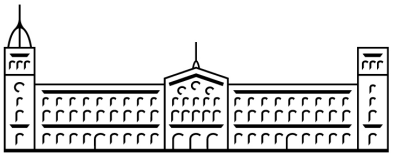

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques
Universitat de Barcelona

Visualización de datos con D3.js y Angular.js

Autor: Alejandro Cortés Cabrejas

Director: Santi Seguí
Realitzat a: Dept. de Matemàtica Aplicada i Anàlisi

Barcelona, 26 de junio de 2015

Abstract

Every second, massive amount of data are generated around us. In an era where data is one of the most valuable resources that humans can own, the extraction of knowledge from this data has improved its relevance. Data science is science that studies the extraction of knowledge from large volumnes of data. Data visualization is one of the subjects included in data science. It's defined as a toolkit with the goal of present this knowledge in a graphical format: a chart. Along development of this project, we'll use a web environment to implement some of them. Therefore we'll use some datasets whose focus is a particular topic, labor inequality, to ask ourselves some questions we'll try to solve using upcoming knowledge and technologies which will allow us to display it.

Resum

Cada segon que pasa, es generen quantitats massives de dades al nostre voltant. En una època on es considera que la informació és un dels recursos més valuosos que els humans poden tenir, la capacitat d'extreure coneixement d'aquestes dades ha guanyat especial rellevància. Data science és la ciència que estudia la extracció de coneixement de grans volums de dades. Una de les disciplines que inclou data science és la visualització de dades, un conjunt d'eines l'objectiu de les quals és representar aquest coneixement a través de representacions visuals. Durant aquest projecte, utilitzarem un entorn web com a mitjà en el qual implementar-ne unes quantes partint de conjunts de dades sobre un cas concret: desigualtat laboral. Es plantejaran algunes qüestions que intentarem resoldre amb l'ús de tecnologies de visualització de dades.

Resumen

Cada segundo que transcurre, se generan cantidades masivas de datos a nuestro alrededor. En una época en la que se considera la información como uno de los recursos más preciados que los humanos podemos de poseer, la capacidad de extraer conocimiento de estos datos ha ganado especial relevancia. Data Science es la ciencia que estudia la extracción de conocimiento de grandes volúmenes de datos. Una de las disciplinas que incluye Data Science es la visualización de datos, un campo cuyo objetivo es representar este conocimiento mediante representaciones visuales. En este proyecto utilizaremos un entorno web como medio de implementación de múltiples gráficos partiendo de conjuntos de datos sobre un caso concreto: desigualdad laboral. Se plantearán una serie cuestiones que intentaremos resolver utilizando tecnologías de visualización de datos.

Agradecimientos

Quiero agradecer a todos mis seres queridos, en especial a mis familiares, ya que sin su apoyo incondicional no hubiera llegado hasta este punto. Desde aquí aprovecho para expresar mis más sinceras gracias por la guía ofrecida, a mi tutor, Santi, cuya ayuda ha sido sido esencial a la hora de desarrollar este proyecto.

Índice general

Índice de figuras

Capítulo 1

Introducción

Estamos rodeados de cantidades ingentes de datos, posiblemente muchos más de los que podamos comprender. Ordenadores, smartphones, internet, consolas... son solo la cabeza visible de un mundo que observa atónito cómo nosotros, los humanos, hemos encontrado en esta vorágine una preciada fuente de conocimiento.

¿Qué son los datos? Los datos son representaciones simbólicas de variables diversas. Los datos pueden llegar a ser cosas tan básicas como clientes atendidos, colores de un cuadro o días de vacaciones. Probablemente, si analizamos un dato aislado, la información que éste contenga no sea excesivamente relevante pero... ¿y si cogemos conjuntos de datos sobre un contexto?

Con las mejoras más que notorias en el campo de las telecomunicaciones, el tamaño de los conjuntos de datos que se generan diariamente a nuestro alrededor ha crecido de sobremanera, permitiéndonos extraer grandes cantidades de información de ellos, lo que se conoce como *data science*.

> " Data science es la ingeniería civil de los datos. Sus acólitos poseen un conocimiento práctico de herramientas y materiales, junto con una comprensión teórica de lo que es posible." por *Michael E. Driscoll*

Con la creciente popularidad de la ciencia de los datos, el análisis de éstos ha incrementado su presencia en prácticamente todos los ámbitos, destacando principalmente el empresarial. Los grandes beneficios cuya aplicación puede llegar a aportar ha provocado que, aquellos que han dominado estas técnicas, deban sean capaces de representar este conocimiento, permitiendo incrementar su alcance, lo que se conoce como *data visualization*.

1.1. Motivación

Vemos cómo la visualización de los datos juega un papel fundamental dentro del campo del *data science*. Unas buenas técnicas de visualización no solo permiten mostrar y reproducir aquellos conocimientos que hemos sido capaces de extraer de conjuntos de datos, sino que además nos permiten mejorar la capacidad de análisis sobre éstos. Evidentemente, una representación más clara de los datos es de inestimable ayuda a la hora de afirmar ciertos enunciados sobre los datos analizados.

Por lo tanto, la visualización de datos no es únicamente una técnica para transmitir con más facilidad el conocimiento extraído, sino que también muestra la información de una manera más clara y ordenada. Es especialmente útil para simplificar conjuntos de datos relativamente complejos cuyo análisis seria realmente complicado sin estas técnicas.

Ante esto, sólo cabe exponer mi interés en lograr demostrar cómo aplicar estas herramientas de visualización de datos para analizar y transmitir conclusiones extraídas de ciertas cuestiones planteadas sobre conjuntos reales de datos. Concretamente, se estudiarán diversos factores relacionados con la desigualdad de género en el ámbito laboral europeo, que espero que sea de interés para los lectores. Dichos datos se encuentran disponibles al alcance de cualquiera que desee utilizarlos.

1.2. Objetivos

Con Internet estando presente en la mayor parte de nuestras vidas, creo que no existe canal más idóneo en el que aplicar las, ya mencionadas, técnicas de visualización de datos. No olvidemos que uno de los grandes retos que ésta plantea es transmitir conocimientos a terceros.

Partiendo de cero en visualización de datos y con pocos conocimientos previos de *data science*, el primer objetivo con el que toparse es, inevitablemente, aprender lo suficiente de ambos campos (*data science* y *data visualization*). Puesto que el canal de transmisión, en este caso, es importante, el siguiente objetivo claro es aprender las tecnologías que me permitan representar estos conocimientos en dicho canal: Internet.

Llegados a este punto, sólo queda definir qué planteamientos queremos representar mediante técnicas de visualización. Durante la última década, uno de los círculos de lucha más recurrentes ha sido, sin duda, la igualdad entre hombres y mujeres. Partiendo de Europa, será interesante analizar estas cuestiones:

- ¿Han alcanzado las mujeres a los hombres en cuanto a número de trabajadores?
- ¿Influye el PIB en los salarios de los empleados?

- ¿La inversión en educación garantiza el equilibrio de salarios entre hombres y mujeres?
- ¿En qué actividades existe más diferencia de empleados y como han ido evolucionando éstas con el paso de los años?
- ¿Qué actividades han sufrido un mayor descenso de empleados durante los últimos años?¿Qué género lo ha sufrido más?
- ¿Como se distribuye la actividad laboral en función de los estudios alcanzados? En las actividades que requieren más educación... ¿existe más equilibrio entre el número de empleados y de empleadas?
- ¿Las tendencias en algunos sectores se han ido invirtiendo con el paso de los años?

Éstas son algunas de las hipótesis planteadas a las que se intentará dar respuesta mediante el desarrollo del proyecto, además de plasmar los resultados obtenidos mediante el uso de técnicas de visualización de datos.

1.3. Organización

Tal y como nos deja intuir el índice anterior, este documento estará organizado mediante capítulos. En esta sección definiremos brevemente los contenidos de cada uno de éstos para otorgar una idea general del contenido del proyecto:

- **Capítulo 2: Visualización de datos**. Contiene pinceladas globales de la teoría existente detrás del proyecto que nos ayudará a comprender su posterior desarrollo.
- **Capítulo 3: Planificación**. Contiene tanto la planificación inicial del proyecto como la planificación final resultante, incluyendo algunas reflexiones posteriores derivadas de ambas planificaciones.
- **Capítulo 4: DataSet**. Contiene información sobre los datos en los que se basa el proyecto y de dónde pueden extraerse.
- **Capítulo 5: Desarrollo**. Reproduce los pasos que hemos realizado para construir la aplicación, definiendo los aspectos relativos a la implementación de ésta.
- **Capítulo 6: Resultados**. Contiene las reflexiones extraídas de los datos vinculados a las diferentes gráficas que componen la aplicación junto a la resolución de las cuestiones planteadas previamente.

- **Capítulo 7: Conclusiones**. En este apartado se encontrarán las conclusiones y reflexiones extraídas durante el desarrollo del proyecto y la finalización de éste. Encontramos conclusiones relativas tanto al ámbito tecnológico como al campo de estudio.

Capítulo 2

Visualización de datos

Durante años, los humanos han dependido de representaciones como mapas o dibujos para comprender la información de forma más clara y rápida. Entonces... ¿Por qué no aplicarlo a los grandes conjuntos de datos que somos capaces de generar hoy en día?

La visualización de datos consiste en un conjunto de técnicas de comunicación visual que nos permiten representar información más eficientemente. Esto se consigue mediante el uso de gráficos que enfatizan la tarea analítica planteada sobre conjuntos de datos. Como consecuencia del modo de procesar la información que tiene el cerebro humano, somos capaces de entender más rápidamente el significado de muchos puntos de datos si éstos se encuentran representados sobre los gráficos adecuados.

Es imposible concebir la visualización de datos sin los gráficos y los retos que éstos plantean. Ser capaz de representar adecuadamente los datos requiere comprender qué objetivos existen detrás de éstos y de qué manera se puede utilizar un impacto visual para facilitar la comprensión de los mismos.

Además de determinar qué objetivos tienen dichos datos, hay que plantear qué queremos hacer con nuestro gráfico: ¿Queremos comunicar nuestros conocimientos?¿Queremos identificar patrones, relaciones...?¿Queremos ambas cosas? Todo inicio de una representación visual requiere enfrentarse a estas preguntas y determinar qué tipo de representación emerge como la más óptima.

Observamos como la ciencia y el arte se entrelazan gracias a este campo. Como toda obra artística, un gráfico posee de manera intrínseca cierto grado de subjetividad, aunque existen algunas directrices que nos marcan buenas técnicas de diseño:

- Entender los datos que se están intentando visualizar, particularmente, su tamaño y cardinalidad.
- Determinar el objetivo fundamental de los datos.

- Ser consciente de quién va a ser el observador y cómo éste procesa la información visual.
- Elegir el tipo de representación que mejor se adapte a nuestro contexto.

A la hora de iniciar el diseño del gráfico, debemos partir de la base de que existe uno que, por naturaleza, representa mejor cada tipo de dato y que, por encima de esto, existe una composición visual asociada que permite cumplir con los objetivos mencionados previamente.

Es común encontrar gráficos que resultan complicados de comprender y cuya representación no entendemos a simple vista. Aunque se parta de la premisa *cualquier gráfico siempre es mejor poder entenderlo sin instrucción previa*, conjuntos muy complejos de datos, probablemente, requieran una representación visual que demande al observador conocer la idea que existe detrás de la creación del mismo.

