Qué es Angular

Gustavo García, 19 de abril de 2023.

# Antes de comenzar

Git y GitHub optimizan el tráfico. Por un lado, comprimen todo antes de subirlo. Pero además usan un archivo llamado .gitignore, que indica una cantidad de carpetas o archivos que nunca se subirán desde el repositorio local al remoto, y por lo tanto, no estarán en el remoto. Por ejemplo:

.angular/

node\_modules/

.mvn/

target/

Cuando otra persona clona el repositorio remoto en su PC local, baja lo que hay. En nuestro caso, el proyecto de Angular que nos bajamos del repositorio remoto, no tiene los módulos. Pero los necesitamos, de modo que tenemos que agregarlos al proyecto incompleto que nos descargamos. Abrimos el proyecto en VS Code, vamos a la terminal, y ejecutamos los comandos:

npm install --force

npm audit fix

ng serve --o

# Proyecto Angular viejo

Al cargar un proyecto viejo, puede suceder que nos aparezcan mensajes de error, avisando que nos faltan dependencias, y si las queremos actualizar nos avisan que forzar esas dependencias viejas es peligroso. Podemos usar el comando:

ng update

que nos dirá qué paquetes necesitamos actualizar, y nos sugerirá amablemente el comando preciso que necesitamos ejecutar.

# Angular y React

Angular es una popular plataforma de desarrollo web desarrollada y mantenida por Google. Angular usa TypeScript como su principal lenguaje de programación.

React es como Angular, pero "more minimal", en palabras de la documentación oficial. Hay una cantidad de reviews que los comparan. Todos son sorprendentemente largos y minuciosos. Son realmente distintos. Angular es un framework, React es una biblioteca. Angular es más OOP que React. Fuera de eso, desde cada uno de los numerosos puntos de vista en los que se puede enfocar el análisis, ambos tienen pros y contras. En gran medida, es una cuestión de gustos, o inclinaciones personales.

# Using Angular in Visual Studio Code

<https://code.visualstudio.com/docs/nodejs/angular-tutorial>

Está en inglés. De ser necesario, Google puede traducirlo al español. Sale todo perfecto.

En la terminal de Windows:

node -v o node --version

v16.16.0

npm -v o npm --version

8.19.2

code -v o code –version

1.76.2

ee2b180d582a7f601fa6ecfdad8d9fd269ab1884

x64

# Instalar Angular CLI

Para instalar Angular CLI es necesario haber instalado antes npm. Iniciar PowerShell o la Consola de Windows en Modo Admin, lo que se llama elevado. Ejecutar los siguientes comandos:

npm uninstall -g angular-cli

npm uninstall -g @angular-devkit/architect

npm uninstall -g @angular-devkit/core

npm uninstall -g @angular-devkit/schematics

npm uninstall -g @schematics/angular

npm uninstall angular-cli

npm uninstall @angular-devkit/architect

npm uninstall @angular-devkit/core

npm uninstall @angular-devkit/schematics

npm uninstall @schematics/angular

npm cache verify

npm cache clean --force

npm install -g @angular/cli@latest

Para verificar la versión, ejecutar:

ng version

\_ \_ \_\_\_\_ \_ \_\_\_

/ \ \_ \_\_ \_\_ \_ \_ \_| | \_\_ \_ \_ \_\_ / \_\_\_| | |\_ \_|

/ △ \ | '\_ \ / \_` | | | | |/ \_` | '\_\_| | | | | | |

/ \_\_\_ \| | | | (\_| | |\_| | | (\_| | | | |\_\_\_| |\_\_\_ | |

/\_/ \\_\\_| |\_|\\_\_, |\\_\_,\_|\_|\\_\_,\_|\_| \\_\_\_\_|\_\_\_\_\_|\_\_\_|

|\_\_\_/

Angular CLI: 15.2.4

Node: 16.16.0

Package Manager: npm 8.19.2

OS: win32 x64

Angular: undefined

...

Package Version

------------------------------------------------------

@angular-devkit/architect 0.1502.4 (cli-only)

@angular-devkit/core 15.2.4 (cli-only)

@angular-devkit/schematics 15.2.4 (cli-only)

@schematics/angular 15.2.4 (cli-only)

# [Global Angular CLI version greater than local version](https://stackoverflow.com/questions/44525746/global-angular-cli-version-greater-than-local-version)

Si aparece un mensaje como:

[Your global Angular CLI version is greater than your local version. To disable this warning use "ng config -g cli.warnings.versionMismatch false"](https://stackoverflow.com/questions/57611447/your-global-angular-cli-version-is-greater-than-your-local-version-to-disable-t)

usar la solución de esta respuesta de StackOverflow[[1]](#footnote-1):

npm install --save-dev @angular/cli@latest -g

Esa respuesta, además, explica por qué aparece el mensaje en cuestión.

