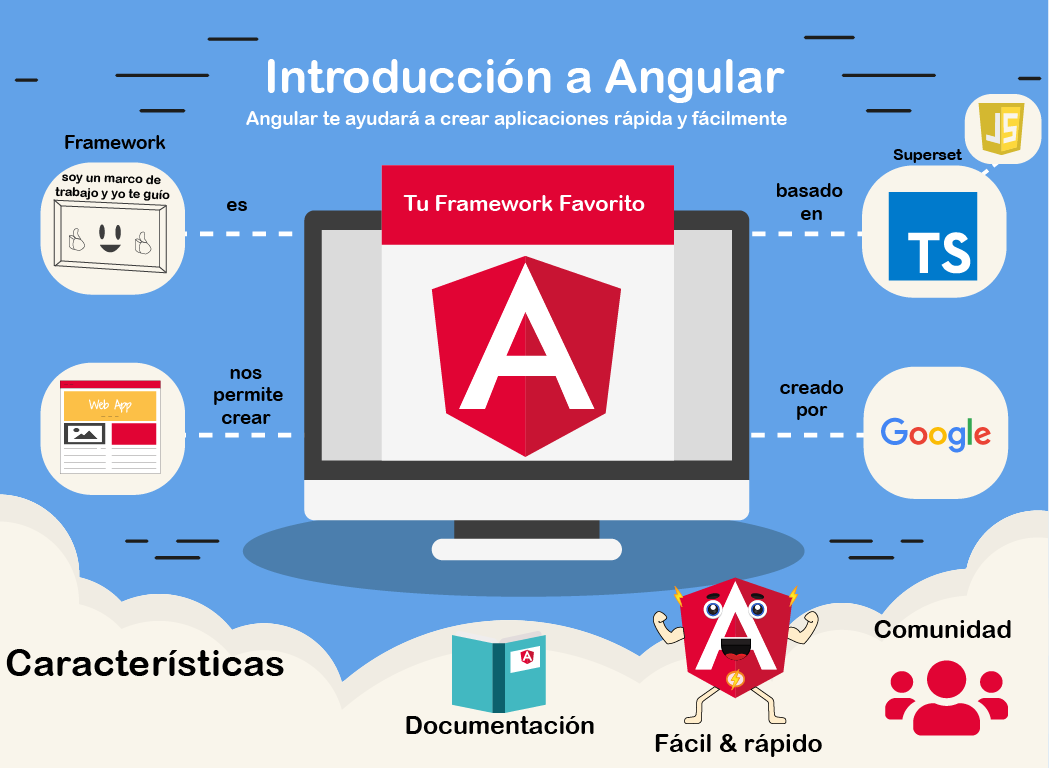
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

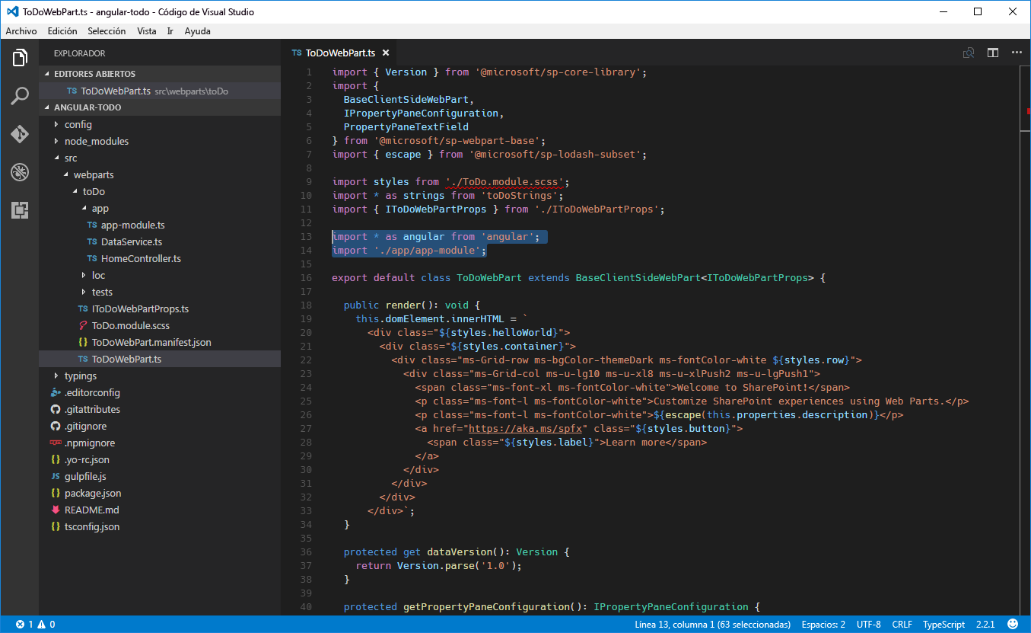
**Curso de Angular**Claudio Ortiz Castillo **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