Pese a la subjetividad que otorga el arte a estas representaciones, existen malos gráficos. La teoría de la visualización cataloga como malos gráficos aquellos que:

- Son construidos sobre datos no válidos.
- Proporcionan una visión equivocada de los datos.
- Desinforman al observador.
- Alteran la realidad de los datos.
- Su resolución formal no es la correcta.

¿Debe una representación priorizar la estética?¿O bien decantarse por la utilidad? Estas preguntas escenifican uno de los mayores debates que rodea la teoría de visualización de datos. El impacto que tiene el sentido de la vista para las personas es evidente e innegable, y un gráfico estéticamente logrado no tiene por que entrar en conflicto con la utilidad del mismo. Un gráfico siempre debe ser útil y evitar a toda costa que únicamente cumpla una función estética.

Todo dato dentro de un conjunto siempre es relevante, aunque no todos en la misma medida. A la hora de diseñar una representación, un ejercicio importante es depurar estos datos y seleccionar para nuestra visualización aquellos que realmente tienen peso sobre nuestro objetivo.

El dominio de la teoría que sustenta la visualización de datos requiere conocer ciertos aspectos de la capacidad perceptiva humana, de entre los cuales son relativamente interesantes, *los principios de Gestalt*:

- **Semejanza**. Nuestra mente tiende a agrupar elementos similares.
- **Proximidad**. Solemos agrupar los elementos que están más próximos entre ellos.
- **Simetría**. Las imágenes simétricas distantes son percibidas como un solo elemento.
- **Continuidad**. Los detalles que mantienen un patrón o dirección tienden a agruparse juntos.
- **Dirección común**. Los elementos que parecen construir un patrón en una misma dirección se perciben como una única figura.
- **Simplicidad**. Organizamos nuestros campos de percepción con rasgos simples, regulares y armónicos.
- **Relación entre figura y fondo**. Nuestro cerebro no puede interpretar un objeto como figura y fondo al mismo tiempo.
- **Igualdad**. Cuando concurren varios elementos de diferentes clases, tendemos a agrupar aquellos que son iguales.
- **Cerramiento**. El cerebro interpreta un conjunto de formas como una única uniendo las líneas exteriores que las rodean.
- **Experiencia**. Nuestra percepción viene condicionada por nuestras propias experiencias.

Además de estos conceptos, comprender la teoría de la visualización conlleva conocer qué tipos de datos existen y qué analogías visuales son las más adecuadas para entenderlos.

2.1. Tipos de datos

Definir los tipos de datos existentes depende efusivamente del contexto en el que nos encontremos. En este caso, nos centraremos en los tipos de datos que encontramos al realizar análisis:

- **Cuantitativos**. Son aquellos datos que se expresan de forma numérica.
  - **Continuos**. Datos cuantitativos que pueden adoptar cualquier valor numérico intermedio en un rango. Por ejemplo: edad, peso, colesterol, etc.
  - **Discretos**. Datos cuantitativos que sólo pueden tomar algunos valores dentro de un rango. Por ejemplo: número de veces que vas al cine, número de hijos, número de mascotas, etc.

- **Cualitativos**. Son aquellos datos que no se pueden expresar numéricamente y que representan cualidades o atributos.
  - **Binarios**. Datos cualitativos que sirven para clasificar en una de dos posibles categorías. Por ejemplo: sano/enfermo, hombre/mujer, casado/soltero, etc.
  - **Categóricos**. Datos cualitativos que sirven para clasificar en más de dos categorías posibles. Por ejemplo: color de ojos, grupo sanguíneo, color de pelo, etc.

2.2. Visualización Interactiva

Ésta rama de la visualización de datos utiliza la interacción entre los humanos y las máquinas para crear gráficos con muchísimo más detalle y que, a través de dicha interacción, son capaces de modificar los datos que se están representando inmediatamente. Para que una visualización sea considerada interactiva debe cumplir dos criterios:

- El observador debe ser capaz de controlar algún aspecto de la representación.
- El tiempo de respuesta a los cambios del observador debe ser prácticamente en tiempo real.

La incorporación de la interactividad abre una nueva ventana al campo de la visualización de datos ya que no contempla al observador como un mero espectador de una representación gráfica, sino que influye activamente en ésta. En las visualizaciones interactivas, se permite al usuario explorar los conjuntos de datos que manejan las representaciones y, al jugar con éstos, personalizar los conocimientos en función de sus propios intereses.

2.3. Tipos de visualizaciones

Existen muchísimos tipos posibles de representación. Para acotar el ámbito, nos centraremos en los tipos de gráficos más comunes cuando hablamos de visualización de datos. Además, también merece la pena comentar algunos gráficos más complejos y no tan populares que irán apareciendo a lo largo del proyecto.

2.3.1. Line Chart

Los *line chart* se forman al dibujar puntos de datos sobre un sistema de coordenadas cartesianas[1] y conectando los puntos mediante líneas.

[1] Tipo de coordenadas usadas en espacios euclídeos que utilizan como referencia ejes ortogonales que se cortan en un punto llamado origen

Este tipo de gráfico se utiliza principalmente para representar variables cuantitativas sobre intervalos de tiempo continuos. Es útil para mostrar tendencias y, cuando tenemos más de una linea en el gráfico, relaciones. Ayudan a dar una visión global de los datos y su desarrollo sobre un intervalo de tiempo.

2.3.2. Area Chart

Los *area chart* parten de los *line chart* pero le añaden textura o color al área que la linea definida genera con los ejes.

Este tipo de gráfico es utilizado para representar el desarrollo de variables cuantitativas sobre un intervalo de tiempo.

2.3.3. Bubble Chart

Los *bubble chart* se construyen al dibujar puntos sobre un sistema de coordenadas cartesiano en el cada eje representa una variable independiente. Además, estos puntos son capaces de representar una tercera variable utilizando su área como círculo. Habitualmente se utilizan diversos colores para diferenciar entre los diferentes puntos de la representación.

Este tipo de gráfico suele ser utilizado para comparar las relaciones entre los círculos mediante el uso de su posición y proporción. Muestran un gran potencial para el análisis de patrones.

2.3.4. Histograma

Los *histogramas* están formados por barras cuya superficie representa la frecuencia de una variable sobre un intervalo continuo.

Este tipo de gráfico es muy útil para valorar como se concentran los valores y nos ofrece una pequeña idea de su distribución de probabilidad[2].

2.3.5. Bar Chart

Los *bar chart* utilizan barras horizontales o verticales para mostrarnos comparaciones numéricas discretas entre diferentes categorías. Un eje identifica las categorías específicas que se comparan, mientras que el otro eje representa valores discretos.

[2]Función que asigna a cada suceso de una variable la probabilidad de que dicho suceso ocurra

2.3.6. Stacked Bar Chart

Los *stacked bar chart* tienen su origen en los *bar chart* pero, a diferencia de éstos, segmentan sus barras para enmarcar las distintas categorías. Son utilizados para mostrar cómo grandes categorías se dividen en más pequeñas, proporcionando una excelente visión de la relación de cada una de las partes con el total. Uno de los mayores problemas de este tipo de gráfico radica en la dificultad de comparar cada segmento con otro puesto que usualmente no están alineados sobre una base común.

2.3.7. Pie Chart

Los *pie chart* surgen al dividir una circunferencia en segmentos proporcionales donde la longitud de cada arco representa una proporción de una categoría.

Este tipo de gráfico ayuda a ver las proporciones y porcentajes entre diferentes categorías respecto a un total, en gran medida porque que la suma de éstas equivale al cien por cien. Pese a esto, los *pies chart* tienen algunas carencias: sólo pueden representar una cantidad de valores limitada, ocupan bastante más espacio que otras alternativas similares y no permiten hacer comparaciones precisas.

2.3.8. ScatterPlot

Los *scatterplot* utilizan una serie de puntos sobre sistemas de coordenadas cartesianas para representar valores de dos variables. Tienen gran valor para observar si existe correlación entre ambas variables. Añadir lineas puede ayudar al análisis de la influencia que ejerce una variable sobre otra.

2.3.9. Cloropeth Map

Los *cloropeth map* muestran un área geográfica dividida en regiones más pequeñas y asignan un color a éstas en relación a una variable. Ofrece una interesante manera de observar patrones geográficos.

Habitualmente, se utiliza una progresión del color para representar el valor de esta variable sobre cada región. Las grandes desventajas que tiene este tipo de gráfico radican en la dificultad de analizar con precisión los valores existentes detrás del gráfico y el exceso de protagonismo que reciben las regiones más grandes.

2.3.10. Chord Diagram

Los *chord diagram* nos ofrecen la capacidad de mostrar las interrelaciones que se producen entre datos, dada una matriz de éstos. Los datos se sitúan alrededor de un círculo, con las relaciones entre los puntos definidas por arcos que los conectan.

2.3.11. Sunburst Diagram

Los *sunburst diagram* son, probablemente, los menos habituales de los gráficos comentados. Están basados en vínculos jerárquicos formados entre un conjunto de nodos. Los nodos se dibujan como áreas solidas y su posición respecto a los nodos adyacentes viene determinada por los datos que éstos representan. Se organizan mediante círculos concéntricos en los que el círculo central representa el nodo raíz de la jerarquía.

2.4. Tecnologías

Una vez definido el contexto en el que se encuentran *data visualization* y *data science*, es momento de ver como aplicar estos conocimientos en un entorno web.

2.4.1. Programación Web

Antes de hablar de alguna tecnología más concreta y especializada, hay que comentar ciertos aspectos de la programación web que son posteriormente utilizados por éstas:

- HTTP, *Hypertext Transfer Protocol*, es el protocolo[3] utilizado en cada transacción que se realiza por la web. Estamos ante un protocolo sin estado, con una comunicación basada en petición-respuesta.
- HTML, *HyperText Markup Language*, es un lenguaje basado en marcas que se utiliza para elaborar páginas web.
- CSS, *Cascading Style Sheets*, es un lenguaje utilizado para definir la capa de presentación de un documento elaborado en HTML.
- JS, *JavaScript*, es un lenguaje de programación interpretado orientado a objetos[4]. Se utiliza mayoritariamente para la interpretación de éste por parte del navegador, permitiendo mejoras sustanciales tanto en la interfaz como en la lógica de la aplicación.

[3] Conjunto de reglas que define la comunicación de datos por internet a través de paquetes conmutados
[4] Paradigma de programación que basa sus interacciones en objetos

- DOM, *Document Object Model*, es una API[5] que proporciona al programador tanto acceso como manipulación del contenido estructurado en lenguaje HTML.

2.4.2. D3.js

D3.js constituye la tecnología sobre la que se edifica el proyecto. Nos encontramos ante un librería basada en JavaScript que nos permite vincular datos a elementos del DOM. Gracias a D3, podemos crear visualizaciones de datos interactivas y dinámicas en navegadores web.

Esta librería se basa en funciones predefinidas escritas en lenguaje javaScript que mejoran la selección de elementos, nos permiten crear y personalizar elementos SVG y añadirles efectos como transiciones, dinamismo o tooltips. Tiene una gran capacidad para manipular grandes volúmenes de datos y vincularlos a dichos elementos para lograr crear gráficos de gran riqueza.

Los formatos más estandarizados de conjuntos de datos a nivel de web son CSV y JSON. D3 permite la lectura de datos en ambos formatos de forma asíncrona y eficiente.

