# Resultado de imagen de simbolo de angular 8 navegar por las páginas de mi aplicación Angular



GH [Gustavo Hernán Dohara](https://gustavodohara.com/blogangular/)

**Navegar por las páginas de nuestro sitio web es una funcionalidad básica en cualquier aplicación web, pero con Angular se puede hacer mucho mucho más que sólo navegar.**

**Podemos**

* **definir múltiples Routers,**
* **pasar datos a nuestros Routers,**
* **recargar dato para nuestras Vistas,**
* **agrupar nuestros Routers,**
* **agregar validaciones de acceso a nuestros Routers,**
* **agregarle estilo y animaciones, e incluso**
* **mejorar la performance sincronizando la carga de nuestros Routers.**

¿Qué es Routing****?****

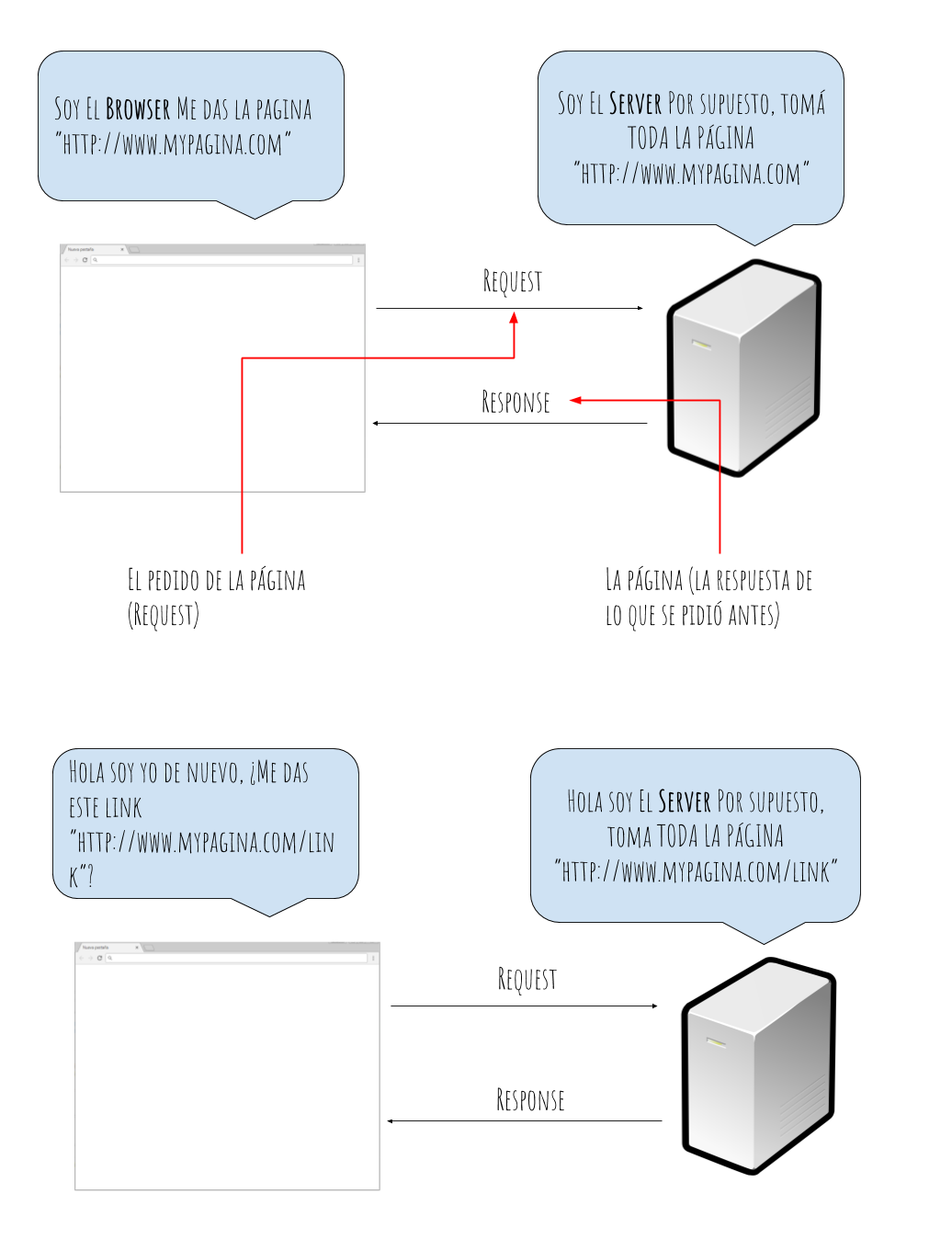
Es el mecanismo que permite a nuestra aplicación navegar entre diferentes ****Componentes****, enviándole parámetros (o sea datos) si los necesita. Y ****Router,**** es el encargado de hacerlo.

## Un poco de historia…

Al inicio de los tiempo no existía ****Angular**** ni ninguna tecnología parecida, los ****browsers**** eran mucho mas básicos y las cosas eran mas simples.

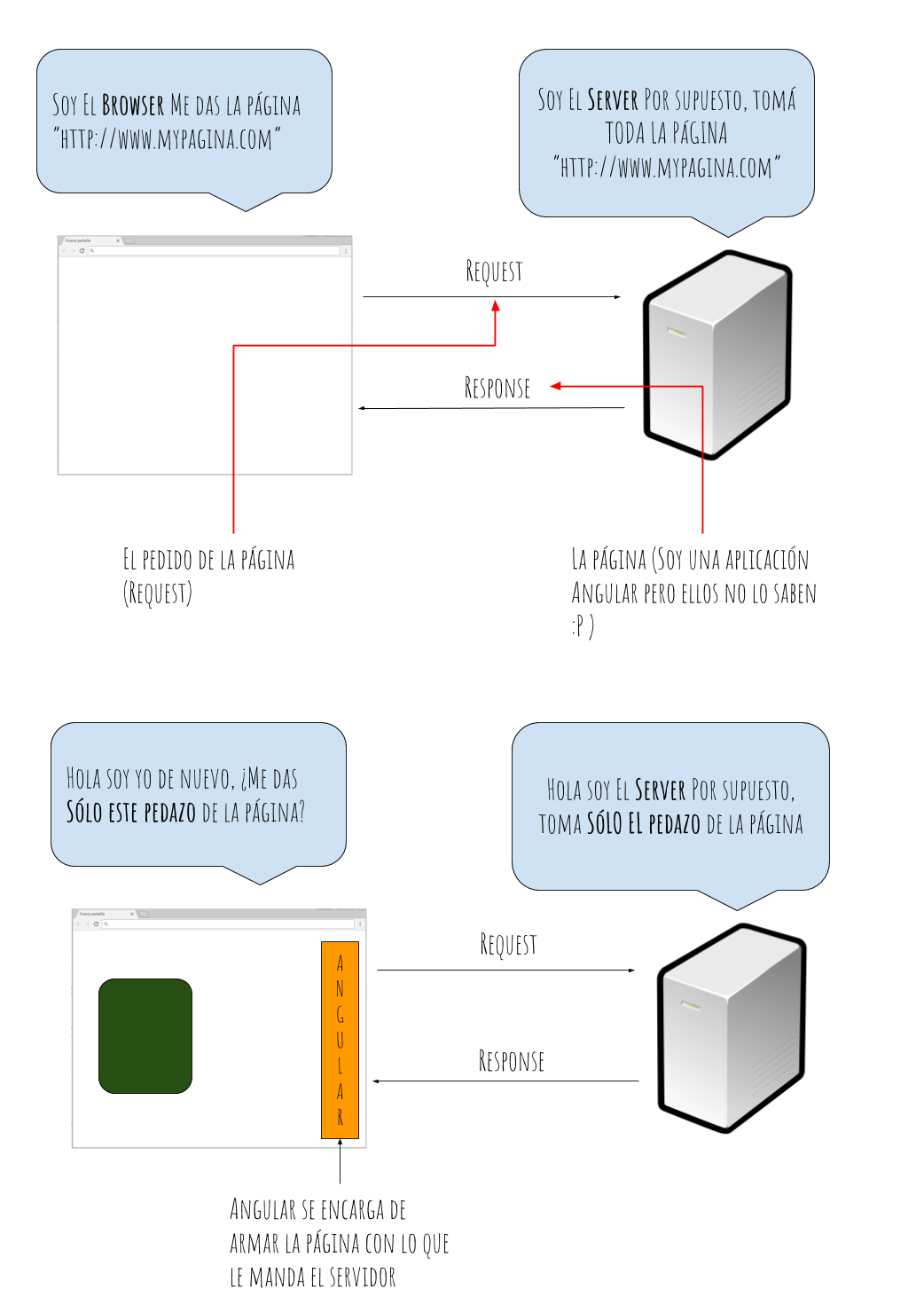
Cuando un usuario quería ver una página, la ingresaba en el ****browser****, este se conectaba en el ****Servidor**** (o ****Server)**** -la máquina donde estaba tu página web- y finalmente, este último te mandaba toda la página de vuelta (en uno o mas archivos).