**Índice:**

# Introducción:

**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).



# Conceptos generales para empezar con Angular y TypeScript:

* Generalidades de TypeScript  
    
  TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases. Anders Hejlsberg, diseñador de C# y creador de Delphi y Turbo Pascal, ha trabajado en el desarrollo de TypeScript.1​ TypeScript es usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor, o extensiones para programas (Node.js y Deno).

Extiende la sintaxis de JavaScript, por tanto, cualquier código JavaScript existente debería funcionar sin problemas. Está pensado para grandes proyectos, los cuales a través de un compilador de TypeScript se traducen a código JavaScript original.

TypeScript soporta ficheros de definición que contengan información sobre los tipos de librerías JavaScript existentes, similares a los ficheros de cabeceras de C/C++ que describen la estructura de ficheros de objetos existentes. Esto permite a otros programas usar los valores definidos en los ficheros como si fueran entidades TypeScript de tipado estático. Existen cabeceras para librerías populares como jQuery, MongoDB y D3.js, y los módulos básicos de Node.js.

El compilador de TypeScript está escrito asimismo en TypeScript, compilado a JavaScript y con Licencia Apache 2.

TypeScript se incluye como lenguaje de programación de primer nivel en Microsoft Visual Studio 2013 Update 2 y posteriores, junto a C# y otros lenguajes de Microsoft. Una extensión oficial permite a Visual Studio 2012 soportar también TypeScript.

* ¿Por qué Angular usa TypeScript?
* Mitos y realidades de Angular

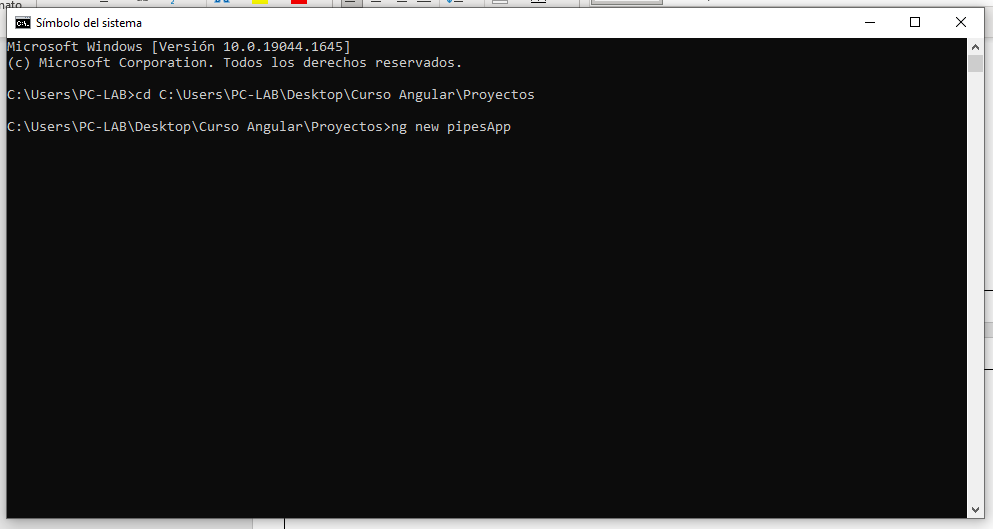
# Base de TypeScript

* Tipos básicos
* Objetos, arreglos e interfaces
* Funciones y sus argumentos
* Desestructuración de arreglos y objetos
* Importaciones y exportaciones
* Clases, constructores
* Tipos genéricos
* Decoradores
* Encadenamiento opcional

# Introducción a Angular

* Crear proyectos de Angular:

Debemos ir a la terminal cmd y colocar:



Cd C:\Users\PC-LAB\Desktop\Curso Angular\Proyectos

ng new pipesApp

* Explicar cada archivo y directorio de un proyecto
* Componentes
* Módulos
* One way data binding
* Uso del AngularCLI - Angular Command Line Interface
* Directivas creadas por Angular
* ngIf y ngIf-else
* Y más...

