



# Cocinando

Proyecto Fin de Ciclo  
“Desenvolvemiento de Aplicaci3ns Web”  
para IES San Clemente



Sandra M<sup>a</sup> Acha Nine

## [ Introducci3n ]

Aplicaci3n web donde se puede ver la carta de productos, sus datos y caracter3sticas.

En funci3n del tipo de login podremos a1adir unidades del producto deseado a un “carrito”, donde debe confirmar el pedido para que sea enviado a los empleados del restaurante para su elaboraci3n y entrega en la mesa del establecimiento.

Si el tipo de login es privado se podr3 consultar la lista de los pedidos y el detalle de estos.

# [ Requerimientos y Restricciones según Tipos de Usuarios ]

**-Usuario Público:** No necesita aportar datos, accederá a la web públicamente y tendrá acceso a toda la información sobre los productos exceptuando el precio de los mismos, únicamente para la consulta de la misma, en ningún momento podrá realizar pedidos.

**-Usuario con Ticket:** Debe informar el número de ticket y tendrá acceso a toda la información sobre productos y a la posibilidad de seleccionar elementos y realizar el pedido.

**-Usuario Privado:** Debe introducir sus credenciales y tendrá acceso al listado de pedidos y los productos seleccionados en cada uno de estos, no a la información de los productos.

# [ Tecnologías y Herramientas ]

-Utilizamos el framework **Angular** para aplicaciones web desarrollado en **TypeScript** (basado en Javascript), de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página basadas en el Modelo Vista Controlador.

-**AngularCLI** es la herramienta de línea de comandos para crear, depurar y publicar aplicaciones Angular.

-Como entorno en tiempo de ejecución multiplataforma **Node.js**

-**Html** como lenguaje de marcas

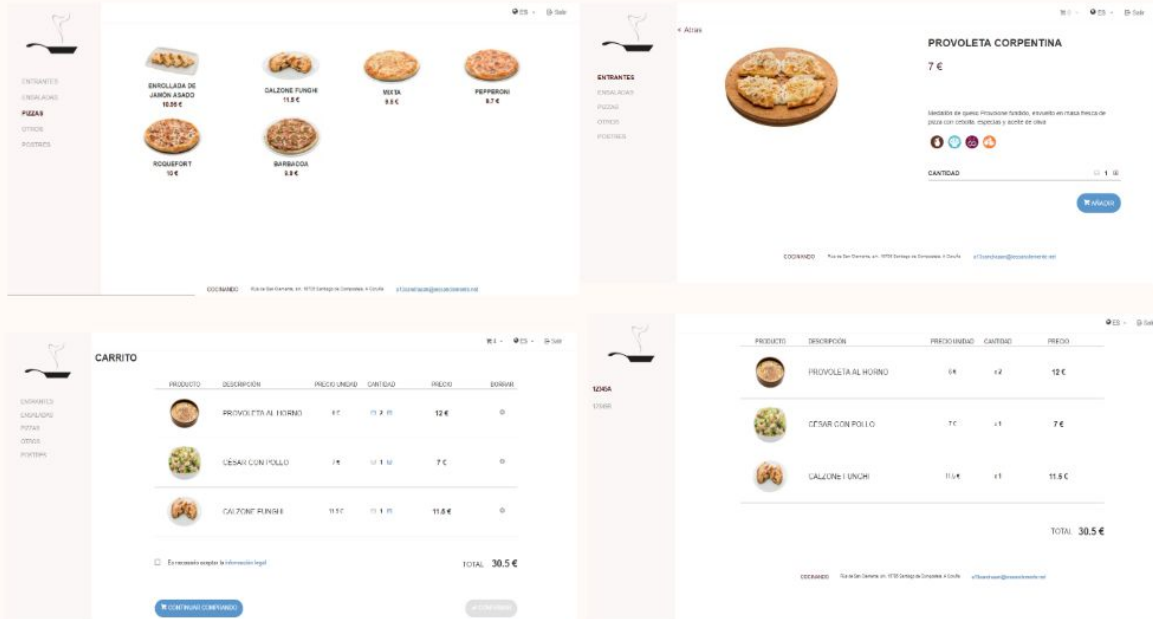
-**Sass** como lenguaje de hojas de estilo, que es un preprocesador de css con responsive web design

-**Bootstrap** biblioteca de plantillas de diseños basada en HTML, CSS y funcionalidades en Javascript.

