



TUTORIAL ANGULAR

Trabajo de Teoría ADI



14 DE ENERO DE 2021
UNIVERSIDAD DE ALICANTE

Contenido

¿Qué es Angular?	2
Características de Angular	2
Como Instalar Angular	3
NodeJS	3
Linux/OS X	3
Windows	3
Angular CLI	3
Creando nuestra primera aplicación Hello World	6
Estructura de un proyecto de Angular	8
E2E folder – (End To End)	8
Node-Modules	8
Src	9
App	9
Assets	9
Environments	10
Lista de la compra en Angular	11
Servidor	11
Cliente	12
Vista	17
Ejecución	18

¿Qué es Angular?

Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript.

Angular se basa en clases tipo "Componentes", cuyas propiedades son las usadas para hacer el binding de los datos. En dichas clases tenemos propiedades (variables) y métodos (funciones a llamar).

Angular es la evolución de AngularJS, aunque incompatible con su predecesor.

Angular permite la creación de aplicaciones para el lado del cliente en HTML, CSS y JavaScript/TypeScript.

Características de Angular

- Ofrece una estructura clara para el desarrollo de aplicaciones
- Incluye código reusable, lo que permite acelerar los procesos de desarrollo
- Permite crear tests para comprobar el correcto funcionamiento de la aplicación

Como Instalar Angular

Para empezar a instalar Angular, primero comenzaremos con unos requisitos previos para poder desarrollar nuestra aplicación de prueba.

NodeJS

Antes de comenzar con nuestra aplicación en Angular, deberemos instalar NodeJS.

Linux/OS X

Su instalación es realmente sencilla, para sistemas Unix Utilizaremos:

```
- curl -L https://git.io/n-install | bash
```

Windows

Para instalar NodeJS en Windows, iremos a su página web

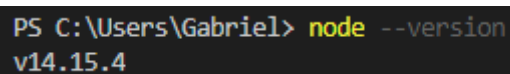
- <https://nodejs.org/es/download/>

Una vez dentro, seleccionaremos la versión que corresponda con nuestro sistema.

Para conocer la versión de NodeJS que tenemos en nuestro sistema, iremos a una terminal y pondremos

- node --version

En nuestro caso tendremos la siguiente salida



```
PS C:\Users\Gabriel> node --version
v14.15.4
```

Angular CLI

Lo siguiente que necesitaremos será Angular CLI, el cual lo utilizaremos para la creación de proyectos en Angular

CLI significa: Command-Line Interface

Para la instalación deberemos poner en la terminal:

```
- npm install -g @angular/cli
```

Para comprobar que la instalación ha sido correcta, podremos usar la siguiente línea

```
- ng --version
```

Si la instalación es correcta, tendremos lo siguiente en consola

```
Angular CLI
Angular CLI: 11.0.7
Node: 14.15.4
OS: win32 x64

Angular:
...
Ivy Workspace:

Package                          Version
-----
@angular-devkit/architect        0.1100.7 (cli-only)
@angular-devkit/core             11.0.7 (cli-only)
@angular-devkit/schematics       11.0.7 (cli-only)
@schematics/angular             11.0.7 (cli-only)
@schematics/update               0.1100.7 (cli-only)
```

Como se puede ver, nos muestra la versión de Angular CLI, la versión de Node y la del sistema Operativo que estamos usando.

Puede surgir que tengamos el siguiente problema a la hora de ejecutar Scripts dentro de nuestro sistema Windows:

```
PS C:\Users\Gabriel\OneDrive - UNIVERSIDAD ALICANTE\Ingenieria\2020-2021\1er Cuatrimestre\VDI\Teoria\Ejercicio Grupal - Angular Tutorial> ng --version
ng : No se puede cargar el archivo c:\Users\Gabriel\AppData\Local\Temp\ng-psl porque la ejecución de scripts está deshabilitada en este sistema. Para obtener más información, consulta el tema about_Execution_Policies en
https://go.microsoft.com/fwlink/?LinkID=135170.
En línea: 1 Carácter: 1
+ ng --version
+ ~~~~~
+ CategoryInfo          : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
```

Esto se debe a que la política de ejecución de scripts está deshabilitada, para habilitarla, simplemente ejecutaremos una ventana de Windows Powershell como administrador y pondremos lo siguiente:

