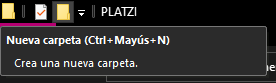
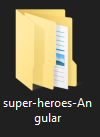
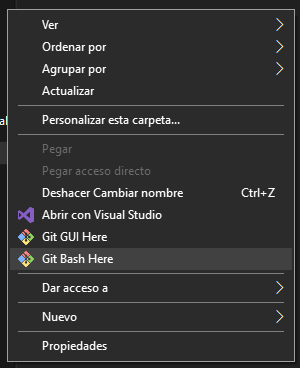
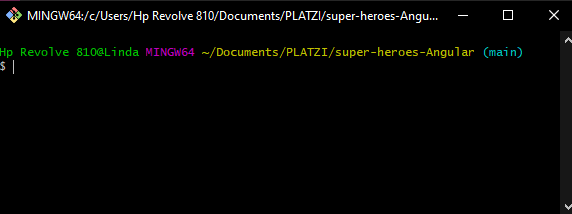
1. Creamos una nueva carpeta, escribimos el nombre de nuestra carpeta  
2. Damos doble clic para entrar a nuestra carpeta recién hecha, en seguida, adentro de nuestra carpeta damos clic derecho, y seleccionamos donde dice git bash here.



1. En seguida se abrirá una terminal, aquí vamos a configurar nuestro proyecto en angular y seguiremos una serie de pasos.

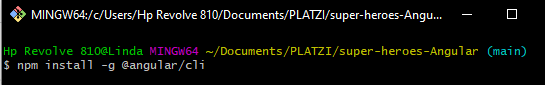


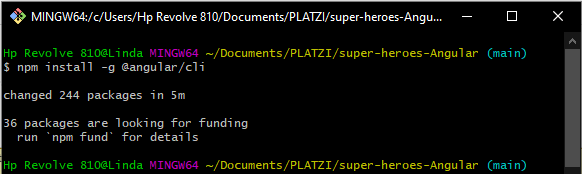
## **Instalar la CLI de Angular**

**npm package manager**

Angular, CLI de Angular, y las aplicaciones de Angular dependen de [paquetes npm](https://docs.npmjs.com/getting-started/what-is-npm) para muchas funcionalidades y funciones. Para descargar e instalar paquetes npm, necesitas un administrador de paquetes npm.

* 1. **Primer paso:** vamos a instalar la última versión del CLI Angular (¿*que es el CLI de Angular? es una herramienta de línea de comandos que se utiliza para inicializar, desarrollar, estructurar y mantener aplicaciones de Angular directamente desde la terminal.*) Instalar el CLI usando el gestor de paquetes npm: npm install -g @angular/cli



* 1. Lo escribimos en la terminal cómo se muestra en la imagen y damos enter, esperamos unos segundos la instalación y una vez instalado nuestro CLI nos debe aparecer como se muestra en la siguiente imagen.

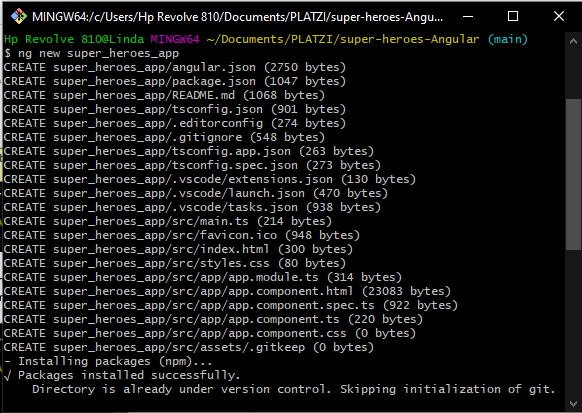
## **Crea un espacio de trabajo y una aplicación inicial**

* 1. Creamos un nuevo espacio de trabajo y una aplicación inicial: crear una nueva carpeta que será nuestra nueva área de trabajo en Angular, en donde se generará una nueva estructura de la aplicación (*tendrá múltiples aplicaciones y librerías*).

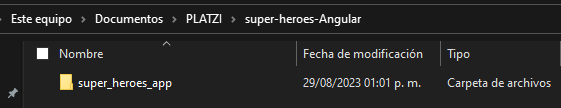
En la terminal escribiremos el siguiente comando: ng new super\_heroes\_app



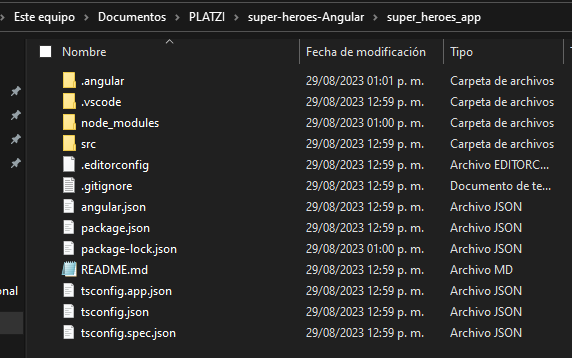
* 1. Damos enter y esperamos y esperamos unos segundos la instalación de los paquetes. Una vez instalados nos debe aparecer los siguiente cómo se muestra en la siguiente imagen imagen.



* + 1. Podemos visualizar que se creó una nueva carpeta



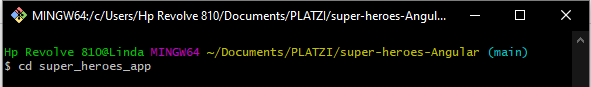
* + 1. Damos doble clic y podemos visualizar hay más carpetas y archivos que nos facilitarán para hacer nuestro proyecto



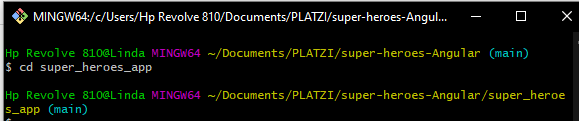
## **Ejecutar la aplicación**

* 1. Ahora entraremos a nuestra carpeta anterior mente hecha desde la terminal usando el comando cd y el nombre de nuestra carpeta y damos enter.

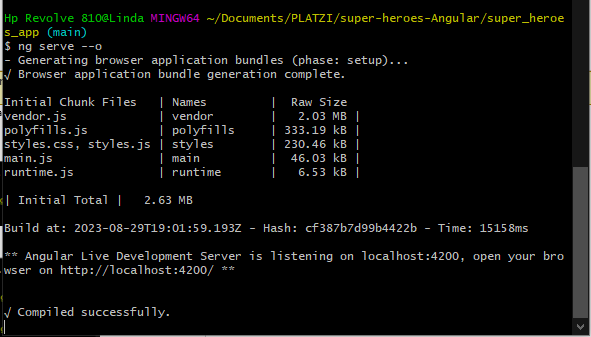
Escribimos el comando: cd super\_heroes\_app



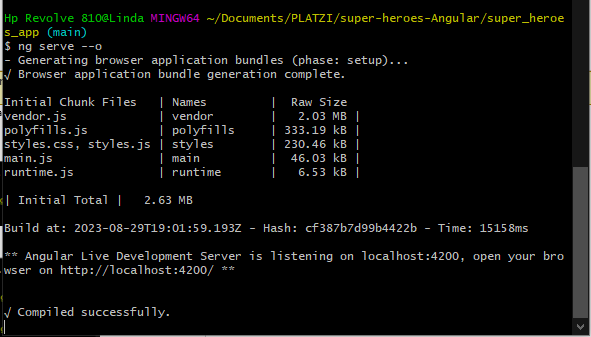
Y así es cómo se visualiza el cambio al entrar a la carpeta.



