Práctica Angular – Parte 6

Tic-Tac-Toe

Objetivos

- De manera tutorizada extender la anterior versión del juego de las tres en raya.
- Posteriormente, sin ayuda, el alumno deberá realizar una serie de actividades con tal de ampliar aún más la funcionalidad de la aplicación.

Introducción

Para poner en práctica los conceptos vistos en el tutorial anterior, vamos a ampliar la funcionalidad de la aplicación de las tres en raya:

- Implementaremos rutas, para lo cual:
  - Incluiremos la posibilidad de navegar (mediante pestañas):
    - Página de bienvenida, para iniciar una partida, para continuar con una partida previa
- Crearemos una conexión al backend mediante un servicio
  - Recuperaremos del servidor partidas anteriores para poder continuarlas
- Crearemos un formulario mínimo para que el jugador introduzca su nombre
  - El formulario usará la característica del doble *binding*

Funcionalidad para la versión 2.0

Como se aprecia en la figura anterior, en la parte superior del tablero se muestra la barra de navegación de la aplicación, mediante la cual el usuario puede navegar a una de las tres páginas: bienvenida, nueva partida y continuar con una partida previa. Además, mejoraremos la estética de la aplicación con CSS.

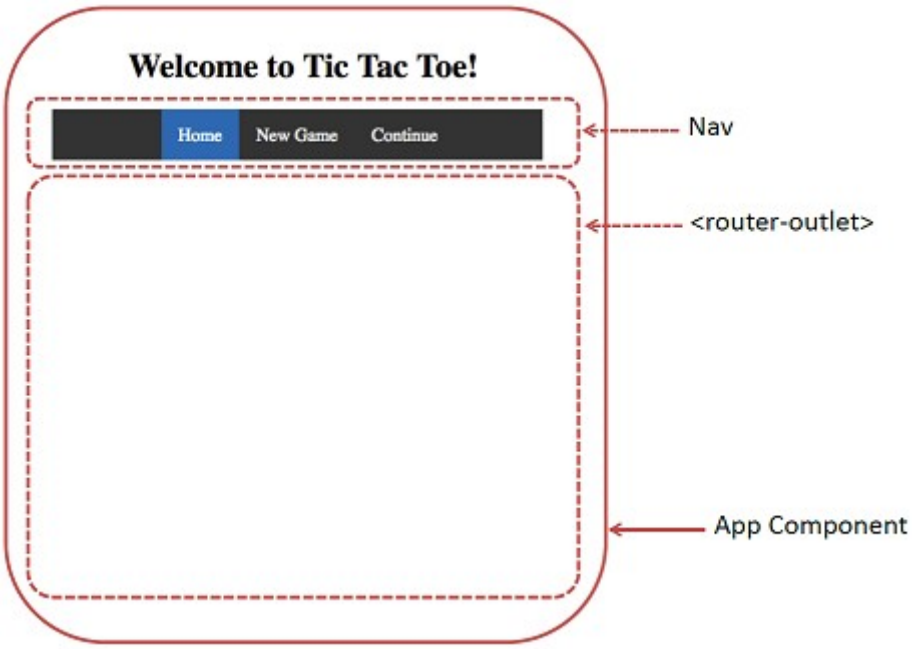
La aplicación

Tomaremos como punto de partida el proyecto final de la parte A (versión 1) de esta serie de prácticas.

