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)

Con el auge tecnológico, la cantidad de datos que somos capaces de generar ha aumentado considerablemente.

Data science estudia la capacidad de extraer información de estos grandes volúmenes de datos y aprovecharla en la toma de decisiones.

Uno de los ámbitos pertenecientes al data science es la **visualización de datos**, cuyo objetivo mostrar esta información visualmente mediante el uso de representaciones visuales.

Universitat de Barcelona

Planteamiento (I)

Se podría definir la base del proyecto como una **aplicación web** de visualización de datos implementada con **D3.js** y **Angular.js**.

La aplicación estará formada por un conjunto de gráficas que nos mostrarán información sobre la **desigualdad de género** existente en el marco laboral europeo.

El objetivo principal del proyecto es aplicar técnicas de visualización de datos sobre este caso concreto para acabar respondiendo una serie de cuestiones interesantes del mismo.

El usuario debe ser capaz de interactuar con las representaciones. La **visualización de datos interactiva** permite al usuario modificar los datos que se están representando en las diferentes gráficas.

Universitat de Barcelona

Planteamiento (II)

¿Por qué realizar este proyecto?

Hoy en día, la propia extracción de la información es tan importante como ser capaz de **transmitirla** a terceros de la forma adecuada. Al fin y al cabo, es muy probable que personas ajenas a la extracción deban entender esta información y actuar en consecuencia. Si algo nos ha enseñado nuestro sistema de percepción, es el hecho de que los humanos comprendemos más intuitivamente la información se representa adecuadamente con herramientas visuales.

En cuanto al caso de uso, el motivo principal por el cual se ha escogido se debe a que, durante la última década, se ha hecho bastante hincapié en intentar eliminar los rescoldos de la desigualdad de género y creo que es interesante observar si, al menos a nivel laboral, se ha tenido éxito.

Universitat de Barcelona

Funcionamiento de la aplicación (I)

El resultado deseado es un **DashBoard** de visualización de datos que nos permita conseguir nuestros objetivos. Para lograrlo, además de incluir las tecnologías web más comunes (HTML, CSS y JS), se han utilizado otras dos especialmente relevantes: Angular.js y D3.js.

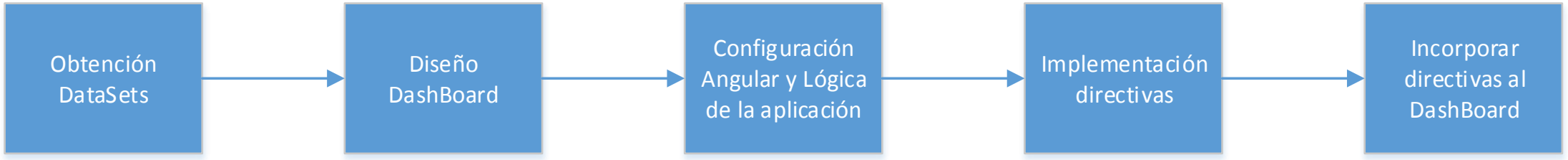
- **Angular.js** es un framework MVC de código abierto basado en JavaScript. Su base consiste en la interpretación de etiquetas incrustadas en el código HTML. La aportación de Angular ha contribuido principalmente a lograr la interactividad del usuario con las gráficas del DashBoard y que estos cambios se propaguen.

- **D3.js** es un librería basada en JavaScript que nos permite vincular datos a elementos del DOM. Esto nos permiten crear y personalizar elementos SVG de forma dinámica. La mayor contribución de D3 afecta al diseño de las gráficas resultantes.

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.

Gracias a estos DataSets podemos configurar las gráficas sobre la desigualdad de género en el marco laboral.

Además de estos, también necesitaremos un fichero **TopoJSON** que contenga las coordenadas de geolocalización que nos permitan representar un mapa de la Unión Europea.

Universitat de Barcelona

Diseño DashBoard

Para crear el DashBoard se ha hecho uso del framework **Bootstrap** cuya mayor virtud es facilidad que nos aporta a la hora de crear de interfaces web **responsives**.

Las características más notorias de nuestro DashBoard son:

- Uso de **paneles** para contener las gráficas
- **Botones de transiciones** en la sección del título de algunos paneles
- **Navbar lateral** con la que el usuario puede interactuar
- Uso del sistema **Grid** de Bootstrap: se define el ancho en columnas de cada elemento y la altura queda variable en función del contenido

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>: **<ng-app>.**

Universitat de Barcelona

Lógica de la aplicación

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

Primero de todo necesitamos un componente controller cuya existencia se comunica con la etiqueta

<ng-controller>.