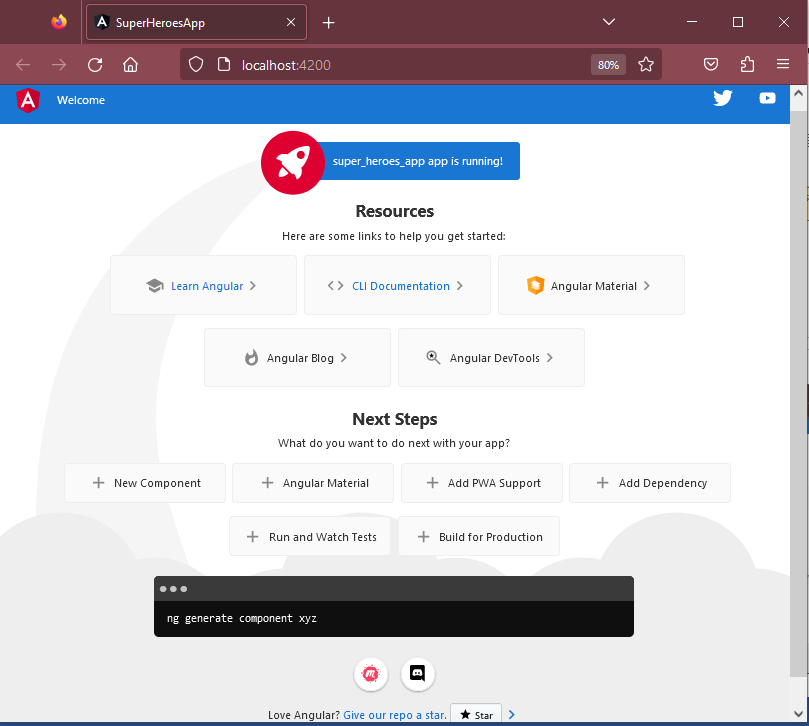
* 1. En la terminal escribiremos el comando para compilar nuestro proyecto en el navegador: ng serve –o



Y esperamos unos segunditos para que cargue.

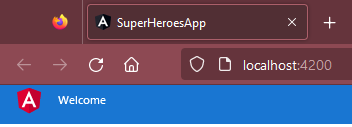


Una vez que cargue, se abrirá nuestro navegador y nos mostrará una pantalla predeterminada.



¡¡¡Felicidades!!! Has creado tu primer proyecto en Angular.

1. En tu navegador, abre <http://localhost:4200/> para ver como se esta ejecutando la nueva aplicación. Cuando usas el comando [ng serve](https://docs.angular.lat/cli/serve) para construir una aplicación y servirla localmente, el servidor automáticamente reconstruye la aplicación y recarga la página cuando tu cambias cualquiera de los archivos base.



Configuración del área de trabajo y proyecto.

Se crea un único archivo de configuración de área de trabajo, angular.json, en el nivel superior del área de trabajo. Aquí es donde estableces los valores predeterminados por proyecto para las opciones de los comandos CLI y especificar configuraciones que se usan cuando el CLI construye un proyecto para diferentes targets.

El comando [ng config](https://docs.angular.lat/cli/config) te permite establecer y recuperar valores de configuración desde la línea de comandos, o puedes editar directamente el archivo angular.json. Tenga en cuenta que los nombres opcionales en el archivo de configuración debe usar [camelCase](https://docs.angular.lat/guide/glossary" \l "case-types), mientras que los nombres opcionales que se suministra a los comandos pueden ser tanto camelCase o dash-case.

# **Aplicación y tutorial Super héroes**

PRIMEROS PASOS

En este tutorial, crearás tu propia aplicación desde cero, proporcionando experiencia con el proceso de desarrollo típico, así como una introducción a los conceptos básicos de diseño de aplicaciones, herramientas y terminología.

Este tutorial de *Super héroes* te muestra cómo configurar tu entorno de desarrollo local y desarrollar una aplicación utilizando la [Herramienta CLI de Angular](https://docs.angular.lat/cli)

* La aplicación Super héroes que construyes ayuda a una agencia de personal a administrar su grupo de héroes. La aplicación tiene muchas de las características que esperarías encontrar en cualquier aplicación basada en datos. La aplicación final adquiere y muestra una lista de héroes, edita los detalles de un héroe seleccionado y navega entre diferentes vistas de datos heroicos.

1. Empezamos con nuestro proyecto

Este tutorial de *Tour de héroes* te muestra cómo configurar tu entorno de desarrollo local y desarrollar una aplicación utilizando la [Herramienta CLI de Angular](https://docs.angular.lat/cli), y proporciona una introducción a los fundamentos de Angular.

La aplicación *Tour de héroes* que construyes ayuda a una agencia de personal a administrar su grupo de héroes. La aplicación tiene muchas de las características que esperarías encontrar en cualquier aplicación basada en datos. La aplicación final adquiere y muestra una lista de héroes, edita los detalles de un héroe seleccionado y navega entre diferentes vistas de datos heroicos.

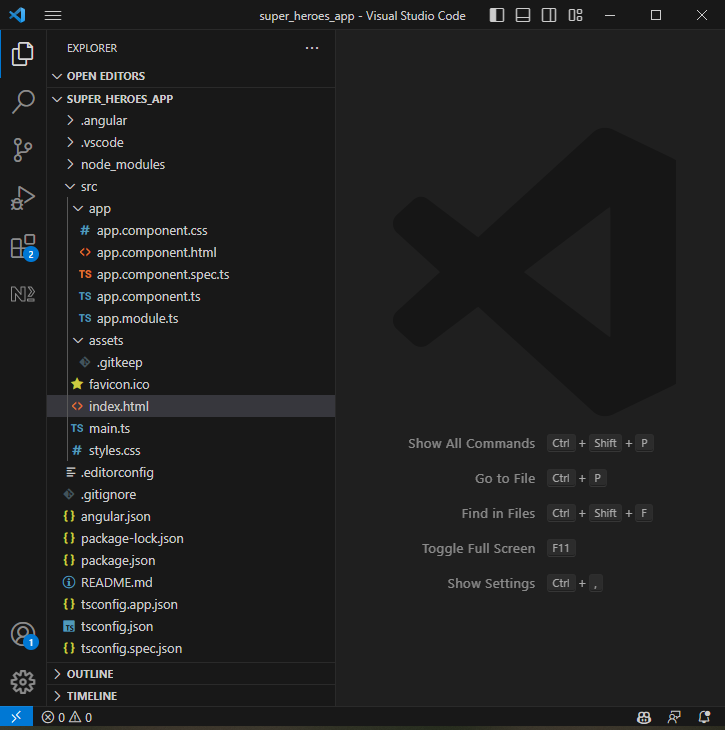
Encontrarás referencias y expansiones de este dominio de aplicación en muchos de los ejemplos utilizados en toda la documentación de Angular, pero no necesariamente necesitas trabajar en este tutorial para comprender esos ejemplos.

Al final de este tutorial, podrás hacer lo siguiente:

* Utilizar las [directivas](https://docs.angular.lat/guide/glossary#directive) Angular integradas para mostrar y ocultar elementos y mostrar listas de datos de héroes.
* Crear [componentes](https://docs.angular.lat/guide/glossary#component) Angular para mostrar los detalles del héroe y mostrar una lista de héroes.
* Usar el [enlace de datos](https://docs.angular.lat/guide/glossary#data-binding)(data binding) unidireccional para datos de solo lectura.
* Agregar campos editables para actualizar un modelo con enlace de datos bidireccional.
* Enlazar métodos de componentes a eventos de usuario, como pulsaciones de teclas y clics.
* Permitir a los usuarios seleccionar un héroe de una lista maestra y editar ese héroe en la vista de detalles.
* Dar formato a datos con [pipes](https://docs.angular.lat/guide/glossary#pipe).
* Crear un [servicio](https://docs.angular.lat/guide/glossary#service) compartido para reunir a los héroes.
* Utilizar [enrutamiento](https://docs.angular.lat/guide/glossary#router) para navegar entre diferentes vistas y sus componentes.