Funcionalidad y componentes

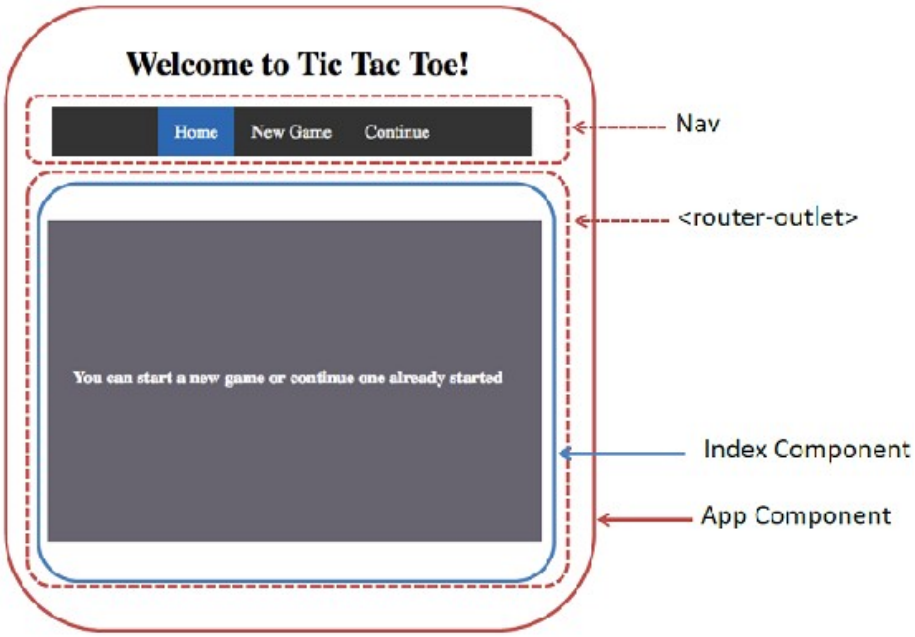
Tendremos dentro de AppComponent dos nuevos elementos:

- Una barra de navegación basada en Bootstrap.
- <router-outlet>, el *placeholder* de Angular para mostrar la página que toque en cada momento, según la navegación que realice el usuario.



Veamos las vistas que se mostrarán en <router-outlet>:

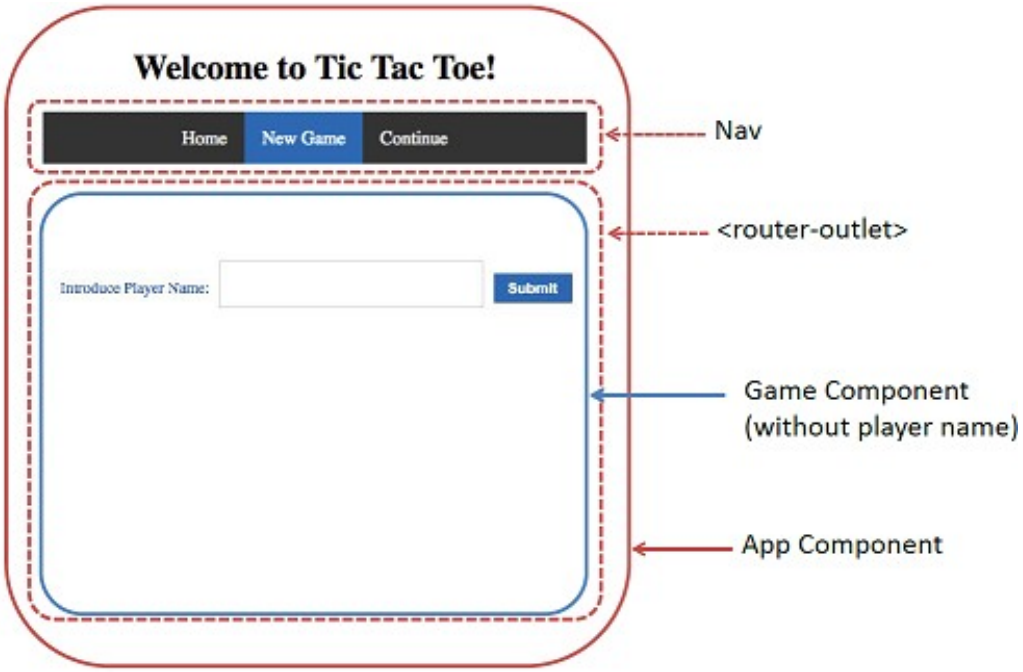
Al cargar la aplicación o cuando el usuario acceda a la pestaña “Home” se mostrará la página de bienvenida (**IndexComponent**):