La optimización de D3 respecto al uso de JavaScript plano se acentúa primordialmente en cuatro factores: selección, dinamismo, asociación y transiciones.

La selección permite al programador utilizar selectores de estilo CSS para seleccionar un nodo del DOM y manipularlo de manera similar a como lo hacen otras librerías como JQuery.

Las transiciones permiten la interpolación[6] de atributos y valores de elementos con datos asociados a lo largo de un periodo de tiempo. La interpolación de D3 soporta primitivas, números, composiciones de cadenas de caracteres con números y números compuestos.

La asociación de datos requiere que éstos se organicen en un array[7] y que se realice por correspondencia de índice. Una vez establecido el vínculo, los elementos siempre van a recordar dichos datos sin necesidad de volver a forzarlo posteriormente.

Una de las grandes ventajas que te permite D3 es la capacidad de asociar datos a elementos que todavía no han sido creados para otorgar a dichos datos la capacidad de crearse dinámicamente. Asimismo, también se pueden aplicar manipulaciones, selecciones y transiciones sobre los elementos que todavía no tienen datos vinculados o no han sido generados.

[5] Conjunto de procedimientos ofrecido por una librería para ser utilizado por otro software
[6] Obtención de nuevos puntos a partir de un conjunto definido de éstos
[7] En JS, variable que guarda múltiples valores

2.4.2.1. Tipos básicos de elementos SVG de D3.js

- **Rectángulos**. El tamaño de sus lados se define mediante las propiedades 'width' y 'height'. Además se posicionan mediante la coordenada de la esquina superior izquierda del rectángulo utilizando las propiedades 'x' e 'y'.
- **Círculos**. El radio determina el tamaño del círculo, definido en la propiedad 'r'. Se posiciona a través del centro de éste con las propiedades 'cx' para el eje horizontal y 'cy' para el vertical.
- **Lineas**. Una línea viene definida por dos puntos. D3 usa las coordenadas de éstos para crear este tipo de elemento. Se pueden crear caminos con la combinación de múltiples lineas.

2.4.3. Angular.js

Angular.js es un framework[8] MVC[9] de código abierto basado en JavaScript. Es especialmente útil en la gestión de sitios web SPA, *Single Page Application*[10]. La base de este framework consiste en la interpretación de etiquetas personalizadas incrustadas en el código HTML. Angular relaciona las entradas y salidas de una página HTML con un modelo construido sobre código JavaScript. Gracias a esta asociación bidireccional entre modelo y vista, así como a su sincronización automática, Angular nos permite distribuir el contenido de forma dinámica y fiable.

Pese a encontrarnos ante un framework *front-end*, su punto fuerte no radica en la manipulación del DOM, sino en la disgregación entre la lógica de la aplicación y el cliente, en este caso, el navegador web. Esto permite que se pueda considerar como uno de los mejores frameworks a la hora de diseñar test unitarios[11] sobre sus aplicaciones. Aquí podemos observar (2.1) la idea y el funcionamiento que está detrás de dicho framework, tal como hemos explicado anteriormente.

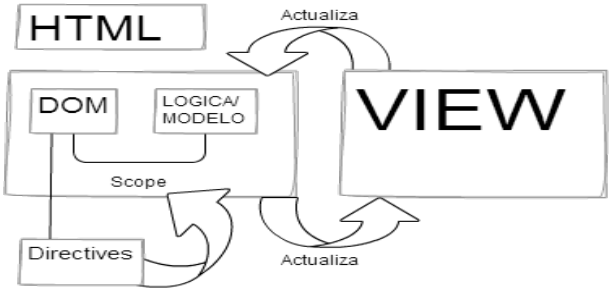

FIGURA 2.1: Diagrama de comunicación de Angular.js

[8] Estructura de soporte que facilita el desarrollo de software
[9] Patrón de arquitectura de software basado en la separación de los datos, la lógica y la interfaz de usuario
[10] Aplicación web construida sobre una única página
[11] Prueba para comprobar el correcto funcionamiento de un módulo de código

Para poder comprender la implicación de Angular en el proyecto, es necesario comprender los siguientes conceptos: $scope, directive, controller i service.

2.4.3.1. $Scope

Un $scope en Angular es un objeto con una estructura jerárquica que sirve de enlace entre la lógica de la aplicación y la vista. Un $scope permite almacenar información en él y aprovechar esta estructura jerárquica para lograr el intercambio de información entre las diferentes directivas y la lógica. Toda aplicación Angular tiene un $rootscope del que descienden el resto de $scopes asociados a las diferentes directivas. Los $scopes también permiten observar cambios en algunos de sus valores y propagarlos a las vistas.

2.4.3.2. Directive

Las directivas de Angular son elementos HTML personalizados que tienen asociado un comportamiento específico que se ejecuta al compilarlas[12]. Angular tiene definidas un conjunto de directivas con sus propios comportamientos, pero su mayor potencial se explota en el momento en que permite al usuario la creación de directivas y comportamientos personalizados. Además, las directivas permiten vincular valores definidos en el scope a sus atributos para lograr la intercomunicación de lógica y vista.

2.4.3.3. Controller

Un controller en Angular es una función constructor[13] escrita en JavaScript que se asocia a un elemento del DOM mediante la directiva ng-scope. Se utilizan principalmente para inicializar los valores del $scope, así como para añadirle comportamientos, por ejemplo, vinculándole un valor a una interacción del usuario.

2.4.3.4. Service

Un servicio en Angular define un objeto *singleton*[14] sustituible que utiliza la capacidad de inyección de dependencias de Angular para modular y estructurar mejor el código. Estos objetos únicamente se instancian en el momento en que la aplicación los demanda.

La inyección de dependencias es un patrón de diseño de software que define cómo los componentes obtienen sus dependencias. Con este patrón, son las clases las que reciben el objeto del que dependen en vez de crearlo ellas mismas.

[12] Traducción de un lenguaje de alto nivel a binario
[13] Función que se ejecuta automáticamente al crear una nueva instancia de la clase
[14] Patrón de diseño que restringe el número de instancias de una clase a uno

2.4.4. Otros

La implementación del proyecto ha requerido la incorporación de otras tecnologías que, aunque no tengan tanto peso como las dos mencionadas anteriormente, son especialmente útiles.

- **Underscore.js**. Librería JavaScript que ofrece funcionalidades comunes muy útiles al programador, especialmente en lo que refiere a manipulación de estructuras de datos como arrays, JS Objects y JSON.
- **Bootstrap**. Framework de desarrollo web que permite crear interfaces web con CSS y JavaScript que se adaptan al tamaño del dispositivo en el que se visualizan.
- **JQuery**. Librería JavaScript que simplifica la interacción con los documentos HTML y los elementos del DOM. Además, permite gestionar eventos y animaciones, así como agregar AJAX[15] de forma simple a la web.
- **TopoJSON**. Extensión de GeoJSON[16] que codifica características geográficas. Su principal ventaja radica en el tamaño de los ficheros, ya que ocupan aproximadamente un 80 por ciento menos que su predecesor. Esto se consigue gracias a la eliminación de los elementos redundantes y la compactación de la información.
- **Tooltipster**. Plugin[17] de JQuery que simplifica el diseño de tooltips[18].

[15] Técnica de desarrollo web utilizando JS que permite mantener una comunicación constante entre cliente y servidor sin tener que recargar la página
[16] Formato estándar para almacenar conjuntos de características geográficas
[17] Aplicación que se annexa a otra para dotarla de nuevas funcionalidades
[18] Herramienta de ayuda visual que aparece en el momento en que se interactúa sobre un elemento

Capítulo 3

Planificación

La planificación inicial del proyecto podemos observarla en (3.1). En general, destaca cómo las etapas están claramente definidas y no se produce ningún solapamiento entre ellas.

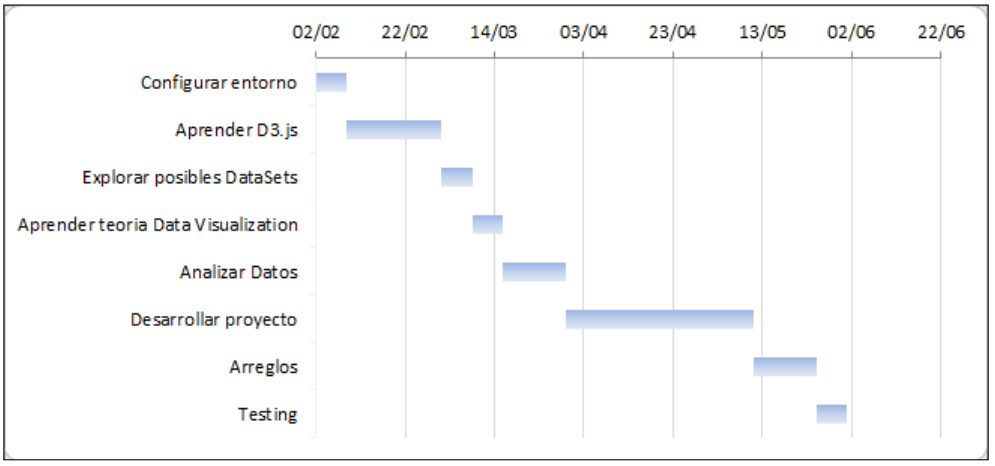

Figura 3.1: Planificación Inicial

Una vez elaborado el proyecto, podemos ver la inversión de tiempo resultante en las diferentes tareas aquí (3.2).

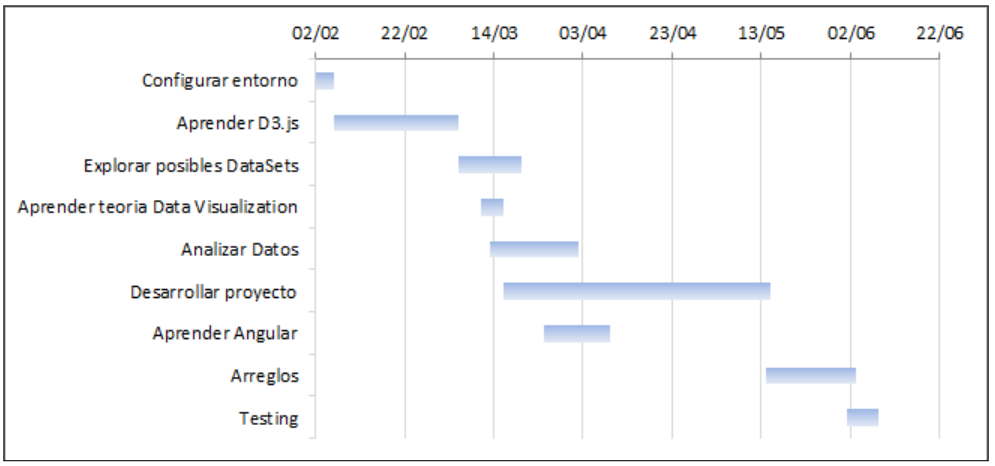

FIGURA 3.2: Planificación Final

Observamos cómo la tendencia de las tareas en las que se divide el proyecto es alargarse más de lo previsto en la mayoría de los casos. Esto se debe, principalmente, a la aparición de contratiempos que no se preveían, como el surgimiento de *bugs*[1], circunstancias externas al proyecto o, sencillamente, previsiones demasiado optimistas.