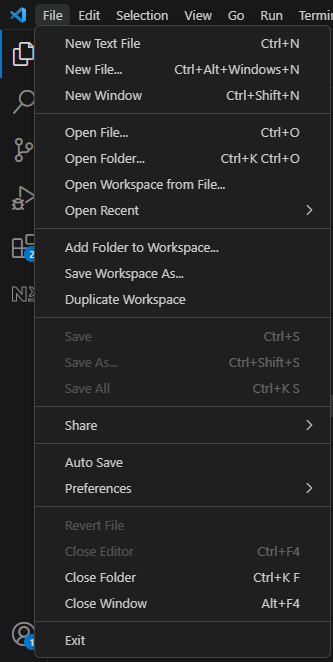
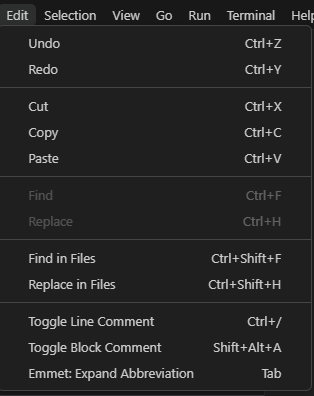
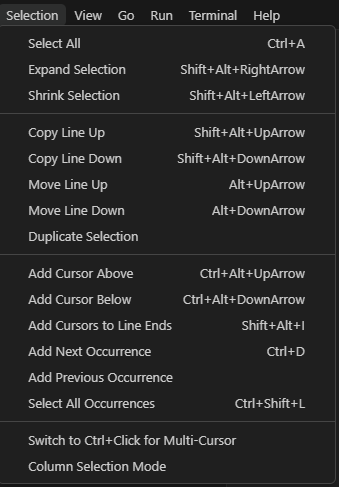
Aprenderás suficiente Angular para comenzar y ganarás la confianza de que Angular puede hacer lo que tú necesites que haga.

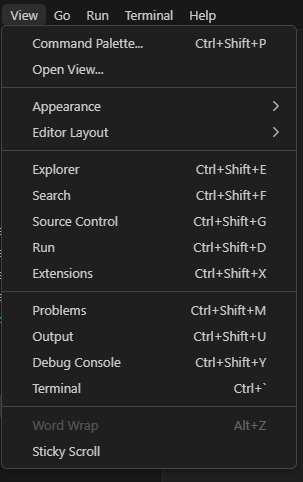
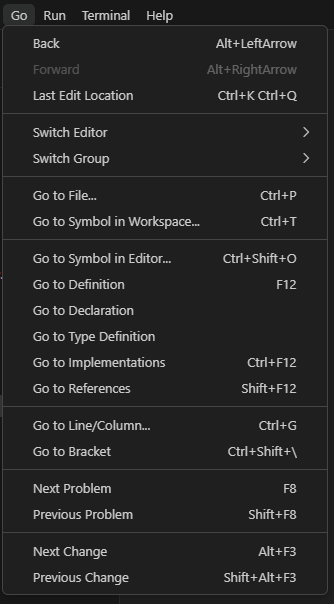
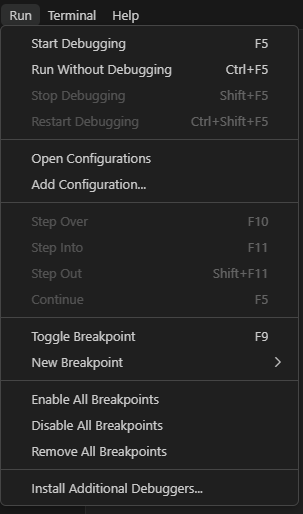
Abrimos nuestro proyecto en Visual Estudio 

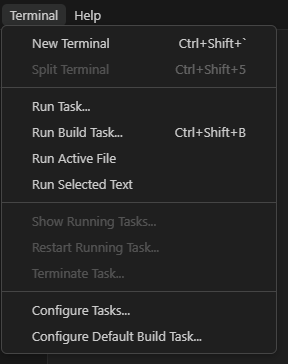
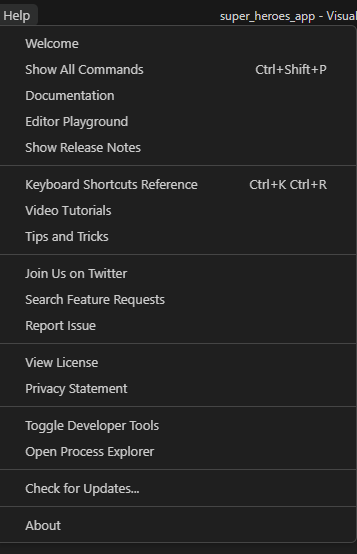


1. Cuando se abre nuestro archivo podemos ver un menú en la parte de arriba

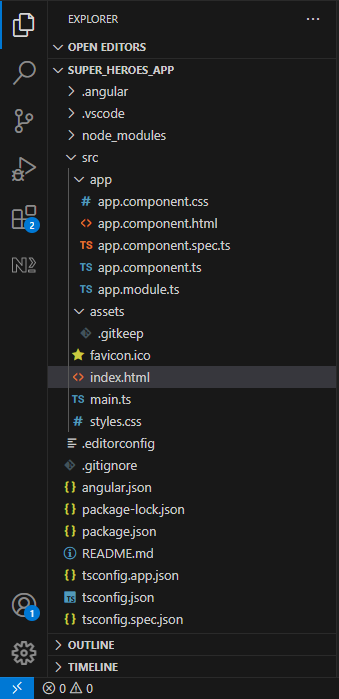


1. Al igual que del lado derecho podemos ver varias carpetas, esas carpetas se crearon al momento de crear nuestro archivo de angular, también se creó una carpeta con varias carpetas en su interior.



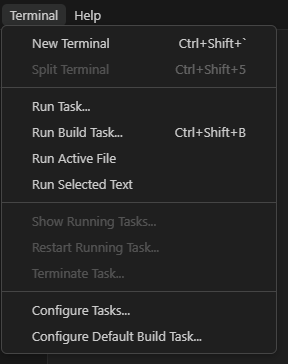
Crear un componente de héroes

Usa la CLI angular para generar un nuevo componente llamado heroes.

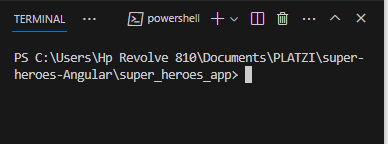
content\_copyng generate component heroes

CLI crea una nueva carpeta llamada src/app/heroes/, y genera tres archivos sobre HeroesComponent junto un archivo de prueba.

1. Ahora vamos a seleccionar la pestaña de **Terminal**, para crear nuevos componentes (carpetas con nuevos archivos) que vamos a utilizar en nuestro proyecto, y seleccionamos **New Terminal**.



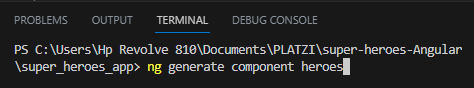
* 1. Y nos va a salir una terminal en nuestro programa.



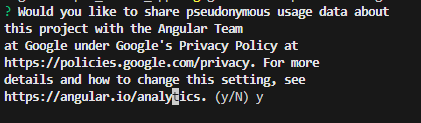
* + 1. En esta terminal vamos a escribir el siguiente comando seguido del nombre de la nueva carpeta.

ng generate component heroes

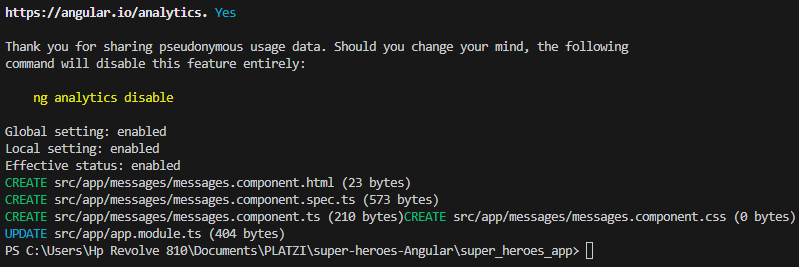
donde *ng generate component* es el comando para crear nuestro componente y *heroes* es el nombre de nuestro componente, y damos enter.

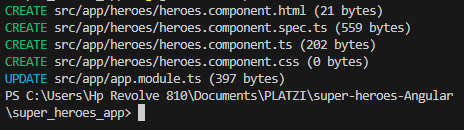


* + 1. Después nos sale un mensaje donde pondremos yes(y) o no(n), y nosotros ponemos Y, damos enter.



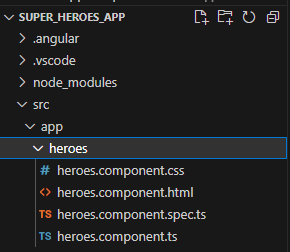
* + 1. Y se crea nuestro componente, así cómo se muestra en la imagen.



