Visualización de Datos

Con D3.js y Angular.js

Autor: Alejandro Cortés Cabrejas

Tutor: Santi Seguí

Julio 2015, Barcelona

Índice

- Introducción
- Planteamiento
- Funcionamiento de la aplicación
  - Obtención DataSets
  - Diseño DashBoard
  - Configuración Angular
  - Lógica de la aplicación
  - Diseño directivas
  - Incorporar directivas al DashBoard
- Resultados
- Conclusiones y opinión personal

Universitat de Barcelona

Introducción (I)

La cantidad de datos que se generan a nuestro alrededor ha aumentado con las mejoras en tecnologías.

- **Data science.** Estudia la capacidad de extraer información de grandes cantidades de datos.
- **Visualización de datos.** Uso de representaciones visuales para:
  - Facilitar la extracción de la información.
  - Mostrar resultados.

Universitat de Barcelona

Planteamiento (I)

Aplicación web de visualización de Datos:

- Conjunto de gráficas dinámicas.
- Estudio de la desigualdad laboral de género.
- Uso de D3.js.
- Interacción entre el usuario y las representaciones.

Universitat de Barcelona

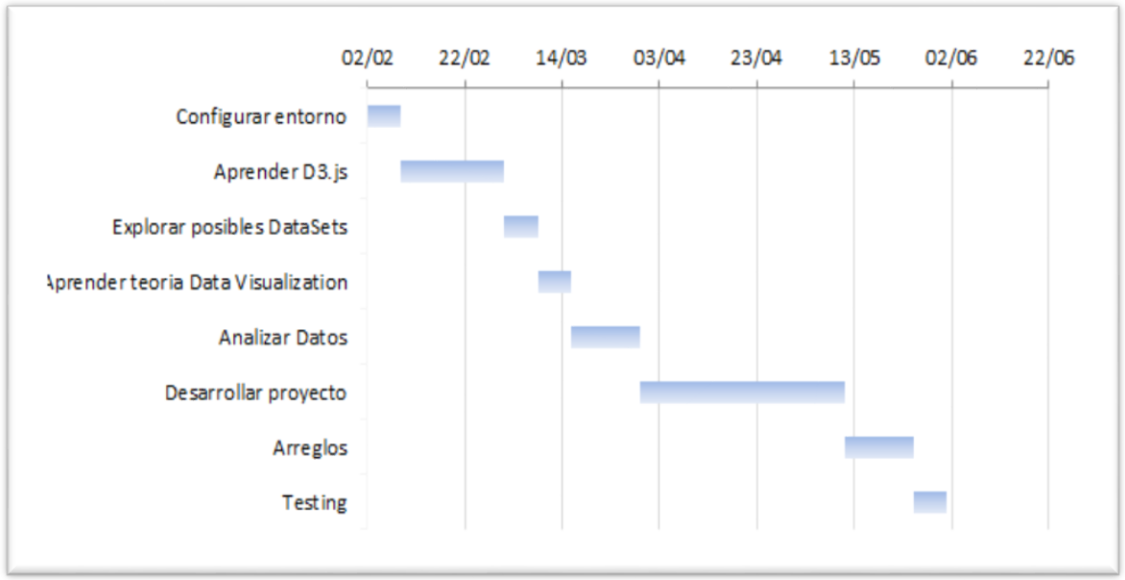
Planteamiento (II)

¿Por qué realizar este proyecto?

