

Práctica Frontend

DummyJSON

Get dummy/fake JSON data to use as placeholder in development or in prototype testing.

Recomendable la extensión de JSON Viewer si usas Google Chrome o Edge

Objetivo: Montar una página web con Angular que mapee ciertos recursos de la API de [DummyJSON](#).

Aprenderás a trabajar con **VS Code** y con las tecnologías de **TypeScript** y **Angular**. También conocerás como tratar con **JSON**, listados paginados, búsqueda de productos... entre otros. Del mismo modo, se deberá pensar en un diseño para enseñar toda la información requerida.

La **idea** inicial es similar a la de una tienda online, donde se gestionarán productos, usuarios y carritos.

Para el HTML se requiere hacer uso de etiquetas adecuadas [etiquetas HTML](#), no abusar del <div> [¡Deja de usar la etiqueta DIV para todo en HTML!](#)





Estructura del proyecto

El proyecto de Angular se estructurará de la siguiente forma (todo a continuación dentro de src):

- app
 - core
 - model – Mapeo de los objetos JSON
 - services – Servicios de angular
 - constants.ts – Fichero de constantes
 - modules
 - Todas las componentes irán bajo esta carpeta
- assets
- environments – carpeta especial donde se almacena la base-url
 - environment.prod.ts
 - environment.ts

✓ Crear la estructura del proyecto

Frameworks CSS (elegir uno)

-  [Bootstrap](#)
-  [TailwindCSS](#)
-  [Angular Material](#)
-  [PrimeNG](#)

- ✓ Elegir un framework CSS

¿Has pensado como maquetar la página web?

Sección de Productos

Primero hay que centrarse en mapear correctamente el JSON y guardarlo en la carpeta *model* del proyecto. Usaremos una interfaz de TS para apoyarnos en la creación de objetos:

Click derecho en la carpeta → New File → xyz.model.ts

Siendo xyz el nombre del objeto (por ejemplo, *product.model.ts*). Nos podemos apoyar en una herramienta online muy útil: [json-to-typescript](#)

En esta sección se pide:

- **Listado de todos los productos paginados**
 - Enseñar todos los productos. Mostrar el título, descripción, precio, marca, categoría y una imagen.
 - Para el paginado hay que añadir los parámetros *limit* y *skip*.
 - También se pide un buscador por título del producto.
- **Detalle de un producto completo**
 - Un detalle es mostrar toda la información del objeto.
- **Operaciones de añadir, actualizar y eliminar.**
 - A tener en cuenta, estas acciones no modifican el servidor como tal, si no que devuelven un objeto modificado con un flag distintivo.

Observaciones: Las acciones de añadir, actualizar y mostrar detalle pueden ir en la misma pantalla, porque contienen la misma información.

- ✓ Crear una página de productos completa y funcional.

Sección de usuarios

Se seguirá un proceso similar a la sección de productos. Para esta sección no se requiere mostrar toda la información del usuario, si no que una cierta parte, ya que hay demasiada información.

Se pide:

- **Listado de todos los usuarios paginados**
 - Enseñar todos los usuarios. Mostrar el primer nombre, apellido, edad, email, usuario, fecha de nacimiento y foto de perfil.
 - Para el paginado hay que añadir los parámetros *limit* y *skip*.
 - Buscador de usuarios.
 - Filtro de usuarios
- **Detalle de un usuario completo**
 - Mostrar la información principal del usuario (elegir al menos 10 campos)
- **Operaciones de añadir, actualizar y eliminar.**
 - A tener en cuenta, estas acciones no modifican el servidor como tal, si no que devuelven un objeto modificado con un flag distintivo.

✓ Crear una página de usuarios completa y funcional.

Sección de carrito

Esta sección simula el carrito de la compra de un determinado usuario.

Se pide:

- **Listado de todos los carritos paginados**
- **Buscador de un carrito a partir de un usuario.**
- **Operaciones de añadir, actualizar y eliminar.**
 - A tener en cuenta, estas acciones no modifican el servidor como tal, si no que devuelven un objeto modificado con un flag distintivo.

✓ Crear una sección de productos completa y funcional.