Cuando querías ver una segunda página (haciendo click en un ****link,**** por ejemplo) el ****browser**** le volvía a pedir la otra página y el ****Server**** te volvía a mandar TODA la pagina de nuevo; sí sí, como leés, te mandaba toda la página aunque hubiera cosas repetidas, todo lo que estaba en la pagina se volvía a enviar -había mecanismos para guardar cosas que se repetían en la página pero, para simplificar el la explicación, vamos a decir que te enviaba todo de nuevo ;) -.



Como por esa época las páginas eran livianas y no se necesitaba enviar mucha información, este mecanismo funcionaba bien (ponéle…) pero, con el paso del tiempo, las páginas se volvieron más y más complejas, su volumen empezó a crecer  y se inventaron mecanismos para disminuir la cantidad de intercambios entre el ****browser**** y el ****Servidor****.

Lo que hacen ahora las aplicaciones Web Modernas es cargar una sola página (generalmente ****index.html****) y después, todas las otras páginas se refrescan usando ****Javascript****; pero sólo ****index.htm****l es una página completa, el resto de las páginas son sólo porciones de ****HTML**** que su código ****Javascript**** va cambiando dinámicamente. Esto hacía que la aplicación del ****browser**** funcionara mucho más rápido, ya que una de las cosas más lentas de una aplicación web es el intercambio de información entre el ****browser**** y el ****Server.**** Es más, ¡muchas veces ni siquiera es necesario un intercambio entre el ****browser**** y el ****Server,****ya que ****Angular**** se encarga de cambiar las páginas que tiene cargadas en el ****browser!.****



## Ahora si, veamos cómo funciona el ruteo en Angular

Para que este «****mecanismo Angular**** » funcione, hay que hacer varias cosas.

Lo que queremos hacer es «actualizar» solo un «cachito» de una página, en lugar de la página completa, como se hacía antiguamente; el cachito que le indiquemos va cambiando, pero el resto de la página sigue igual, sin modificación. Para eso se usa el ****tag <router-outlet>****

****Angular**** tiene un mecanismo que escucha los cambios de ****URL**** del ****browser**** y, dependiendo de la configuración, va a cambiar ese ****<router-outlet>****

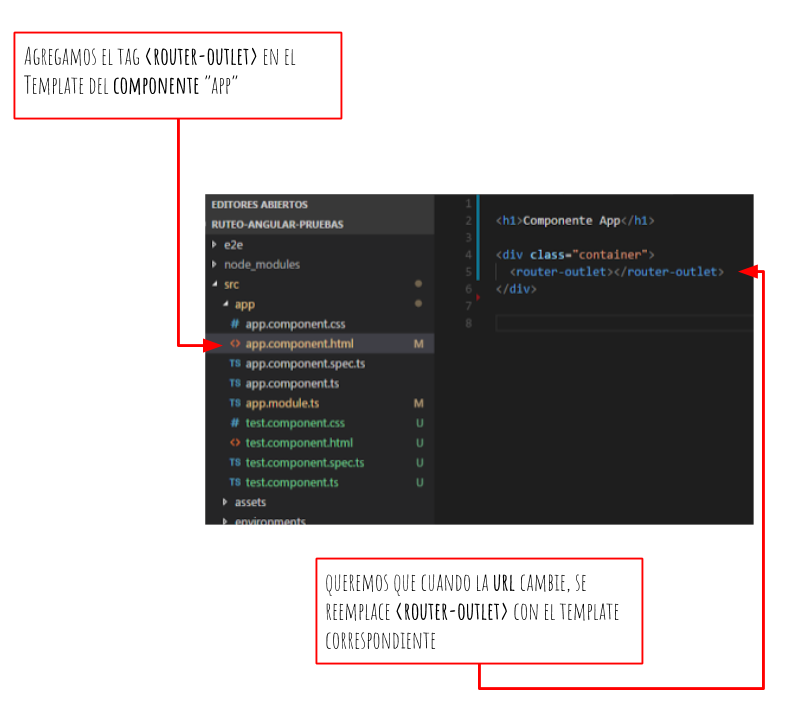
por el ****Template**** correspondiente a la ****URL****

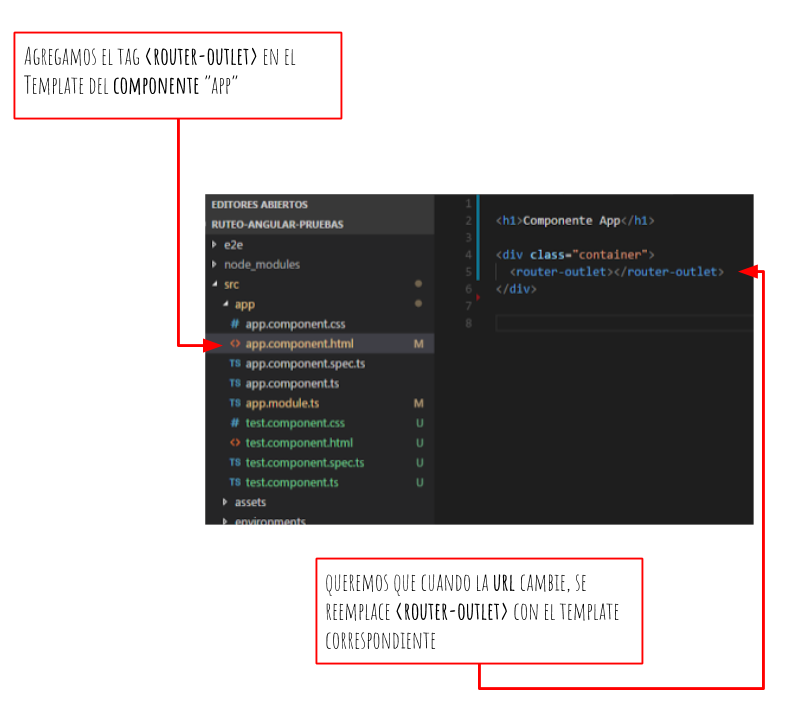
Recordemos primero que las aplicaciones ****Angular**** están formadas por ****Módulos Angular****, y que cada ****Módulo Angular**** puede tener uno o más ****Componentes****.