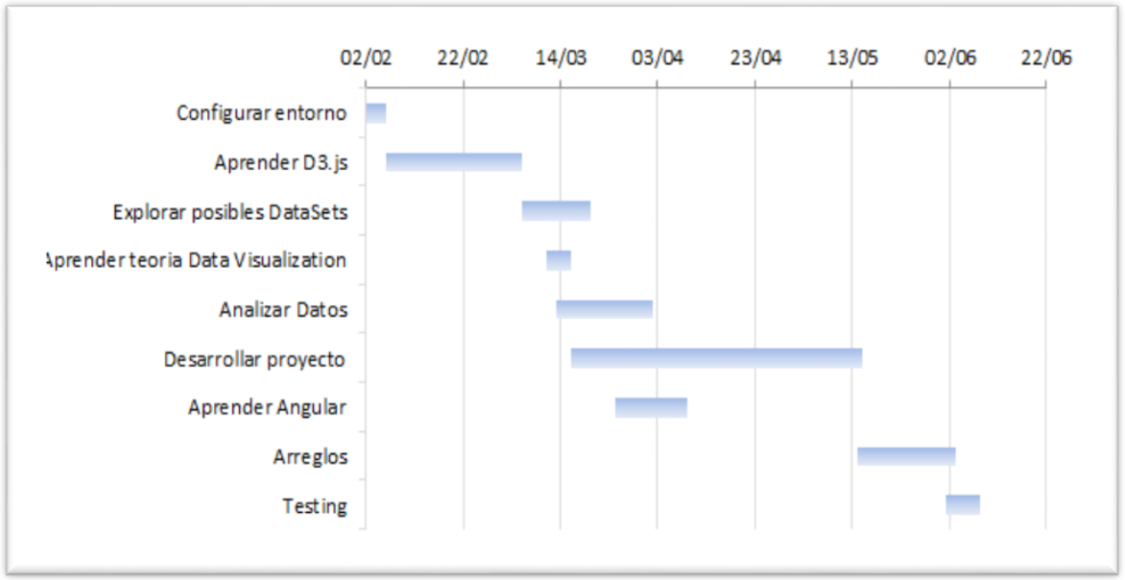
- Campo de Visualización de Datos → Tecnologías de implementación en web.
- La importancia de la transmisión de información a terceros.
- Influencia visual en la percepción de los datos.
- Aplicar técnicas de visualización a un caso de estudio concreto.

Universitat de Barcelona

Planteamiento (III)



Inicial



Final

Universitat de Barcelona

Funcionamiento de la aplicación (I)

DashBoard de Visualización de Datos:

- Tecnologías básicas web (HTML, CSS y JS).
- Uso de D3.js para dibujar las gráficas dinámicas en formato SVG.

Durante el desarrollo de la aplicación:

- Uso de Bootstrap para la interfaz.
  - **R**esponsive **W**eb **D**esign (RWD).
- Observamos la necesidad de que las gráficas interactúen entre si:
  - Incorporación Angular.js → Nos permite tener variables que afectan a múltiples gráficas y automatiza la propagación de cambios.

Universitat de Barcelona

Funcionamiento de la aplicación (I)

Angular.js

- Framework basado en JavaScript.
- Aplicación *front-end* con arquitectura **M**odel-**V**iew-**C**ontroller(MVC).
- Interpretación de etiquetas incrustadas en el HTML.

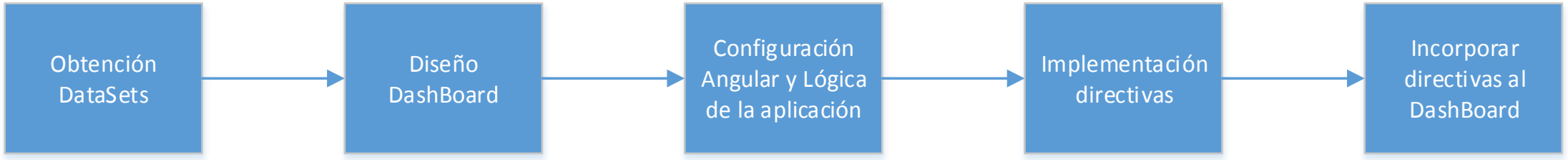
D3.js

- Librería en lenguaje JavaScript.
- Creación y personalización de elementos en formato SVG.
- Vincular datos a los elementos del DOM.

Funcionamiento de la aplicación (II)

¿Cómo se ha creado la aplicación?

Podemos fraccionar su elaboración en 5 etapas:



Universitat de Barcelona

Obtención DataSets

Se utilizan un conjunto de **DataSets** en formato **JSON** proveídos por **Eurostat** que abarcan las siguientes áreas:

- Educación.
- Demográficos.
- Gasto en educación.
- Datos de empleo.
- PIB.
- Salarios.