Llama la atención cómo durante la implementación del proyecto van apareciendo nuevas tareas que durante la planificación no se consideraban, por ejemplo, el aprendizaje de Angular. Esto suele deberse a la necesidad de incorporar nuevas tecnologías que a la larga nos aporten beneficios o a la aparición de dependencias externas de las cuales no teníamos conocimiento previo.

Pese a la aparición de estos contratiempos, en términos generales, se han cumplido bastante decentemente los tiempos que se preveían en la planificación inicial. Si bien es cierto que la idea inicial era centrarme en un solo objetivo hasta que lo dominase para pasar al siguiente, ha habido veces en que se ha tenido que trabajar en múltiples objetivos y dejar de lado la idea de centrar esfuerzos.

[1]Error de sotfware

Capítulo 4

DataSet

Un DataSet es un conjunto de datos que habitualmente se corresponde con una única tabla de una base de datos. Estas tablas están formadas por columnas que representan variables y filas que representan los miembros de un conjunto de datos. En definitiva, los DataSets contienen los valores de una serie de variables que corresponden a diferentes miembros de un conjunto.

En el ámbito del *data science*, es muy habitual el uso de DataSets concretos en detrimento de bases de datos convencionales. Particularmente, consideramos como propensos a ser utilizados aquellos que poseen las siguientes características:

- Son suficientemente grandes como para ser atractivos de procesar sin necesidad de *clusters*[1] ni computación paralela[2].
- Requieren un filtrado previo de los datos antes de ser procesados y analizados.
- Su riqueza de datos permite diferentes vías de análisis sobre éstos.

La idoneidad del uso de DataSets concretos en *data science* nos lleva a determinar cual será el origen de los datos que queremos analizar. Dicho esto, entran en escena dos formatos de datos posibles para el proyecto: CSV y JSON. El primero se considera el mejor formato para representar datos de una tabla, mientras que el segundo se ha erigido, junto a XML[3], como un estándar en transferencia y almacenamiento de datos web.

[1] Conjunto de ordenadores construidos sobre hardware común cuyo comportamiento simula ser un único ordenador

[2] Forma de cómputo en la que muchas instrucciones se ejecutan al mismo tiempo

[3] eXtensible Markup Language, lenguaje de marcas utilizado para almacenar datos de forma que su lectura sea fácil

4.1. CSV

CSV es un formato de tipo abierto de ficheros que representa datos organizados en tablas. En este tipo de documentos, la separación de las columnas se determina mediante comas, mientras que cada fila finaliza con un salto de linea.

El motivo principal de la gran popularidad de la que goza CSV recae en la simplicidad de éste: no requiere definir juegos de caracteres ni posicionamiento de bytes ni formato de salto de línea. Para poder analizar el contenido de estos ficheros de forma eficiente, se necesita que el orden de las columnas sea idéntico en todas las filas, además de definir el contenido de cada columna mediante el uso de *labels*.

4.2. JSON

JSON, *JavaScript Object Notation*, es un formato de ficheros ideado para el intercambio de datos por la web. Junto a XML, constituye el formato más utilizado en transferencia de información entre aplicaciones web, particularmente en el uso de AJAX.

Su simplicidad y facilidad de uso, así como su mimetización con el lenguaje JavaScript ha propiciado su generalización como alternativa a XML. Destaca por ser un formato compacto y realmente eficiente, ideal para representar estructuras jerárquicas y manejar grandes volúmenes de datos gracias, especialmente, a la sencillez con la que se analizan sintácticamente.

Este formato está construido alrededor de dos tipos de estructuras:

- Conjuntos de parejas clave-valor, interpretados como objetos.
- Listas ordenadas de valores, interpretadas como arrays.

4.3. Eurostat

Eurostat es la oficina estadística de la Unión Europea cuya principal meta radica en proveer a ésta de datos estadísticos sobre los países y regiones que la constituyen. Gracias a estos datos, se permite la comparación entre las diferentes regiones y países. Además, la estadística objetiva que aporta suele influir en las decisiones políticas estructurales de la Unión Europea.

Hay que puntualizar que Eurostat no se encarga de la recolección de los datos que publica, sino que es cada país quien lo hace y los verifica. Una vez hecho esto, es Eurostat

quien se encarga de consolidar los datos, asegurando que puedan compararse sin riesgo alguno.

Eurostat toma mayor relevancia en el aspecto económico, sobretodo a raíz de la aparición del euro como moneda única de la Unión Europea. Muchas de las decisiones tomadas por el Banco Central Europeo[4] son apoyadas por los datos macroeconómicos que produce la entidad.

Los datos se encuentras organizados en ocho áreas temáticas:

- Estadísticas generales y regionales.
- Economía y finanzas.
- Población y condiciones sociales.
- Industria, comercio y servicios.
- Agricultura y pesca.
- Comercio exterior.
- Transportes.
- Ambiente y energía.

[4] Administración encargada de la gestión político-económica de la Unión Europea

Capítulo 5

Desarrollo

5.1. Estructura del proyecto

Una aplicación de *data visualization* debe estar compuesta principalmente por datos y representaciones. Dicho esto, el uso de un *DashBoard*[1] para la gestión de los gráficos se antoja básico para el desarrollo del proyecto.

La aplicación desarrollada es completamente *front-end*[2], en la que todo el coste computacional recae en el cliente. Al no tener un servidor como tal se reducen considerablemente los fallas de seguridad ya que, al fin y al cabo, todos los datos permanecen en la máquina del usuario.

El uso de D3, tanto para el *parsing*[3] de datos de forma asíncrona como para el diseño y elaboración de las gráficas, unido al uso de Angular.js para la creación de la lógica de la aplicación, nos da la posibilidad de crear este tipo de aplicación. En cuanto al diseño del *DashBoard*, se hace uso del framework Bootstrap, cuyo punto fuerte radica en la adaptabilidad del diseño.

Así pues, observamos cómo, gracias a Angular, podemos construir una aplicación web *front-end* utilizando un patrón MVC. Por un lado tenemos la vista, formada por los ficheros HTML y las hojas de estilo, encargada de la interfaz de interacción con el usuario. La lógica de la aplicación viene definida por los elementos de Angular.js y se implementa en el controller, las directivas y sus servicios. Finalmente, también tenemos una base de datos formada por ficheros en formato JSON que constituyen los conjuntos de datos sobre los que trabajará la aplicación.

[1] Interfaz gráfica de usuario que proporciona mucha información de diversas fuentes en un espacio reducido con el objetivo de que el usuario lo entienda y actue en consecuencia

[2] Parte del software que interactua con el usuario

[3] Analizar sintácticamente una cadena de símbolos y actuar en consecuencia

La naturaleza de Angular nos define de forma implícita dos modelos a la hora de organizar nuestro código de forma eficiente en función de la extensión de la aplicación:

- Si la aplicación es basta, la mejor manera de estructurar el código desarrollado en Angular es por temática: agrupando aquellos componentes que actúen sobre los mismos elementos.
- Si la aplicación no es excesivamente grande, una manera más eficiente de organizar el código consiste en agrupar los componentes por su naturaleza.

Como nuestra aplicación no es excesivamente extensa, se utiliza la segunda opción de organización dando el siguiente resultado (5.1).

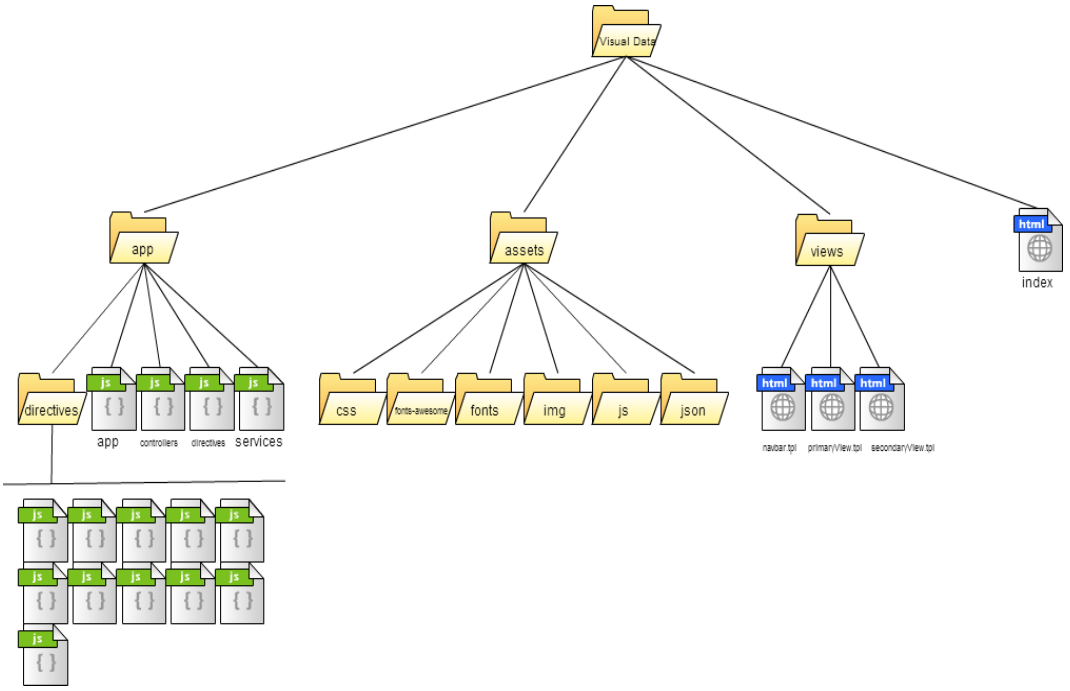

FIGURA 5.1: Estructura de la app

Podemos observar la estructura final en cuanto a organización de ficheros y directorios que tiene el proyecto. Por un lado, tenemos los componentes lógicos de Angular.js (modulos, controllers, directives...) en la carpeta app. Estos ficheros son los encargados de definir los $scopes y manipular los datos que representaremos. La carpeta assets contiene todo lo referente al diseño de la web y funciones de soporte JavaScript (hojas de estilo, fuentes, imágenes, bootstrap...). En cuanto a views, es evidente que contienen los templates[4] que conformarán nuestro *DashBoard*. Index contiene el *head*[5] donde se realizan las importaciones y define con qué combinación de templates se crea el mencionado *DashBoard*.

[4]Plantilla que determina la estructura de un sitio web
[5]Elemento HTML donde se describen los metadatos del propio documento

Vemos rápidamente cómo una de las grandes características de la aplicación es ser una de las ya mencionadas SPA. Este tipo de aplicación destaca por estar formada por una única página cuyos componentes se generan de forma dinámica en función de la interacción con el usuario. En nuestro caso, los *templates* y las directivas con sus respectivos $scopes determinarán nuestra view, que siempre se construirá sobre el fichero index.

5.2. Obtención de Datos

Es inconcebible una aplicación de visualización de datos sin conjuntos de éstos sobre los que actuar. Como se ha mencionado anteriormente, la idea sobre la que se construye la aplicación es analizar la desigualdad de género en el ámbito laboral europeo. Aprovechando el enorme potencial de los DataSets que ofrece Eurostat, se han recopilado una serie de conjuntos de datos que permiten efectuar los análisis planteados anteriormente en los objetivos. Así pues, se han utilizado seis DataSets en formato JSON que contienen los datos necesarios para el desarrollo del proyecto y que cubren las siguientes áreas:

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

Una de las grandes ventajas que te ofrece Eurostat es la capacidad de filtrar los DataSets que provee. Considerando que la aplicación debe gestionar de manera simultánea la lectura de múltiples conjuntos de datos, se ha utilizado esta herramienta para eliminar la información menos relevante para nuestro caso de estudio.