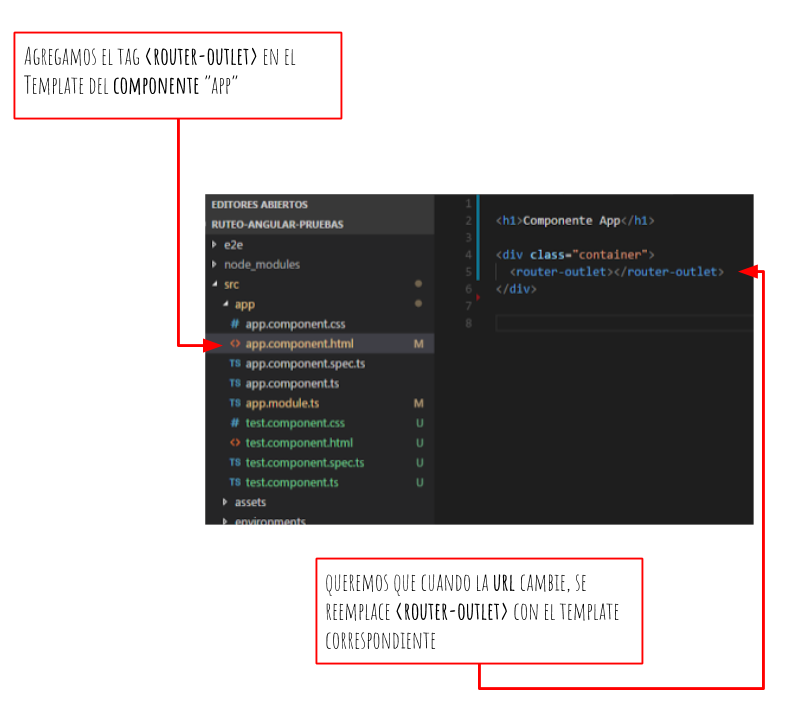
En el componente que vamos a poner el tag <router-outlet> también hay que configurarle  en su Modulo Angular el modulo

****RouterModule****

Dicho de otra forma, en el módulo ****RouterModule,**** vamos a mapear una url con su ****Componente**** correspondiente. En este ejemplo vamos a agregar en el ****Template**** del componente «app» (app.component.html) el ****tag <router-outlet>**** de esta forma:







### Buenísimo, ¿con eso ya es suficiente?

No, todavía no, ahora hay que configurar el ruteo en tu aplicación. O sea, si cambia la url de

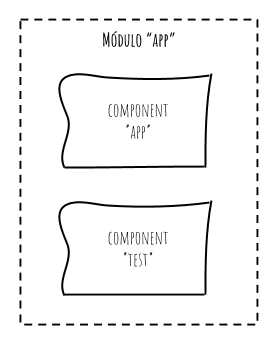
«http://localhost:4200/»

a

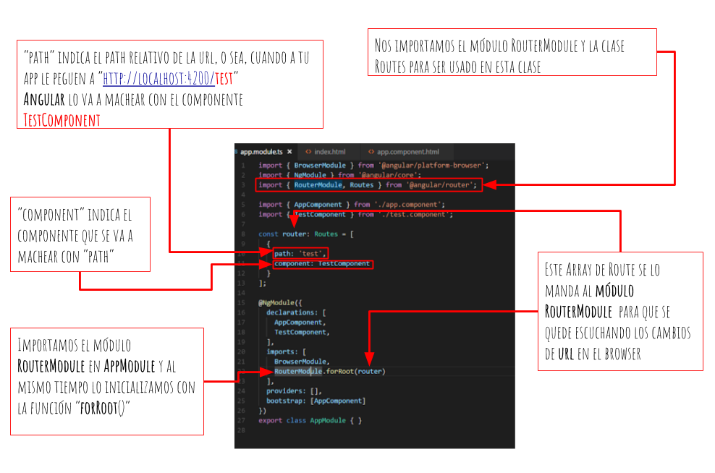
«http://localhost:4200/test»

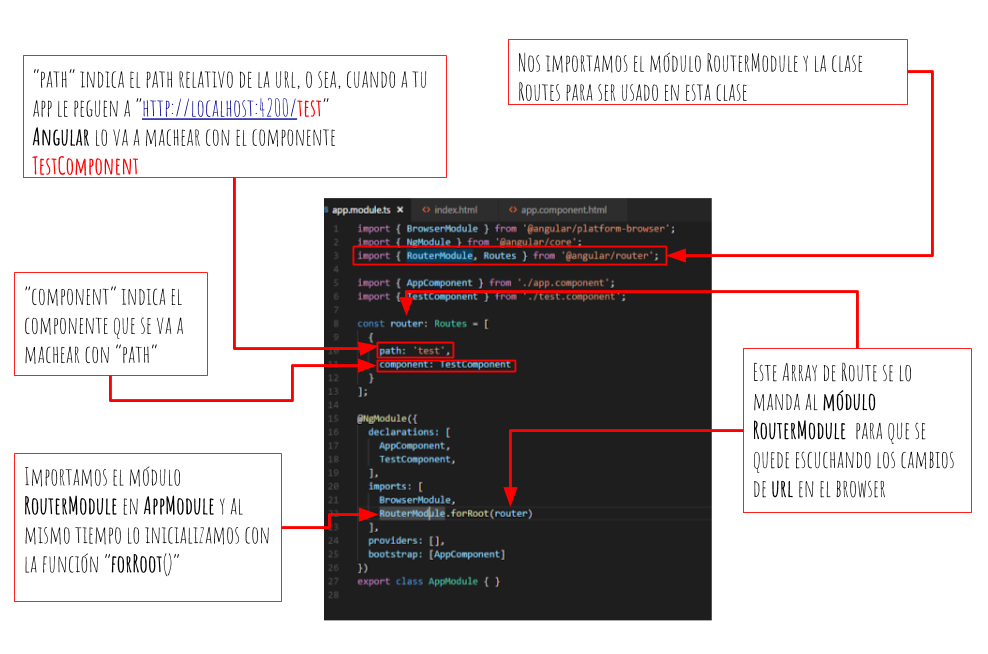
quiero que se actualice la aplicación con el ****Componente**** correspondiente.

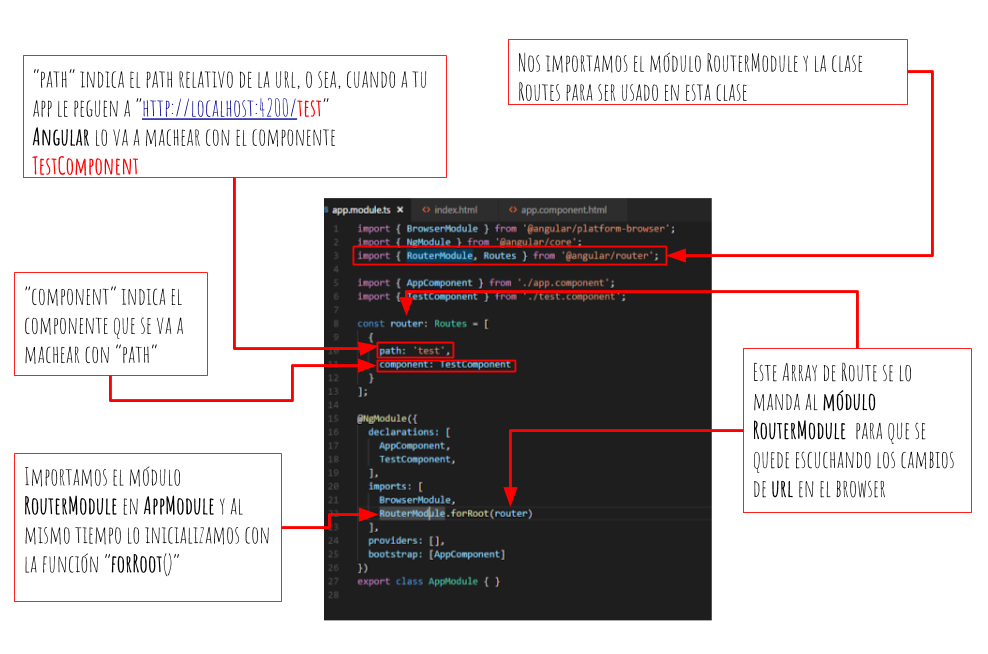
Supongamos que tenemos el ****ModuloApp**** que tiene dos ****Componentes:**** «app» y «test». Queremos ,que al darle clicKs en un ****link,**** se nos actualice un cachito de la página con el ****Template**** del ****Componente**** «test»



