog.nrwl.io/nx-is-modern-angular-bda6cf10746d>



Nx: Extensible Dev Tools for Monorepos


Nx is a suite of powerful, extensible dev tools to help you architect, test, and build at any scale - integrating seamlessly with modern technologies and libraries while providing a robust CLI, caching, dependency management, and more. It has first-class support for many frontend and backend

 <https://nx.dev/>



migration: Migrating existing code bases | Nx angular documentation

Migrating into an Nx workspace can seem intimidating. While every codebase is unique, we can offer recommendations for how to proceed based on the Nrwl team's years of experience. The key to success is an incremental approach. You don't need to migrate your entire codebase at once.


 <https://nx.dev/latest/angular/migration/overview>



Avanzado...

Bulletproof Angular. Angular strict mode explained - Angular inDepth


Angular uses TypeScript because TypeScript provides us with the tooling to create more robust applications. I'm talking about tools for type safety. But tons of developers aren't using the provided tools. They just create applications as they did 10 years ago using JavaScript.

 <https://indepth.dev/posts/1402/bulletproof-angular>




Angular Injection Token for sharing environment across Nx libraries & applications.

Instead, you will create beautiful and reusable libraries!! 🚀 But as you already know, with great power comes great responsibility. Your libraries MUST stay isolated. Meaning you cannot import anything from the actual app into the library. This is a one-way street. But then... How do I use my application-specific

 <https://medium.com/shopstyle-engineering/angular-injection-token-for-sharing-environment-across-nx-libraries-applications-befb9d3c6c67>



►► 1 - Instalación y configuración de Nx

 Nx = CLI ++

 Consola gráfica

<https://marketplace.visualstudio.com/items?itemName=nrwl.angular-console>

 Práctica

Creación de un espacio de trabajo preparado para Angular

```
npm i -g @angular/cli@latest
npm i -g @nrwl/cli@latest
ng v
# short version interactive
npx create-nx-workspace --preset=angular
# long version automated
# set [REPOSITORY, APP_NAME, ORGANIZATION]
npm init nx-workspace REPOSITORY --appName=APP_NAME --defaultBase=main --interactive=false --linter=eslint --nxCloud=false --npmScope=ORGAN
# Here we will use [angularbuilders, www, angular-builders]
npm run update
npm start
```

La creación de espacios de trabajo y su configuración se puede scriptar para facilitar su reutilización.

angularbuilders/ab-generators

Scripts, schematics and generators to build Angular apps with Narwhal Nx dev tools Generate the scaffolding for large Angular applications See the proposed workspace structure Angular Nx Dev Tools Bulma (optional) Universal (optional) (for bash scripts version) Customize and run the scripts Use your <https://github.com/angularbuilders/ab-generators>

angularbuilders/ab-generators

Generators for Angular applications



1 Contributors 0 Issues 0 Stars 0 Forks



Ejemplo de generador y configurador de aplicaciones:

<https://github.com/angularbuilders/ab-generators/blob/main/tools/scripts/0-ab-ng-init.sh>

<https://github.com/angularbuilders/ab-generators/blob/main/tools/scripts/1-ab-ng-app.sh>

<https://github.com/angularbuilders/ab-generators/blob/main/tools/scripts/2-ab-ng-cfg.sh>

2 - Estructura de soluciones multi proyecto mono repositorio



¿Por qué un **Mono Repositorio** de **Múltiples Proyectos** de aplicaciones y librerías?

 Para compartir código entre aplicaciones

Desarrollos similares para un mismo cliente

Desarrollos de un mismo servicio para distintos clientes

 Para dividir aplicaciones enormes en pequeñas librerías

Compilado, prueba y despliegue atómicos

Equipos dispersos manteniendo control de integración y estándares

 Para establecer reglas de dependencia

El sistema de módulos es útil para esconder o publicar su contenido, pero nada más

El sistema de librerías permite establecer jerarquías mediante permisos de dependencias

Workspace



Workspace propuesto: <https://github.com/angularbuilders/ab-generators/blob/main/docs/workspace.md>

Ejemplo de una estructura propuesta para aplicaciones de tamaño medio. (Entre 2-4 devs durante 2-4 años)