# Identificadores en Angular

Ver el proyecto A\_c-123.

Cuando generamos una aplicación con el asistente de Angular, la guardará en una carpeta con el nombre que hayamos elegido. Es nombre puede tener letras del alfabeto inglés, mayúsculas o minúsculas, dígitos, underscore (\_) y guión (-). Es posible usar otros caracteres, pero mejor no complicarnos por ahora. No puede comenzar con un dígito.

El componente raíz se llama app-root. Los archivos que lo definen están en la carpeta app dentro de la carpeta src. Estos archivos se llaman app.component.x, donde x es la extensión: css, html o ts.

Para cada componente que definamos dentro de la aplicación deberemos elegir un nombre. El nombre del componente puede tener letras minúsculas y dígitos. Los archivos que lo definen estarán guardados en una carpeta con el nombre que le dimos al componente, dentro de la carpeta app, que es la que contiene toda la aplicación.

# Imports, declarations, providers

La siguiente entrada de Stacj Overflow lo explica:

<https://stackoverflow.com/questions/39062930/what-is-the-difference-between-declarations-providers-and-import-in-ngmodule>

* Los imports hacen que las declaraciones exportadas de otros módulos estén disponibles en el módulo actual.
* Las declaraciones son para hacer que las directivas (incluidos los componentes y las tuberías) del módulo actual estén disponibles para otras directivas en el módulo actual. Los selectores de directivas, componentes o conductos solo se comparan con el HTML si se declaran o importan.
* Los proveedores son para dar a conocer los servicios y valores a DI (inyección de dependencia). Se añaden al ámbito raíz y se inyectan a otros servicios o directivas que los tengan como dependencia.

# Diferencia entre ngOnInit y constructor

## El constructor()

Es el método que se ejecuta cuando se crea una instancia de una clase.

El constructor es propio de una clase en EcmaScript6 y por ende JavaScript llama al constructor antes que a ninguno, lo que significa que no es un buen lugar para avisarle a Angular que ha terminado de inicializar el componente. Es aquí, dentro del constructor, donde podemos aprovechar y decirle qué dependencias necesitamos cargar.

## ngOnInit()

El método ngOnInit() es propio de Angular y se ejecuta después del constructor. A diferencia del constructor, ngOnInit() pertenece al ciclo de vida propio de Angular y es aquí donde le decimos que el componente ya está listo para darle uso. Este es un lugar apropiado para hacer cualquier tarea de inicialización: el componente está creado, pero no ha sido renderizado para que el usuario lo vea.

# CSS box model

La explicación de border y padding (palabra que nuestro superhéroe mispronounces[[2]](#footnote-2)) es confusa. La siguiente página da una explicación simple, clara y correcta:

<https://www.w3schools.com/css/css_boxmodel.asp>

Las unidades de medida que se usan en CSS están explicadas en la siguiente página:

<https://www.w3schools.com/cssref/css_units.asp>

El siguiente artículo de la Wikipedia resume todo el tema:

<https://en.wikipedia.org/wiki/CSS_box_model>

The Cascading Style Sheets (CSS) specification describes how elements of [web pages](https://en.wikipedia.org/wiki/Web_page) are displayed by graphical browsers. Section 4 of the CSS1 specification defines a "formatting model" that gives block-level elements—such as p and blockquote—a width and height, and three levels of boxes surrounding it: padding, borders, and margins.[[4]](https://en.wikipedia.org/wiki/CSS_box_model#cite_note-CSS-19990111-4) While the specification never uses the term "box model" explicitly, the term has become widely used by web developers and web browser vendors.

All HTML elements can be considered "boxes", this includes div tag, p tag, or a tag. Each of those boxes has five modifiable dimensions:

* the height and width describe dimensions of the actual content of the box (text, images, ...)
* the padding describes the space between this content and the border of the box
* the border is any kind of line (solid, dotted, dashed...) surrounding the box, if present
* the margin is the space around the border

According to the CSS1 specification, released by W3C in 1996 and revised in 1999, when a width or height is explicitly specified for any block-level element, it should determine only the width or height of the visible element, with the padding, borders, and margins applied afterward. Before CSS3, this box model was known as **W3C box model**, in CSS3, it is known as the content-box.

The total width of a box is therefore left-margin + left-border + left-padding + width + right-padding + right-border + right-margin. Similarly, the total height of a box equals top-margin + top-border + top-padding + height + bottom-padding + bottom-border + bottom-margin.

For example, the following CSS code

.**myClass** {

**width**: 200px;

**height**: 100px;

**padding**: 10px;

**border**: **solid** 10px **black**;

**margin**: 10px;

}

would specify the box dimensions of each block belonging to 'myClass'. Moreover, each such box will have total height 160[px](https://en.wikipedia.org/wiki/Pixel) and width 260px.