Para agregar el mapeo de esos componentes, lo hacemos con la función «****forRoot()»**** del módulo ****RouterModule****. El mapeo se hace de esta forma:







Lo que nos dice esta configuración es que cuando el ****path**** cambie a «****test****» en la URL del browser, Angular va a actualizar lo que este en

****<router-outlet>****

con el ****Componente****

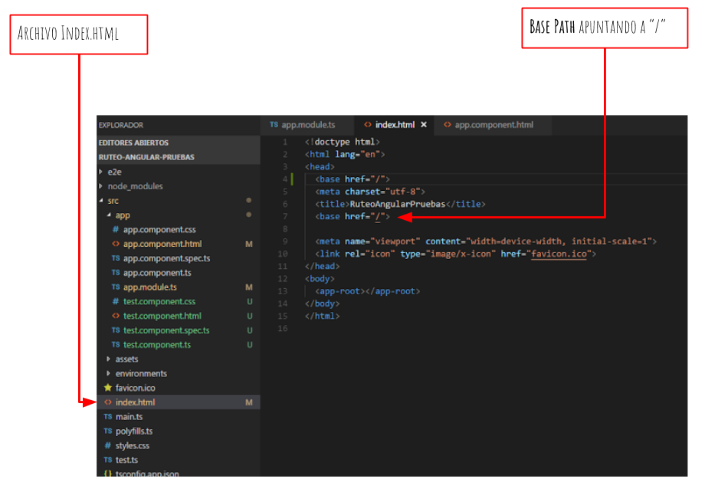
«TestComponent»

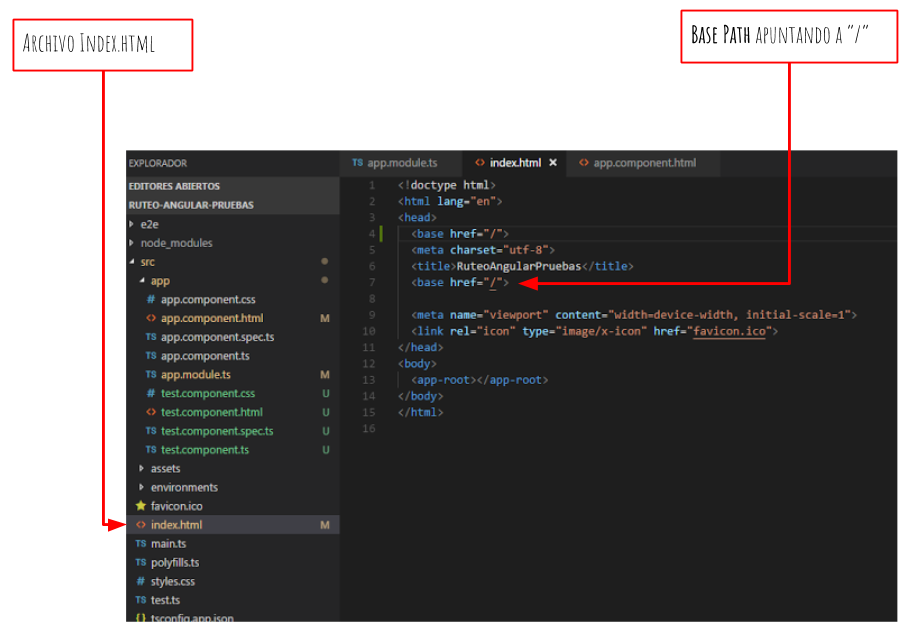
Nota: cualquier ****<router-outlet>**** puede rutear un solo ****Template**** de un componente al mismo tiempo.

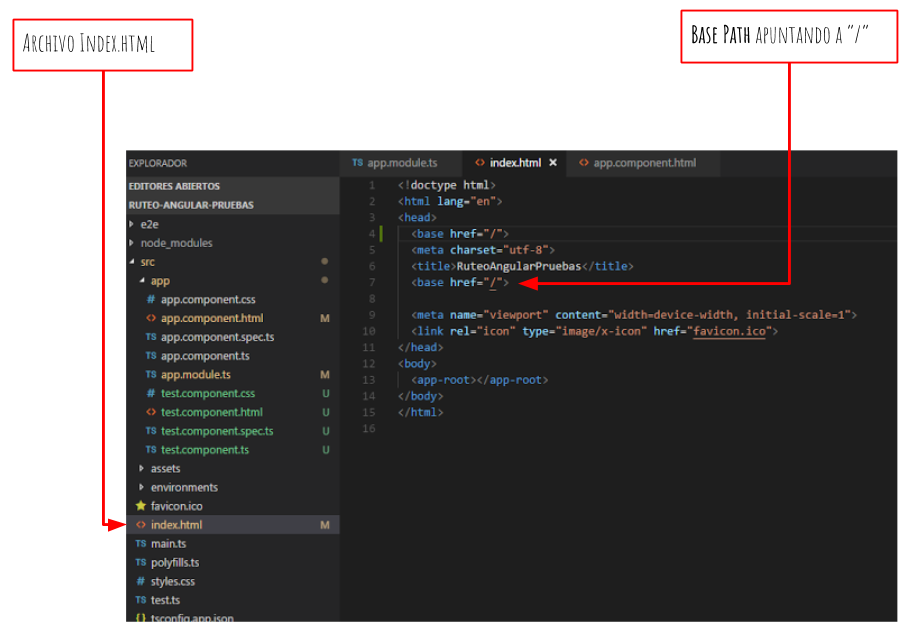
### Ahora sí, ¿con esto es suficiente?

Nop, falta una cosita más: agregar el ****Base Path**** en nuestro archivo ****index.html**** en la sección ****<head>****. Para simplificar el desarrollo, lo configuramos de esta forma en nuestro ****index.html:****