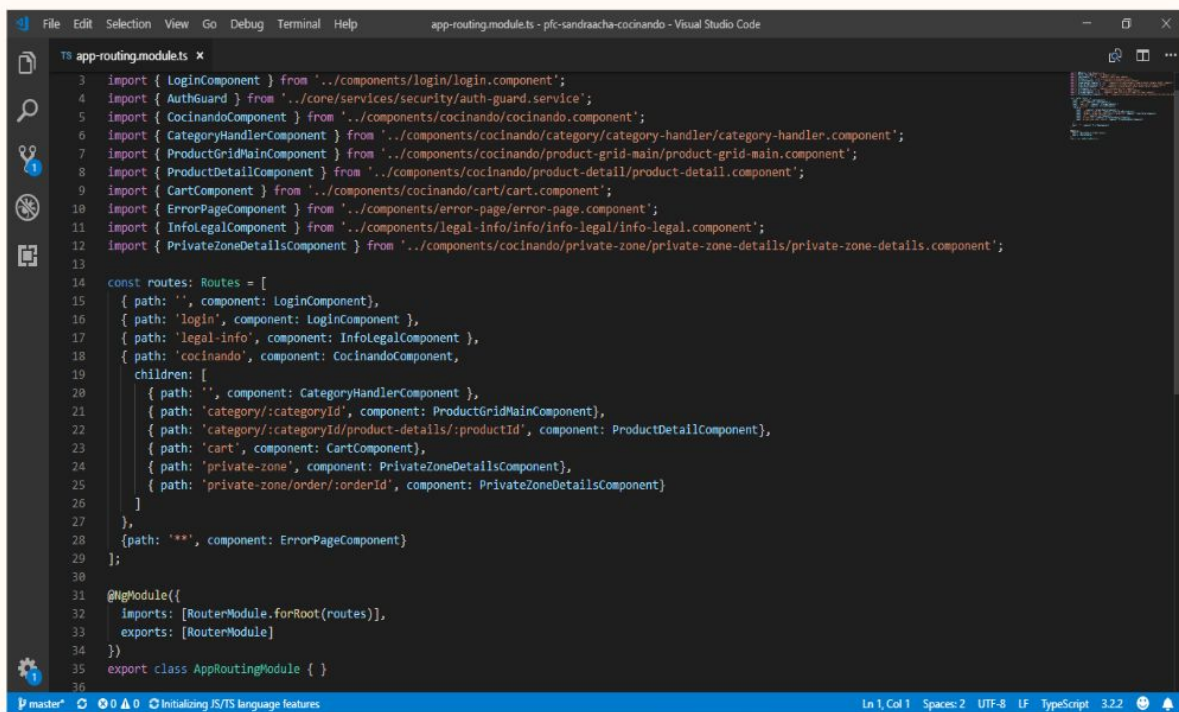
-Validación de formularios y datos mediante **Regex**, patrón que describe el formato de un texto

-**Visual Studio Code** editor de código fuente con soporte para la depuración, Git integrado...