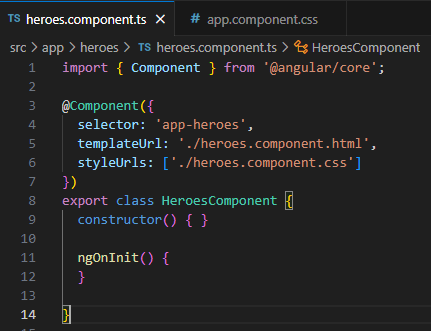


También se puede visualizar en las carpetas que se muestran del lado izquierdo de nuestro archivo.

Podemos ver que se crearon los siguientes archivos css, html, spect.ts y ts



1. El archivo de la clase HeroesComponent es el siguiente.



Siempre Importa el símbolo [Component](https://docs.angular.lat/api/core/Component) de la biblioteca pricipal de Angular, y realiza la anotación a la clase del component con @[Component](https://docs.angular.lat/api/core/Component).

@[Component](https://docs.angular.lat/api/core/Component) es una decoradoro que especifica metadatos Angular para un componente.

La CLI generó 3 propiedades de metadatos:

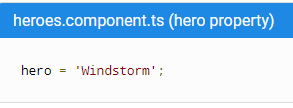
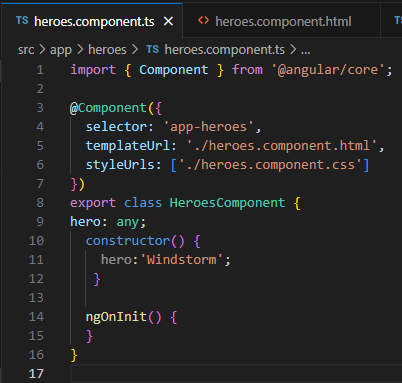
1. selector— El selector de elementos CSS para el componente
2. templateUrl— La ubicación del archivo plantilla para el componente
3. styleUrls— La ubicación de los estilos CSS privados del componente.

El [Selector de elementos CSS](https://developer.mozilla.org/en-US/docs/Web/CSS/Type_selectors) 'app-heroes', coincide con el nombre del elemento HTML que identifica este componente en el componente padre Plantillas.

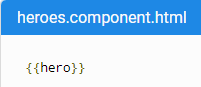
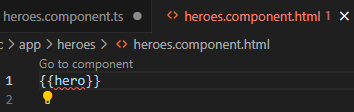
El ngOnInit() es un [gancho de ciclo de vida](https://docs.angular.lat/guide/lifecycle-hooks#oninit) ("lifecycle hook") . Angular llama a ngOnInit() inmediatamente después de crear el componente. Adecuado para poner la lógica de inicialización.

Siempre exporta la clase de componente, por lo que siempre puede importarla en otro lugar, como un AppModule.

* 1. Agrega la propiedad hero
     1. Agrega una propiedad hero al HeroesComponent para un héroe llamado "Windstorm".

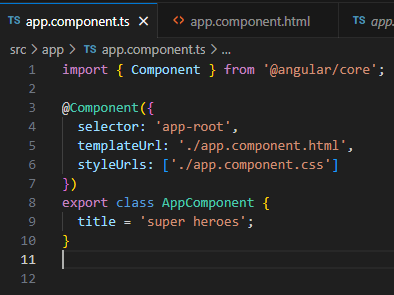
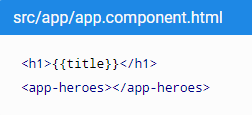
 

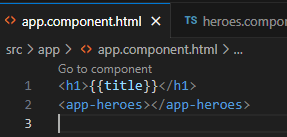
* 1. Mostrar el héroe
     1. Abre el archivo de plantilla heroes.component.html. Elimina el texto predeterminado generado por CLI angular, Reemplaza con un enlace de datos a la nueva propiedad hero.

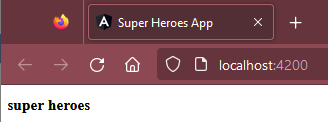
* 1. Mostrar la vista HeroesComponent
     1. Para ver el HeroesComponent, debe agregarlo a las Plantillas en el AppComponent del shell de tu aplicación.

Recuerda que app-heroes es el selector de elemento del HeroesComponent. Entonces, en el archivo Plantillas de AppComponent, agrega el elemento <app-heroes> directamente debajo del título.

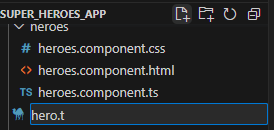
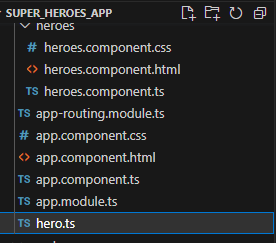


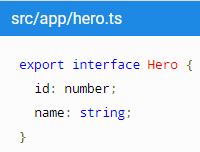
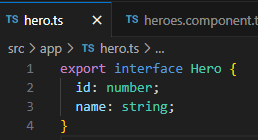
* + 1. Si el comando CLI ng serve todavía se está ejecutando, El navegador se actualiza para mostrar el título de la aplicación y el nombre del héroe.



1. Crear interfaz de héroe
   1. Un héroe es más que un nombre.
      1. Creamos el nuevo archivo en la carpeta app llamado hero.ts.

Crea una interfaz Hero en su propio archivo en la carpeta src/app. Dale una propiedad id y una propiedad name.

* 1. Regresa a la clase HeroesComponent e importe la interfaz Hero.

Refactoriza la propiedad de héroe del componente para que sea del tipo 'Héroe'. Inicialízalo con un id de 1 y un nombre de Windstorm.

El archivo de clase revisado HeroesComponent se ve así:

Agregamos en el export class HeroesComponente

 hero: Hero = {

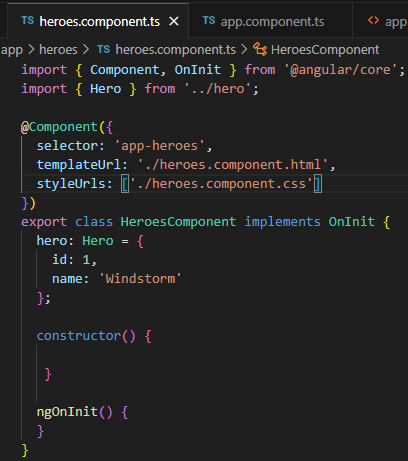
    id: 1,

    name: 'Windstorm'

  };

Después agregamos hasta arriba en import

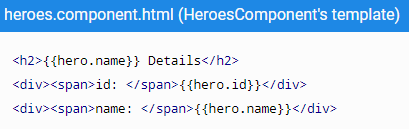
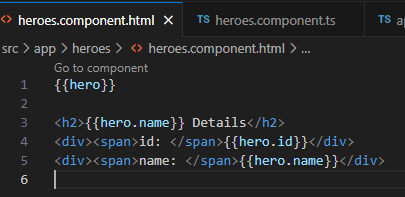
import { Hero } from '../hero';

Cambió el héroe de texto a un objeto, lo que provocó que la página se mostrara incorrectamente.

* 1. Mostrar objeto de héroe

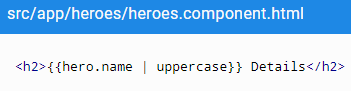
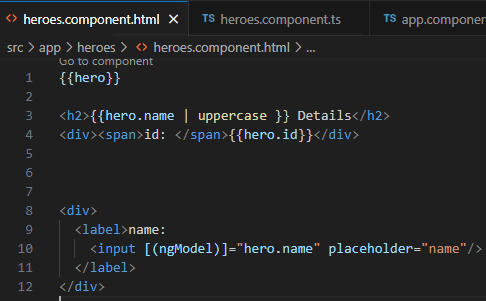
Actualiza los enlaces de Plantillas para anunciar el nombre del héroe, Muestra tanto el id como el name con un diseño detallado como este:

El navegador se actualiza para mostrar la información del héroe.

## **Formatea con UppercasePipe**

* 1. Modifica el enlace para hero.name de esta manera:

El navegador se actualizará para mostrar el nombre del héroe en mayúsculas.

En el enlace de interpolación, la palabra mayúscula inmediatamente después del operador pipe (|) es Inicie el 'UppercasePipe' incorporado.

[pipe](https://docs.angular.lat/guide/pipes) ("pipe") Es adecuado para formatear cadenas, importes monetarios, fechas y otros datos de visualización. Angular viene con múltiples pipes incorporadas, y puede crear las suyas propias.