CSS3 introduced the **Internet Explorer box model** to the standard, known referred to as border-box.

# CSS: detectar clases sin usar

Se recomienda instalar la siguiente extensión de VS Code:

Unused CSS Classes for JavaScript/Angul

Una vez instalada esta extensión, cuando abrimos un archivo css, todas las clases que no usan aparecen grisadas, como deshabilitadas.

# Curso de Angular desde cero - Instalaciones necesarias 2022 - #1

<https://youtu.be/csTZcCAE8aw>

11 minutos.

Spoiler alert: los tutoriales son excelentes, pero este muchacho pronuncia mal muchas palabras del inglés. You've been warned.

Descargar Visual Studio Code. [https://code.visualstudio.com/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbkl6dnAxcU9qY3UzNEZTVmZPTlZiSVI5N21sQXxBQ3Jtc0traXVYMTJNMWVDcVYtNFZfQVpkbl84MkV0UjJPaU1TMGV6RmJwR3VwRTd1NllKd0Jyb1JZSUZ3TkttUmZMTkYyalJZWmxTOGZQUkNPRy03ZDNpZjlVZmt3cDUzLXpNRHVZdmJZODlranZpSXVsckEycw&q=https%3A%2F%2Fcode.visualstudio.com%2F)

1º Instalar NodeJs: [https://nodejs.org/es/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbFp4X3hmajBGTHpsMC16emlJVFdMSlR3TGt1QXxBQ3Jtc0tsWVQ0dTM1LXhrc2dQY2Z1MDRCQ3pJTEVLcE1qZzJWZGduWUdqOExvSmdiYkZsLTV4cDl6eExFRmxLc0p5dnBQUVBQNG5GQl9xZkZPTW9LemhCT3V1MFE1NFRjZDMzZjVjVEZHMEs2YV9jR3lxaE8tTQ&q=https%3A%2F%2Fnodejs.org%2Fes%2F)

2º Instalar Angular Cli: [https://angular.io/cli](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqbUJVSzBVWW9HTDBLOGEtSkQ2aHFvaGVWNkFmZ3xBQ3Jtc0tuTWdDUFpxUmExVjJySnR4ZFNUMXd3bUtjcGxRdGtZY0hxaGRDN3luZVRFUWxhcDRIRk5QV2NfUU5DTW83Y2R1aDBnRDBMTkt1dHd4bXBuazVOUmRXQmVBdlF1YmtlNTZFTmpNSEpXR0pCMk53ZHM1Zw&q=https%3A%2F%2Fangular.io%2Fcli)

npm install -g @angular/cli

## Versiones que tenemos

node --version

v16.16.0

npm --version

8.19.2

Lo anterior es lo que da en mi PC hoy.

## Crear nueva aplicación proyecto my-app

Angular contruye SPA, Single Page Applications. Una aplicación que tiene una sola página. Esa página es index.html. Todo lo que hay en una aplicación de Angular es un componente. Un componente puede contener otros componentes. Pero no hay nada que no sea un componente, salvo index.html.

Si hemos instalado Angular desde cero, las primeras veces tardará más, porque necesita descargar bibliotecas. El proyecto para este tutorial está en:

20-que-es-angular\my-app\

Abrir la terminal, y navegar hasta el directorio padre, es decir aquel donde queremos que esté la carpeta que se va a generar y que contendrá todo nuestro proyecto. En la Terminal ejecutar los comandos:

ng new my-app

cd my-app

ng serve -–o

El flag -o es sinónimo de --o y de --open. Es para que la aplicación se abra automáticamente en el browser. Para cortar el servicio desde Git Bash, en la misma ventana en la que arrancamos la aplicación con ng serve, ingresamos Ctrl + c. Vemos el prompt del sistema, lo que nos confirma que el servicio fue terminado.

Si abrimos en el browser la URL:

<http://localhost:4200/>

veremos la aplicación.

Para limpiar el puerto 4200, ejecutar en la consola de Windows el comando:

for /f "tokens=5" %a in ('netstat -aon ^| find /i "LISTENING" ^| find ":4200"') do taskkill /f /pid %a

Una vez que cerramos la ventana del browser, volvemos a la terminal, y puede ser que está como bloqueada, sin respuesta. En ese caso, apretar ctrl + c.

# Curso de Angular - Primer Proyecto - Plugin - 2022 - #2

<https://www.youtube.com/watch?v=R90EqA-O37U&ab_channel=YouTics>

En esta segunda mini clase creamos nuestro primer proyecto, lo ejecutamos en un servidor local, generamos algunos cambios y además instalamos algunos plugin: vscode-icons y ESLint

OJO: al crear el proyecto ponerle que SÍ queremos el router. Él dice que no, pero después sí lo usa.