# Expandiendo nuestras bases

* Profundizar un poco más en los módulos
* FormsModule
* ngModel
* @Inputs
* @outputs
* Servicios
* Métodos en servicios
* Depuraciones

# Despliegues rápidos a producción

Este es un breve listado de los temas fundamentales:

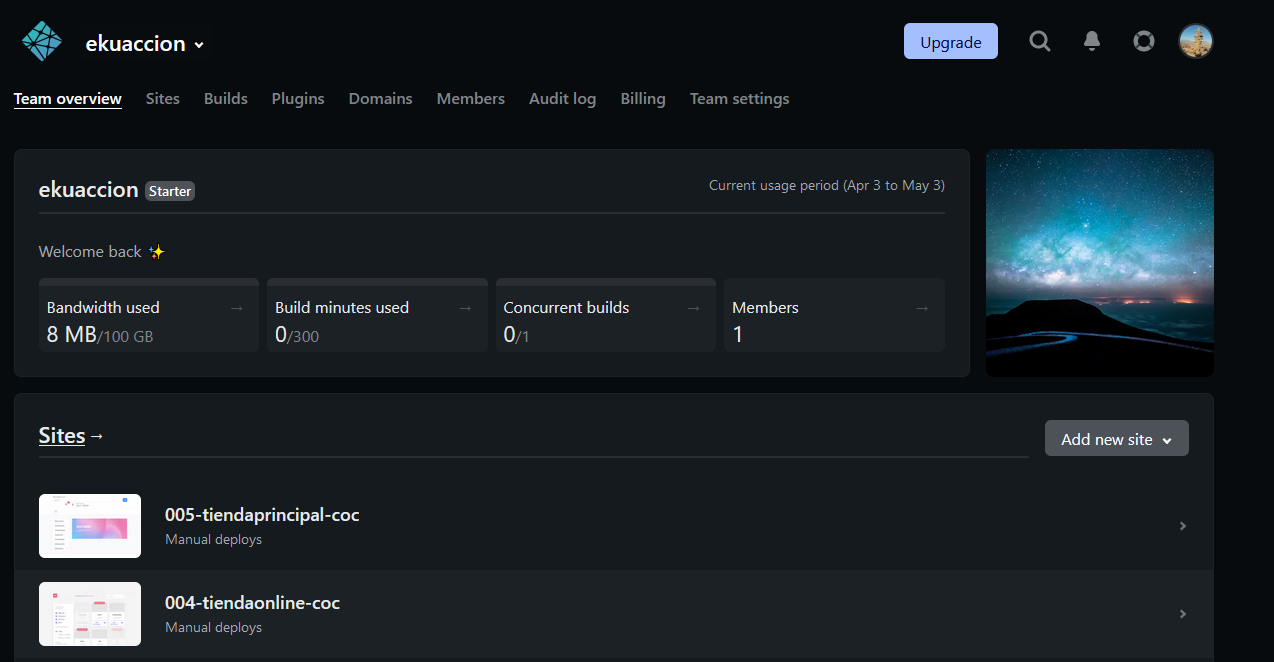
* Generar build de producción

**Cd directorio de proyecto**

* Desplegarlo rápidamente

**ng build --prod**

* Netlify: subir página …



Aquí aprenderemos como generar el build de producción de nuestra aplicación y la desplegaremos en la web rápidamente usando Netlify, el proceso de despliegue en otros servidores es virtualmente el mismo, tomar nuestra carpeta DIST (que contiene la aplicación con archivos HTML, CSS y JS) y desplegarla mediante FTP (preferiblemente sFTP) en el hosting deseado.