## **Editar el héroe**

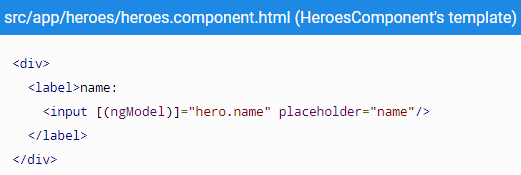
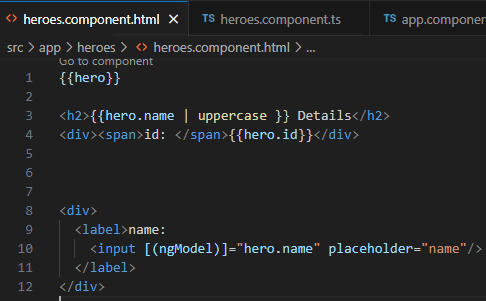
* 1. El usuario debe poder editar el nombre del héroe en el cuadro de texto <input>.

En el cuadro de texto, la propiedad name del héroe se muestra , La propiedad se actualiza según los tipos de usuario. Esto es de la clase de componente a \_screen, Y significa el flujo de datos desde la pantalla a la clase de componente.

Para automatizar ese flujo de datos, configure un enlace de datos bidireccional entre el elemento de formulario <input> y la propiedad hero.name.

### Enlace de datos bidireccional

Refactorizando el área de detalle de las Plantas HeroesComponent se ve así:

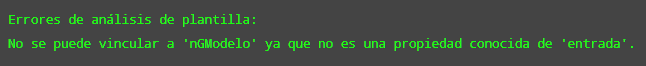
[(ngModel)] Es la sintaxis de enlace de datos bidireccional de Angular.

Esto vinculará la propiedad hero.name al cuadro de texto HTML, por lo que Puede pasar datos en ambas direcciones desde la propiedad hero.name al cuadro de texto y desde el cuadro de texto a la propiedad hero.name.

### FormsModule No encontrado

Observa que la aplicación dejó de funcionar cuando agregué el [([ngModel](https://docs.angular.lat/api/forms/NgModel))].

Para ver el error, abre las herramientas de desarrollo de su navegador, Busca mensajes como el siguiente en la consola,



[ngModel](https://docs.angular.lat/api/forms/NgModel) Es una directiva angular válida pero no está disponible por defecto.

Pertenece al [FormsModule](https://docs.angular.lat/api/forms/FormsModule) opcional y debe optar por ese módulo para usarlo.

*AppModule*

En Angular, cómo encajan las partes de la aplicación, Necesita saber qué otros archivos y bibliotecas necesita su aplicación. Esta información se llama *metadata*.

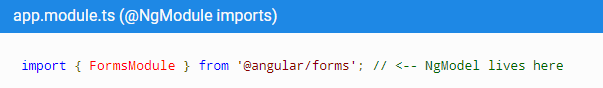
Algunos de los metadatos se encuentran en el decorador @[Component](https://docs.angular.lat/api/core/Component) que agregó a su clase de componentes. Otros metadatos importantes son[@NgModule](https://docs.angular.lat/guide/ngmodules)Está en el decorador.

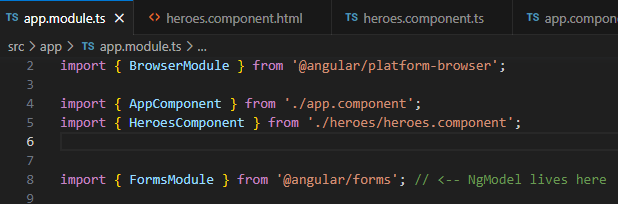
El decorador más importante @[NgModule](https://docs.angular.lat/api/core/NgModule) anota la clase AppModule de nivel superior.

Angular CLI creó la clase AppModule en src/app/app.module.ts al crear el proyecto. Ahora opta por el [FormsModule](https://docs.angular.lat/api/forms/FormsModule).

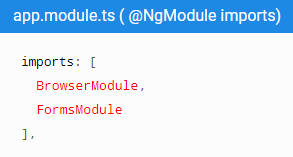
### Importar FormsModule

Abre AppModule (app.module.ts) e importe el símbolo [FormsModule](https://docs.angular.lat/api/forms/FormsModule) desde la biblioteca @angular/forms.





A continuación, agregue el [FormsModule](https://docs.angular.lat/api/forms/FormsModule) a el arreglo imports de los metadatos @ [NgModule](https://docs.angular.lat/api/core/NgModule). Esta matriz contiene una lista de módulos externos que requiere su aplicación.

La aplicación debería funcionar nuevamente cuando se actualice el navegador. Puedes editar el nombre del héroe y ver los cambios reflejados inmediatamente en el <h2> arriba del cuadro de texto.

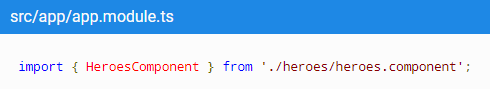
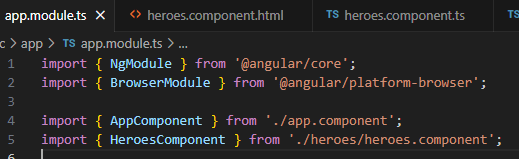
### Declarar HeroesComponent

Todos los componentes deben declararse con exactamente uno [NgModule](https://docs.angular.lat/guide/ngmodules).

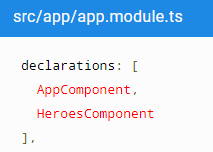
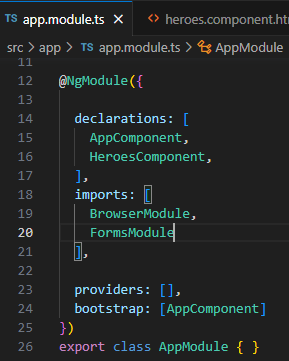
No has declarado HeroesComponent`. Entonces, ¿por qué funcionó la aplicación?

La aplicación funcionó porque Angular CLI declaró el componente en el AppModule cuando generó el HeroesComponent.

Abra src/app/app.module.ts y encuentre el HeroesComponent importado cerca de la parte superior.

HeroesComponent se declara en la matriz@[NgModule.declarations](https://docs.angular.lat/api/core/NgModule#declarations).

AppModule declara los componentes de aplicación AppComponent y HeroesComponent.

## **Resumen**

* Creo un segundo HeroesComponent usando el CLI.
* Agregó HeroesComponent al shell de AppComponent y lo mostró.
* Aplico 'UppercasePipe' para formatear el nombre.
* Utilizo el enlace de datos bidireccional en la directiva [ngModel](https://docs.angular.lat/api/forms/NgModel).
* Aprendío sobre AppModule.
* Importó [FormsModule](https://docs.angular.lat/api/forms/FormsModule) en AppModule para reconocer y aplicar la directiva Angular [ngModel](https://docs.angular.lat/api/forms/NgModel).
* Aprendío la importancia de declarar un componente en un AppModule y me di cuenta de que la CLI está haciendo esa declaración por usted.

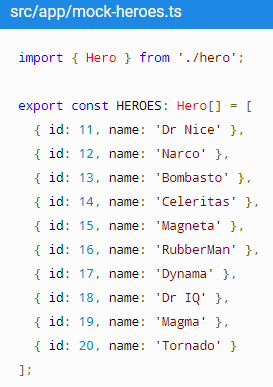
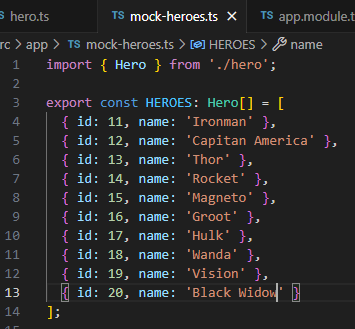
# **Mostrar una lista de selección**

En esta página, *ampliaremos la aplicación* Tour de Héroes para mostrar una lista de héroes, Permite al usuario seleccionar un héroe y ver los detalles del héroe.