## Crear proyecto MiPrimeraApp

El proyecto para este tutorial está en:

C:\GIT\20-que-es-angular\MiPrimeraApp\

Si al crear el proyecto nos sale un mensaje como:

Directory is already under version control. Skipping initialization of git.

eso se debe a que la carpeta que contiene a la carpeta del proyecto ya está inicializada como repositorio de Git. Ir a la carpeta raíz del repositorio y pegar el comando:

git add --all

para que agregue todo el proyecto que acabamos de crear, carpetas y subcarpetas.

# Cambiar el título del documento HTML

No se trata de inyectar una variable en un tag <h1>...</h1>, sino de cambiar en run time el tag <title> dentro de la sección <head> de index.html, programáticamente. Me voy a basar en el siguiente artículo:

<https://www.tektutorialshub.com/angular/set-page-title-using-title-service-angular-example/>

Comienzo por crear una nueva aplicación, sobre la cual haré luego los cambios.

En Git Bash navego hasta la carpeta GIT\20-que-es-angular. Pego el comando:

ng new title

Agrego routing. Elijo SCSS. Tarda un poco.

cd title

code .

En la carpeta GIT\20-que-es-angular\title está el proyecto completo. Los comentarios explican todo detalladamente. Está bastante cambiado respecto del original que bajé. Mejor usar esta versión.

## Modo de trabajo

Voy a ir citando los archivos en un cierto orden. Hay que ir al proyecto, abrir ese archivo y leer los comentarios. Es casi seguro que la primera vez no vamos a entender realmente todo. Es natural. No asustarse. Leer con atención, y tratar de memorizar lo que se pueda. Quizá deberemos hacer todo el proceso dos o tres veces. En algún momento el panorama se aclara, y todo cobra su significado. A eso vamos a llegar. Pero para llegar hay que trabajar. No nos va a caer mágicamente en las manos.

## index.html

Es un buen lugar para empezar. Vemos que es una página de inicio en html, muy simple. De hecho, el body solo tiene el tag <app-root></app-root>, que es el selector del componente principal. Se llama selector porque es el nombre por el cual se identifica ese componente. El componente principal es lo que veremos a continuación.

## app.module.ts

Contiene el código relativo a la aplicación en sí misma. Una vez más, recordamos que la arquitectura de Angular esta basada en componentes: todo es un componente, que puede estar contenido en otro componente, o contener otros componentes. La idea es que todo esté concentrado lo más cerca posible de donde se usa, para que sea más fácil encontrarlo.

## app-routing.module.ts

Este módulo se encarga del routing. Recordar que Angular hace aplicaciones que se miran en el browser. Esto significa que tenemos que poner una URL. En esa URL puede haber parámetros.

## Un componente de Angular

Desde el punto de vista del sistema de archivos, un componente está formado por tres archivos que se guardan en una cierta carpeta. A continuación vemos los archivos que pertenecen al componente app-root.

## app.component.html

El template del componente. Esto determina el contenido. Para simplificar, tiene dos tipos de elementos. Lo más fácil es el html puro. Lo no tan fácil son los router links y router outlet. Por ahora, leer los comentarios, registrarlos en la memoria y no tratar de entender ya a fondo, porque es un tema avanzado que veremos más adelante.

## app.component.scss

La hoja de estilos propia de este componente. Cada componente tiene su propia hoja de estilo. Esto tiene varias ventajas. En primer lugar, no tenemos que andar pensando dónde estará definido el estilo que queremos cambiar, ni tenemos que buscar entre miles de líneas. Está en la hoja de estilo del componente. Y la hoja de estilo es corta, tiene pocas líneas.

En esta hoja de estilo definimos el estilo que le vamos a dar los componentes one, two y three, cuando sean renderizados. Les vamos a dar el mismo estilo. No sería inteligente copiar y pegar la definición en la hoja de estilo propia de cada componente. Es más simple definirlo en esta, y que se herede.

## app.component.ts

La parte de programación de este componente. Los comentarios explican las características más destacables. Por ahora no se preocupen de la inyección de dependencias. Es un tema avanzado que veremos más adelante. Simplemente, lean y registren como para que les vaya quedando en la memoria.

## Crear un componente

En la Terminal ejecutamos el comando:

ng g c one

que es una abreviatura de ng generate component one. Es posible que nos aparezca la siguiente pregunta:

? Would you like to share pseudonymous usage data about this project with the Angular Team

at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more

details and how to change this setting, see https://angular.io/analytics. No

Yo le respondí que no (n).