```
PS C:\Windows\system32> Set-ExecutionPolicy RemoteSigned

Cambio de directiva de ejecución
La directiva de ejecución te ayuda a protegerte de scripts en los que no confías. Si cambias dicha directiva, podrías
exponerte a los riesgos de seguridad descritos en el tema de la Ayuda about_Execution_Policies en
https://go.microsoft.com/fwlink/?LinkID=135170. ¿Quieres cambiar la directiva de ejecución?
[S] Sí [O] Sí a todo [N] No [T] No a todo [U] Suspender [?] Ayuda (el valor predeterminado es "N"): o
PS C:\Windows\system32> Get-ExecutionPolicy -list

Scope ExecutionPolicy
-----
MachinePolicy Undefined
UserPolicy Undefined
Process Undefined
CurrentUser Undefined
LocalMachine RemoteSigned

PS C:\Windows\system32>
```

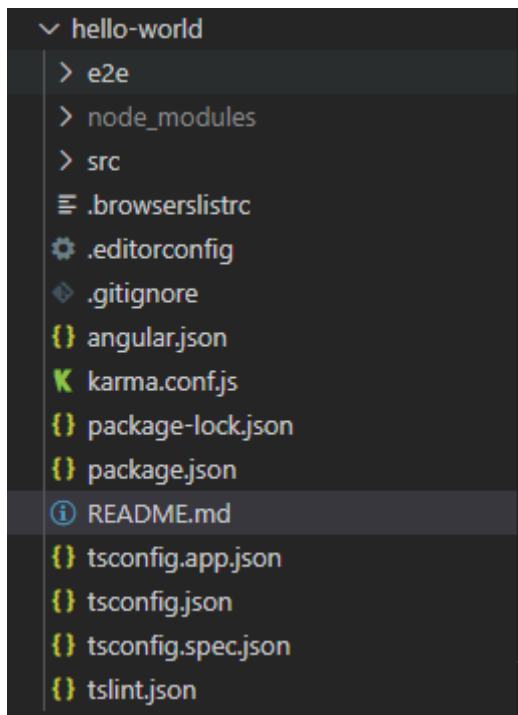
Una vez hecho esto, podremos crear nuestro primer proyecto de Angular

Creando nuestra primera aplicación Hello World

Para crear nuestra primera aplicación de Angular, teclearemos lo siguiente en una terminal

- `ng new hello-word`

Una vez hecho, tendremos lo siguiente



Una vez generado el proyecto, para arrancarlo haremos lo siguiente dentro de su carpeta:

- `ng serve`

Una vez compilado, tendremos lo siguiente en la terminal

```
PS C:\Users\Gabriel\OneDrive - UNIVERSIDAD ALICANTE\Ingenieria\2020-2021\1er Cuatrimestre\ADI\Teoria\Ejercicio Grupal - Angular Tutorial\hello-world> ng serve
✓ Browser application bundle generation complete.