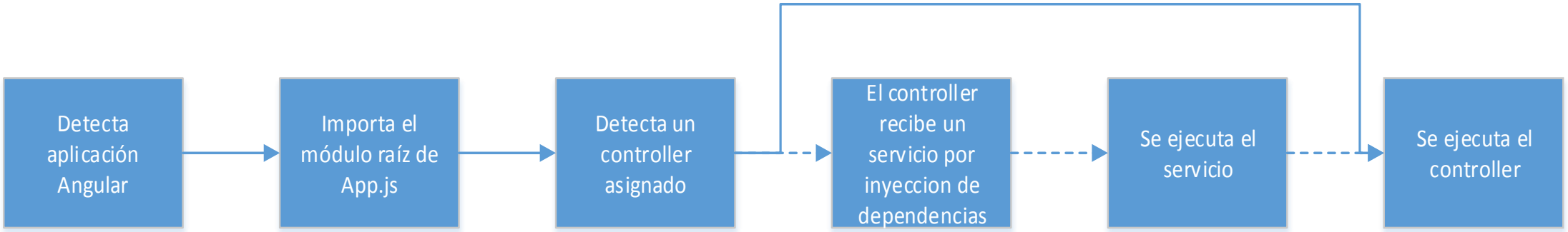
La función principal del controller es la **inicialización** de las variables globales de la aplicación que deban ser accesibles desde todas las gráficas. Estas variables se almacenan en su **$scope**. Además, también recibe la inyección de nuestros servicios implementados.

Lógica de la aplicación (II)

Los servicios se encargan de la instanciación de aquellos objetos que requiere la aplicación, en nuestro caso se encargan de:

- La carga de datos de los JSON.
- La creación de la tabla de correspondencia NUTS-PAÍS.

El flujo de ejecución es el siguiente:



Universitat de Barcelona

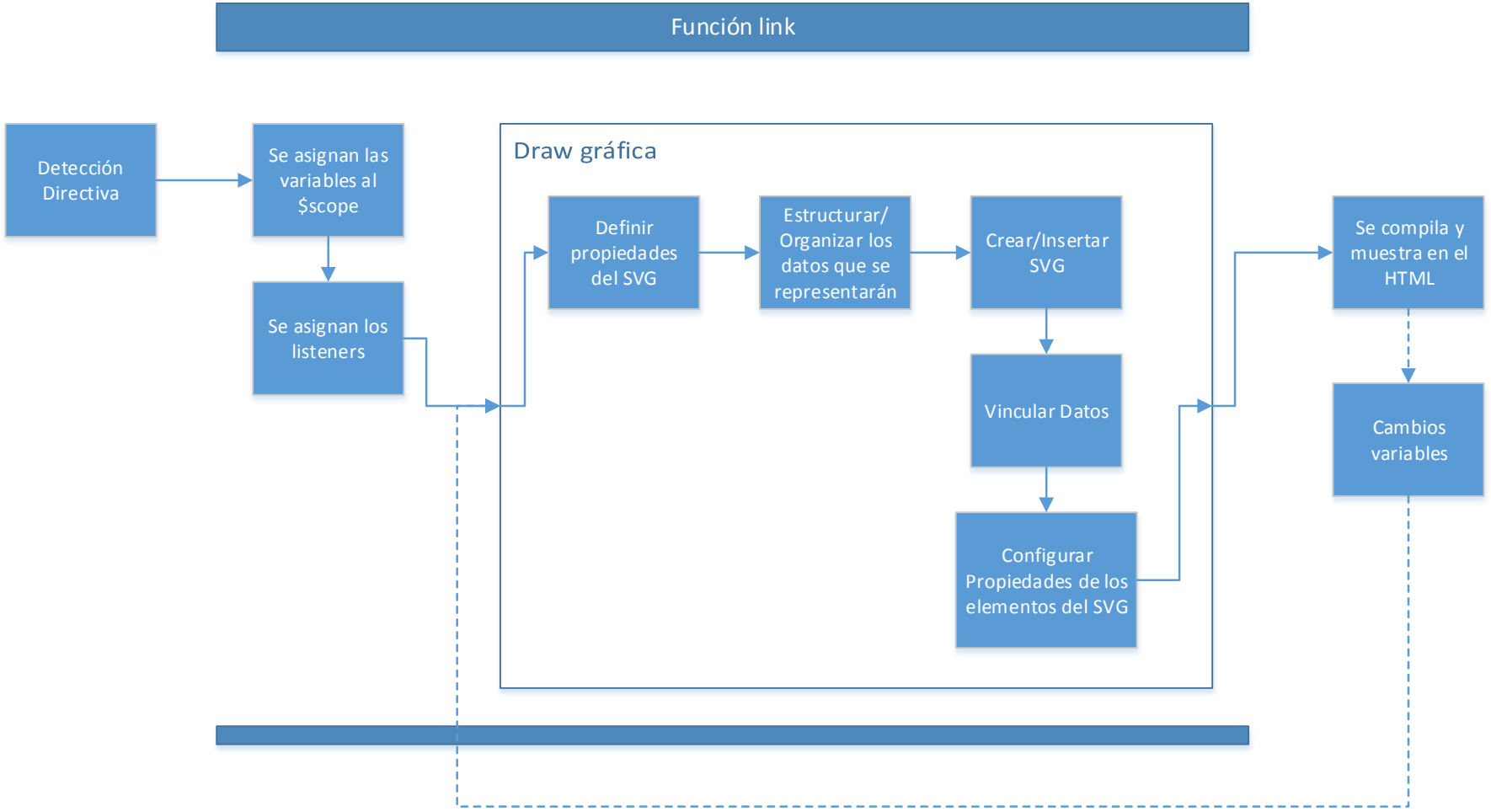
Implementación directivas

Las directivas son los componentes de Angular encargados de la creación de las gráficas que conformarán el DashBoard. Todas las directivas tiene acceso a los valores del $scope del controller, además de tener uno propio en el que se guardaran las variables que gestionan los datos del gráfico.

Las directivas tienen asociado un comportamiento a su compilación, en el que inicializamos las variables de la gráfica y asignamos **listeners** a éstos.

Angular nos permite asignar listeners a las variables del $scope y definir qué se ejecutará cuando se detecten cambios sobre éstas. En nuestro caso, cambios sobre las variables de interacción desembocaran en *redraws* de las gráficas.

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

Integrar directivas al DashBoard

La composición final del DashBoard se consigue mediante el uso de templates. En este caso, los templates son ficheros HTML que únicamente contienen un fragmento de código HTML y que, con el uso de la propiedad **<ng-include>** de Angular, podemos incrustar directamente en el fichero HTML.

Tanto el NavBar como el contenido del DashBoard se implementan con el uso de templates. Tenemos dos templates diferentes que para el contenido que se de los cuales se cargará uno de los dos en función de los datos que estamos visualizando:

- Si mostramos todo el **conjunto de la Unión Europea**, el template escogido mostrará 12 gráficas en el DashBoard.
- Si mostramos datos específicamente de **un país**, el template escogido mostrará 9 gráficas en el DashBoard.

Resultados

¿Qué mejor que observarlo directamente?

http://calexx.github.io

Universitat de Barcelona

Conclusiones (I)

Gracias a la visualización de datos, somos capaces de analizar ciertos comportamientos de los datos que, observándolos directamente en una tabla, serían mucho más difíciles de percibir.

Una de las grandes características de este campo es su capacidad para mostrar a terceras personas información extraída a través del análisis de conjuntos de datos.

El hecho de que el creador de las representaciones visuales sea capaz de influir en su interpretación, constituye un riesgo importante. Si bien es cierto que unas correctas técnicas de visualización de datos descartan completamente esta posibilidad, es interesante tener en cuenta esta posible e indeseable sugestión hacia el observador.

Conclusiones (II)

A nivel de tecnologías, comentar algunas de las sensaciones que me han transmitido tanto D3 como Angular:

- Angular.js. Considerando que, al inicio del proyecto, no estaba previsto su uso, su incorporación ha resultado gratamente valiosa. Muy útil y prácticamente indispensable si lo que quieres es crear **específicamente** una aplicación **SPA**, de lo contrario yo me plantearía seriamente si su incorporación, en términos generales, merece la pena. Su modularidad y estructura simplifica las **tareas de testing**. Su método de funcionamiento basado en **directivas** y la gran capacidad de configuración de éstas es bastante interesante y, en este caso, me ha venido como anillo al dedo.
- D3.js. La librería más extendida en cuanto a SVG con JavaScript se refiere. Uno de sus mejores puntos es la gran aportación de la comunidad que repercute directamente en la **facilidad de aprendizaje**. Desde mi punto de vista, es una librería bastante completa que permite hacer SVGs con diferentes niveles de complejidad y que destaca, sobretodo, a la hora de **vincular los datos** a los elementos del SVG i a la rapidez con la que se **actualizan los cambios** sobre la gráfica al modificar los datos. No todo es perfecto y tiene algún punto mejorable, como el encadenamiento de transiciones y la falta de implementación de eventos propios de gráficos (como Drag and Drop o autoTooltips).