A continuación se genera el componente:

Global setting: enabled

Local setting: disabled

Effective status: disabled

Estos son los archivos creados o modificados en el proceso:

CREATE src/app/one/one.component.html (18 bytes)

CREATE src/app/one/one.component.spec.ts (578 bytes)

CREATE src/app/one/one.component.ts (190 bytes)

CREATE src/app/one/one.component.css (0 bytes)

UPDATE src/app/app.module.ts (710 bytes)

Los 4 primeros, que están dentro de la carpeta one, corresponden estrictamente al componente que acabamos de crear. El último es app.module.ts, del que hablamos antes. Si lo editamos, veremos que hay dos modificaciones: un import y una declaración. Estas dos modificaciones las hace la plataforma automáticamente por nosotros. Lo más seguro es hacerlo por medio del asistente. Pero si alguna vez necesitamos agregar manualmente un componente, este es el modo.

## El componente one

Los archivos que forman el componente one están guardados físicamente en la carpeta one, que está a su vez dentro de la carpeta app.

Notar que la clase del componente implementa la interfaz OnInit. La siguiente respuesta:

https://es.stackoverflow.com/a/172913

aclara que es optativa, no obligatoria. Pero se considera una buena práctica.

Como se dijo antes, el estilo de este componente es todo heredado del componente que lo contiene.

# Curso de Angular Crear Componentes + CSS - 2022 - #3

Lo mismo que antes, conviene usar el proyecto que está en este repositorio, porque está muy cambiado respecto del que usa el autor del tutorial:

<https://www.youtube.com/watch?v=lW_zfBm5t4A&ab_channel=YouTics>

Mi recomendación es que vayan siguiendo al mismo tiempo el tutorial de YouTube y este documento. El video, porque las explicaciones son claras, y además de allí saqué la idea. Pero en este documento yo voy explicando un montón de cambios. Sobre todo, omito un par de cosas que hace ese autor, porque al final no las usa para nada, y solo complican el panorama.

En este capítulo aprenderemos a:

* crear un componente,
* llamar ese componente dentro de otro utilizando un selector,
* crear clases de css y
* aplicarlas dentro de un html.

Lo más importante que tenemos que aprender, al terminar este capítulo, es cómo se arma el frontend, una aplicación completa de Angular, a partir de los componentes.

## Generar el proyecto

Creamos el proyecto para este tutorial. Necesitamos una ventana de Git Bash en la carpeta GIT\20-que-es-angular. Pegamos el comando:

ng n tutorial-3

Contestamos que no queremos routing, elegimos css. Cuando termina de generar el proyecto, cerramos la ventana de Git Bash, y abrimos la carpeta del proyecto recién creado con VS Code.

## Generar componentes

Ya dentro de VS Code, en la Terminal escribimos:

ng g c header

lo cual va a generar un componente llamado header. No generamos los otros dos componente que menciona el autor, porque luego no los usa para nada, y solo agregarían confusión.

## Editar los archivos del proyecto

Vamos a ver los archivos, en un orden que me parece apropiado.

## src\favicon.ico

Este favicon lo puse solo para mostrar cómo se hace, y dónde hay que ponerlo. Claro que hay otras posibilidades, esta es solo una de ellas.

## src\index.html

Es el típico. Notar que tiene idioma español, y un favico distinto. Leer con cuidado el comentario, que explica las URL.

## src\assets\img\212 x 256.jpg

Es el logotipo de la UTN, que después usaremos. Por ahora, simplemente tomamos nota de su ubicación.

## src\app\app.component.ts

El módulo de este componente. Leer los comentarios.

## src\app\app.component.html

Template del componente raíz. Leer los comentarios.

## src\app\app.component.css

La hoja de estilo de este componente.

## src\app\app.module.ts

El módulo de la aplicación. Notar que no es del componente raíz, sino de la aplicación.

## src\app\header\header.component.css

La hoja de estilo para este componente. Leer los comentarios.

## src\app\header\header.component.ts

El módulo de este componente. Leer los comentarios.

# Curso Angular - Data binding interface - array - \*ngFor - 2022 - #4

ESTE TUTORIAL ES REALMENTE DIFÍCIL. TIENE MUCHOS CONCEPTOS DIFERENTES, QUE TRABAJAN COLABORATIVAMENTE PARA CONSEGUIR UN RESULTADO MUY COMPLEJO. RECOMENDACIÓN: LEER TODOS LOS DOCUMENTOS, EN SECUENCIA, PRESTANDO ATENCIÓN, PERO SIN PRETENDER ENTENDER TODO A FONDO EN LA PRIMERA LECTURA. LEER TODO, LLEGAR AL FINAL, Y LUEGO LEER DE NUEVO TODO EN SECUENCIA. LA SEGUNDA VEZ SE ENTENDERÁ MÁS Y MEJOR. LUEGO, REPASAR LAS VECES QUE HAGA FALTA. QUE NO SERÁN POCAS. PERO EL RESULTADO ES SORPRENDENTE.