<base href="/">





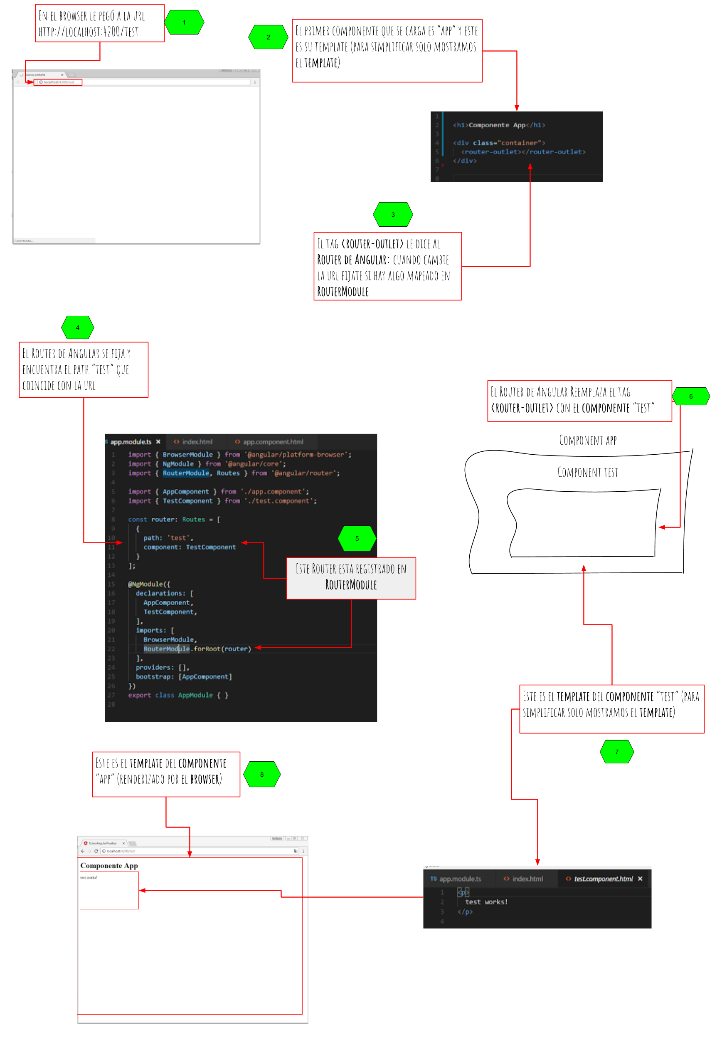


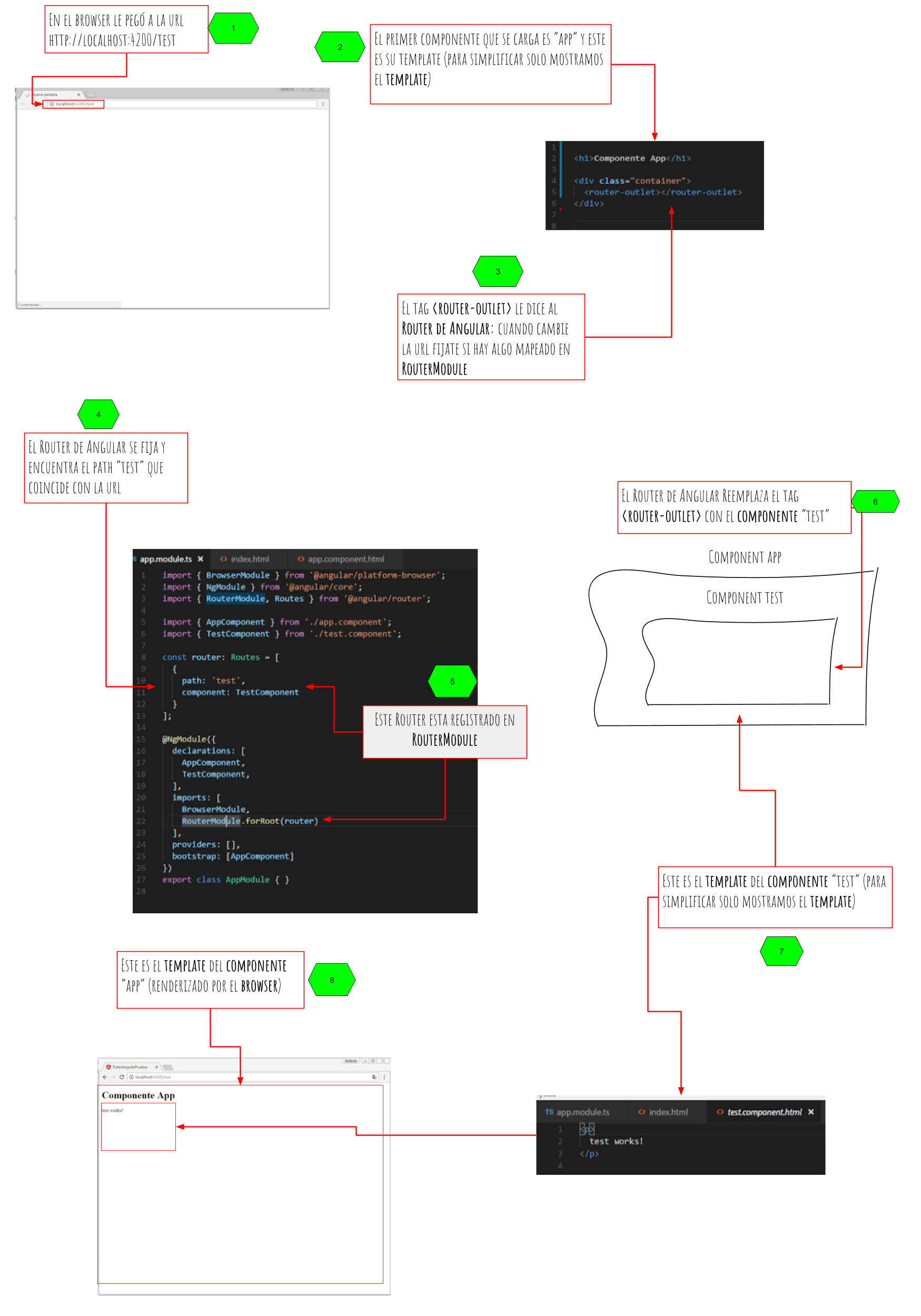
#### ¿Y que es el ****Base Path****?

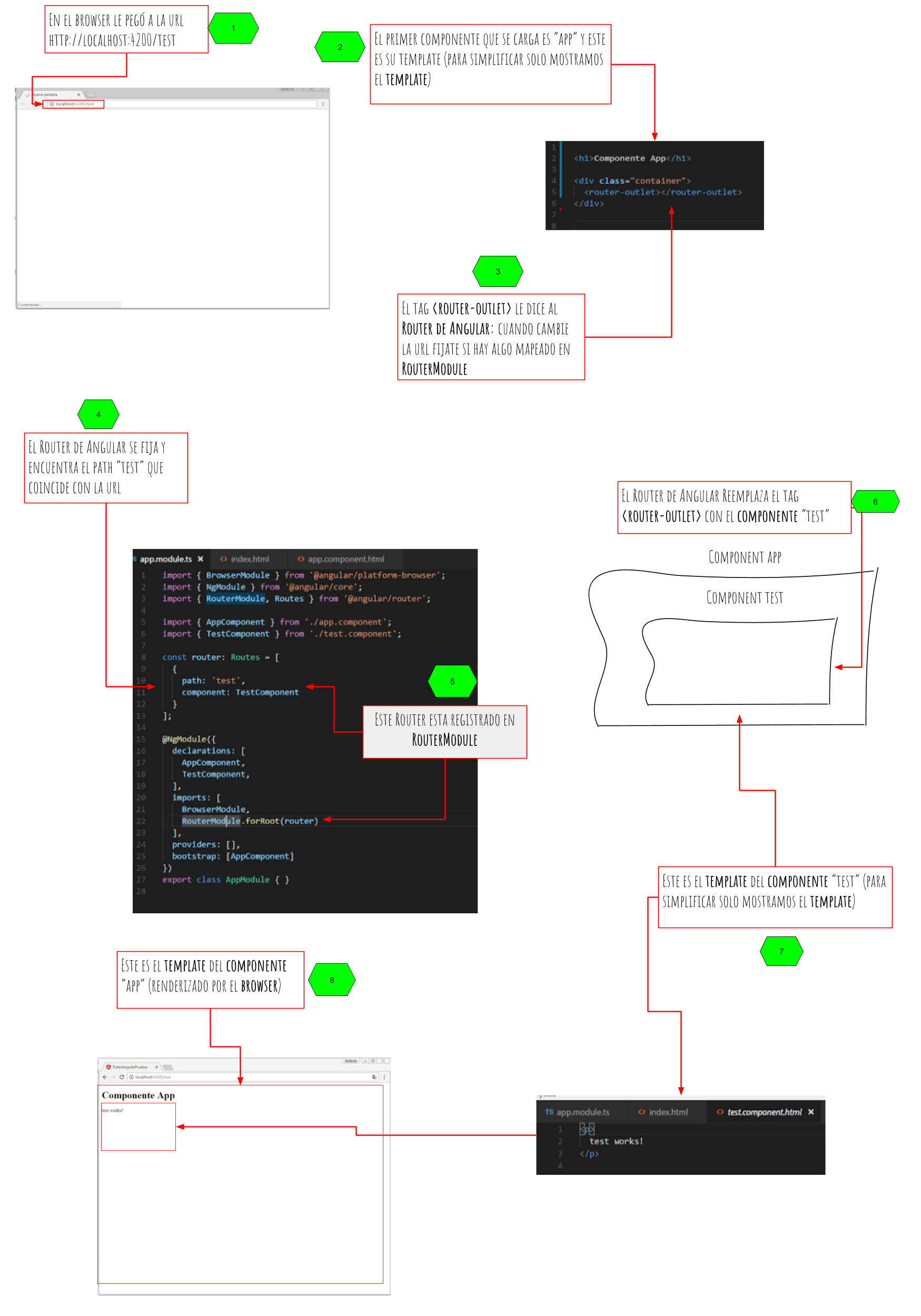
****Base Path**** es la parte de la ****URL**** que me indica la sub-carpeta de la aplicación en el ****Server****. ****Angular Router**** usa este ****Base Path**** para construir las ****Url**** para la navegación.