# GifsApp – app para buscar imágenes

# SPA – PaisesApp

# Continuación aplicación de Países – Sugerencias, debounce y más

Aquí continuaremos la aplicación de países, pero enfocados en la parte de las sugerencias y auto-completado, este es un breve listado de los temas fundamentales:

* ngClass y diferentes formas de manipular clases
* CSS condicionales
* Optimizaciones a peticiones HTTP
* Animaciones de CSS
* Auto-completado en la búsqueda de países

No es una sección muy larga, pero dejaremos las bases de cómo funciona un auto-completado en general.

# Pipes de Angular

Esta sección es de mis favoritas del curso, veremos temas sobre Pipes y transformación visual de la data, pero también decidí mezclarlo con otro tema muy solicitado que es PrimeNG (Más adelante hay también una sección con muchos componentes de Angular Material), pero esta sección pretende enseñarles muchas cosas interesantes, aparte de los Pipes de Angular

Los pipes son trasformaciones ala visual de los datos sin afectar su lógica y nunca modifican al data con la que están trabajando.

Todos los Pipes de Angular a la fecha:

<https://angular.io/api?query=pipe>

* Uppercase Lowercase
* TitleCase: coloca en modo título el texto.
* Date: cambia el formato de la fecha.
* Decimal: cambia el formato de un número.
* Currency: modo de vizualizción como moneda.
* Percent: trabaja como porcentajes los decimales.
* i18nSelect: arregla el género de la frase.
* i18Plural: arregla el numero referenciado de la frase o texto.
* SlicePipe: se utiliza para cortar el arreglo que quiero mostrar.
* KeyValue: se ocupa para recorrer un arreglo y poder mostrarlo en angular.
* Json: pipe útil para hacer depuraciones. Si el objeto tiene funciones que imprimen otros objetos. En ese caso marca error por no poder imprimir un objeto.
* Async: necesita un observable o una promesa.
* PrimeNG: es una librería completa que tiene componentes prefabricados.  
    
  <https://www.primefaces.org/primeng/setup>

npm install primeng --save

Agregamos estilos a través del archivo general css. Con variables alojadas en el root.

* Módulo especializado para módulos de PrimeNG

Creamos un modulo tonto donde contener y exportar todos estos módulos de prime.

PrimeNG es un paquete de componentes estilizados que contienen funcionalidades basadas en Angular para crear aplicaciones elegantes, hermosas y funcionales rápidamente.

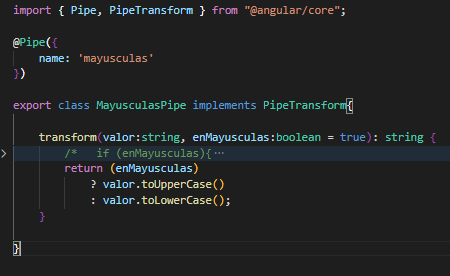
Recuerden que PrimeNG no es el único, hay muchos, pero la mayoría funcionan de formas similares, importando módulos, por eso mi objetivo de entrenarlos bastante bien con el manejo de módulos en todo momento para que nos familiaricemos con esta metodología.

# Pipes Personalizados

* Pipes personalizados: son pipes creados de forma particular con el fin de mejorar nuestra app. La creación se puede hacer de forma automática de la siguiente forma:

ng g pipe ventas/pipes/ordenar --skipTests

* Argumentos hacia los Pipes. Los argumentos son mas aplicaciones de nuestro pipe.



* Toolbar Table manual y usando PrimeNg

Es una sección pequeña, pero sumamente ilustrativa y necesaria para poder expandir todos los pipes que no existen