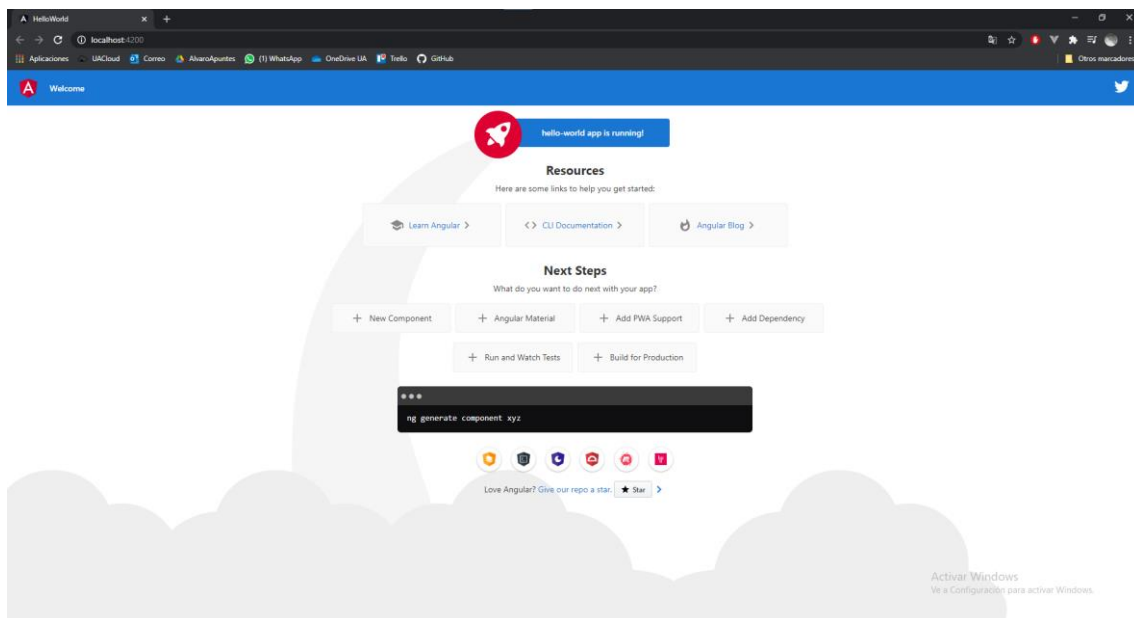
Initial Chunk Files | Names | Size
vendor.js           | vendor | 2.66 MB
polyfills.js        | polyfills | 484.82 kB
styles.css, styles.js | styles | 344.38 kB
main.js             | main | 59.76 kB
runtime.js          | runtime | 6.15 kB
                    | Initial Total | 3.54 MB