Cada uno de estos DataSets ofrece un enfoque particular al proyecto. Mirando la naturaleza del análisis, es evidente que deben existir ciertas variables comunes sobre las que se entrelazan los distintos conjuntos de datos que se utilizan. Observamos que en todos los DataSets encontramos los países de la unión europea, la variable temporal en intervalos anuales y el género sobre el que recae cada uno de los datos.

- El DataSet de Educación nos ofrece información sobre el porcentaje de hombres y mujeres que tienen diferentes niveles de educación. Cuando hablamos de educación, se suele utilizar los niveles ISCED[6] para clasificarla. Esta clasificación divide

[6]Estructura de clasificación para organizar la información sobre la educación de UNESCO

la educación en nueve niveles: 0-2 para la educación elemental o inferior (analfabetismo, primaria y primeros años de secundaria), 3-4 para la educación básica obligatoria y 5-8 para la educación posterior no obligatoria.

- El DataSet de Demográficos nos indica cuantos hombres y cuantas mujeres viven en cada país.
- El DataSet de Inversión en educación nos aporta una idea sobre los números de la inversión en euros que realiza cada país por alumno en cuanto a educación se refiere.
- El DataSet de Datos de empleo nos ofrece cifras de empleo y desempleo por género con bastante detalle. Podemos ver desde la cantidad de trabajadores por actividad laboral hasta cómo se distribuyen en función del nivel de educación adquirida. Este nivel de detalle lo convierte en uno de los pilares de la aplicación.
- El DataSet de PIB nos da una visión del *Producto Interior Bruto* que tiene cada país por habitante.
- El DataSet de Salarios nos sirve para hacernos una idea de cómo esta la media de los salarios de hombres y mujeres según su nivel de educación.

Además de estos DataSets, también contaremos con un fichero en formato TopoJSON creado a partir de un GeoJSON proveído por Eurostat. Este fichero contiene coordenadas obtenidas mediante localización de posición geográfica que nos servirá para poder crear un mapa de Europa en formato SVG[7] con D3. Cada uno de los países definidos en este fichero cuenta con un identificador NUTS[8] que nos permitirá asociarlo con los datos de los DataSets anteriores.

5.3. Desarrollo de la implementación

El primer paso a la hora de desarrollar el proyecto es definir la estructura de ficheros y directorios de éste. Una vez hecho esto, es momento de empezar a definir y diseñar nuestro *DashBoard*, cuya base la forma el fichero index. Este fichero deberá contener todas las importaciones que requiere el proyecto, tanto de scripts como de librerías y estilos. Generalmente, los frameworks y librerías los importamos desde sus respectivos repositorios, a excepción de bootstrap y tooltipster, que requieren ser descargados y guardados en nuestra carpeta assets. Definidas las importaciones que harán posible construir la aplicación, es momento de diseñar el *DashBoard*.

Después de ver una serie de *DashBoards* a modo de inspiración, llegamos a la conclusión de que éste debe organizarse mediante el uso de paneles, el tamaño de los cuales

[7] Especificación que describe gráficos vectoriales bidimensionales
[8] Conjunto de demarcaciones territoriales utilizada por la Unión Europea

dependerá de la representación que contengan. Para organizar el espacio central de la página se utilizará el sistema Grid de Bootstrap, que permitirá organizar la posición de los paneles con el uso de filas y columnas. Bootstrap define la anchura máxima de una fila como doce columnas y determinando cuantas quieres que ocupe el panel, determina la anchura de éste en consecuencia. Así pues, se define una fila como un contenedor de clase row, mientras que para las columnas, aprovechándonos del sistema de bootstrap, las definimos como col-lg-x, donde x determina el tamaño de la columna hasta un máximo de 12. Con las etiquetas que distribuirán el contenido de la página ya claras, es momento de diseñar los paneles que contendrán nuestras representaciones.

Los paneles de los *DashBoards* se suelen caracterizar por tener una región para un título que describe la representación y otra que contiene la gráfica a mostrar. La altura de la zona de representación del panel vendrá determinada por la altura del gráfico que contendrá. Podemos observar el diseño de uno de los paneles creados sin contenido alguno (5.2).

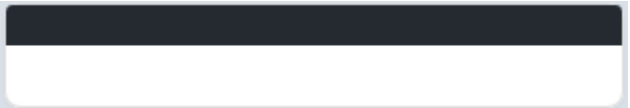

FIGURA 5.2: Ejemplo de panel con contenido vacío

También aprovecharemos para crear el contenedor que actuará como botón de inicio de transición. Este contenedor aparecerá en la región título de los paneles de aquellas gráficas que permitan transiciones. Este contenedor estará compuesto básicamente por un *background-image*[9] para simular la estética de un botón de play común en todo reproductor. Al utilizar un contenedor para esto, deberemos definir y asignarle un evento *onClick* para conseguir el comportamiento deseado. Una transición se define sobre un elemento SVG con el uso de la función .transtition. Además, el uso de esta función permite anillar múltiples funciones adicionales para configurar esta transición. Entre otras cosas, podemos definir los atributos finales del elemento, la duración de la transición y el *delay*.

Una vez definidos los elementos que nos permitirán organizar el *DashBoard*, llega el momento de obtener los datos mencionados en la sección anterior. Así pues, una vez tenemos los ficheros con los conjuntos de datos en nuestra carpeta assets/json/, el siguiente paso es analizar estos datos para plantear de qué manera se podrá interactuar con ellos en nuestra aplicación. Para analizar esto, lo primero que debemos hacer es mirar que elementos en común comparten todos nuestros DataSets. Observamos como absolutamente todos los DataSets tienen, obviamente, los países de la unión europea. Además, podemos ver como ésta no es la única variable que comparten, puesto que todos los DataSets contienen una variable temporal. Este factor ya nos da una ligera

[9]Propiedad css que define una imagen de fondo

idea de cual es nuestra intención a la hora de seleccionar y analizar los DataSets, puesto que la aparición continua de la variable temporal va estrechamente ligada al análisis de tendencias y evoluciones.

Mirándonos las hipótesis planteadas y los DataSets que deben dar respuesta a éstas, podemos dividirlos en dos tipos. Por un lado tenemos aquellos que tienen un impacto insustituible sobre el conjunto de hipótesis, mientras que por otro lado tenemos aquellos que sirven de contexto para los primeros. Este hecho se observa más claramente al ver qué DataSets serán requeridos obligatoriamente por las gráficas. Teniendo esto en cuenta, tanto el DataSet de Salario como el de Empleo se clasificarían en el primer grupo. Estos DataSets nos dan la información vital que requerimos analizar para resolver las hipótesis, mientras que el resto de ellos nos sirven para especificar y/o complementar la información que los primeros aportan.

Este hecho gana relevancia si nuestro planteamiento sobrepasa la interactividad local de una gráfica para crear un *DashBoard* en el que exista interacción global sobre éste. Es decir, que una interacción con una visualización concreta no genere únicamente un impacto sobre ella misma, sino que también pueda ser capaz de modificar el resto de elementos. Además de esto, también es necesario definir a estas alturas del desarrollo si nuestras gráficas van a tener transiciones. Si nuestras representaciones van a ser estáticas, probablemente nos interesaría extraer la variable temporal para definirla como posible interacción global, es decir, que al cambiar el año, se cambien todas las gráficas. Por otra parte, si nuestra idea es utilizar las gráficas con transiciones, no hay mejor variable sobre la que basar la transición que la temporal. En nuestro caso, al no querer únicamente visualizaciones estáticas, la mejor idea es plantear la variable temporal con interacción local o, directamente, sin interacción, puesto que no todas las gráficas reaccionan positivamente a las transiciones y algunas deben ser estáticas por naturaleza. Para definir qué variables son aptas para interacción global, debemos ver qué variables aparecen en la mayoría de los gráficos. En nuestro caso, observamos como tanto los países como el nivel de estudio ISCED son buenas candidatas.

Decididas qué variables provocarán cambios en gran parte del *DashBoard*, hay que plantear el diseño que lo haga posible, momento en el que entra en escena Angular.js. Gracias a Angular, vamos a ser capaces de implementar de manera eficiente y sencilla el planteamiento anterior y acabar generando un *DashBoard* dinámico con actualización continua de modelo y vista. Antes de incorporar angular a nuestro proyecto, hay que definir cómo se realizará la interacción global. Como tenemos dos variables interesantes para recibir este tipo de interacción, las realizaremos de la siguiente manera:

- Con un elemento del propio *DashBoard*, externo a cualquier representación, en nuestro caso un *navigation bar*.
- Con interacción sobre un elemento de una gráfica, más concretamente al lanzar un evento *onClick()* sobre ciertos elementos concretos.

En definitiva, la aplicación permitirá al usuario modificar el contenido de sus visualizaciones especificando un nivel de estudio concreto mediante un navbar[10] que se situará en la parte izquierda de la página mientras que la selección del país sobre el que se centrarán las gráficas se realizará al pulsar ciertos elementos de éstas. Podemos observar que interacción se asignará a cada gráfica y los cambios que ello originará en la siguiente figura(5.3).

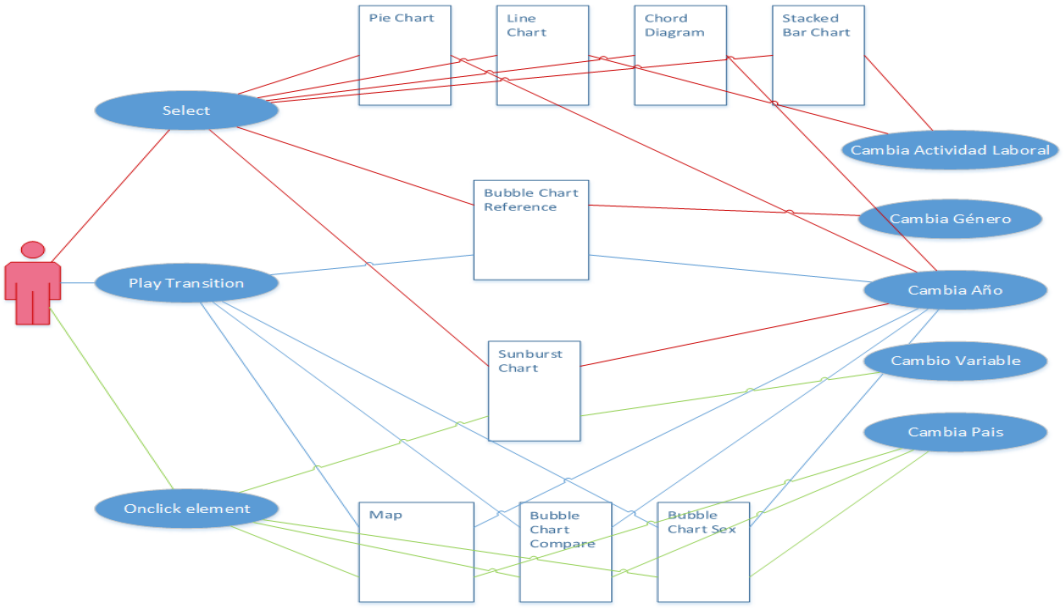

FIGURA 5.3: Diagrama de interacción con el usuario

El primer paso para crear una aplicación basada en Angular.js consiste en comunicar al navegador que la página que va a mostrar está construida sobre dicho framework. Para ello, lo primero que debemos hacer es asociar la directiva ng-app al body de nuestro HTML. Antes de comenzar la implementación, hay que entender el funcionamiento de los módulos con los que trabaja Angular. Similar a la importación de packages en Java[11], Angular sustituye a éstos por módulos que debemos definir e importar para poder utilizar las funcionalidades que nos aportan. El primer módulo que debemos incorporar se corresponde con el nombre con el que se interpretará la aplicación: visualDataApp. La declaración de cualquier modulo debe encontrarse en un fichero con extensión JavaScript que debe ser importado por el fichero HTML que use Angular. Estos módulos utilizan una estructura jerárquica de importaciones, es decir, declararemos módulos que contendrán otros módulos y así sucesivamente hasta definir la organización de los componentes Angular de nuestra aplicación. Parece evidente pues, que el primer módulo que debemos definir y que servirá de base para el resto de módulos es visualDataApp.

