ANGULAR

Jose Alberola Torres
Javier García Lillo
Raquel Martínez García
Pedro López Mellado

Índice

¿Qué es?

Angular es un framework opensource desarrollado por Google para facilitar la creación y programación de aplicaciones web de una sola página, las webs SPA (Single Page Application).

Angular separa completamente el frontend y el backend en la aplicación, evita escribir código repetitivo y mantiene todo más ordenado gracias a su patrón **MVC** (Modelo-Vista-Controlador) asegurando los desarrollos con rapidez, a la vez que posibilita modificaciones y actualizaciones.

En una web SPA aunque la velocidad de carga puede resultar un poco lenta la primera vez que se abre, navegar después es totalmente instantáneo, ya que se ha cargado toda la página de golpe.

Solamente es una ruta la que se tiene que enviar el servidor, y Angular lo que hace 'por debajo' es cambiar la vista al navegar para que dé la apariencia de una web normal, pero de forma más dinámica.

Entre otras ventajas, este framework es modular y escalable adaptándose a nuestras necesidades y al estar basado en el estándar de componentes web, y con un conjunto de interfaz de programación de aplicaciones (API) permite crear nuevas etiquetas HTML personalizadas que pueden reutilizarse.

El lenguaje principal de programación de Angular es TypeScript, y así toda la sintaxis y el modo de hacer las cosas en el código es el mismo, lo que añade coherencia y consistencia a la información, permitiendo por ejemplo, la incorporación de nuevos programadores, en caso de ser necesarios, ya que pueden continuar su trabajo sin excesiva dificultad.

Ventajas de usar Angular

Algunos de los beneficios de usar Angular son los siguientes:

- Ahorrar tiempo → cuando uno empieza a pensar en cómo crear una aplicación web, debe tomar una serie de decisiones (la arquitectura de la aplicación, su organización, etc), y Angular se encarga de eso.
- Aplicaciones fáciles de mantener → al usar TypeScript, cualquier cambio que deba hacerse en la aplicación podrá llevarse a cabo rápidamente y sin errores.
- Uso de componentes web → son porciones de código que pueden reutilizarse en otros proyectos implementados con Angular.
- Futuro estable → igual que con JavaScript, por ejemplo, se sufren muchos cambios a menudo, esto no sucede con Angular, que sufrirá muchos menos cambios grandes de versión en versión.
- Gran soporte de herramientas → las plantillas de Angular almacenan por separado el código de la interfaz de usuario y el de la lógica de negocio, por lo que puedes sacarle partido a las muchas herramientas ya existentes para editar este tipo de archivos.
- Flexibilidad en cuanto a la plataforma para desarrollar → por ejemplo, si usamos Ionic (librería basada en Angular), podemos realizar aplicaciones tanto para iOs como para Android.

Conceptos de Angular

La arquitectura de una aplicación en Angular se basa en algún término fundamental. Vamos a explicar los tres más importantes brevemente.

En primer lugar están los *módulos*, conocidos como NgModules y encargados de recopilar código relacionado en conjuntos funcionales. En cada aplicación hay un módulo raíz, denominado con AppModule, que permite su arranque.

A continuación tenemos los *componentes*, donde cada uno de ellos define una clase con datos y lógica de la aplicación, además están asociados a plantillas que definen unas vistas. Estas plantillas combinan HTML con markup de Angular que puede modificar elementos HTML antes de que se muestren.

Por último estarían los *servicios* que proporcionan una funcionalidad específica que no está relacionada con las vistas directamente y se desean compartir con otros componentes.

¿Por qué usar Angular?

Angular surgió hace unos años y es desarrollado y mantenido por Google. Este framework ofrece una serie de características importantes que destacan ante otros frameworks.

La primera a destacar es su velocidad y rendimiento tanto a la hora de ejecutarlo en el navegador, como al realizar cambios en la aplicación, ya que su diseño se basa en componentes reutilizables que hacen que no sea necesario estar recargando la página continuamente.

Por otro lado, cabe mencionar su gran productividad. Angular permite crear vistas de interfaz de usuario con una sintaxis simple, esto es gracias a las plantillas mencionadas anteriormente. Además, hay una serie de herramientas de línea de comandos que permiten tanto empezar a desarrollar código rápidamente como añadir nuevos componentes o realizar tests.

Por último, y a pesar de que es una característica que muchas veces no se tiene demasiado en cuenta, es todo lo relacionado con el testing y la accesibilidad. Este framework previene la detección de errores gracias a su sistema de testing unitario, integración y e2e (End To End, término que se utiliza para indicar que un proveedor de software, además de suministrar una solución, estará presente en todas las fases de interacción de un cliente con esa solución).

Todas estas características mencionadas han hecho que el uso de Angular haya aumentado de forma exponencial en los últimos años y, por tanto, su demanda sea imparable.

Introducción a TypeScript

Como hemos comentado anteriormente, Typescript es el lenguaje principal de programación en Angular. Comentaremos las características principales de este lenguaje de programación.