Build at: 2021-01-14T18:07:38.267Z - Hash: 33faf7543e70edc4a859 - Time: 4643ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
```

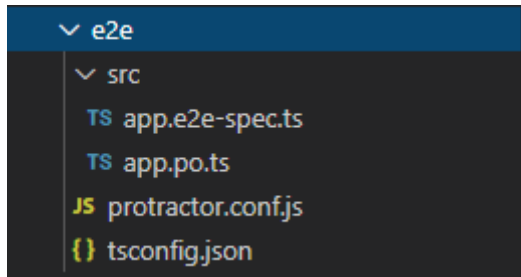
Como se puede ver, nos indica que nuestra aplicación se encontrará desplegada en localhost:4200, si vamos a dicha dirección en nuestro navegador:



Con esto, daremos por concluido el Arranque de una aplicación de Angular

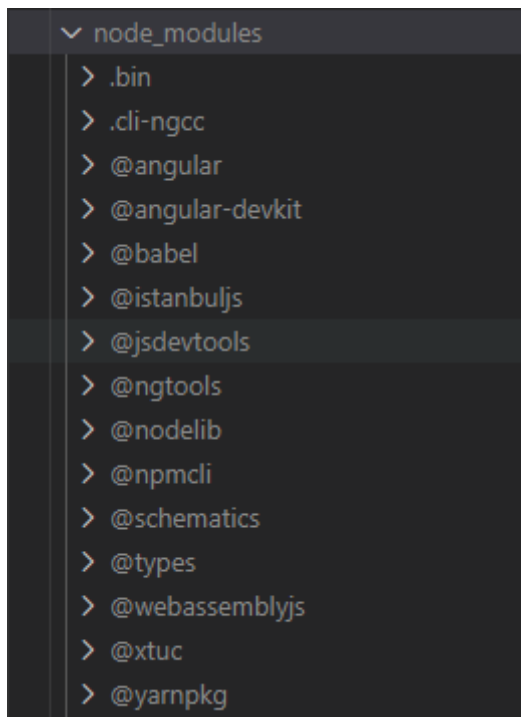
Estructura de un proyecto de Angular

A continuación, miraremos los directorios creados por Angular CLI recién creado el proyecto
E2E folder – (End To End)



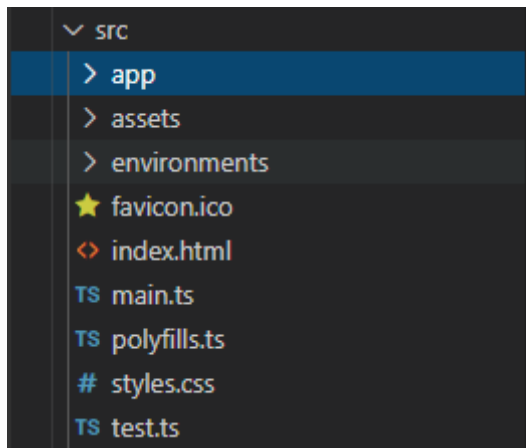
En esta carpeta se encuentra todo lo relacionado al E2E testing de la aplicación, que simula la creación de un usuario de la página y permite testear el correcto funcionamiento de esta. Para el enfoque de este tutorial no es de especial importancia, si se quiere saber más sobre End To End, se pueden consultar los siguientes vínculos:

Node-Modules



En esta carpeta se almacenan todas las librerías requeridas para la ejecución de nuestro proyecto.

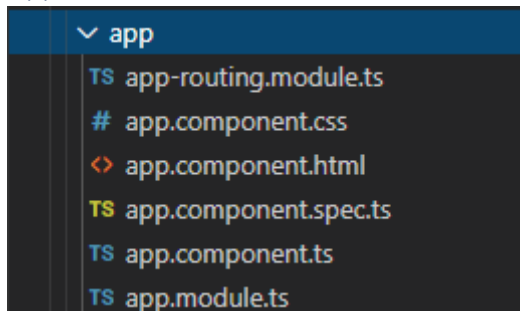
Src



Dentro de la carpeta SRC tendremos todo el código fuente relativo a nuestra página.

En esta carpeta, podremos encontrar las siguientes carpetas adicionales:

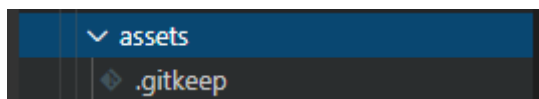
App



Dentro de la carpeta APP tendremos diferentes elementos:

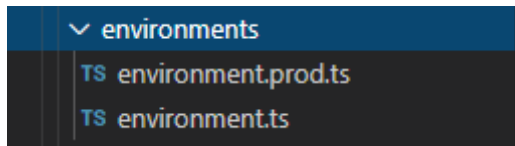
- En caso de que lo hayamos seleccionado, Angular CLI generará un router automático para nuestra aplicación el cual se encuentra dentro del fichero `app-routing.module.ts`
- Tendremos una suma de componentes y un módulo, por obligación, todas las aplicaciones que creemos tendrán, al menos, un módulo y un component

Assets



Dentro de la carpeta Assets, tendremos todos los elementos estáticos de nuestra aplicación, como por ejemplo imágenes.

Environments



Dentro de la carpeta Environments tendremos configuración referente al proyecto, para nuestro proyecto no será importante

Lista de la compra en Angular

El proyecto completo se puede encontrar y descargar en el siguiente enlace

<https://github.com/raymc94/ADI-Practica-Grupal-Teoria>

Como proyecto base sencillo, se ha decidido implementar una lista de la compra en Angular donde se podrán consultar, añadir y eliminar objetos que se enviarán a un servidor y que se mostrarán al cliente vía API-REST

Servidor

El servidor implementado almacena los datos de la siguiente manera:

```
var express = require("express");
var app = express();

var bp = require('body-parser');
var cors = require('cors');

var lista = new Map();
var itemIndex = 1;

app.use(bp.json());
app.use(cors());
```

El servidor se crea mediante una instancia de Express, y la lista de la compra consiste en un `Map` que contiene todos los datos, el índice de los objetos introducidos relativos al ID se almacena junto a una variable que va aumentando a medida que se introducen los ítems.

Para arrancar el servidor, será necesario realizar lo siguiente

- Npm i express
- Npm i cors
- Npm i body-parser

Una vez instaladas las dependencias, se lanza con:

- Node ./index.js

Cliente

El apartado del cliente es el que nos interesa, vamos a comenzar explicando lo que se ha introducido para la realización de la lista

Para empezar, crearemos el proyecto base de la misma manera que con el Hello World

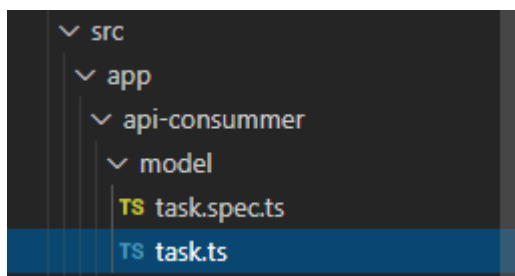
- Ng new adi-teoria-angular

Una vez creado todos los elementos base, continuaremos creando una clase para el modelo de la entidad con la que nos comunicaremos con la API, es decir, solo tendrá los atributos que tendrá tanto para el envío como recepción de datos con la API.

Para crear la clase utilizaremos:

- ng generate class api-consumer/model/task

Una vez creada la clase, la editaremos con los datos que necesitaremos , el archivo es el siguiente



Lo modificaremos para que contenga los datos de un elemento de la lista de la compra:

```
export class Task {
  itemIndex: string;
  item: string;

  constructor(itemIndex:"",item:""){
    this.itemIndex = itemIndex;
    this.item = item;
  }
}
```

El ítem será el ítem al que representa y el itemIndex será el Identificador de cada Objeto.

A continuación, deberemos instalar el siguiente paquete

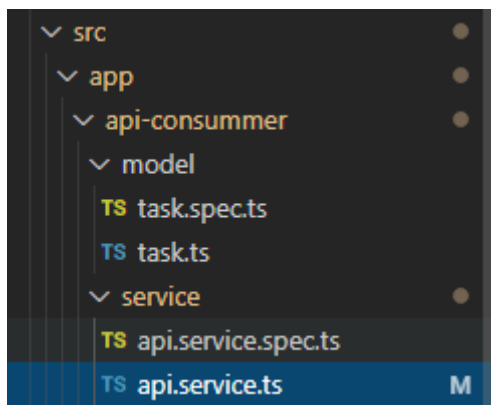
- npm install --save rxjs-compat

Este modulo sirve para el mapeo automático de una respuesta en JSON a nuestro objeto TASK, que representa los ítems de la lista

Una vez instalado, procederemos a crear el servicio consumidor de la API

- ng g service api-consumer/service/api

Ahora, con este servicio, modificaremos para poder interactuar con el servidor de la siguiente manera, el archivo es el siguiente:



Comenzaremos añadiendo los imports necesarios para el programa

```
import { Injectable } from '@angular/core';
import { Task } from "../model/task";
import { HttpClient } from "@angular/common/http";
import { Observable } from "rxjs";
```

- El import de TASK importa el objeto dentro del servicio
- HttpClient es el que se encarga de comunicarse con la API Rest mediante POST GET ...
- Observable es para poder mapear las respuestas en un objeto Task

Injectable es un import por defecto de Angular necesario para el funcionamiento

```
export class ApiService {

    private task: Task = { itemIndex: "", item: ""};
    private apiUrl = "http://localhost:9000"; // URL to web api
```

Dentro de la clase, implementaremos los Task (Objetos de la lista) y la URL a nuestro servidor donde se almacenan los objetos