[10] Barra de navegación que usualmente esta formada por una serie de enlaces
[11] Contenedor de clases que permite su organizarlas en agrupaciones de funcionalidades comunes

A su vez, este módulo deberá importar cada uno de los diferentes módulos encargados de definir los componentes Angular que utilizará nuestra aplicación. Mirando la organización estructural definida anteriormente, vemos que nuestra aplicación utilizará tres tipos de componentes de Angular: services, directives y controllers.

Una vez definido el sistema de modulación en el que se basa Angular, es hora de aplicarlo a nuestra implementación. El primer paso es crear el fichero app.js. Este fichero contendrá el modulo raíz de la aplicación Angular que a su vez importará los módulos de services, directives y controllers y se definirá de la siguiente manera:

```
angular.module('visualDataApp',['visualDataApp.controllers',
    'visualDataApp.services','visualDataApp.directives']);
```

El primer parámetro corresponde al nombre del módulo (al tratarse del módulo raíz, debe corresponderse con el valor de ng-app en nuestro fichero HTML), mientras que en el segundo tenemos las importaciones que se derivan de este módulo. Considerando que nos encontramos en el primer fichero compilado y ejecutado por el navegador, este script también nos es útil para incorporar aquellas funciones JavaScript que queremos o necesitamos que se ejecuten al inicio la aplicación.

A partir de aquí, el navegador ya interpretará la aplicación como aplicación Angular y podemos utilizar sus múltiples funcionalidades. El siguiente paso es comenzar a tratar la naturaleza de los componentes que incluirá nuestra aplicación y como trabajamos con ellos para acabar construyendo nuestra página de visualización de datos:

- **Directives**. Cada una de las directivas que definiremos se corresponderá a un tipo de gráfico que contendrá nuestra aplicación. Estos gráficos estarán construidos gracias a D3.js y vinculados a la vista.
- **Controllers**. Pese a que una aplicación Angular puede tener diversos controllers organizados de manera jerárquica (de forma similar a los scopes), nuestra aplicación únicamente contará con un controller que se ejecutará al inicio de la aplicación.
- **Services**. A diferencia de los otros dos, se trata de componentes menos intuitivos y que requieren más complejidad para introducirlos a la aplicación. En nuestro caso, utilizamos los *services* para inyectar aquellos objetos que debamos tener instanciados una única vez en el controller (por ejemplo, los datos que visualizaremos).

Al igual que ocurría con la directiva ng-app, para comunicar al navegador la existencia de un controller se utiliza la directiva ng-controller. El valor de esta etiqueta deberá contener el nombre con el que se identifica el controller de nuestra aplicación. Esta directiva no va asociada al body del index tal y como pasaba con la anterior, sino que deberá vincularse a otra etiqueta HTML. Teniendo en cuenta que la aplicación diseñada

únicamente contará con un controller, nos interesa ligar esta directiva a un contenedor que englobe todo nuestro *DashBoard*. Dicho esto, es momento de crear y definir el módulo asociado:

```
angular.module('visualDataApp.controllers',[])
```

Vemos como, a diferencia del módulo principal, este módulo no importa sus propios módulos. Esto se debe a que como únicamente tenemos un controller, éste se definirá dentro de este módulo, ahorrándonos sucesivas importaciones. Una vez hecho esto, necesitaremos los datos que el controller se encargará de manipular y gestionar antes de comenzar a implementarlo, lo que nos lleva inevitablemente a definir previamente los services que nos proporcionarán dichos datos.

Los services se utilizaran en el controller gracias a la inyección de dependencias de Angular por lo que, a diferencia de todos los componentes vistos hasta ahora, no requieren asociarlos a ninguna etiqueta HTML. Será el propio controller el que comunicará que debe interpretarse y ejecutarse un service en el momento en el que detecte que tiene una dependencia inyectada. Primero de todo, debemos definir el módulo que contendrá los services, en este caso:

```
angular.module('visualDataApp.services',[])
```

Igual que con el módulo de controllers, definiremos los services utilizados en el propio módulo puesto que no utilizaremos gran cantidad de estos. Antes de comenzar a definir cualquier service, hay que distinguir entre las diferentes posibilidades que éstos nos otorgan:

- **Service**. Los services proveen al controller con la instancia de la función que implementa el service, con lo que cada inyección es una instanciación con su propia *keyword*.
- **Factory**. Las factorías proveen al controller con el objeto y las propiedades que se retornan de la función que implementa la factoría.
- **Provider**. Constituyen el único servicio que se puede pasar a la función *config()*. Útiles a la hora de proveer configuraciones de módulos.

Para cargar los datos de los JSON que contienen los DataSets sobre los que construiremos la aplicación, utilizaremos una factoría. Esta factoría se encargará de cargar los datos sobre los que trabajará la aplicación, construirá el objeto que los contendrá y inyectará dicho objeto en el controller para poder configurar la disposición inicial de nuestro *DashBoard*. Esta factoría se define de la siguiente manera:

```
    .factory('loadJSON', function);
```

Esta factoría utilizará la funcionalidad d3.json de la librería D3.js múltiples veces para cargar el contenido JSON de los diferentes ficheros que conforman nuestros DataSets y los almacenará en objetos JavaScript. Como se ha mencionado anteriormente, está funcionalidad de D3.js carga los datos de manera asíncrona, dicho de forma más clara, la aplicación no parará su flujo de ejecución hasta que la carga de los ficheros se haya completado, como suele ser habitual en los lenguajes de programación, sino que la ejecución sigue adelante y, progresivamente, éstos se van cargando. Este hecho aumenta la eficiencia de nuestra aplicación de forma considerable (el resto de componentes de la aplicación no restan a la espera hasta que se ha realizado la carga de todos los ficheros), pero por otra parte nos originará un conflicto extra con su inyección. Así pues, se realiza una carga de ficheros anillada en la que cada DataSet constituye el valor asociado a una key dentro del objeto que retornará esta factoría, se definen métodos para seleccionar cada uno de los DataSets y se retorna el objeto para que se pueda inyectar en el controller. El problema comentado radica en el contenido de este objeto: puesto que la carga se hace de forma asíncrona, el objeto retornado es de tipo 'undefined' hasta que ésta se realiza. Esta situación nos originará errores al intentar obtener estos datos en el controller y asignarlos al $scope. Nos encontramos ante una situación bastante frecuente cuando se trabaja con AJAX y para solucionarlo existen implementaciones de Angular y JQuery llamadas *deferred* y *promises*.

Estos dos nuevos objetos incorporados en JQuery a partir de la versión 1.5 nos permiten trabajar con los callbacks resultantes de las peticiones AJAX de forma más eficiente y confeccionar aplicaciones más robustas. Un objeto *deferred* representa un objeto que todavía no ha realizado su trabajo y, por lo tanto, cuyo valor es desconocido. Además, este objeto tiene asociado otro, llamado *promise*, en el cual se guardará el valor resultante del trabajo realizado por el objeto *deferred*. Para potenciar dichos callbacks, también se encuentra implementada la función $then, que sostiene la ejecución del código que contiene hasta que el trabajo se ha realizado y conocemos el valor del *promise*. Para poder utilizar los objetos deferred i promise en la factoría es necesario que se pase como parámetro a la función de ésta el servicio $q implementado por angular. El siguiente diagrama (5.4) ofrece una visión más clara del funcionamiento del sistema deferred/promise y del flujo de ejecución que sigue.

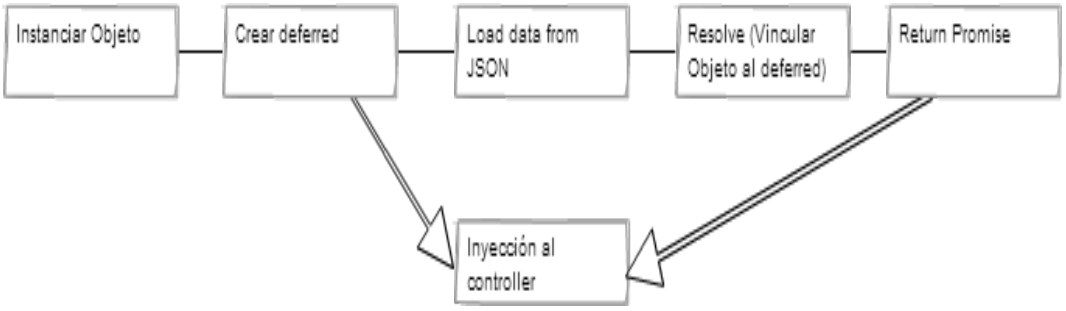

FIGURA 5.4: Diagrama de flujo de deferred/promise

Además, necesitaremos definir otro *service* adicional que nos provea de un diccionario que funcione como una tabla de correspondencia entre el identificador NUTS, que nos definirá un país en el mapa SVG, y el nombre del país utilizado en los DataSets. Este servicio también se inyectará, como la factoría anterior, en el controller.

```
.service('diccionarioEuropa',function);
```

La inyección se realiza introduciendo el nombre del *service* como parámetro en el controller. Además este deberá recibir también el $scope para poder utilizarlo y guardar variables en él. Como todo lo que hagamos en el controller necesita que los datos de los JSON estén cargados, todo su código se encapsula dentro del ya mencionado $then. Para poder utilizar los datos de manera eficiente en nuestras representaciones, debemos organizar la información de los objetos que contienen los datos extraídos mediante d3.json utilizando jerarquías y agrupaciones. Los ficheros JSON están constituidos por un array de objetos en el que cada elemento del array es un valor de la tabla. Para poder gestionar estos datos de forma eficiente, es necesario que agrupemos estos valores en función del país al que pertenecen, el año al que corresponden, la unidad de medida utilizada, etc. Para poder estructurar los datos que utilizará la aplicación se implementará una función utilizando el enorme potencial de Underscore.js. Esta función recibe como parámetros las keys por las que se agruparán los datos y mediante el uso de *groupBy* y *slice* podremos obtener estructuras similares a:

```
{Pais1:
 {Year1:value,
  Year2:value,
  Year3:value
 },
 Pais2:
 {Year1:value,
  Year2:value,
  Year3:value
 },
 Pais3:
 {Year1:value,
  Year2:value,
  Year3:value
```

```
 }
}
```

Así pues, nuestro controller almacenará en su $scope, accesible desde cualquier gráfica, los datos extraídos de los DataSets ya organizados gracias a la función anterior, los datos para crear el mapa SVG y el diccionario de correspondencia. Las variables de interacción global, las que afectan a múltiples gráficas, deberán controlarse con este $scope ya que podrá ser leído por cualquier directiva creada. Así pues, también debemos almacenar en el $scope los valores de visualización inicial de país y de nivel de educación. Se ha considerado inicializar estos valores con los más globales para después ir especificando interactivamente, con lo que inicializamos con el conjunto de la Unión Europea y todos los niveles de educación. Además de todo esto, también guardaremos en el $scope, por comodidad, las diferentes keys por las que hemos agrupado los datos con los que trabajaremos.