Fichero en formato **TopoJSON**

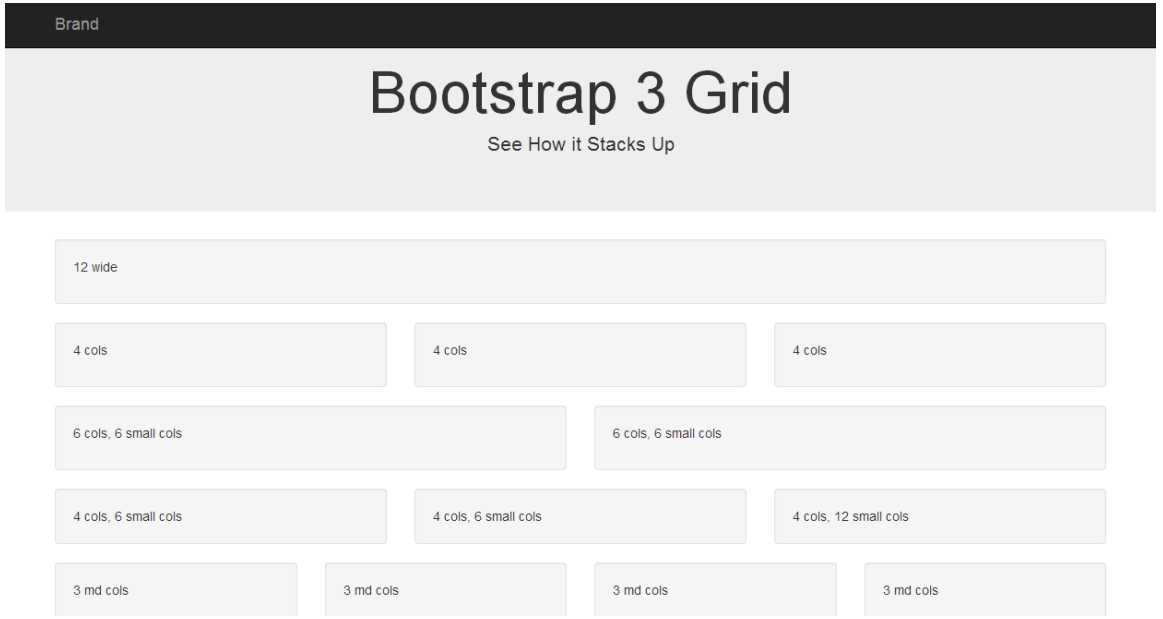
- Coordenadas de geolocalización → Representar un mapa de la Unión Europea en formato SVG con D3

Universitat de Barcelona

Diseño DashBoard

Uso de Bootstrap para el diseño del **DashBoard**:

- Sistema Grid para organizar el contenido.
- Estructurado en paneles.
  - Cada panel contendrá una gráfica.
  - Botón de inicio de transición.
- Navbar lateral → Interacción con las gráficas.



Universitat de Barcelona

Configuración Angular

Usamos tres tipos de componentes:

- **Servicios**.
- **Controllers.**
- **Directivas**.

Estos componentes se estructuran en **módulos** (definidos en ficheros de extensión JavaScript) que deben ser importados por el fichero HTML.

Para que la aplicación detecte el uso de Angular se debe incorporar la siguiente etiqueta al body

```
<body ng-app="visualDataApp">
```

Universitat de Barcelona

Lógica de la aplicación

¿Cómo definimos la inicialización de la aplicación y su funcionamiento?

Componente Controller.

- Un controller para todo el DashBoard.
- Se ejecuta al inicio.
- Inicializa las variables globales.
- Inyección servicios.
- $scope
  - Elemento del DOM asociado al componente.
  - Permite guardar variables y objetos .
  - Accesible desde las jerarquías inferiores.
- Requiere el uso de la siguiente etiqueta.

```
<div id="wrapper" ng-controller="mainCtrl">
```

Universitat de Barcelona

Lógica de la aplicación (II)

Componente Service.

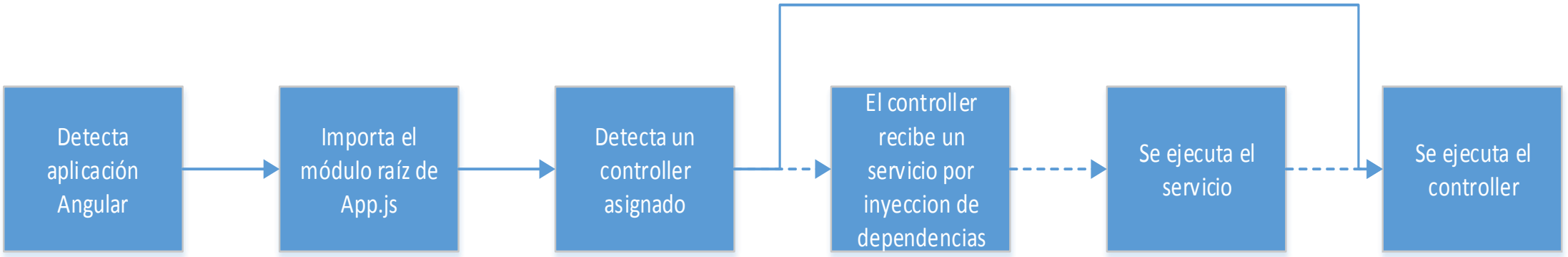
- Instancia un objeto que requerirá la aplicación.
- Sólo bajo demanda.
- Se utilizan en la aplicación para:
  - La carga de datos de los JSON asíncrona → defer & promise.
  - La creación de la tabla de correspondencia NUTS-PAÍS.
- Inyección de dependencias en el controller.

```
.factory('loadJSON',function($q){
.service('diccionarioEuropa',function(){
.controller('mainCtrl',function($scope,loadJSON,diccionarioEuropa){
```