```
    constructor(private http: HttpClient) { }

    public getTasks(): Observable<Task[]> {
        return this.http.get<Task[]>(this.apiUrl)
        //Esto es para controlar que existan errores, evitar que bloquee la pagina
        //.pipe()
    };

    public postTask(task: Task): Observable<Task> {
        return this.http.post<Task>(this.apiUrl, { item: task.item });
    }

    public deleteTask(itemIndex: string): Observable<Task> {
        return this.http.delete<Task>(this.apiUrl+"/"+itemIndex);
    }
}
```

Las funciones, en orden realizan lo siguiente:

- El constructor permite crear una instancia del servicio ApiService el cual ya preparará el elemento HttpClient para poder utilizarlo en nuestra llamada
- El GetTask obtiene del servidor todos los registros de tareas almacenados

- PostTask sirve para almacenar un ítem dentro del Servidor
- DeleteTask sirve para eliminar un elemento del servidor

El siguiente paso será modificar el archivo

TS app.component.ts

```
import { Component } from '@angular/core';
import { ApiService } from "../api-consumer/service/api.service";
import { Task } from "../api-consumer/model/task";
```

Añadiremos 2 imports:

- ApiService, que será para realizar las llamadas al servidor
- Task que es el modelo de los ítems de la lista y que se devolverán a la Vista

```
export class AppComponent {
  public task: Task = { itemIndex: "", item: "" };
  public tasks: Task[];
  public taskNameForm : string = "";
  title = 'adi-teoria-angular';
```

La variable TASK representa un solo ítem, mientras que TASKS representa un conjunto de estos.

TASKS se utilizará para la vista del cliente, mientras que TASK se utilizará para la comunicación con el servidor

TASKNAMEFORM es una variable que se vinculará a un Input de la Vista para poder dar de alta objetos a la lista

Aquí crearemos las variables necesarias para la comunicación, tanto con la Vista del cliente como con el Service

```
deleteTask(id : string):void {  
  console.log("Delete button is clicked. id" + id);  
  this.apiservice.deleteTask(id).subscribe();  
  window.location.reload();  
}  
  
addTask(taskName : string):void {  
  
  console.log("Add task button is clicked. name" + this.taskNameForm);  
  var taskAux : Task = {itemIndex:"", item : this.taskNameForm};  
  this.apiservice.postTask(taskAux).subscribe();  
  window.location.reload();  
}  
  
ngOnInit(): void {  
  this.apiservice.getTasks().subscribe((tasks: Task[]) => {this.tasks = tasks;});  
  console.log("Consumido");  
}
```

Las funciones son bastante auto explicativas, tanto la de delete como la de ADD, estas dos son funciones que se llamarán a través de eventos de clic en la vista

La función ngOnInit() es la función que se carga por defecto cuando se carga la página que consume este componente, en nuestro caso la raíz del proyecto

Vista

En el archivo :

`<> app.component.html`

Tendremos el HTML que se mostrará en la vista del cliente

```
<!-- Resources -->
<h2>Lista</h2>

<table *ngIf="tasks" id="customers">
  <thead class="thead-dark">
    <tr>
      <th scope="col">ID</th>
      <th scope="col">Item</th>
      <th scope="col">Accion</th>
    </tr>
  </thead>
  <tr *ngFor="let t of tasks">
    <td>{{ t.itemIndex }}</td>
    <td>{{ t.item }}</td>
    <td><button type="button" (click)="deleteTask(t.itemIndex)">Eliminar</button></td>
  </tr>
</table>
```

En la vista añadiremos un table para tratar todos los ítems Task recibidos de la API , tal y como se muestra en pantalla, añadimos una columna más con un button en el cual , con el clic llamaremos al componente encargado de el borrado de los ítems, pasándole como parámetro el ID del ítem.

Añadiremos también un Input que relacionaremos con el elemento TaskNameForm del componente

```
<div class="card-container">
  <div class="card card-small" (click)="selection.value = 'component'" tabindex="0">
    <input #taskNameInput type="text" id="taskName" name="taskName" (input)="taskNameForm = taskNameInput.value" />
  </div>