TypeScript es un lenguaje de programación de alto nivel que implementa muchos de los mecanismos más habituales de la programación orientada a objetos, pudiendo extraer grandes beneficios que serán especialmente deseables en aplicaciones grandes, capaces de escalar correctamente durante todo su tiempo de mantenimiento.

La característica fundamental de TypeScript es que compila en Javascript nativo, por lo que se puede usar en todo proyecto donde se esté usando Javascript. Dicho con otras palabras, cuando se usa TypeScript en algún momento se realiza su compilación, convirtiendo su código a Javascript común. El navegador, o cualquier otra plataforma donde se ejecuta Javascript, nunca llegará a enterarse que el código original estaba escrito en TypeScript, porque lo único que llegará a ejecutar es el Javascript resultante de la compilación.

En resumen, TypeScript es lo que se conoce como un "superset" de Javascript, aportando herramientas avanzadas para la programación que traen grandes beneficios a los proyectos. Un superset es un lenguaje escrito encima de otro lenguaje o mejor dicho, que compila a otro lenguaje. En el caso de TypeScript es un lenguaje que compila a JavaScript, pero que incluye muchas facilidades y ventajas.

¿Qué necesitamos para usar TypeScript?

Básicamente necesitamos descargar dos programas. El primero es NodeJS, no porque necesitemos desarrollar con Node, sino porque el compilador de TypeScript está desarrollado en NodeJS.

Desde el sitio de NodeJS encontramos las opciones para la instalación en nuestro sistema operativo, por medio de un típico instalador con su asistente.

Luego necesitarás el TSC (Command-line TypeScript Compiler), la herramienta que nos permite compilar un archivo TypeScript a Javascript nativo. Este software es el que está realizado con NodeJS y su instalación se realiza vía npm con el siguiente comando:

```
npm install -g typescript
```

Implementación del código

En este apartado vamos a explicar cómo hemos realizado nuestra aplicación usando Angular. Va a ser un código sencillo para poder entender fácilmente su funcionamiento, por lo que se va a tratar de un CRUD (create, read, update and delete) de tareas.

En primer lugar, para crear el proyecto hay que ejecutar *ng new nombreproyecto* en una terminal, lo que nos generará una serie de ficheros y directorios.

A continuación vamos a centrarnos en los dos conceptos más importantes cuyos ficheros hemos modificado para implementar la aplicación: la plantilla html y el component.

Html

El html de la aplicación se encuentra en el fichero *app.component.html.* Vamos a dividir la vista en tres secciones principales:

- Crear nueva tarea: cada tarea tendrá un solo atributo, su nombre, y al darle al botón de Crear se llamará al método addTarea().
- Actualizar una tarea: se podrá modificar el nombre de una tarea, y al darle al botón de Actualizar llamará al método updateTarea(). Para ello habrá que seleccionar la tarea antes dándole al botón de Editar correspondiente.
- Lista de tareas: en la parte inferior de la página podremos ver un listado con todas las tareas existentes, puesto que no tenemos una base de datos, las hemos creado de forma manual. Cada tarea tendrá dos botones: editar (como ya hemos explicado en el punto anterior) y borrar (al seleccionarlo se llamará al método deleteTarea() y se mostrará un pop-up para confirmar el borrado de la tarea seleccionada).

  Aquí podemos ver un fragmento de código para mostrar el listado de las tareas:

```
<tr *ngFor="let tarea of tareas; let i = index">
  <td>{{tarea.tarea}}</td>
  <td style="text-align: center;">
    <a style="margin-right: 10px;" (click)="editTarea(i)" class="btn btn-info">
      Editar
    </a>
    <a (click)="deleteTarea(i)" class="btn btn-danger">
      Borrar
    </a>
  </td>
</tr>
```

Component

En el archivo app.component.ts crearemos los métodos CRUD de las tareas. En primer lugar, debemos definir un **@Component**, es decir, un decorador**.**

Al decorador tenemos que pasarle un objeto de JS para configurarlo y hacerlo válido. La información que vamos a incluir aquí se almacenará como meta data asociada a esta clase, que le servirá a Angular para saber qué hacer con dicha clase. De momento, esta información será:

- Un selector con el nombre que tendrá nuestra etiqueta de HTML personalizada. Debe ser un string, y lo habitual es que usemos el prefijo -app más el nombre de nuestro componente.
- Una referencia a una template, con la que podemos tomar una de estas opciones:
    - Apuntar a un archivo HTML externo → templateUrl (debemos usar un relative path)
    - Escribir un inline HTML

El proyecto usa una base de datos cargada en memoria por lo que debemos definir las tareas que queramos que aparezcan por defecto en nuestra aplicación:

```
tareas = [
  {'tarea': 'Hacer la compra'},
  {'tarea': 'Pasear al perro'},
  {'tarea': 'Trabajo ADI'}
]
```

Crearemos los siguientes métodos para el CRUD de las tareas:

- addTarea: el método realiza un push del model al array de tareas.

```
addTarea():void{
  this.tareas.push(this.model);
  this.msg = "Tarea creada";
}
```