# [ Pantallas “una sola página” ]

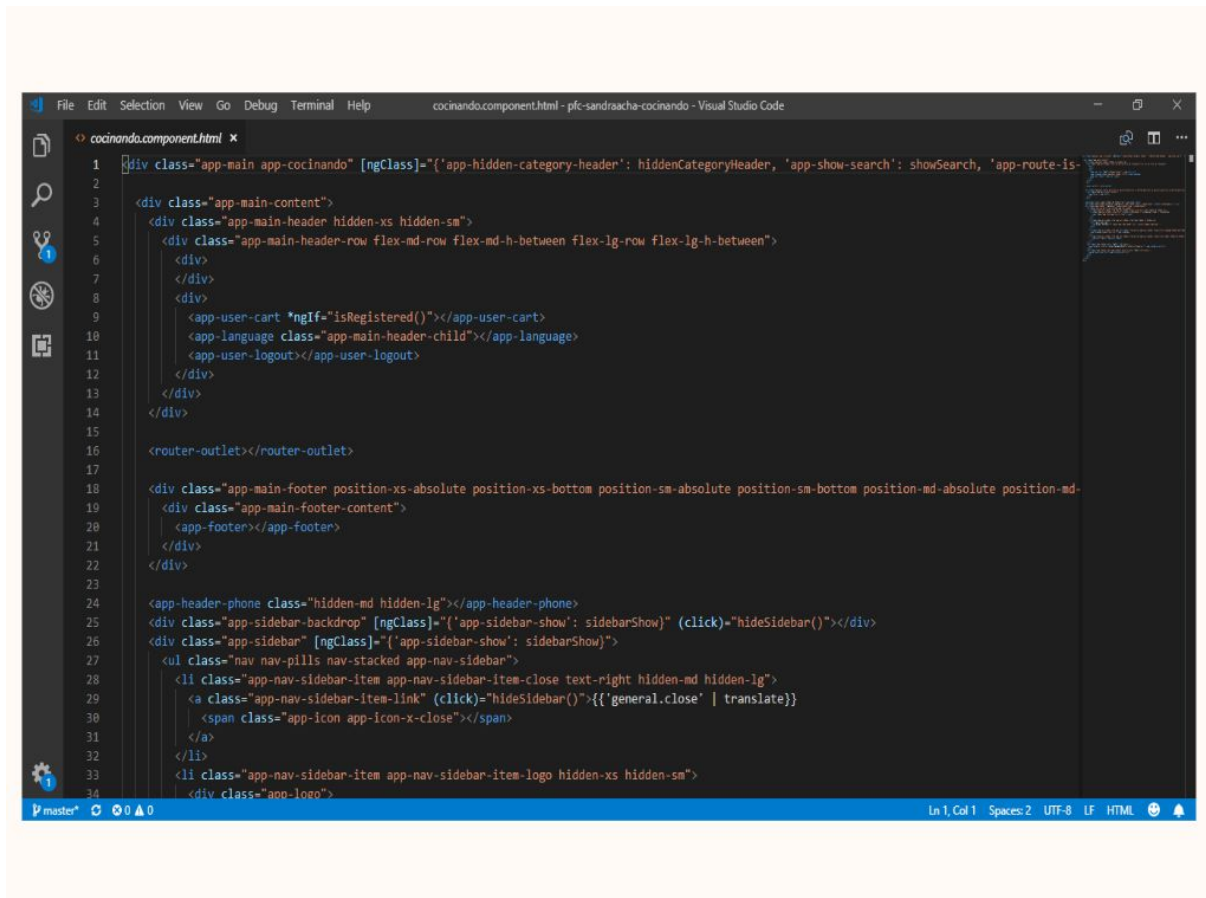


# [ Explicación “una sola página” ]



Esta es la estructura de urls de la aplicación, cada una tiene asignada su componente principal, como se puede apreciar la url cocinando tiene asignados “children”, esto es lo que hace que todas las páginas dentro de esta sección mantengan la estructura que hemos definido en el componente “Cocinando”, a diferencia de las otras que son independientes.