  <div class="card card-small" (click)="addTask('hola')" tabindex="0">
    <svg class="material-icons" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">
      <span>Nuevo Item</span>
    </div>
</div>
```

Por último, añadimos al div la propiedad de ser clickable y así comunicarse con la función `addTask` del componente, la cual se encargará de recoger el valor del `TaskNameForm` y dar de alta el objeto con ese nombre.

Con esto ya tendríamos preparado nuestro proyecto, se ha ignorado la parte de estilos de la página para evitar añadir información irrelevante

Ejecución

Para ejecutar el proyecto, en la carpeta del servidor-api haremos:

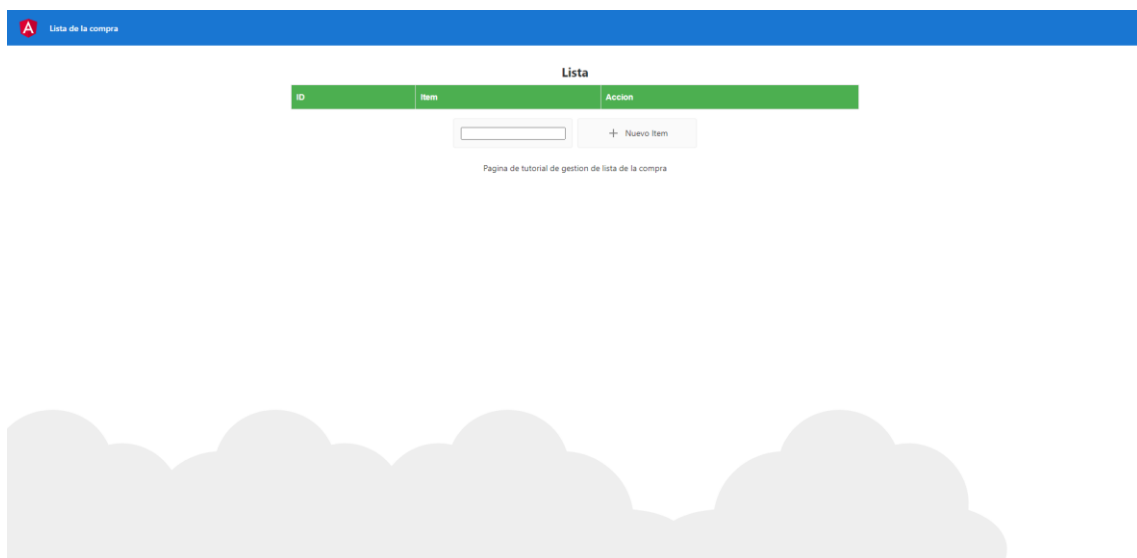
- `node ./index.js`

Se desplegará en el puerto 9000

El proyecto del cliente lo arrancaremos con

- `ng serve`

Se desplegará en el puerto 4200



Como se puede ver la lista inicial está vacía, a continuación, si añadimos un ítem:

A

Lista de la compra

Lista

ID	Item	Accion
4	Patatas	Eliminar

+ Nuevo Item

Página de tutorial de gestion de lista de la compra

Si clickamos en botón de eliminar

A

Lista de la compra

Lista

ID	Item	Accion
----	------	--------

+ Nuevo Item

Página de tutorial de gestion de lista de la compra

Se eliminará el ítem

Video de explicación

A continuación se encuentra un link de youtube que lleva al video explicativo del proyecto:

https://youtu.be/_Ee1iXiokHQ

Con esto concluye el tutorial, si se quiere aprender más de Angular, se recomiendan los siguientes enlaces:

<https://material.angular.io/components/categories>

<https://angular.io/cli>

<https://angular.io/start>

<https://angular.io/guide/event-binding>

<https://stackoverflow.com/questions/43572313/how-do-i-get-the-value-of-an-html-element-in-angular-2>

<https://stackoverflow.com/questions/59552387/how-to-reload-a-page-in-angular-8-the-proper-way>