- deleteTarea: realizaremos un splice del array de tareas.

```
deleteTarea(i:number):void{
  var borrar = confirm('¿Deseas eliminar la tarea?');

  if(borrar){
    this.tareas.splice(i, 1);
    this.msg = "Tarea borrada";
  }
}
```

- editTarea: carga en el label de Actualizar tarea el título de la tarea a editar.

```
editTarea(i: number):void{
  this.ocultarFormUpdate = false;
  this.model2.tarea = this.tareas[i].tarea;
  this.myValue = i;
}
```

- updateTarea: si hemos seleccionado una tarea a editar, cambiaremos el nombre de la tarea al nuevo.

```
updateTarea():void{
  let i = this.myValue;

  if(i < 0){
    this.err = "Debes seleccionar una tarea a editar";
  }
  else{
    for(let j = 0; j < this.tareas.length; j++){
      if(i == j){
        this.tareas[i] = this.model2;
        this.model2 = {};
        this.msg = "Tarea actualizada";
      }
    }
  }
  this.myValue = -1;
}
```

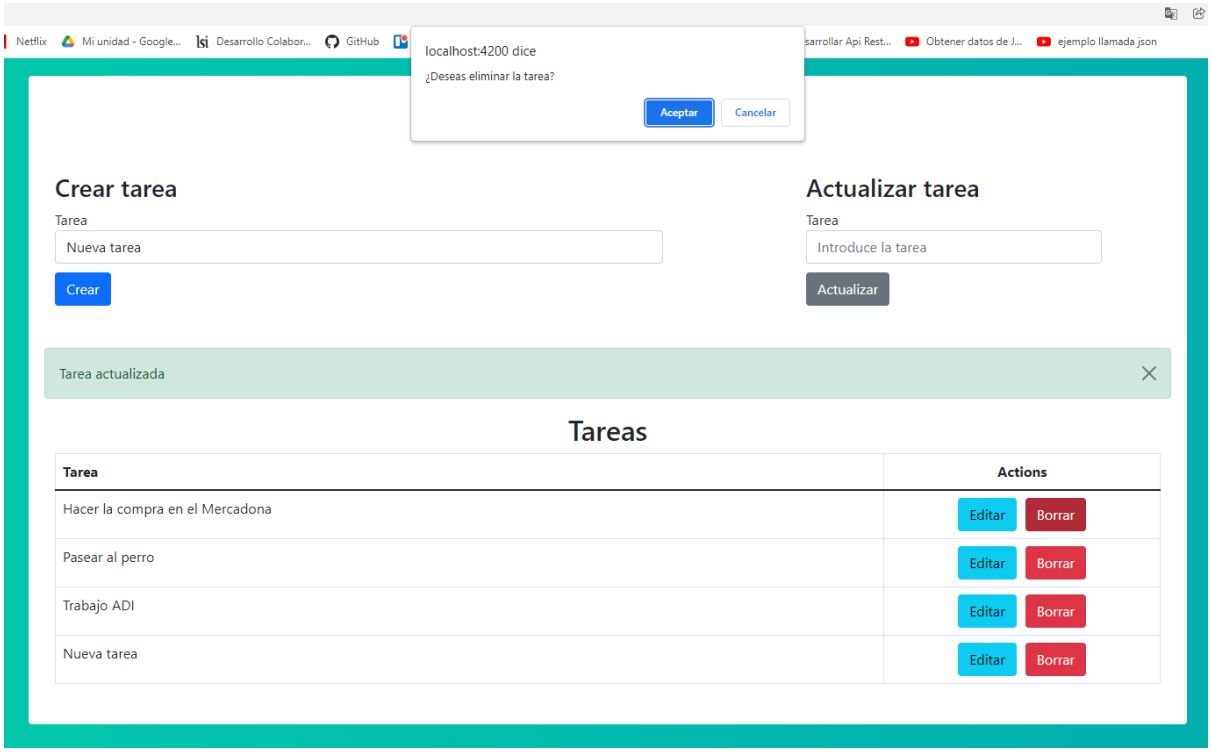
Aplicación

Finalmente vamos a ver una imagen de cómo vería el usuario la página inicial:

Al crear una tarea nos saldrá el siguiente mensaje:

Al modificar una tarea nos saldrá el mensaje correspondiente:

Cuando clickemos sobre Borrar saldrá un popUp con el mensaje de confirmación:



El mensaje que se mostrará al borrar una tarea será el siguiente:

Referencias

https://angular.io
https://desarrolloweb.com/home/angular
https://www.hostinger.es/tutoriales/que-es-angular
http://blog.enriqueoriol.com/2018/10/ciclos-de-vida-en-angular-la-guia-definitiva.html
https://kinsta.com/es/blog/php-vs-angular/
https://www.acontracorrientech.com/claves-para-entender-angular-que-es-y-como-se-utiliza/
https://www.campusmvp.es/recursos/post/las-5-principales-ventajas-de-usar-angular-para-crear-aplicaciones-web.aspx
https://www.typescriptlang.org