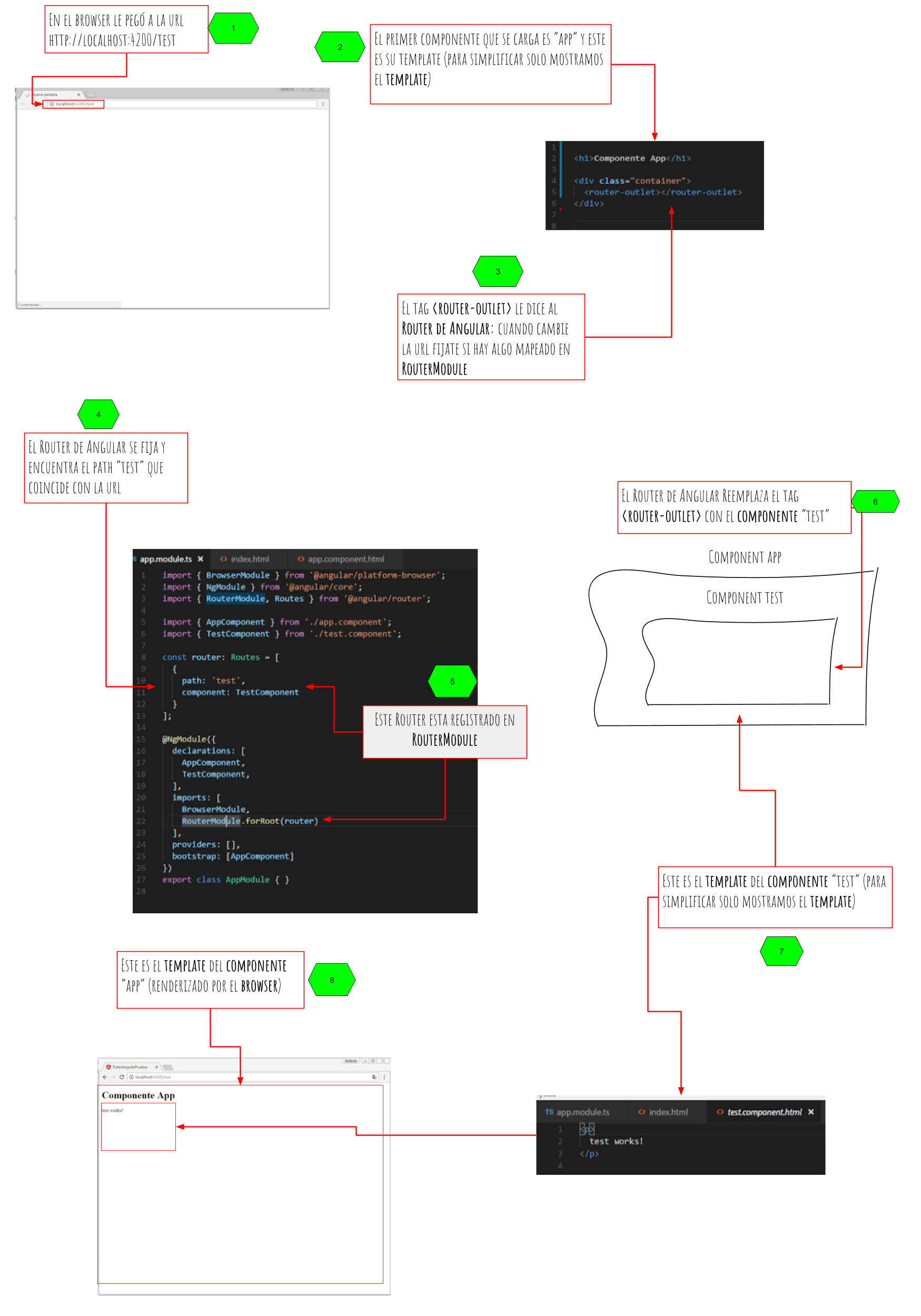
### Y ahora sí, ¡todo listo!

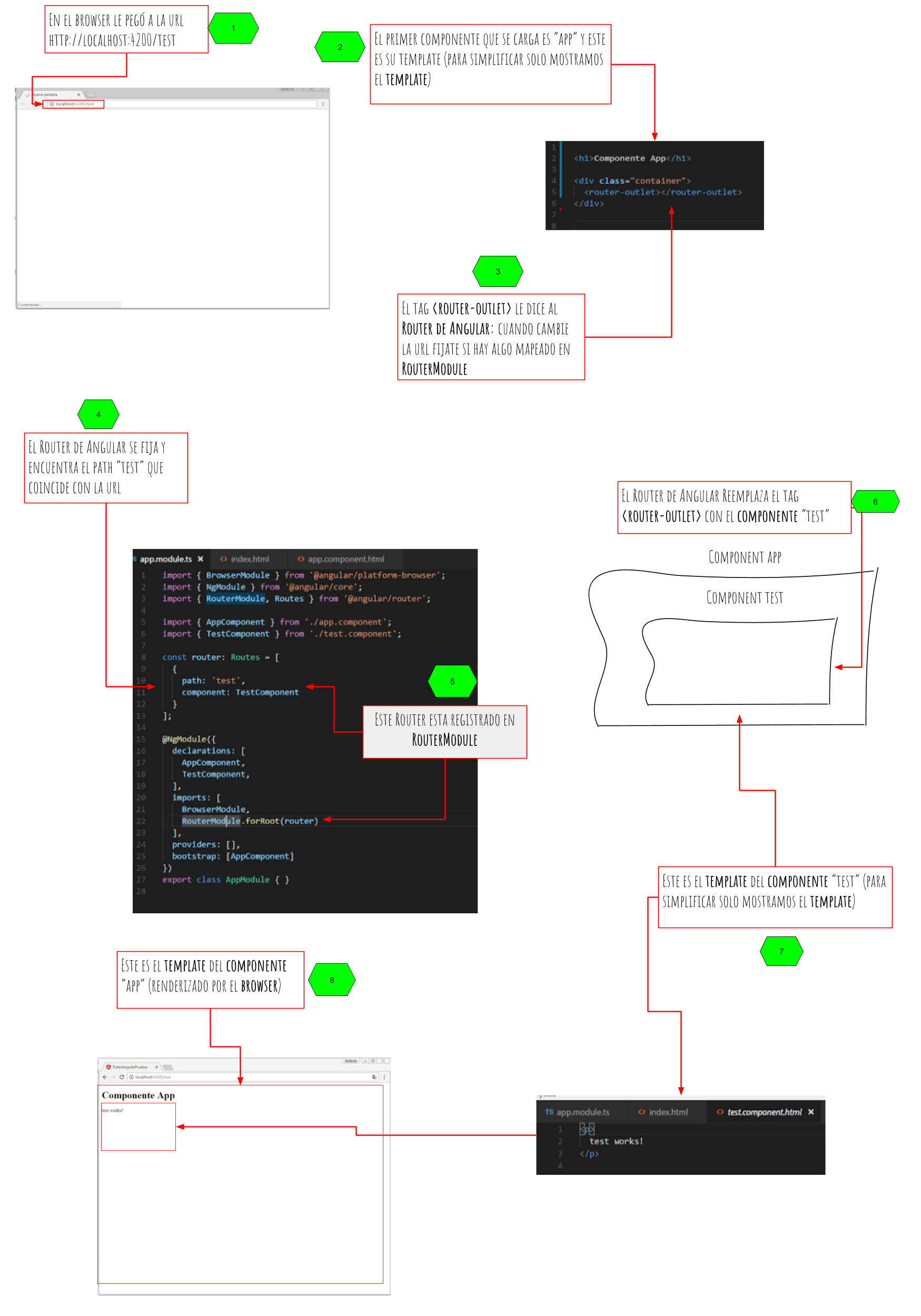
En este gráfico se muestra lo que hace internamente ****Angular Router**** (el ****Router de Angular****)