# HeroesApp – Rutas hijas y Lazyload

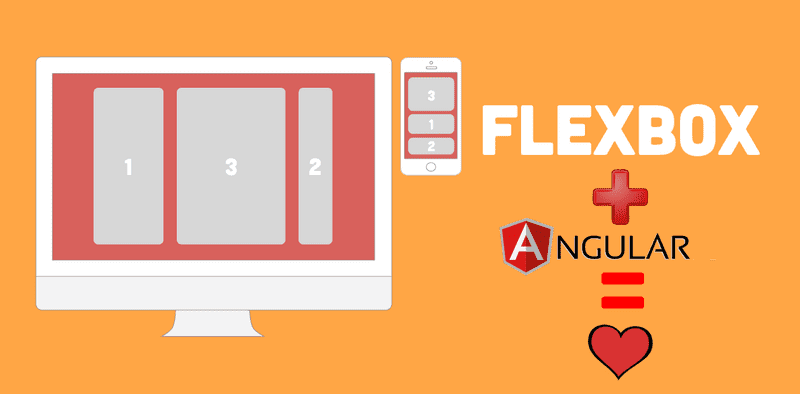
* Rutas Hijas: o children, se hace un routing por cada módulo.
* Rutas Principales: son las rutas mas generales del proyecto, ej: ‘404’
* LazyLoad: El scriptLazyLoad, mencionado más arriba, ofrece una solución compacta que no requiere jQuery. Además de la capacidad de configurar la carga diferida nativa para imágenes, videos e iFrames, este script de 2,4 KB (kilobytes) admite imágenes responsivas, entre otros elementos. Tambienj conocido como carga perezosa.
* Múltiples estilos en la misma SPA: par esto generamos un router out para cada módulo.

Esta sección es fundamental para seguir el curso, ya que, de aquí en adelante, implementaremos la carga perezosa en cada módulo principal de las futuras aplicaciones que haremos.

Comprender el Lazyload no es difícil, y la forma como lo veremos aquí nos ayudará a manejar los dos posibles casos de uso. Uno de ellos es cuando la ruta hija no tiene estilo especial y otra en la que requiere un estilo diferente por cada módulo.

# Angular Material y Flex

Angular Flex:



Backend: Json server. Al cual accederemos con servicios Rest.

<https://www.npmjs.com/package/json-server>

<https://github.com/typicode/json-server>

* Angular Material
* Interfaces y tipado
* Pipes personalizados
* Variables de entorno
* Autocomplete de AngularMaterial
* Peticiones HTTP
* JSON-Server
* Angular Flex y Flexbox
* Y más...

Esta sección tiene por objetivo principal aprender a utilizar Angular Material, es la primera de varias secciones donde lo usaremos. Al final del día trabajaremos con muchos componentes de Angular material que les ayudará a ver cualquier otro tipo de paquete modularizado de la misma manera y ustedes sabrán aplicarlo.

En la siguiente sección después de esta, continuaremos la aplicación, pero realizaremos un CRUD.

# HerosApp – CRUD (continuación con angular material)

Este es un breve listado de los temas fundamentales:

* CRUD
  + Create
  + Read
  + Update
  + Delete
* Pipes puros e impuros: la diferencia esta en que uno se dispara constantemente y le otro solo cuando se actualiza la variable.
* Snacks
* Dialogs

Inyección de servicios manualmente

Esta sección les dará las bases para poder realizar CRUD completos hacia cualquier backend basado en servicios web, mediante comunicación JSON

# Protección de Rutas

Este es un breve listado de los temas fundamentales:

* Protección de rutas
* Rutas privadas
* Rutas públicas
* Servicio de autenticación
* Guards
* CanActivate
* Can Load
* Mantener la sesión del usuario

Esta es una sección muy importante, donde controlaremos de forma básica una autenticación, mucho más adelante en el curso, realizaremos autenticación mediante JWT, pero antes de llegar a eso que son temas de Backend, necesito que comprendamos cómo Angular nos puede servir para proteger nuestras rutas.

# Formularios - Template y Lazload

Template driven

ViewChild

Two way databinding

Formularios dinámicos

Checks, radios y switches

Directivas personalizadas - Nota: este tema se cubre a profundidad después en una sección especializada

Manejo del formulario y validaciones

Encapsular módulos y scope de los mismos

Este es un tema fundamental, pero no es el único, hay varias formas de manejar el estado de un formulario, principalmente tenemos Template driven y Model Driven o formularios reactivos, en esta sección trabajaremos con formularios por template.

1. **Formularios Reactivos**
2. Backend

Npm init

Node index.js

npm install -g nodemon

nodemon index.js

1. Otros:

<https://www.chakray.com/es/que-diferencias-hay-entre-rest-y-soap/>