Apps

-  www
-  www-e2e

Libs

-  domain
 -  home
 -  not-found
 -  search
 -  search-box
-  shared
 -  auth
 -  data
 -  form
 -  global
 -  layout
 -  ui

Tools

Práctica

Aquí tienes los scripts propuestos para generar esta estructura



Online Code:

<https://github.com/angularbuilders/ab-generators/blob/main/tools/scripts/3-ab-ng-shared.sh>

<https://github.com/angularbuilders/ab-generators/blob/main/tools/scripts/4-ab-ng-pages.sh>

<https://github.com/angularbuilders/ab-generators/blob/main/tools/scripts/5-ab-ng-widgets.sh>

<https://github.com/angularbuilders/ab-generators/blob/main/tools/scripts/6-ab-ng-auth.sh>

Veamos algunos ejemplos en detalle...

 Libs  Domain  Home

Esta librería gestiona **rutas** de carga diferida. Le creamos un componente de tipo **Page** otro componente presentacional y un servicio para manejo de datos

```
nx g library home --importPath=@ab/home --prefix=ab --routing --lazy --parentModule='apps/www/src/app/core/core-routing.module.ts' --tags='  
nx g c home --project=domain-home --flat --skipTests=false --skipSelector --type=Page  
nx g c home --project=domain-home  
nx g s home --project=domain-home --flat
```

Libs Shared Data

Librería especializada en **servicios** de acceso remoto y gestión local de datos

```
nx g library data --directory=shared --importPath=@ab/data --tags='shared, core'  
nx g interceptor adapter --project=shared-data  
nx g interceptor tracker --project=shared-data  
nx g class store --project=shared-data
```

Libs Shared UI

Librería especializada en **componentes (y otros artificios)** reutilizables. Es común encontrarse con algo como esto...

```
nx g library ui --directory=shared --importPath=@ab/ui --prefix=ab-ui --tags='shared, ui'  
nx g c components/breadcrumb --project=shared-ui --export=true  
nx g c components/header --project=shared-ui --export=true  
nx g c components/menu --project=shared-ui --export=true  
nx g c components/message --project=shared-ui --export=true  
nx g c components/notification --project=shared-ui --export=true  
nx g c components/tabs --project=shared-ui --export=true  
nx g c templates/box --project=shared-ui --export=true --type=Template  
nx g c templates/card --project=shared-ui --export=true --type=Template  
nx g c templates/modal --project=shared-ui --export=true --type=Template  
nx g c templates/page --project=shared-ui --export=true --type=Template  
nx g c templates/panel --project=shared-ui --export=true --type=Template  
nx g c templates/section --project=shared-ui --export=true --type=Template  
nx g directive directives/track --project=shared-ui --export=true  
nx g pipe pipes/truncate --project=shared-ui --export=true
```

3 - Ecosistema y utilidades de Nx

Las  **extensiones de Narwhal (Nx)** aportan herramientas y utilidades para programadores listas para usar.

Estilo

Prettier: guía de estilo web casi estándar

ESLint: en lugar del obsoleto par tsLint + codelyzer

Compilación

Diferencial: sólo se recompilan los cambios

Cloud: usar sus servidores para compilar (*puede costar dinero*)

Pruebas

Jest: más rápido que karma+jasmine , perfecto para pruebas de servicios y controladores

Cypress: solución más usada en lugar del obsoleto Protractor

StoryBook: para las pruebas de componentes, directivas y pipes

👉 Práctica propuesta (Agregar Storybook a Shared/UI)

```
ng add @nrwl/storybook
ng generate @nrwl/storybook:configuration --name=shared-ui --uiFramework=@storybook/angular
```

Apps www

```
<nav class="navbar" role="navigation" aria-label="main navigation">
  <div class="navbar-brand">
    <a class="navbar-item" href="/"> <strong class="ml-4 is-size-2">{{ title }}</strong> </a>
  </div>
</nav>
<main>
  <router-outlet></router-outlet>
</main>
<footer class="footer">
  <div class="content has-text-centered">
    <p> by <a href="https://twitter.com/albertobasalo">By Alberto Basalo</a>. </p>
  </div>
</footer>
export class AppComponent {
  title = 'Angular.Builders'
}
```



Alberto Basalo