## **Crea un simulacro de héroe**

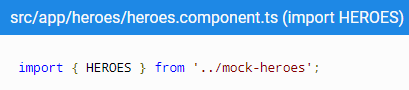
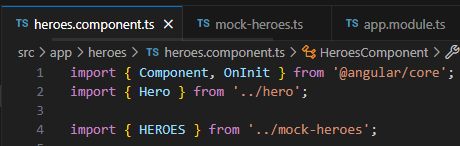
Primero, necesitarás algunos héroes para mostrar. Eventualmente los obtendrá de un servidor de datos remoto. Por ahora, creará algunos héroes simulados y pretenderá que provienen del servidor.

Crea un archivo llamado mock-heroes.ts en la carpeta src/app/. Define la constante HEROES como un conjunto de 10 héroes y expórtala. El archivo se verá así:

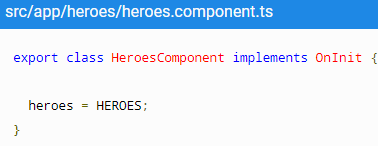
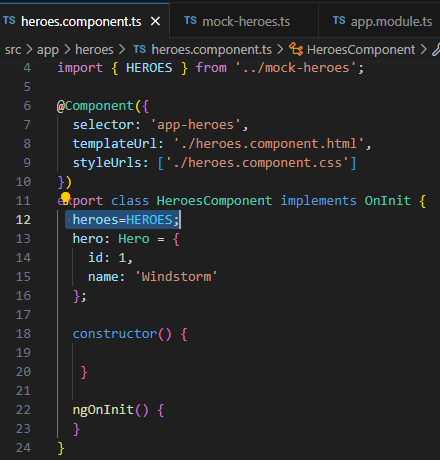
 

## **Mostrar Héroes**

Abre el archivo de clase HeroesComponent e importe el mock HEROES.

En el mismo archivo (clase HeroesComponent), define una propiedad de componente llamada heroes para exponer el array HEROES para la vinculación.

* 1. Enumerar héroes con \*[ngFor](https://docs.angular.lat/api/common/NgForOf)

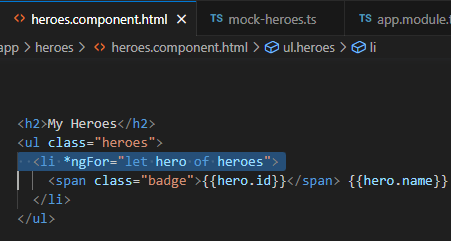
Abre las Plantillas HeroesComponent y realiza los siguientes cambios:

* Agrega <h2> al principio
* Agrega una lista HTML desordenada (<ul>) debajo de ella
* Inserta <li> dentro del <ul> que muestra la propiedad hero
* Espolvoreé algunas clases CSS al estilo (agregaremos estilos CSS en breve)

Se parece a esto:

Esto muestra un héroe. Para enumerarlos a todos, agrega \*[ngFor](https://docs.angular.lat/api/common/NgForOf)\* a <li> para iterar sobre la lista de héroes.

[\*ngFor](https://docs.angular.lat/guide/built-in-directives#ngFor) es la directiva de *repetición* de Angular.

Esto repite el elemento host para cada elemento de la lista.

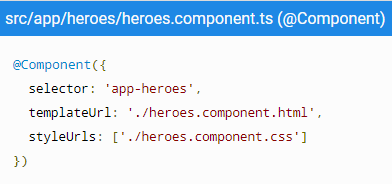
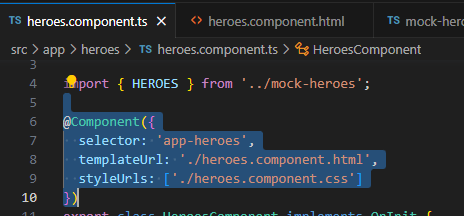
La sintaxis para este ejemplo es:

* <li> es un elemento host
* heroes es una lista de clases HeroesComponent que contiene la lista de héroes simulados
* hero contiene el objeto héroe actual en una lista para cada ciclo

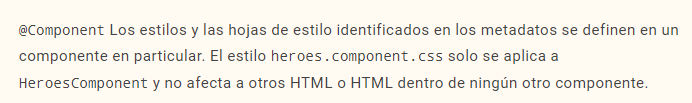
Actualiza tu navegador para ver la lista de héroes.

### Agregar estilo a los héroes

* La lista de héroes debe ser atractiva y visualmente prominente cuando el usuario coloca el cursor y selecciona un héroe de la lista.
* En el [Primer tutorial](https://docs.angular.lat/tutorial/toh-pt0#app-wide-styles), configuro el estilo básico de toda la aplicación en styles.css.
* No incluí el estilo de la lista de héroes en esta hoja de estilo.
* Puedes agregar más estilos a styles.css y seguir expandiendo esa hoja de estilo a medida que agrega componentes
* Es posible que prefieras definir un estilo privado para un componente en particular y mantener todo lo que el componente necesita — código, HTML, CSS— en un solo lugar.
* Este enfoque facilita la reutilización del componente en otro lugar y aún así proporciona al componente la apariencia deseada, incluso cuando los estilos aplicados globalmente son diferentes.
* Los estilos privados se definen en línea dentro de la matriz @[Component.styles](https://docs.angular.lat/api/core/Component#styles) o como un archivo de hoja de estilo identificado en una matriz particular @[Component.styleUrls](https://docs.angular.lat/api/core/Component#styleUrls) como un archivo de hoja de estilo.
* Cuando la CLI crea un HeroesComponent, se crea un heroes.component.css vacío para el HeroesComponent. @[Component.styleUrls](https://docs.angular.lat/api/core/Component#styleUrls) se señala de esta manera.

Abre heroes.component.css y pega el estilo privado para HeroesComponent. Puedes encontrarlos en la [Revisión del código final](https://docs.angular.lat/tutorial/toh-pt2#final-code-review) al final de esta guía.



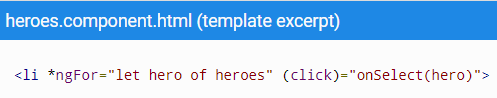
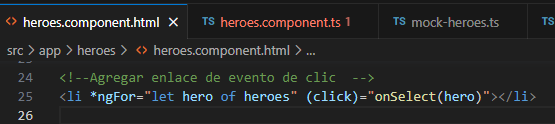
## **Maestro/Detalle**

Cuando haces clic en un héroe en la lista **maestro**, el componente debe mostrar los **detalles** del héroe seleccionado en la parte inferior de la página.

En este capítulo, esperemos a que se haga clic en el elemento del héroe y luego actualiza los detalles del héroe.

### Agregar enlace de evento de clic

Agrega el enlace de evento click a su <li> así:

Este es un ejemplo de la sintaxis de Angular [enlace de eventos](https://docs.angular.lat/guide/event-binding) .

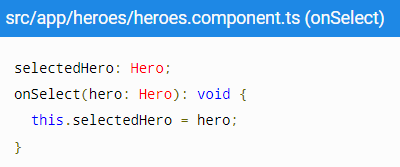
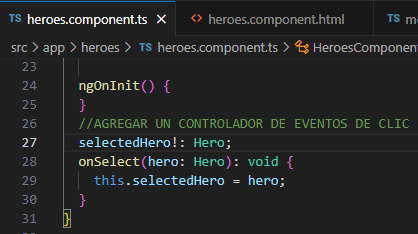
Los paréntesis alrededor del clic le dicen a Angular que es un evento clic para el elemento <li>. Cuando el usuario hace clic en <li>, Angular ejecuta la expresión onSelect(hero).

En la siguiente sección, definiremos el método onSelect() en el HeroesComponent para mostrar los héroes definidos por las expresiones \*[ngFor](https://docs.angular.lat/api/common/NgForOf).

### Agregar un controlador de eventos de clic

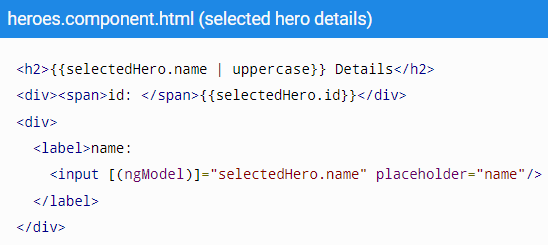
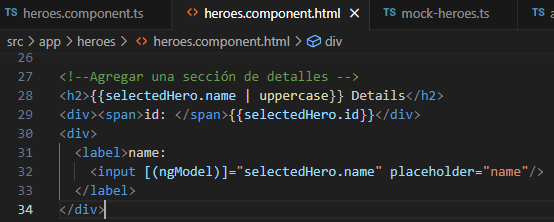
Cambia el nombre de la propiedad hero del componente a selectedHero, pero no la asignes todavía. No hay héroe seleccionado cuando se inicia la aplicación.