## Conceptos

Vamos a usar los siguientes conceptos:

* interpolación,
* data binding,
* expresiones de template,
* event binding,
* inyección de dependencias,
* creación de proveedores de servicios,
* flujo de datos de contenedor a contenido, y de contenido a contenedor,
* recorrido de una colección,
* propiedades opcionales,
* arrow functions.

Proyecto en la carpeta tutorial-4-7.

## Cambios

En este proyecto nos vamos a concentrar en la funcionalidad que se logra con la colaboración de todos los componentes. No vamos a hablar de cómo se crean los componentes, porque eso lo vemos en otros proyectos. Acá nos vamos a concentrar en la colaboración y el intercambio de datos.

Este proyecto está muy cambiado respecto de versiones anteriores.

Leer con detenimiento los abundantes comentarios que hay en el código fuente. He tratado de poner las explicaciones exactamente en el lugar en el que está el código. He reunido materiales de distintas fuentes, tratando de unificar todo. Son muchos conceptos difíciles. Yo creo que este es un modo más fácil de llegar a entenderlos.

<https://youtu.be/rOsd3OI9dOA>

**Seguimos avanzando desde cero en la programación web utilizando el framework de Angular. En este capítulo** seguiremos comunicando entre componentes y aprenderemos a crear interface, arrays, Property data binding y la directiva \*ngFor. También seguiremos analizando las propiedades de css e implementadolas en nuestro proyecto. Vamos a ver dos conceptos difíciles al mismo tiempo, funcionando armónicamente. Por un lado vamos a recorrer un array de datos con la directiva \*ngFor, y por otro vamos a provocar un flujo de información, o sea de datos, desde el array de datos a las propiedades del componente. El resultado será que vamos a generar, a partir del componente y del array de datos, un array de componentes, en el cual cada instancia del componente tendrá sus propiedades cargadas con los datos que le vinieron del array de datos que recorrimos.

## src\index.html

La única página de la aplicación. Recordar que Angular construye Single Page Applications. Leer los comentarios. Notar el modo de indicar el path donde está el favicon. Y notar que el directorio raíz de la aplicación ha quedado ordenado: lo único que hay es index.html.

## src\app\app.component.css

Hoja de estilo del componente raíz. Acá se define el estilo grid, que permite mostrar en la pantalla una grilla de cards.

## src\app\app.component.html

Template del componente raíz.

### Listado de elementos con la directiva \*ngFor

Vamos a analizar la siguiente línea de código:

<div \*ngFor="let item of items">{{item}}</div>

La directiva \*ngFor se utiliza en Angular para iterar sobre una lista de elementos. Por ejemplo:

\*ngFor="let item of items"

El objeto items debe ser iterable, por ejemplo un array o colección. Cada uno de los elementos de la colección items es un item. Los nombres de la colección y de cada uno de los ítems que la componen son arbitrarios. La directiva \*ngFor recorre la colección items. Es decir que, de a uno por vez, toma todos y cada uno de los ítems que forman la colección, cada uno de ellos una y solo una vez. No omite ni repite ninguno. Cada vez que toma un ítem, lo hace disponible en la variable item, que se ha declarado con la palabra clave let.

La directiva \*ngFor está dentro de un tag div de HTML:

<div></div>

Dentro del tag div hay una interpolación:

{{item}}

Poniendo todo junto, queda:

<div \*ngFor="let item of items">{{item}}</div>

La directiva \*ngFor recorre la colección items. Cada uno de los elementos de la colección queda temporalmente almacenado en la variable item. Esa variable item, poblada con los valores tomados del elemento de la colección items, es hecha disponible para el template por Angular. En la interpolación se renderiza la propiedad item en el browser. Como la directiva \*ngFor recorrió una colección, y por cada item se renderizó una div, al final tendremos renderizada un array de divs.

Para obtener más información, consulte el [resumen](https://angular.io/guide/structural-directives#shorthand) de [Directivas estructurales](https://angular.io/guide/structural-directives).

## src\app\app.component.ts

El módulo del componente raíz. Leer todos los comentarios. En particular, lo que explica en la sección previa, sobre la directiva \*ngFor, se ve aplicado en este módulo.

## src\app\app.module.ts

El módulo de la aplicación.

## src\app\servicio-favoritos.service.ts

Este módulo define una clase que es un proveedor de servicio. Este servicio está pensado para ser inyectado en otra parte, como una inyección de dependencias. Este archivo es uno de los más complicados de este proyecto.

## src\app\card\card.component.css

Hoja de estilo del componente card. Define todos los estilos que se usan en el template.

## src\app\card\card.component.html

Template del componente card. Leer despacio todos los comentarios.

## src\app\card\card.component.ts

El módulo de este componente. Acá es donde se implementa la inyección de dependencia.

## src\app\favoritos\favoritos.component.css

Hoja de estilo de este componente. Define varias clases que se usan en el template.