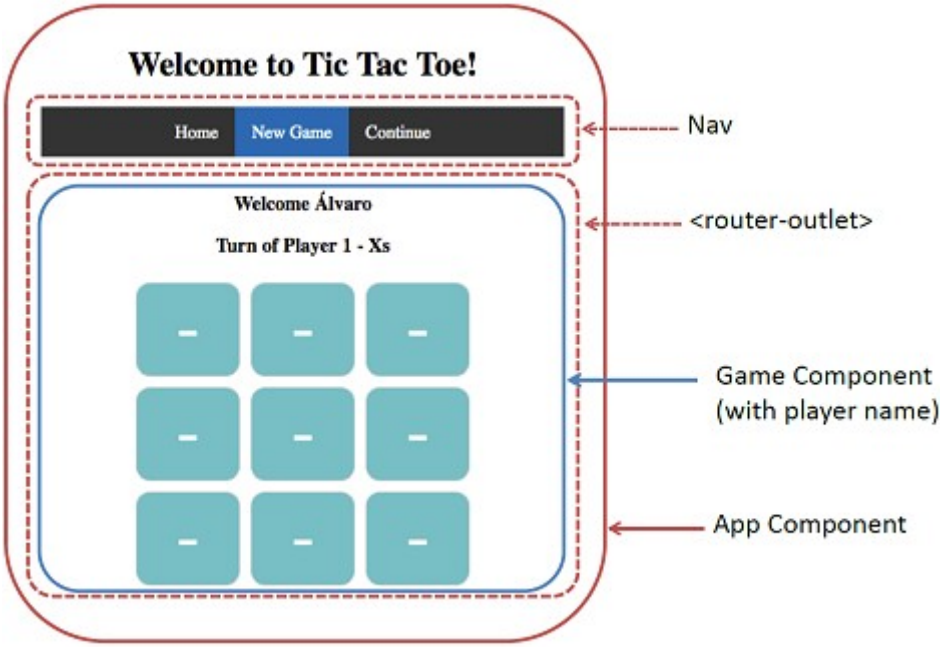
IndexComponent aún no existe; se trata de un componente muy sencillo que tan solo muestra un mensaje de bienvenida.

Cuando el usuario seleccione “New Game” o “Continue” mostraremos el mismo componente: GameComponent, que ya lo tenemos del proyecto anterior. Aprovechamos el mismo componente porque la funcionalidad básica es la misma para un nuevo juego o para continuar una partida previa. Mediante el control de parámetros y otros factores podremos mostrar una u otra vista.

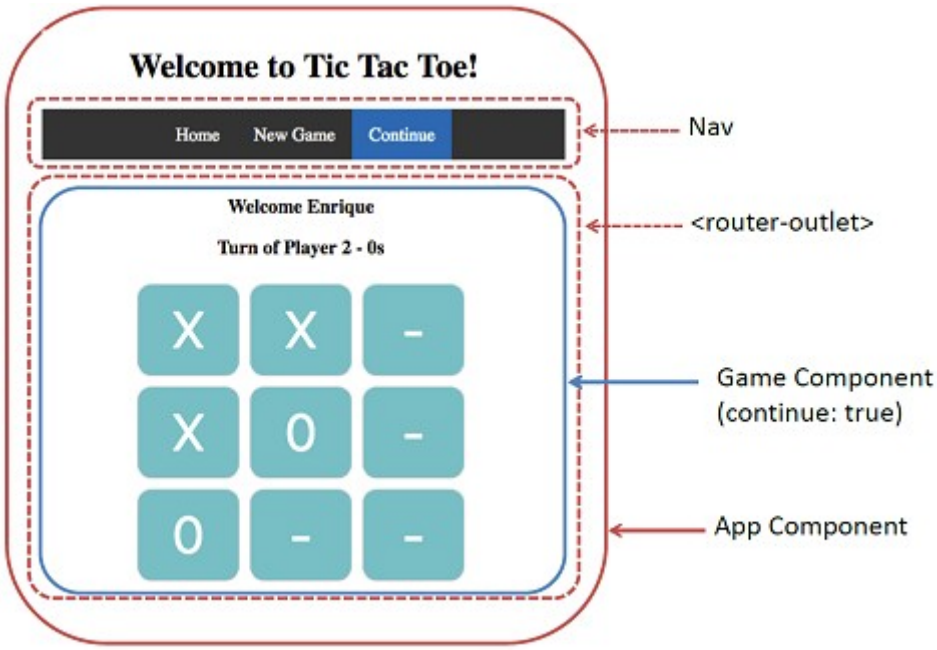
Al seleccionar “New Game” mostraremos un formulario muy simple en el que el usuario introducirá su nombre:



No se mostrará el tablero hasta que no introduzca un nombre y se pulse el botón de envío.
Una vez que el nombre esté establecido se mostrará el tablero para comenzar la nueva partida:



Finalmente, al seleccionar la opción “Continue” pasaremos a GameComponent un parámetro extra (continue = true) para desencadenar la recuperación de una partida previa desde el servidor y cargarla en el tablero:



Pasos a seguir

Organizaremos el trabajo de la siguiente manera:

1. Creación de rutas
    - Carga *fake* (hardcoded) de una partida previa
2. Conexión con el *backend* para obtener una partida previa real
3. Creación del formulario para el nombre del jugador
4. Mejora del aspecto de la aplicación mediante estilos

Paso 1. Creación de las rutas

Este paso se compone de las siguientes fases:

- Crear un nuevo componente llamado IndexComponent para mostrar la página de bienvenida.
- Configurar las rutas en AppModule.
- Modificar AppComponent para mostrar el navbar.
- Modificar GameComponent para cargar una partida *fake* cuando se seleccione “Continue” en el navbar, o resetear el estado de la aplicación cuando se seleccione “New Game”.

Creación de IndexComponent

Esta nueva versión de la aplicación necesita una página de bienvenida para la opción “Home” del navbar. Por tanto, vamos a crear un componente llamado **IndexComponent** que se encargue de esto. Dado que este componente no corresponde con la lógica del juego, lo crearemos fuera del módulo GameModule:

```
ng g c index
```

Lo anterior nos habrá creado una carpeta llamada “index” al mismo nivel que “game”. Ahora vamos al fichero src → app → index → index.component.html para establecer el mensaje de bienvenida:

index.component.html

```
<p>
  Hello! You can start a new game or load an already started one.
</p>
```

Creación y configuración de las rutas

A continuación crearemos las rutas para navegar de una página a otra. Cuando se seleccione en el navbar lo que queremos es lo siguiente:

- *'Home' → la URL contendrá 'index' → se ejecutará IndexComponent y veremos la página de bienvenida.*
- *'New Game' → la URL contendrá 'new' → se ejecutará GameComponent y veremos el formulario para introducir el nombre.*
- *'Continue' → la URL contendrá 'continue' → se ejecutará GameComponent y cargaremos la partida fake.*

Por tanto, creamos el array de rutas en AppModule, después de los *imports*. También nos aseguramos de tener los *imports* de Routes, IndexComponent y de GameComponent:

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { GameModule } from './game/game.module';

import { AppComponent } from './app.component';
import { IndexComponent } from './index/index.component';
import { GameComponent } from './game/game/game.component';
import { Routes } from '@angular/router';

const appRoutes: Routes = [
  { path: 'index', component: IndexComponent },
  { path: 'new', component: GameComponent },
  { path: 'continue', component: GameComponent, data: {continue: true} },
  { path: '', redirectTo: '/index', pathMatch: 'full' }
];