Universitat de Barcelona

Lógica de la aplicación (III)

El flujo de ejecución es el siguiente:



Universitat de Barcelona

Implementación directivas

Se encargan de la creación de cada una de las gráficas del DashBoard.

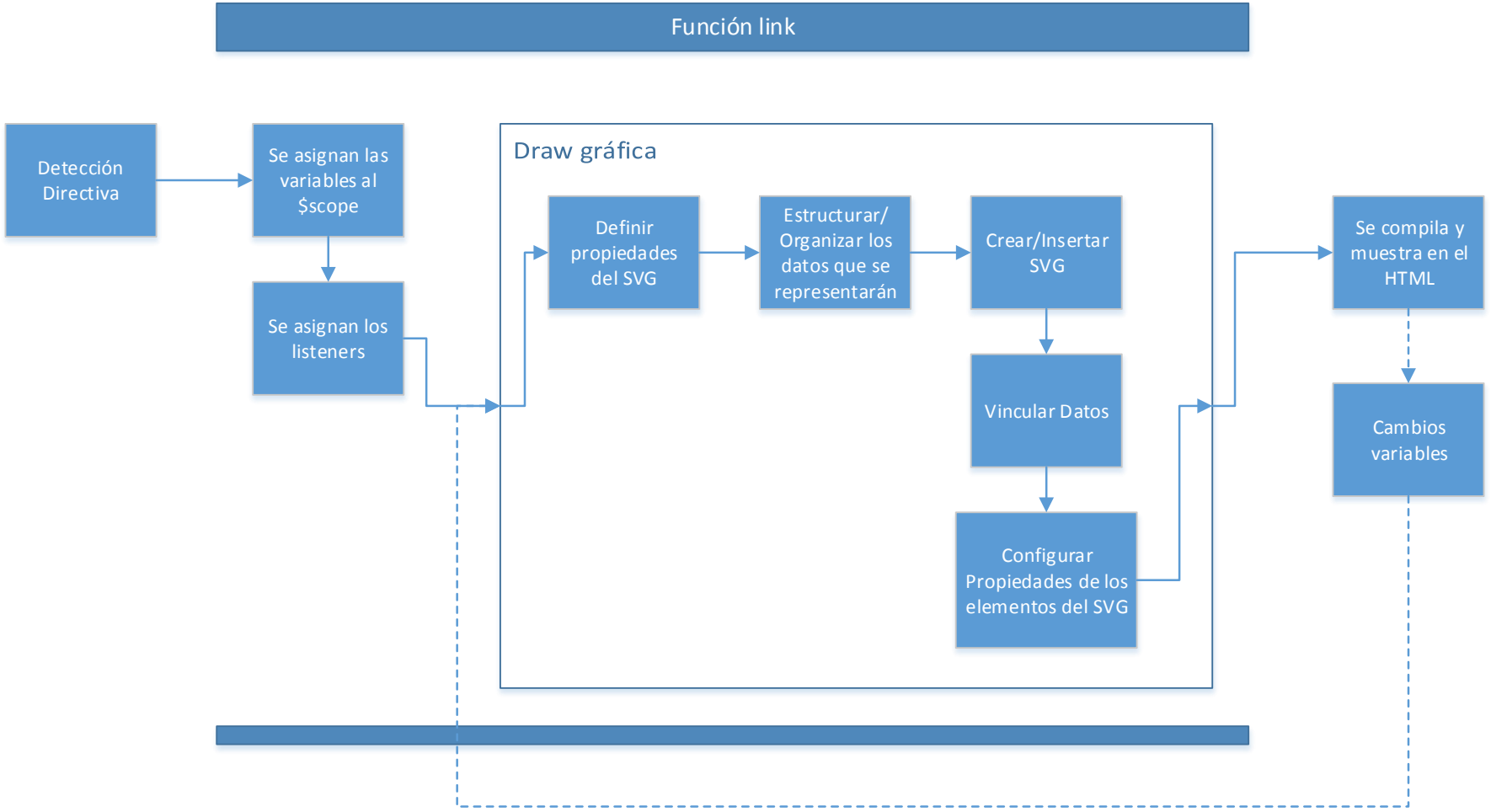
Basadas en el uso del $scope:

- Tienen acceso al $scope del controller.
- Tienen un $scope propio.
- Se ejecuta una función al compilarlas en la que:
  - Inicializamos las variables de esta gráfica.
  - Asignamos listeners sobre las variables de los $scope → Detección de cambios.
- Se añaden en el HTML:

```
<div my-area-chart class="panel panel-default">
```

Universitat de Barcelona

Implementación directivas (II)



Universitat de Barcelona

Implementación directivas (III)

La aplicación implementa 11 directivas que equivalen a 9 tipos de gráficos diferentes.

Los **tipos de gráficos** que encontramos en nuestra aplicación son:

- Cloropeth Map
- Bubble Chart
- Line Bar Chart
- Area Chart
- Pie Chart
- Stacked Bar Chart
- Line Chart
- Sunburst Chart
- Chord Diagram

Tres de las directivas sirven para crear el mismo tipo de gráfico: **Bubble Chart**, único tipo de gráfico que aparece más de una vez en el DashBoard. Cada uno de ellos analiza datos y matices diferentes.

Universitat de Barcelona

Implementación directivas (IV)

Ejemplo de creación de una directiva:

1. Definimos la directiva:

```
.directive('myChart',function(){
```

2. Inicializamos variables y guardamos en $scope.

```
scope.values = values;
```

3. Asignamos listeners a variables de $scope.

```
scope.$parent.$watchGroup(['pais','education'], function(){
    drawChart(scope,el,scope.$parent.employment);
});
```

4. Creamos el contenedor SVG:

```
var svg = d3.select(el[0].children[1])
    .append("svg")
    .attr("width", w)
    .attr("height", h)
    .attr("class","graph")
    .attr("id","svg_s");
```

Universitat de Barcelona

Implementación directivas (V)

5. Vinculamos los datos a los elementos del SVG.

```
var circles = svg.selectAll("circle")
    .data(array)
    .enter()
    .append("circle")
```

6. Definimos eventos sobre estos elementos.

```
.on ("mouseleave",function(d){
    var cr = d3.select(this);
    cr.attr("r",2);
    tooltip.remove();
})
```

Universitat de Barcelona

Integrar directivas al DashBoard

La composición final del DashBoard se consigue mediante el uso de **templates**:

- Ficheros HTML.
- Dividen estructuralmente la página.
- Tenemos 3 templates:
  - Navbar.
  - Contenido de Visualización. Define qué directivas se crean y cómo se organiza el contenido
    - Primaria → Visualización por defecto.
    - Secundaria → Visualización específica de un país al seleccionarlo con el mapa.
- Se añaden a la página con el uso de esta directiva de Angular

```
<div ng-include src="'views/primaryView.tpl.html'"></div>
```

Universitat de Barcelona

Resultados

¿Qué mejor que observarlo directamente?

http://calexx.github.io

Universitat de Barcelona

Conclusiones (I)

Después de trabajar en este proyecto de **Visualización de datos**, algunas de las conclusiones extraídas:

Análisis

- Mejor percepción de comportamientos → Correlaciones y patrones.
- Asociación de elementos → Influencias entre ellos.
- Tipo de gráfica → Enfoque → Mejor análisis.
- Interactividad → Depuración de datos más relevantes.

Representación

- Datos mejor representados → claridad y orden.
- Resultados más intuitivos y fáciles de interpretar.
- No requiere que el usuario haya analizado los datos.
- Interactividad → Usuario más involucrado.
- Requiere estética y utilidad.
- Subjetividad.

Universitat de Barcelona

Conclusiones (II)

Angular.js.

- Útil para aplicación SPA.
- Directivas → Tratar elementos del HTML de forma independiente.
- Scope y listeners → Nos evitan tener que definir los comportamientos de todas las gráficas afectadas sobre cada evento de cambio.
- MVC → Código estructurado y control de flujos de ejecución.
- Facilidad de testeo.
- Definir funciones de post-compilación en las directivas.

D3.js

- Alto nivel de manipulación de SVG.
- Gran aportación de la comunidad → Aprendizaje.
- Vinculación de datos sencilla→ Generación de SVGs dinámicos.
- Actualización de modificaciones sobre los datos vinculados muy rápida.
- Carga de JSONs eficiente y asíncrona.
- Falta implementación de eventos comunes → Drag&Drop, autoTooltip.
- Dificultad de implementar transiciones en iteraciones variables.