Agrega el método onSelect() de la siguiente manera y asigne el héroe en el que se hizo clic desde Plantillas al componente 'seleccionadoHero`.

### Agregar una sección de detalles

Actualmente, el componente Plantillas tiene una lista. Para hacer clic en un héroe en la lista para ver los detalles de ese héroe, necesita una sección de detalles para representarlo en Plantillas. Agrega lo siguiente debajo de la sección de la lista de heroes.component.html.

Cuando actualizo el navegador, la aplicación está rota.

Abre y busca las herramientas de desarrollador de su navegador y busca un mensaje de error como este en la consola:



### ¿Que pasó?

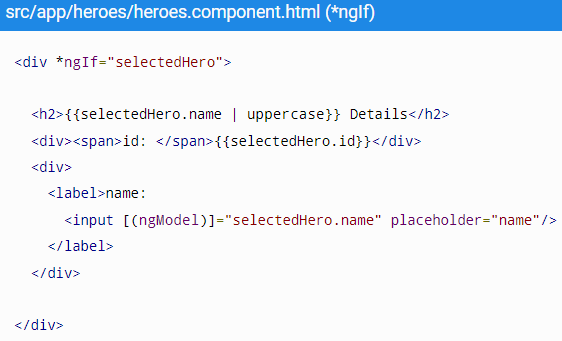
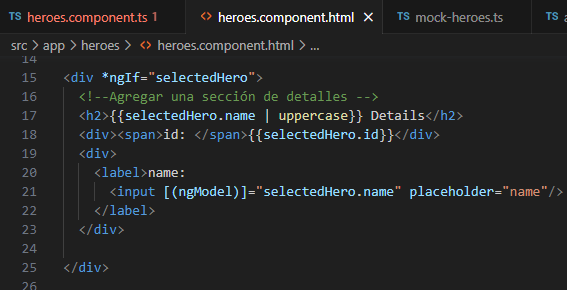
Cuando inicia la aplicación, selectedHero es intencionalmente indefinido`.

Los enlaces de expresión en Plantillas que se refieren a las propiedades de selectedHero - expresiones como{{{selectedHero.name}} deben fallar porque el héroe seleccionado no existe, no.

### Reparemos-use \*ngIf para ocultar detalles vacíos

El componente solo debe mostrar detalles para el héroe seleccionado si selectedHero está presente.

Adjunta los detalles del héroe en HTML <div>. Agrega la directiva angular \*[ngIf](https://docs.angular.lat/api/common/NgIf) a su <div> y configúrelo en selectedHero.

Actualiza su navegador y verá la lista de nombres nuevamente. El área de detalles está en blanco. Hace clic en un héroe en la lista de héroes para ver más detalles. La aplicación comenzó a funcionar nuevamente. Los héroes se muestran en la lista, y los detalles del héroe seleccionado se muestran en la parte inferior de la página.

### Por qué esto funciona

Cuando selectedHero no está definido, [ngIf](https://docs.angular.lat/api/common/NgIf) elimina los detalles del héroe del DOM. No hay obligación de preocuparse por selectedHero.

Cuando el usuario selecciona un héroe, selectedHero tiene un valor y [ngIf](https://docs.angular.lat/api/common/NgIf) inserta los detalles del héroe en el DOM.

### Dar estilo a el héroe seleccionado

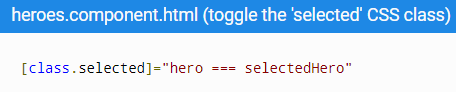
Si todos los elementos <li> se parecen, es difícil identificar al héroe seleccionado en la lista.

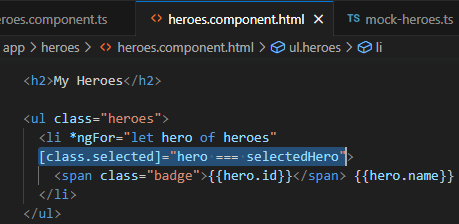
Si el usuario hace clic en "Magneta", el héroe debe dibujarse con un color de fondo prominente como este:

El color del héroe seleccionado es el trabajo de la clase CSS .selected en [el estilo que acaba de agregar](https://docs.angular.lat/tutorial/toh-pt2#styles). Simplemente aplica la clase .selected a <li> cuando el usuario hace clic.

El [enlace de clase](https://docs.angular.lat/guide/attribute-binding#class-binding)de Angular facilita la adición y eliminación de clases CSS condicionales. Simplemente agregue [class.some-css-class] =" some-condition" al elemento que desea decorar.

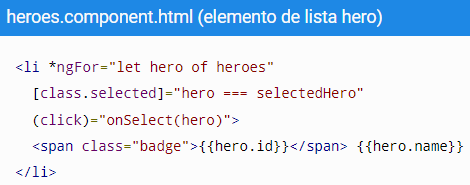
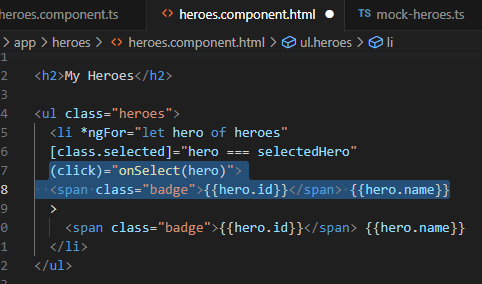
Agrega el enlace [class.selected] al <li> en HeroesComponent Plantillas:



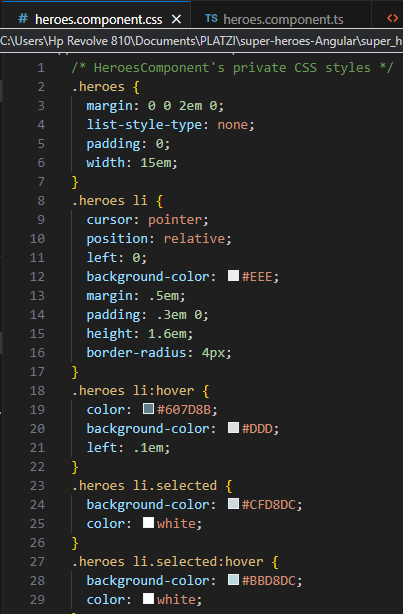
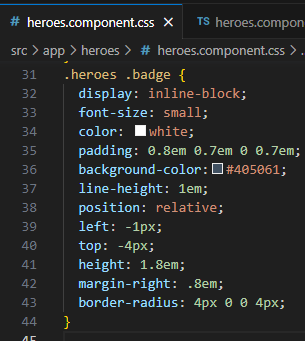


Si el héroe de la fila actual es el mismo que de selectedHero, Angular agrega una clase CSS selected.Cuando los dos héroes son diferentes, Angular eliminará la clase.