Con toda esta preparación previa, ya estamos listos para comenzar a definir nuestras directivas personalizadas y diseñar los gráficos que constituyen, en definitiva, la esencia del proyecto.

5.3.1. Directives

El último de los tres módulos comentados previamente que resta por definir e importar. Todas las directivas que definiremos a lo largo del proyecto se importaran desde este módulo, hasta acabar teniendo:

```
angular.module('visualDataApp.directives',
    ['visualDataApp.directives.myMapDirective',
    'visualDataApp.directives.myChartDirective',
    'visualDataApp.directives.myStackedBarChartDirective',
    'visualDataApp.directives.myPieChartDirective',
    'visualDataApp.directives.myBubbleChartSexDirective',
    'visualDataApp.directives.myBubbleChartCompareDirective',
    'visualDataApp.directives.myBubbleChartReferenceDirective',
    'visualDataApp.directives.myAreaChartDirective',
    'visualDataApp.directives.myLineBarChartDirective',
    'visualDataApp.directives.mySunburstChartDirective',
    'visualDataApp.directives.myChordDiagramDirective'])
```

Vemos como cada una de las directivas será un módulo independiente importado por éste. Esto nos permitirá organizar cada directiva en un fichero y poder trabajar con cada una independientemente del resto. Para definir una directiva utilizamos la siguiente funcionalidad angular:

```
.directive(directiveName,function);
```

Una directiva siempre va a retornar un objeto JavaScript que define, entre otras cosas, la función que se ejecutará al iniciarse la directiva, llamada función *link*. Cada directiva tendrá su propio $scope desde el cual se puede acceder al $scope del controller, gracias al uso de $parent. Otras opciones que te permite configurar una directiva son: especificar el comportamiento del $scope, asignar el template resultante tras la compilación y restringir el modo de notación de la propia directiva. Todas las directivas utilizadas para la definición de gráficas instancian un $scope que hereda del $scope padre, en este caso, del controller, siempre y cuando especifiquemos su comportamiento con el valor booleano true. Esto permite a las directivas poder observar y modificar en cada momento los valores del $scope del controller.

Gracias a los $scopes y la función $watch, podemos asignar listeners a valores de un $scope y lanzar eventos cuando se detecten cambios sobre éstos. Hay que tener en cuenta que la propia creación de un valor nuevo en el $scope también hace saltar el evento asignado al $watch. Cabe destacar que la función *link* debe recibir como parámetros tanto el $scope como el elemento sobre el que se pretende representar la gráfica, en este caso, el panel correspondiente. Esta función se encarga principalmente de tres tareas: inicializar las variables específicas de la gráfica (recordemos que el controller dispone las inicializaciones de las variables de interacción global), asignar listeners a las variables de interacción que tendrán repercusión sobre esta gráfica (estas variables se almacenan en los $scopes) y dibujar la gráfica utilizando D3.js.

Para inicializar los valores del $scope de una directiva, debemos asegurarnos de que se realice después de la carga asíncrona de los datos. En caso contrario, inundaríamos el $scope con *undefineds* que, obviamente, no nos darían los resultados esperados. Así pues, utilizaremos la funcionalidad $watch para ejecutar la función cuando se produzca un cambio sobre alguno de los valores del $scope del controller. Hay que tener en cuenta que la propia creación del valor ya lanza el evento del $watch, así que únicamente debemos ejecutar el código cuando el valor que se capturando no sea *undefined*.

Todas las gráficas que diseñaremos tendrá la anchura del panel que contendrá la representación, definida mediante el sistema grid de bootstrap. A la hora de crear la gráfica, un esquema habitual utilizado suele ser el siguiente (5.5), donde podemos hacernos una idea de los pasos que requiere crear una visualización con D3.js.

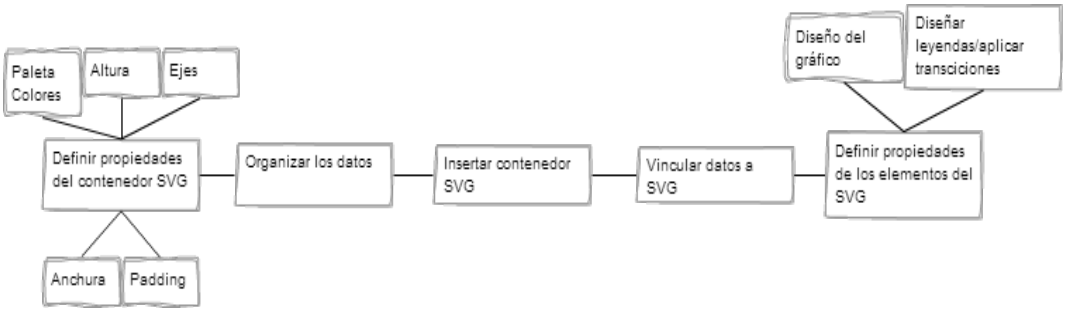

FIGURA 5.5: Esquema creación de gráfica con D3.js

5.3.1.1. Map

Esta directiva creará el mapa SVG que se ha ido comentando anteriormente dentro de uno de los paneles de nuestro *DashBoard*. La representación será un Cloropeth Map interactivo sobre la variable temporal en el que se podrá observar la diferencia porcentual que existe entre el número de hombres y de mujeres que tienen empleo y como ésta va variando a lo largo de los años. El potencial de este tipo de gráfico no sólo nos permitirá ver, de forma bastante intuitiva, la situación de los diferentes países, sino también intuir las tendencias locales y globales que está experimentando Europa así como poder comparar individualmente los países entre ellos. La interacción sobre este gráfico se realizará mediante un botón que iniciará transiciones sobre los elementos que conformarán esta gráfica.

Además, puesto que los elementos que conformarán este mapa se corresponden a los países de la unión europea que aparecen en los DataSets que utilizamos, es el candidato idóneo sobre el que recae la variable de interacción global que nos falta por definir: el país actual. Esta gráfica permitirá al usuario poder seleccionar un país al pulsar sobre éste en el mapa y dicha acción derivará en una serie de cambios sobre el resto de gráficas del *DashBoard*.

Una particularidad de esta directiva es la vinculación de la función *link* para su ejecución previa a la compilación y la función *postlink* para la ejecución posterior.

Además de inicializar los valores del $scope, debemos asignar un listener al otro elemento de interacción global, el nivel de educación, porque un cambio sobre éste también implicará volver a crear la gráfica. Para poder crear un mapa con D3.js hay que configurar propiedades extras que permiten la representación de los arcos definidos en el fichero TopoJSON. Primero de todo, debemos definir la proyección con la que se interpretarán estas coordenadas y el clip de visualización de esta proyección. D3 nos da la posibilidad de configurar diferentes características de nuestra proyección indicando, por ejemplo: centro, escalas y traslaciones.

Una vez definido todo esto, es momento de asignar los datos, crear los elementos del SVG correspondientes y personalizarlos estéticamente. Se utiliza la función .data para asignar un array de datos a un elemento SVG, lo que nos permite la creación dinámica de éstos. La diferencia respecto a un gráfico más estándar, radica en cómo conseguir que D3 interprete y dibuje las coordenadas de *geolocalización*. La librería TopoJSON, que hemos importado previamente, nos ayudará a leer este formato de datos para poder asignarlos al SVG y conseguir representar nuestro mapa con sus respectivos países. Para poder determinar diferentes elementos en el mapa resultante es necesario que el fichero TopoJSON tenga organizados los arcos separados por países y no asignados a un único objeto. Si esto se cumple, podremos definir cada uno de los países como un único elemento dentro del mapa SVG y trabajar sobre ellos individualmente.

Una vez asignados los datos, se utiliza la función *enter* para crear los elementos SVG correspondientes y se les asigna tanto una clase común a todos los países como una propia con el nombre del país para poder identificarlos. Para poder hacer esto, es necesario que en el fichero TopoJSON se haga referencia al país al que pertenecen las coordenadas. Una vez dibujado el mapa, solo queda definir el color con el que se rellenará cada país modificando el valor de la propiedad de estilo *fill*.

Creados todos los países que conforman la Unión Europea y visualizados en nuestro mapa, debemos definir sus fronteras. Las fronteras no van asociadas a cada país, sino que constituye un único elemento SVG adicional que se obtiene mediante el uso de la función *mesh* de la librería TopoJSON. Exactamente como ocurría con los países, procedemos a asignarle una clase y cambiarle el estilo.

Los datos que darán sentido a este mapa los obtenemos del DataSet de datos de empleo. Pese a que este conjunto de datos dispone de muchísima información adicional como tipos de actividad y diferentes unidades de medida, para éste gráfico en particular utilizaremos cifras generales de cuantas personas están trabajando por género y variaremos el nivel de educación escogido en función del $scope del controller.

Una particularidad de este estilo de mapa reside en que los datos asignados a los elementos que lo conforman no son los datos que vamos a utilizar para el análisis, sino que son aquellos que nos permiten dibujarlo. Esto nos fuerza a utilizar el identificador NUTS del país para poder acceder, mediante el diccionario de correspondencia, a los datos de empleo del país que representa el elemento SVG que se está tratando. En el resto de gráficos, cada elemento se crea con sus datos correspondientes asociados, con lo que nos ahorramos determinar el elemento en el que nos encontramos e identificar su *alter ego* en el conjunto de datos a analizar. Para obtener el valor según el cual debemos pintar un elemento, se calcula el porcentaje que representa la diferencia de empleados y empleadas de un país sobre el total de trabajadores. Obtendremos un porcentaje positivo si el nombre de hombres trabajando es superior, mientras que la situación inversa se representa con un porcentaje negativo.

A nivel de diseño estético del mapa, se ha escogido un tono rosáceo para representar superioridad de empleadas mientras que un tono azul hace lo propio con los empleados. La elección de ambos colores es un tanto obvia, ya que esos colores son considerados un estándar y prácticamente todo el mundo los relaciona instintivamente. Además, se utiliza una paleta de colores divergente para definir las distintas diferencias que se establecen utilizando el blanco como punto neutral. Este tipo de rango de colores es especialmente útil cuando queremos visualizar diferentes rangos superiores e inferiores respecto a un punto, en nuestro caso, la igualdad en el número de empleados y empleadas. Observamos 5.6 el resultado del mapa ya dibujado.

FIGURA 5.6: Mapa dibujado con D3.js

Otro punto interesante que se puede aplicar sobre los elementos que conforman el SVG es la incorporación de los eventos propios de JavaScript. En este gráfico, provocaremos una reducción de la opacidad de un país al hacer *mouseover* sobre el, mientras que al hacer *mouseleave*, restauramos su valor por defecto. Puesto que se trata de una gráfica con transiciones, aprovecharemos para incluir, en la función donde se crea el mapa, la inserción del botón de transición. Además de añadir este nodo al DOM, también debemos asignarle el evento *onClick* y definir su comportamiento, en este caso, iniciar las transiciones de los elementos del SVG.