## src\app\favoritos\favoritos.component.html

Template de este componente.

## src\app\favoritos\favoritos.component.ts

Módulo de este componente. Usa las clases definidas en la css.

## Carpetas para los assets

Dentro de la carpeta assets creamos una carpeta img. Dentro de la carpeta img creamos tres carpetas: favicon, logo y personajes.

## src\assets\img\favicon\favicon.ico

El favicon para la aplicación. Tomar nota del path, porque luego vamos a tener que ponerlo correctamente en index.html. Es más prolijo poner este recurso en una carpeta adecuada. Por decirlo de otro modo, tener el favicon en el directorio raíz de la aplicación, es decir en la misma carpeta donde está index.html es poco profesional.

## src\assets\img\logo\212 x 256.jpg

El logotipo de la UTN. Lo vamos a usar en el componente header.

## src\assets\img\personajes\chacha.gif

Esta y las otras dos imágenes las vamos a usar en los componentes card. Tomar nota del path.

HASTA ACÁ

# Curso de Angular - Clonar e Instalar un Proyecto de Angular - Git + GitHub - 2022 - #5

<https://youtu.be/x3nFRQtauEM>

En este capitulo aprenderemos lo básico para guardar un proyecto en la nube y como recuperar ese proyecto con todos los archivos que necesita angular para trabajar.

# Curso de Angular Habilitar Terminal de Windows en Visual Studio Code - 2022 - #6

<https://www.youtube.com/watch?v=4aqA8Ny090o&t=1s&ab_channel=YouTics>

En este capitulo contestaremos a una duda importante acerca de cómo habilitar una terminal en Windows.

## PowerShell's execution policy[[3]](#footnote-3)

La política de ejecución de PowerShell es una característica de seguridad que controla las condiciones bajo las cuales PowerShell carga archivos de configuración y ejecuta scripts. Esta característica ayuda a prevenir la ejecución de scripts maliciosos. En una computadora con Windows, puede establecer una política de ejecución para la computadora local, para el usuario actual o para una sesión en particular. También puede usar una configuración de Política de grupo para establecer políticas de ejecución para computadoras y usuarios. Las políticas de ejecución para la computadora local y el usuario actual se almacenan en el registro. No necesita establecer políticas de ejecución en su perfil de PowerShell. La política de ejecución para una sesión en particular se almacena solo en la memoria y se pierde cuando se cierra la sesión. La política de ejecución no es un sistema de seguridad que restrinja las acciones del usuario. Por ejemplo, los usuarios pueden omitir fácilmente una política escribiendo el contenido del script en la línea de comando cuando no pueden ejecutar un script. En cambio, la política de ejecución ayuda a los usuarios a establecer reglas básicas y evita que las violen sin querer. En equipos que no son Windows, la política de ejecución predeterminada es Unrestricted y no se puede cambiar. El cmdlet Set-ExecutionPolicy está disponible, pero PowerShell muestra un mensaje de consola que indica que no es compatible. Si bien Get-ExecutionPolicy devuelve Unrestricted en plataformas que no son de Windows, el comportamiento realmente coincide con Bypass porque esas plataformas no implementan las Zonas de seguridad de Windows.

## Para configurar

Un modo es cerrar todas las ventanas de VS Code que tengamos abiertas; ejecutar VS Code como Administrador, en VS Code abrir una nueva terminal.

Otro modo es el que recomienda el tutorial. Las mayúsculas no son importantes.

En PowerShell pegar el comando:

PS C:\Users\Gustavo>Get-ExecutionPolicy

Restricted

Esto es lo que hay que cambiar. Para eso, pegar el comando:

Set-ExecutionPolicy Unrestricted

# Curso de Angular - @Input - 2022 - #7

Proyecto en la carpeta tutorial-4-7.

<https://youtu.be/57Z_U6vPIl8>

En este capitulo aprenderemos a comunicar información entre componentes utilizando el decorador @Input.

# Curso de Angular | Databinding - Interpolación - #8

<https://youtu.be/13jsC5nLntE>

Proyecto en tutorial-8.

En este capitulo del curso de angular abordaremos con mas profundidad el concepto de Databinding con ejemplos de interpolación.

La interpolación es one-way binding, parte del ts y se dirige al html.

Property binding es one-way binding, parte del ts y se dirige al html.

Event binding es one-way binding, parte del html y se dirige al ts.

## BootStrap

Vamos a ver que en este tutorial usamos al framework BootStrap, que es muy popular. El sitio oficial es:

<https://getbootstrap.com/>

En esa página, vamos a Include via CDN, elegimos la opción CSS only, copiamos y pegamos en la sección head de index.html.

# Curso de Angular 2022 Property Binding - #9

Fácil.

<https://youtu.be/WiSC5-yr13U>

Proyecto en tutorial-8-9.