El <li> completado se ve así:

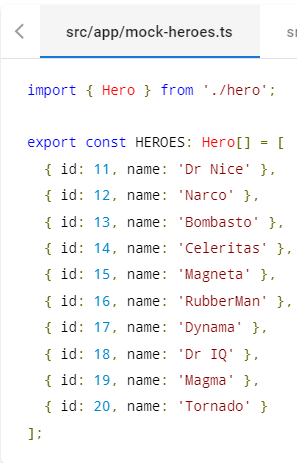
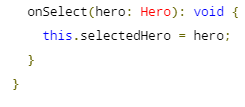
 

* 1. Héroes.component.css

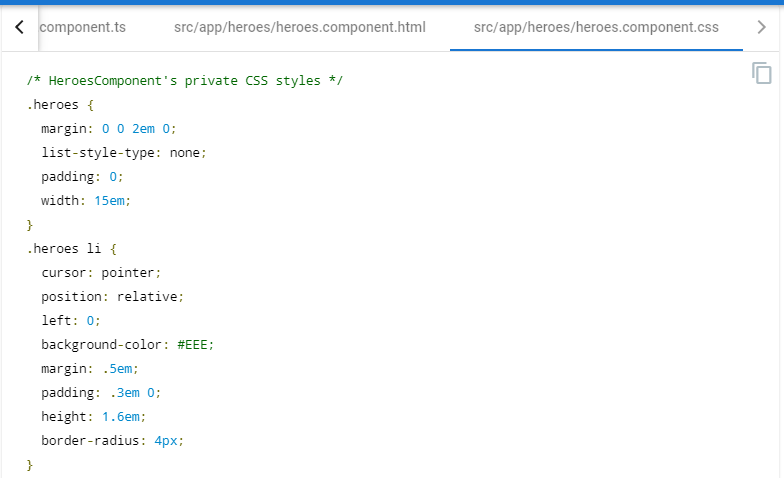
## **Revisión final del código**

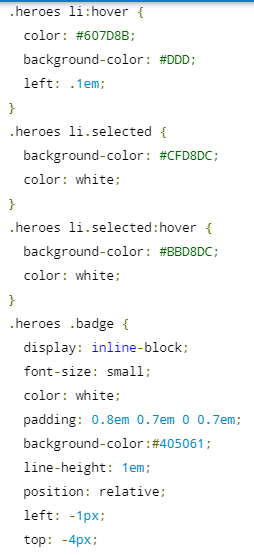
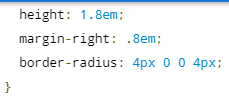
Aquí está el archivo de código en esta página que incluye el estilo HeroesComponent.



HEROES.COMPONENT.CSS



## **Resumen**

* La aplicación "Tour de Héroes" muestra una lista de héroes en la pantalla Maestro / Detalle
* El usuario puede seleccionar un héroe y ver los detalles de ese héroe
* Utilizó \*[ngFor](https://docs.angular.lat/api/common/NgForOf) para mostrar la lista
* Utilizó \*[ngIf](https://docs.angular.lat/api/common/NgIf) para incluir o excluir condicionalmente bloques de HTML
* La clase de estilo CSS se puede cambiar con el enlace class

# **Crear un componente de características**

Por el momento, el HeroesComponent muestra tanto la lista de héroes como los detalles de los héroes seleccionados.

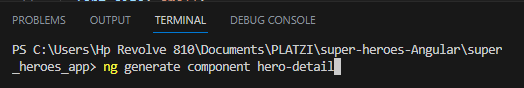
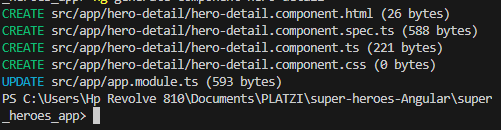
Mantener toda la funcionalidad en un componente se vuelve menos sostenible a medida que la aplicación crece. Deberá dividir un componente grande en subcomponentes más pequeños que se centren en una tarea o flujo de trabajo en particular.

Esta página da el primer paso en ese camino moviendo los detalles del héroe a otro HeroDetailComponent reutilizable.

HeroesComponent solo muestra una lista de héroes. HeroDetailComponent muestra los detalles del héroe seleccionado.

## **Crear HeroDetailComponent**

Usa el CLI de Angular para generar un nuevo componente llamado hero-detail.

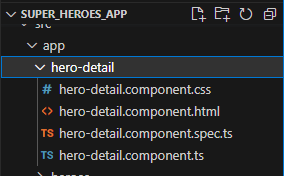
Este comando generará una plantilla para el archivo HeroDetailComponent y declarará este componente en el AppModule. Este comando produce la siguiente plantilla:

* Crear el directorio src/app/hero-detail

Genera cuatro archivos en este directorio:

* Archivo CSS para estilo de componente
* Archivo HTML para la plantilla el componente
* Archivo TypeScript de la clase de componente denominada HeroDetailComponent
* Archivo de prueba de la clase HeroDetailComponent

Este comando también agrega HeroDetailComponent como declaraciones en el decorador @ [NgModule](https://docs.angular.lat/api/core/NgModule) del archivo src/app/app.module.ts.

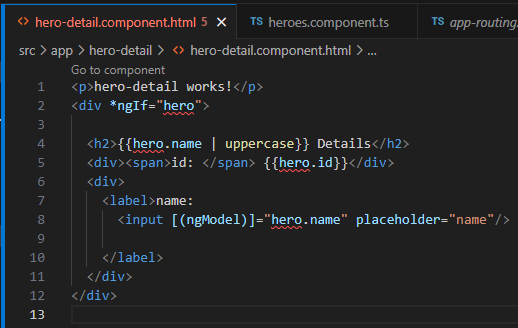


### Escribir plantilla

Corta el HTML de detalles del héroe desde la parte inferior de la plantilla HeroesComponent y pégalo en la plantilla generada en la plantilla HeroDetailComponent.

Las referencias HTML pegadas selectedHero. El nuevo HeroDetailComponent puede mostrar \_cualquier héroe, no solo el héroe seleccionado. Por lo tanto, reemplaza todos los "selectedHero" en la plantilla con "hero".

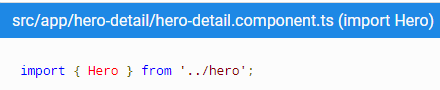
Cuando termines, las plantilla HeroDetailComponent deberían verse así:

### 14.1. Añadir la propiedad @[Input](https://docs.angular.lat/api/core/Input)() al héroe

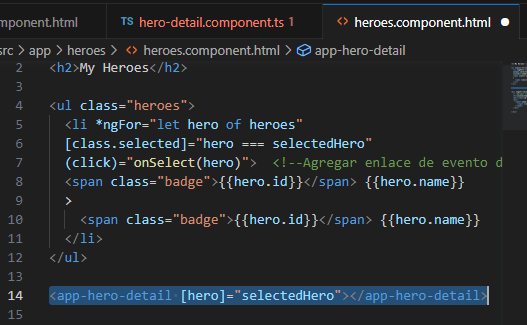
Las plantillas HeroDetailComponent están vinculadas a la propiedad hero de un componente que es del tipo Hero.

Abre el archivo de clase HeroDetailComponent e importe el símbolo Hero.

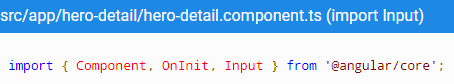
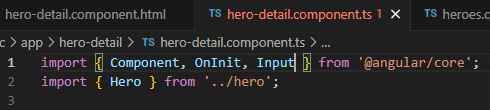


La propiedad hero debe ser una [propiedad de entrada](https://docs.angular.lat/guide/inputs-outputs) -->, anotada con el decorador @[Input](https://docs.angular.lat/api/core/Input)() porque el HeroesComponent externo [se vinculará de esta manera.](https://docs.angular.lat/tutorial/toh-pt3#heroes-component-template)

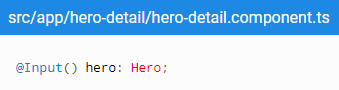


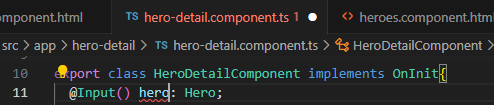


Modifique la declaración de importación @angular/core para incluir el símbolo [Input](https://docs.angular.lat/api/core/Input).

Agrega la propiedad hero antepuesta por el decorador @[Input](https://docs.angular.lat/api/core/Input)().





Este es el único cambio que debe realizar en la clase HeroDetailComponent. No se requieren más propiedades o lógica de visualización. Este componente solo toma un objeto héroe a través de la propiedad hero y lo muestra.

## **Mostrar HeroDetailComponent**

El HeroesComponent todavía está en la vista maestra/detalle.

Hasta que eliminé los detalles del héroe de Plantillas, lo estaba mostrando en este componente. Ahora deleguemos a HeroDetailComponent.

Los dos componentes tienen una relación padre-hijo. Para mostrar un nuevo héroe cada vez que el usuario selecciona un héroe de la lista, El padre HeroesComponent controla al hijo HeroDetailComponent enviándolo.

No cambiarás la clase de HeroesComponent pero cambiarás su template.

### 14.2.1 Actualizar las plantillas HeroesComponent

El selector para HeroDetailComponent es 'app-hero-detail'.

Agrega un elemento <app-hero-detail> a la parte inferior de las plantillas HeroesComponent donde la vista detallada de héroe existió una vez.