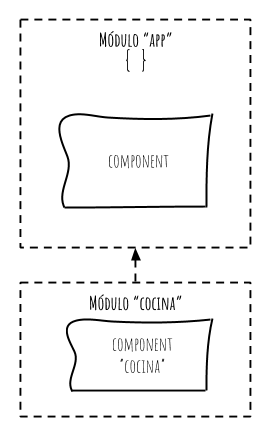




## Vamos con nuestro ejemplo, ¡manos a la obra!

Vamos a seguir con nuestra casa inteligente (podes ver los primeros tres post en estos links: [post 1,](https://gustavodohara.com/blogangular/como-crear-tu-primera-aplicacion-angular-en-una-hora/) [post 2](https://gustavodohara.com/blogangular/como-agregar-un-modulo-angular-en-tu-aplicacion/), [post 3](https://gustavodohara.com/blogangular/agregar-componente-una-pagina-modulo-angular/))

Tenemos una estructura de módulos como la siguiente:



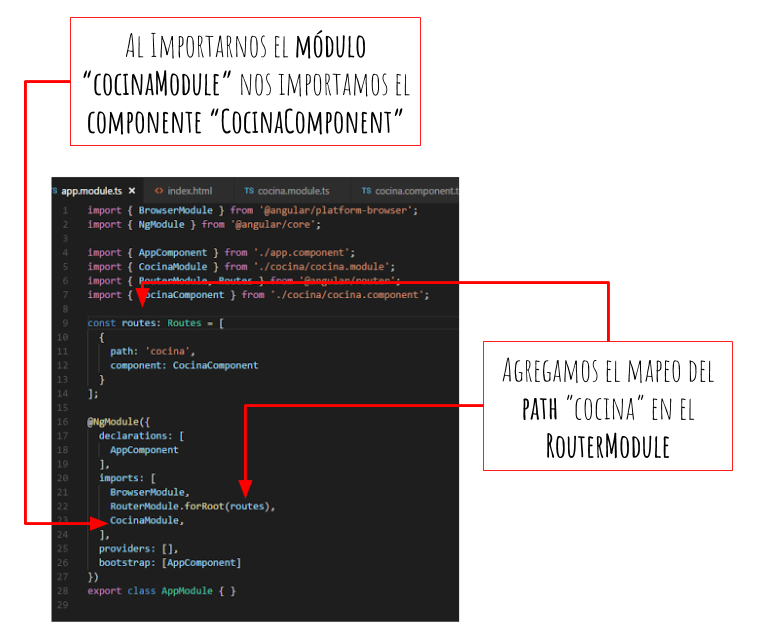
Y lo que vamos a hacer es agregarle un ruteo para que, cuando le peguemos a la ****URL**** «http://localhost:4200/cocina» ,nos muestre lo siguiente en el browser:



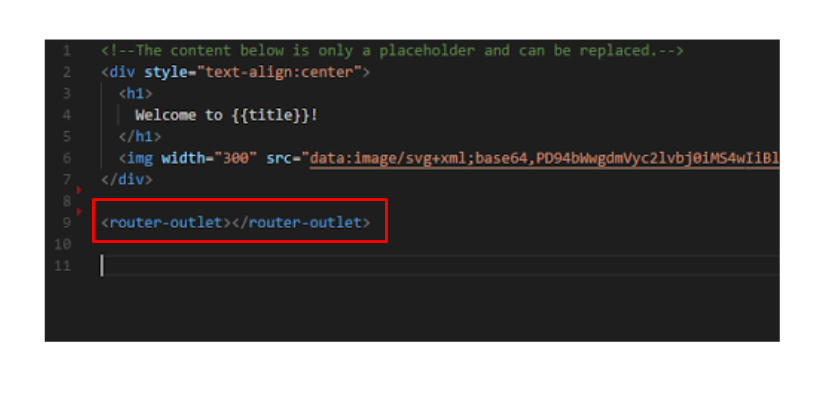
## Toqueteemos un poco el código…

Antes de empezar, cabe aclarar que el mapeo se podría hacer dentro del módulo «cocina» para que quede todo más modularizado, pero para simplificar la explicación, vamos a hacer todo dentro del módulo «app».

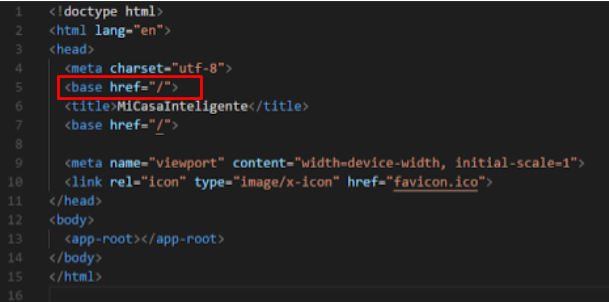
Primero, modificamos el archivo ****app.module.ts****, agregando el modulo ****RouteModule**** con su inicialización, llamando al método «****forRoot****()»



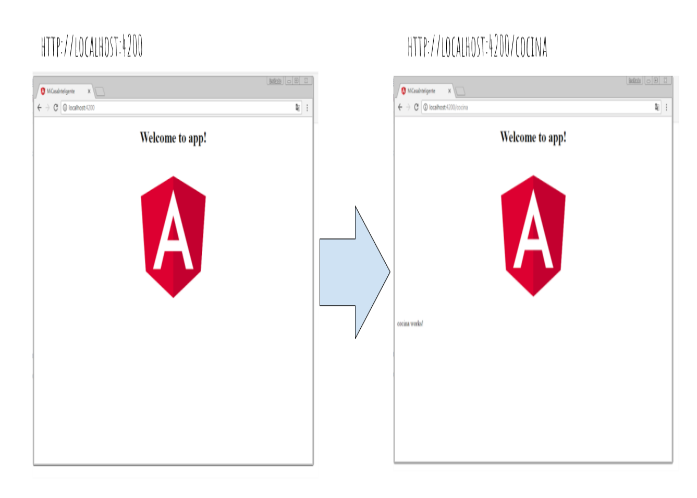
Modificamos el ****app.component**** agregándole el tag ****<router-outlet>**** (y de paso le modificamos un poco el contenido)



Por último, modificamos el archivo index.html por el temita del ****Base Path****



y listo, ya tenemos nuestra Casa Inteligente con Cocina y podemos verla en el browser:



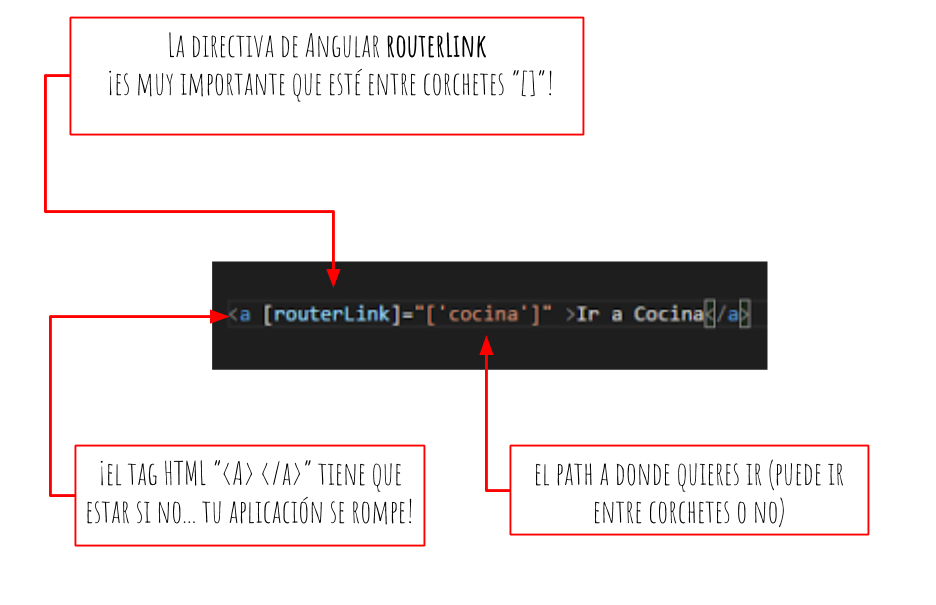
Fácil, ¿no?, en este ejemplo no se terminan de apreciar mucho las ventajas de usar este método, pero cuando más te adentres en el mundo de ****Angular**** y complejices tus aplicaciones, vas a podes verlo mejor.

## BONUS TRACK 1

### Activating Routes (RouterLink)

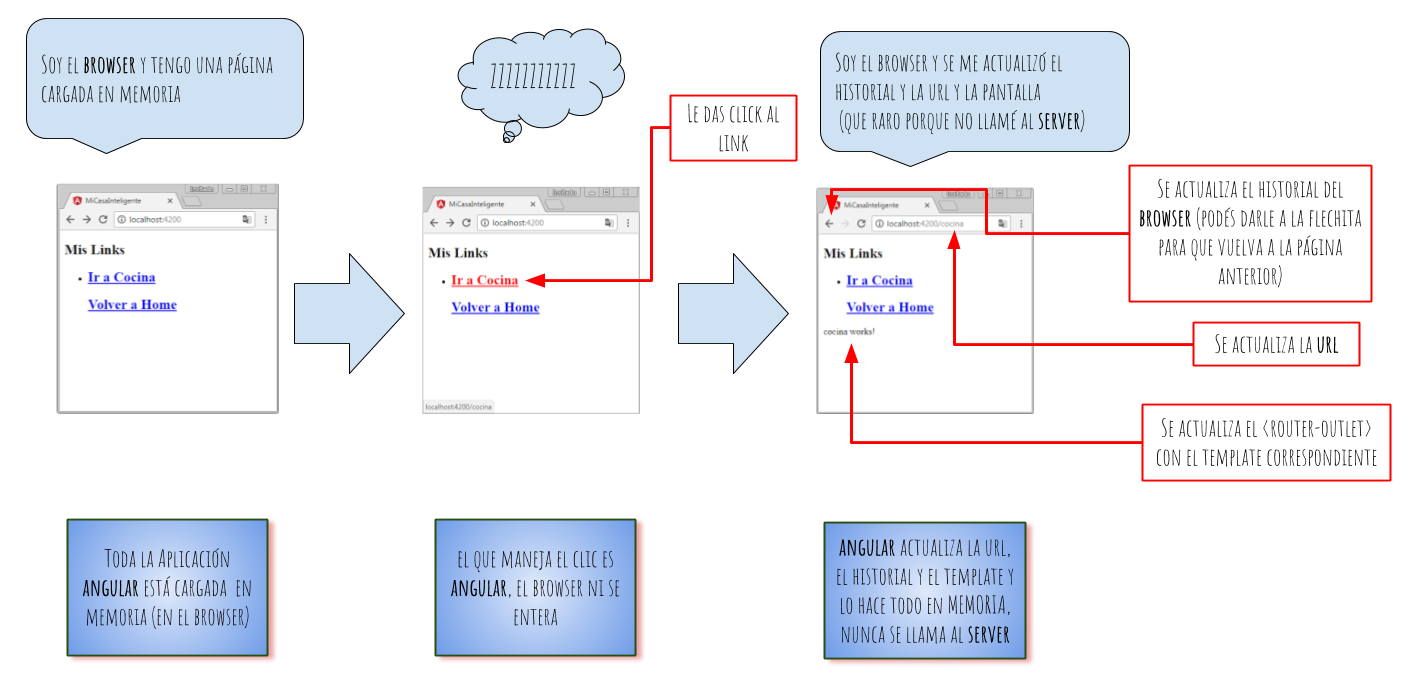
#### ¿Y los links? ¿cómo se manejan los links ahora en Angular?

****Angular**** tiene un mecanismo para eso, para crear un link, se hace algo como lo siguiente



#### ¿Qué pasa exactamente cuando le das clic al link en un routerLink?

****Angular**** ****Router**** actualiza la barra ****url**** del browser y su historial de navegación (del browser). Todo este manejo es local, no se envía nada al ****Server,**** no se recarga la página, en lugar de eso, ****Angular**** ****Router**** busca el ****path**** en su lista de ****path,**** activa el componente asociado y termina mostrando el ****Template**** asociado a ese componente.



Si tu aplicación tiene múltiples «****<router-outlet>****» se activan múltiples componentes con sus ****Templates****correspondiente en cada ****router-outlet****

#### ¿por que no se puede hacer algo tipo

#### <a href=»/cocina»/> ?

Si usas un link con la forma HTML  directa(sin usar ****Angular),**** lo que el ****browser**** hace es pedirle toda la página COMPLETA para después hacer todo el mecanismo de ****Angular**** de mapeo. Pero se pierde todo el manejo interno en memoria.

## BONUS TRACK 2

Por si no te diste cuenta, te engañé un poquito al decirte que no se recargaba la página completa de esta forma (mil disculpas :p) pero fue para simplificar la explicación.

Lo que quiero decir es que, si en la ****url**** de tu ****browser**** le pegás «http://localhost:4200/cocina», lo que el ****browser**** hace es recargar la página COMPLETA, ya que el ****browser**** «cree» que le estás pidiendo una página desde cero y te la vuelve a buscar al ****servidor.****

Para salvar este problema, hay que «engañar» al ****browser**** para que crea que no le estamos pidiendo una página nueva, sino que le estamos pasando parámetros en su ****URL,**** y eso lo hacemos poniendo el caracter ****hash**** (#) justo después de la ****url**** base. O sea, hacemos algo como ésto: «http://localhost:4200/#/»

De esta forma, el ****browser**** cree que le vamos a pasar parámetros y no vuelve a cargar la página.

Para hacer eso, le decimos al ****Router**** ****Angular**** que use el «hash» en los ****imports**** del módulo de la siguiente forma:

RouterModule.forRoot(router, {useHash: true})

Y cuando queremos acceder desde la ****url**** al ****path**** «test», lo hacemos de esta forma «http://localhost:4200/#/cocina»

De esta forma, el ****browser**** no recargará toda la página, y será el ****Router Angular**** el encargado de mostrarte el ****Componente**** correspondiente.

## Conclusión

El tema ****Router**** de ****Angular**** es un mundo, acá sólo tocamos la superficie. Estáte atento, que en futuros post explicaremos los secretos mejor guardados de cómo ****Rutear**** en ****Angular****.