# Angular 12 - Interacción de componentes. @Input() y @Output(). Tutorial en español

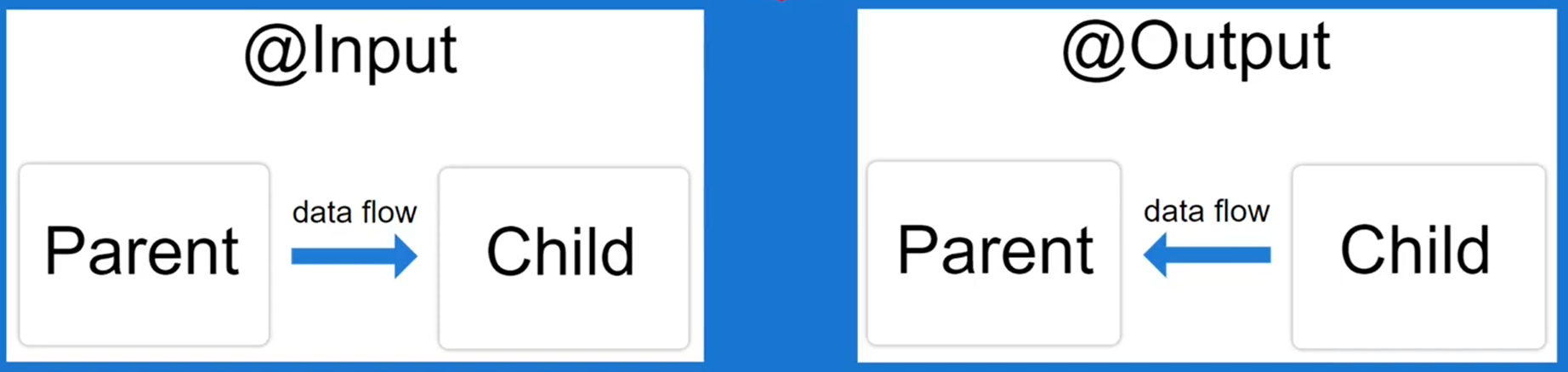
Excelente.

<https://youtu.be/acxmiISz5Ag>

El proyecto se llama sharing-data-between-components.

El código del proyecto tiene muchos comentarios, que tratan de explicar cómo funciona cada uno de estos tres mecanismos.

Angular 12 - Interacción de componentes. @Input() y @Output(). Tutorial en español. Angular 12 - Sharing data between child and parent components.



Enviar datos del padre al hijo con @Input y property binding.

Enviar datos del hijo al padre con @Output y event binding.

00:00 - Intro y configuración del proyecto

04:38 - Enviar datos del padre al hijo

07:52 - Enviar datos del hijo al padre

15:08 - Borrar input

Se puede poner en index.html otros componentes además de app-root, pero hay que agregarlos a la sección bootstrap, como se explica en la siguiente respuesta de StackOverflow:

<https://stackoverflow.com/a/57916874/2740402>

# Comunicación entre componentes de Angular

<https://www.youtube.com/watch?v=df0eH9mM9nU&ab_channel=FernandoHerrera>

# Agregar routing a una SPA de Angular[[4]](#footnote-4)

Editar app.module.ts.

Agregar al principio:

import { [RouterModule](https://angular.io/api/router/RouterModule) } from '@angular/router';

Bajar hasta la sección @[NgModule](https://angular.io/api/core/NgModule)(). Agregar, dentro de la propiedad imports lo siguiente:

RouterModule.forRoot([

{path: 'crisis-list', component: CrisisListComponent},

{path: 'heroes-list', component: HeroesListComponent},

]),

de manera que quede por ejemplo así:

  imports: [

    BrowserModule,

    RouterModule.forRoot([

      { path: 'crisis-list', component: CrisisListComponent },

      { path: 'heroes-list', component: HeroesListComponent },

    ]),

  ],

Grabar.

Editar app.component.html. Agregar la directiva router-outlet:

<[router-outlet](https://angular.io/api/router/RouterOutlet)></[router-outlet](https://angular.io/api/router/RouterOutlet)>

Agregar los siguiente controles de navegación:

<nav>

<a class="button" [routerLink](https://angular.io/api/router/RouterLink)="/crisis-list">Crisis Center</a> |

<a class="button" [routerLink](https://angular.io/api/router/RouterLink)="/heroes-list">Heroes</a>

</nav>

1. <https://stackoverflow.com/a/44526528/2740402> [↑](#footnote-ref-1)
2. <https://dictionary.cambridge.org/dictionary/english/padding> [↑](#footnote-ref-2)
3. <https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-7.2#long-description> [↑](#footnote-ref-3)
4. <https://angular.io/guide/router-tutorial> [↑](#footnote-ref-4)