---



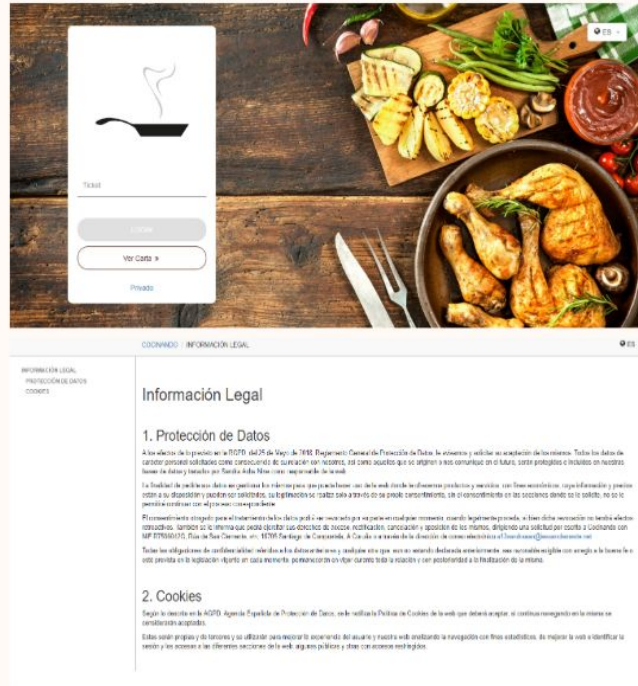
```
1 <div class="app-main app-cocinando" [ngClass]="{'app-hidden-category-header': hiddenCategoryHeader, 'app-show-search': showSearch, 'app-route-is-
2
3 <div class="app-main-content">
4   <div class="app-main-header hidden-xs hidden-sm">
5     <div class="app-main-header-row flex-md-row flex-md-h-between flex-lg-row flex-lg-h-between">
6       <div>
7     </div>
8     <div>
9       <app-user-cart *ngIf="isRegistered()"></app-user-cart>
10      <app-language class="app-main-header-child"></app-language>
11      <app-user-logout></app-user-logout>
12    </div>
13  </div>
14 </div>
15
16 <router-outlet></router-outlet>
17
18 <div class="app-main-footer position-xs-absolute position-xs-bottom position-sm-absolute position-sm-bottom position-md-absolute position-md-
19 <div class="app-main-footer-content">
20   <app-footer></app-footer>
21 </div>
22 </div>
23
24 <app-header-phone class="hidden-md hidden-lg"></app-header-phone>
25 <div class="app-sidebar-backdrop" [ngClass]="{'app-sidebar-show': sidebarShow}" (click)="hideSidebar()"></div>
26 <div class="app-sidebar" [ngClass]="{'app-sidebar-show': sidebarShow}">
27   <ul class="nav nav-pills nav-stacked app-nav-sidebar">
28     <li class="app-nav-sidebar-item app-nav-sidebar-item-close text-right hidden-md hidden-lg">
29       <a class="app-nav-sidebar-item-link" (click)="hideSidebar()">{{'general.close' | translate}}
30       <span class="app-icon app-icon-x-close"></span>
31     </a>
32   </li>
33   <li class="app-nav-sidebar-item app-nav-sidebar-item-logo hidden-xs hidden-sm">
34     <div class="app-logo">
```

Aquí pueden ver que montamos la estructura principal y tenemos el elemento `<router-outlet>` que es la parte de la pantalla que modificamos al movernos por las urls “children”.

---



# [ Pantallas Independientes ]



# [ Gestión de Imágenes y Fuentes ]

Las imágenes principales de la web se gestionan directamente como elementos en el DOM de tipo "img" y las almacenamos en "/assets/images", como el logo, la imagen de fondo del login y las imágenes de los productos.

El resto de imágenes se van a tratar como fuentes, ya que su control y estilos resulta más sencillo. La mayoría de ellas las obtendremos y gestionaremos con la web fontastic.

Las fuentes que no hemos podido conseguir de fontastic, como los alérgenos, se han buscado como svgs vectoriales para poder ser subidas y añadidas a nuestro paquete de fuentes de fontastic. En este caso ha sido necesario modificar dichos svgs ya que se encontraban mal formados y al ser añadidos a fontastic, no se visualizaban o transformaban correctamente, se han modificado con el programa "Adobe Illustrator".

- No pueden contener bordes
  - Se deben formar los grupos de trazos correctamente
  - Las secciones del icono que deben ser un único trazado se unifican
  - Las partes sin color deben ser transparentes por lo que deben ser bocados sobre las zonas de color.
  - El fondo del svg debe ser transparente
  - El tamaño de la imagen debe contener una distancia al borde del svg y tanto el espacio(mesa de trabajo) como la imagen interior deben ser siempre del mismo tamaño
- 

## [ Implementación ]

El proyecto se realizará implementando un acceso a datos doble, utilizando mocks con datos estáticos para la presentación de las vistas y la posibilidad de enlazar el proyecto con un backend programado con rests o microservicios para el acceso a la base de datos, abstrayendo la parte frontal de los datos a los que acceden y de funcionalidades adicionales irrelevantes para la web.

El usuario solo debe disponer de la url de la web y acceder a esta desde el navegador del dispositivo.

# [ Estructura Básica del Proyecto ]

Dentro de la carpeta principal podemos ver varios ficheros y carpetas de configuraciones pero nuestro código irá dentro de **"src"**.

- app**: La web en sí misma, las vistas y las funcionalidades.
- assets**: Recursos extra como fuentes, imágenes, mocks, scss...
- environments**: Ficheros de configuración de entornos.

