https://www.udemy.com/curso-de-angular-2-4-5-avanzado-mean-jwt/

Creando el proyecto:

Ng new angular-avanzado

En angular 7 para verificar la versión es  
ng –version  
Dependiendo de lo que este en el package.json, se puede cambiar el puerto default:

Y empezar el comando con:  
npm run start

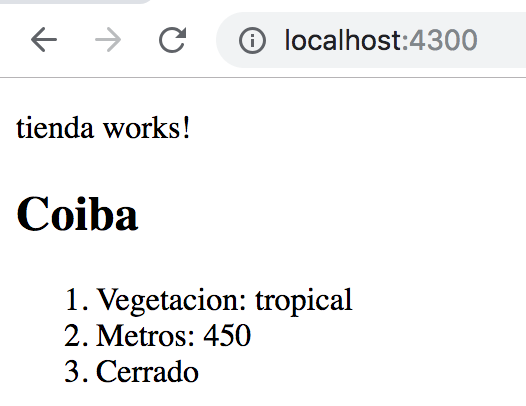


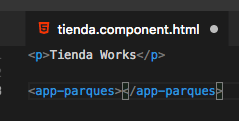
Creando componentes:

Se crea la carpeta componentes, y dentro los que usaremos, en este caso tienda.

Ng g c components/tienda

Parques





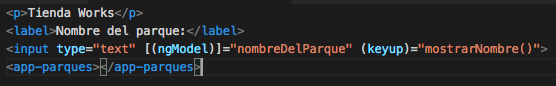


Comunicación entre componentes:

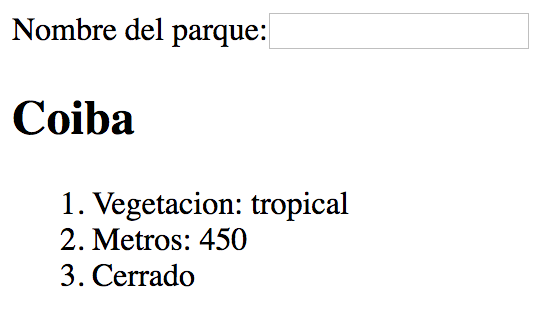
Input y Output

Pasando información desde el componente padre (tienda) Al componente hijo (parque).

Empezando se crea un input en tienda.component y se bindea el valor con un ngModel a una variable del componente tienda. (Importante importar FormsModule en el app.modules.).







Para recibir desde el componente hijo se utiliza el decorador Input() detrás de la variable



Una vez bindeado el valor podemos utilizar esta misma variable para asignarle el valor que tendrá, por ejemplo



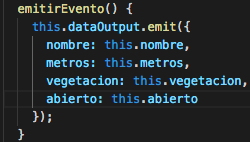


**Output**

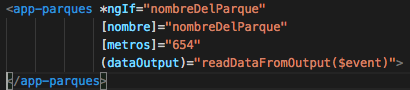
Para enviar datos al componente padre, hay que importar Output desde el componente hijo. Luego crear la variabe siendo una instancia del EventEmitter



Luego dentro de una función se puede declarar el dato de salida con la función emit.

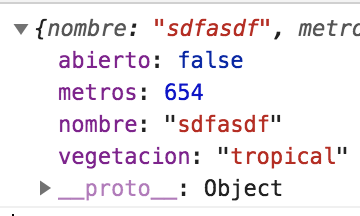


Desde el html del componente padre podemos utilizar el output con la variable declarada, luego llamando una función que recibe como parámetro un event, que realmente son los datos.



y finalmente en el component.ts llamamos a la función donde se le pasa la variable event y ya podemos utilizar el objeto que se recibió.





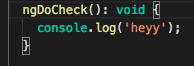
Hooks del ciclo de vida del componente:

Eventos que se lanzan dependiendo del estado del componente, o ciclo de vida.

OnChange, onInit, OnDestroy

Do Check

Se ejecuta luego del onInit, se ejecuta cada vez que pasa algo en nuestro html o componente.



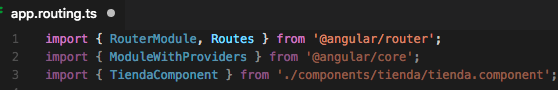
OnDestroy

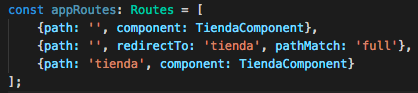
Se ejecuta cuando se destruye un componente. Por ejemplo cuando deja de mostrarse el componente hijo al no tener el nombre del parque en este ejemplo.

Configurando el sistema de rutas.

Se crea un archivo dentro de la carpeta app, app.routing.ts

Luego dentro se importa RouterModule, Routes, ModuleWithProviders y los componentes de la aplicación:

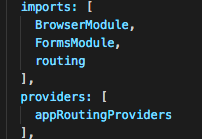


luego se crea una constante de rutas  


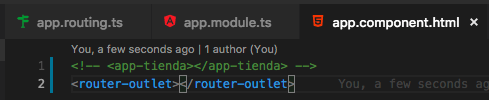
Luego exportando los provider de rutas y el modulo de rutas.:



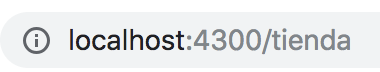
Luego desde el app.module importar estas variables en los imports y en providers:



Luego en el app.html mostramos el tag de router outlet, que mostrara el componente que este definido.



ahora carga por defecto esta ruta, pero si ponemos otra aparece un error



Para no tener error debemos configurar el router y asi cuando se ponga una ruta que no existe cargara por defecto la principal definida.

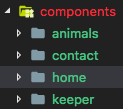


Routers y componentes:

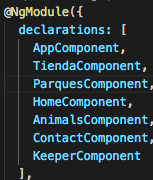
Creando componentes:

Home

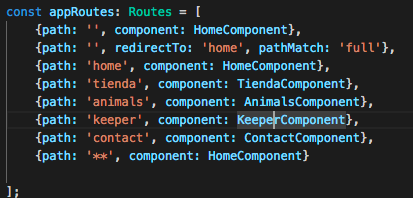




Recordar implementarlos en el app.module en declarations

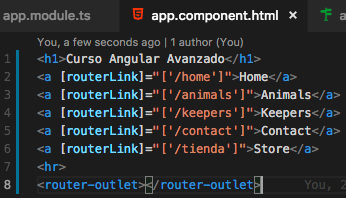


ahora como serán implementados en el nav es necesario incluirlos en el app.routing



Haciendo un menú de navegación

Desde el app componente creamos la navegación utilizando la propiedad routerLink

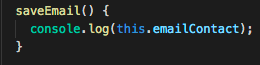




Utilizando localStorage y doCheck:

Nos permite guardar nuestros datos en la navegación, que persiste en la pagina durante la navegación de diferentes paginas.





utilizando el localStorage



si luego de guardarlo se agrega una impresión del localStorage desde el app.Component

se va a imprimir en un ngOnInit y recuperara el valor puesto desde contact.

Para mantener esta información siempre actualizada se puede utilizar el DoCheck ya que cada vez que cambia algo en en nuestra pagina se ejecuta.