@NgModule({
...
```

Pero lo anterior aún no es suficiente, falta referenciar el array de rutas en la propiedad *imports* del decorador, lo que implica añadir el *import* de RouterModule:

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
...
import { RouterModule, Routes } from '@angular/router';

...

@NgModule({
  declarations: [ AppComponent, IndexComponent ],
  imports: [ BrowserModule, GameModule, RouterModule.forRoot(appRoutes) ],
  providers: [],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Y con esto hemos terminado por lo que respecta a las rutas de la aplicación.

Modificar AppComponent para mostrar el navbar

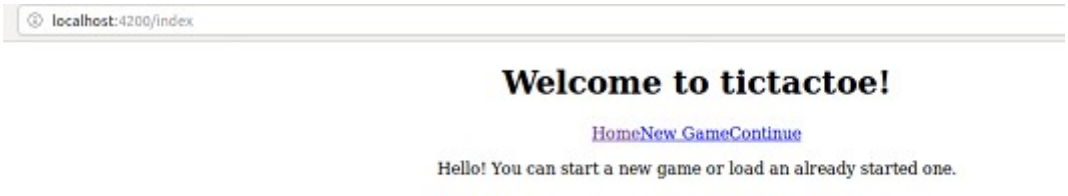
Ahora vamos a modificar la plantilla de **AppComponent** para incluir la barra de navegación. También incluiremos aquí la etiqueta **<router-outlet>** para mostrar de forma dinámica la vista que toque en cada momento. Sustituimos todo el contenido actual por el siguiente:

app.component.html

```
<div style="text-align:center">
  <h1>Welcome to {{ title }}!</h1>
  <nav>
    <a routerLink="/index" routerLinkActive="active">Home</a>
    <a routerLink="/new" routerLinkActive="active">New Game</a>
    <a routerLink="/continue" routerLinkActive="active">Continue</a>
  </nav>
  <router-outlet></router-outlet>
</div>
```

Notad que anteriormente utilizábamos la etiqueta <app-game> para incluir el componente que contenía el juego, mientras que ahora todo se controla de forma dinámica mediante el navbar y <router-oulet>.

Si lanzamos la aplicación deberíamos ver algo como lo siguiente:



Notad que por defecto, al no especificarse ninguna ruta, se va al index.
Si seleccionamos las diferentes opciones del navbar veremos que la URL va cambiando y que se muestran las vistas correspondientes en el lugar donde tenemos la etiqueta <router-outlet>.

Modificar GameComponent

Por el momento “New Game” y “Continue” hacen lo mismo. Para cambiar esto tenemos que modificar GameComponent, así, en función de un parámetro que llegue en la requets haremos una cosa o la otra, según corresponda.

game.component.ts

```
import { Component, OnInit } from '@angular/core';
import {StateService} from '../state.service';
import {ActivatedRoute} from '@angular/router';

@Component({
  selector: 'app-game',
  templateUrl: './game.component.html',
  styleUrls: ['./game.component.css']
})
export class GameComponent implements OnInit {

  private _stateService: StateService;
```

```
  constructor(route: ActivatedRoute, stateService: StateService) {
    this._stateService = stateService;
    /*
      'continue' es una parametro que viene en el array de rutas solamente
      cuando el usuario selecciona en el navbar la opción "Continue"
     */
    if (route.snapshot.data.continue) {
      // Partida fake
      stateService.state = {
        "turno": "PLAYER0",
        "valores": [
          ['X', 'X', '-'],
          ['X', '0', '-'],
          ['0', '-', '-']
        ]};
    } else {
      stateService.reset();
    }
  }

  ngOnInit() {
  }

  handleResetButton() {
    this._stateService.reset();
  }
}
```

Con lo anterior ya tendríamos completada la funcionalidad propuesta para este tutorial.

Conclusión

Hemos implementado un sistema de navegación en nuestra aplicación que nos permite navegar por las diferentes páginas. En la siguiente parte de esta práctica crearemos un servicio para poder recuperar del backend partidas guardadas previamente, en lugar de cargar un partida *fake*.