En **"app"** creamos una estructura para separar los diferentes ficheros según las funcionalidades que desempeñan y las pantallas a las que pertenecen, y los ficheros principales de la aplicación encargados del flujo entre pantallas, idioma y comunicación entre todos los elementos de la estructura...

Angular nos crea el componente principal **"app.component"**, todos se formaran en el interior de este, es lo primero que se carga, aquí deben ir las configuraciones iniciales como el idioma por defecto.

# [ Estructura Interna "app" ]

-**components**: Controlan una pantalla o parte de una y su vista, con funcionalidades específicas. Se estructuran en cascada.

-**core**: Clases con propósitos específicos abstraídas de los componentes para fomentar la reutilización y modularidad.

-**directives**: Comportamientos definidos que podemos utilizar en todos los elementos DOM.

-**model**: Modelos de los objetos para mostrar y utilizar los datos en vistas.

-**pipes**: Funcionalidades específicas y muy concretas para utilizarse en toda la aplicación.

-**routes**: Fichero de rutas configuradas.

-**services**: Clases para el acceso a los datos de la aplicación, desde el servicio hasta el modelo pasando por las conversiones de los datos. Aquí controlamos a que datos accedemos, si mock o rest comprobando en la configuración del fichero "environment" correspondiente el atributo "use-mock".



## [ Política de Código ]

En un **"tslint.json"** configuramos las reglas de código, pueden ser preestablecidos o personalizadas.

Estas se aplican a los ficheros **".ts"**, podemos especificar el nivel de gravedad y configuraciones de cada una.

Las reglas disponibles, su descripción y configuraciones se pueden encontrar en <https://palantir.github.io/tslint/rules/>.

Algunas que hemos configurado:

**-curly:** Controla las llaves de las sentencias if, for...

**-deprecation:** Avisa cuando se está utilizando un API deprecada.

**-eofline:** Asegura que el archivo termina con una nueva línea.

**-forin:** Requiere que un for-in sea controlado con una sentencia if.

**-prefer-const:** Asegura que se utilicen declaraciones de variables como const en lugar de let y var cuando es posible.

## [ Pruebas ]

Todas las pruebas las realizaremos como **"Pruebas de Regresión"** de forma que al detectar un error o una mejora en la aplicación, lo corregiremos y volveremos a ejecutar todas las pruebas para ver si la aplicación responde correctamente y no hemos provocado ningún otro error o problema en su resolución.

Para realizar las pruebas únicamente será necesario iniciar la aplicación y acceder a ella desde el navegador, se realizarán en varios, donde mediante el inspeccionador del mismo podemos revisar lo que ocurre en todas las pantallas, las peticiones, los loggers, los puntos de parada en el código, activar el modo device y responsive permitiéndonos simular otros dispositivos y tamaños de pantalla.

Para las pruebas de usabilidad solicitaremos a otra persona que pruebe la aplicación y nos dé su opinión.



-Pruebas de Usabilidad: Comprobaremos el comportamiento como usuarios dentro de la aplicación web, si resulta fácil de usar, si se pierde en algún paso del flujo. En definitiva, si interactúa de manera fácil y sencilla con ella.

-Pruebas de Interfaz de usuario: Evaluamos el diseño de la aplicación web, el tamaño de las ventanas, tipos de letras, colores, ubicación de los controles, el responsive...

-Pruebas de Rendimiento: Evaluamos el tiempo de respuesta de nuestra aplicación web, de nuestros componentes y funcionalidades ante diferentes situaciones.

-Pruebas de Compatibilidad: Validamos que nuestra aplicación web funciona correctamente para distintos tipos de dispositivos (ordenador, tablet, móvil...) y navegadores web (firefox, chrome, internet explorer...).

---

# Conclusiones

Al inicio de un proyecto se debe realizar un análisis exhaustivo de los paquetes y librerías y que las versiones son compatibles y su estado.

Hay que evitar crear componentes que se vayan a utilizar a posteriori, saliendonos del flujo.

.Se debería reservar un tiempo extra al dar por finalizado un proyecto para modificaciones y mejoras.

El código debe ser lo más reutilizable posible.

Las cosas no siempre salen como uno espera, pero eso aumenta la experiencia, las capacidades...