La idea de las transiciones sobre esta gráfica es conseguir un efecto similar a una reproducción en la que los años vayan avanzando y las regiones del mapa vayan variando de color en función de los datos asociados. Una transición en D3 se aplica a un elemento del SVG y, tras definir su duración y *delay* correspondiente, utiliza la interpolación de las propiedades de estilo del elemento al inicio de la transición y las que determinamos que tendrá al final de ésta para lograr el efecto deseado.

En definitiva, necesitamos tantas transiciones como años tengamos en el conjunto de datos que queremos representar. En este caso, necesitamos que al acabar una transición, se genere otra con el año siguiente. Este comportamiento se puede configurar gracias al uso de la funcionalidad .each, que nos permite definir una función que se ejecutará para cada elemento del SVG. Esto, combinado con el evento *end* que se lanza cuando una transición finaliza, nos permite encadenar transiciones.

```
element.transition().each('end',function)
```

Así pues, implementando las transiciones en bucle con estos recursos, únicamente debemos definir la función que se encarga de aplicarla y controlar cuando no se deben realizar más: cuando el año actual de la última transición ya sea el último del DataSet. Hay que tener en cuenta que la función se llamará cada vez que un elemento del SVG tenga que hacer una transición extra. Finalmente, para lograr el efecto de Cloropeth Map dinámico que deseamos, sólo queda definir, en cada transición, el color con el que se deberá rellenar cada elemento del SVG analizando los datos correspondientes a esa transición.

Finalmente, únicamente nos queda definir un segundo SVG fundamental para completar e interpretar el mapa: la leyenda. Ésta constará de una serie de rectángulos dispuestos en fila, debajo de los cuales encontraremos los porcentajes con los que se interpretan los diferentes colores de la paleta divergente con las que están pintados los elementos del mapa. La leyenda creada resultante podemos observarla aquí (5.7).

El último retoque sobre este gráfico consiste en la incorporación del año que se está visualizando en la transición actual. Obviamente, éste también deberá implementar las mismas transiciones que los elementos del mapa e ir modificando el propio $scope que contiene el valor actual de la variable temporal.

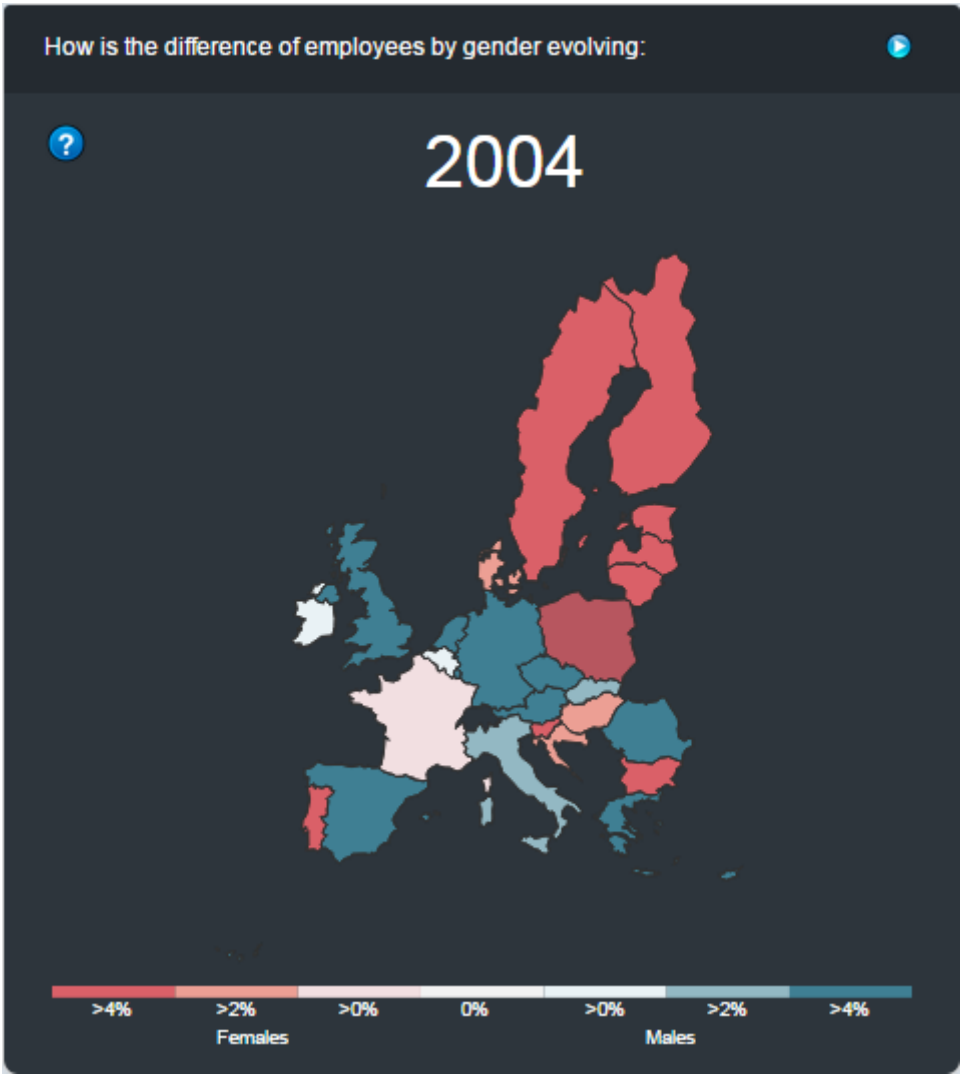

FIGURA 5.7: Resultado Final del gráfico con el nivel de educación más alto

El botón con el signo de interrogación que podemos ver en el panel completo (5.7) se crea mediante la librería tooltipster en la función *postlink* mencionada anteriormente. El tooltip que aparece al pulsar sobre el icono nos ofrece cierta información adicional sobre el gráfico y cómo interactuar con él.

5.3.1.2. LineChart

Esta directiva creará un *Line Chart* 2D dentro de uno de los paneles de nuestro *DashBoard*. Esta representación contará con la variable temporal en el eje horizontal y el número de empleados sobre el eje vertical. Mediante un selector HTML, el usuario podrá interactuar con esta gráfica y, además, modificará su contenido en función de los valores de las dos variables globales de nuestro *DashBoard*. En el selector, el usuario podrá indicar qué actividad laboral quiere visualizar.

Esta gráfica estará formada por dos lineas independientes formadas por sus respectivos puntos: una linea representará al género masculino mientras que la otra hará lo propio con el femenino. Considerando las variables que forman los ejes, podemos resolver que la gráfica nos mostrará las tendencias en cuanto al número de empleados y empleadas que hay en una región durante un periodo de tiempo concreto. Además, gracias a la interacción con el usuario, también se podrá determinar el país, la actividad laboral y el nivel de educación a los cuales hace referencia la visualización. Los colores utilizados para dibujar ambas lineas se corresponden con los dos colores que originaban la paleta divergente de la directiva anterior.

Esta directiva únicamente contará con una función, *link*, que asignará los listeners a aquellas variables cuya modificación implique volver a crear la gráfica: nivel de educación y país, controlados por el $scope del controller, y, la actividad laboral, por el $scope de la propia directiva. A partir de aquí, debemos crear las escalas con las que trabajará nuestra representación en los ejes vertical y horizontal. D3 tiene la función scale que define un objeto de tipo escala al que únicamente hay que configurarle el rango y el dominio de los valores sobre los que trabajará, además de especificar que tipo de escala desea: lineal, logarítmica, identidad, etc. Después de crear ambas escalas, es el turno de los ejes cartesianos con los que contará nuestra gráfica. Al igual que con las escalas, D3 también dispone de un objeto de tipo eje, creado mediante la función axis que, para utilizarlo adecuadamente, debemos configurar indicándole que escala de las que hemos creado previamente hay que asignarle. Además, también se pueden personalizar indicando la orientación, la cantidad de marcas(*ticks*) que debe tener y el formato de éstas.

Para asignar los datos a elementos SVG mediante la función .data, es necesario que dichos datos estén almacenados en un array de JavaScript, donde cada posición del array se corresponderá con un elemento del gráfico. Así pues, es necesario adaptar los datos que vayamos a asociar e intentar desechar aquellos que no serán utilizados en nuestra gráfica para reducir cualquier coste computacional que nos podamos ahorrar. En particular para esta gráfica, que muestra los datos para el país actual ($scope del controller), los datos deberán tener una estructura similar a la siguiente:

```
datos = [
 {year:{'value':valor,'sexo':Males'}},
 {year:{'value':valor,'sexo':Males'}},
 {year:{'value':valor,'sexo':Females'}},
 {year:{'value':valor,'sexo':Females'}}
]
```

Una vez creado e insertado el contenedor SVG en el panel, los primeros elementos que debemos definir serán los puntos que definen ambas rectas. Dibujaremos estos puntos como círculos a los que asignaremos eventos tal como hicimos con los elementos de la directiva anterior. Así pues, puesto que estos círculos son en realidad los que contienen

los datos de nuestra gráfica (de hecho, las lineas únicamente toman distintos puntos y los unen) también son los elementos que deben tener asociados los datos que queremos visualizar. Cada uno de los puntos creados tiene asignado como dato uno de los objetos contenidos en el array comentado anteriormente. Con la key del año y el valor, definimos el posicionamiento de cada uno de los elementos en nuestro espacio de coordenadas, considerando siempre la escala correspondiente.

Una vez tenemos los puntos dibujados en nuestra gráfica el siguiente paso es crear la sucesión de lineas que combinen estos puntos, teniendo en cuenta que una linea debe unir únicamente los puntos referentes al género masculino y, la otra, al femenino. Cada linea es un objeto de tipo path que recibe las coordenadas del conjunto de puntos que debe unir mediante la función que define una linea, .line, y la función .interpolate de ésta. Con los dos path ya insertados en el SVG con sus respectivos estilos estéticos, sólo queda añadir a la gráfica los ejes para tener el diseño completo. Cada eje se incorpora al SVG utilizando un objeto de tipo agrupación ('g') que utiliza la función .call con el eje como parámetro. Una vez hecho esto, este es el gráfico resultante (5.8).

Pero todavía no esta todo hecho, queda añadir las opciones a escoger en el selector HTML e implementar los comportamientos de respuesta a los eventos de los círculos, tal y como habíamos mencionado anteriormente. Las opciones disponibles de este selector también las crearemos utilizando D3, ja que, como éste permite crear cualquier tipo de elemento HTML (no únicamente SVG), utilizaremos la asociación de datos junto la función .append para insertar los <option> dinámicamente dentro de la etiqueta <select> definida en el HTML. Después de esto, se le asigna al <select> el evento *onchange* para detectar cambios sobre la opción escogida y utilizamos la propiedad value para obtener el valor del selector, que se guardará en el $scope e influirá en el gráfico.

Para potenciar la inmersión del usuario y, al mismo tiempo, ofrecerle más información de manera simple y minimalista, se va a mostrar un tooltip que mostrará información adicional de los datos que contiene el círculo sobre el que se haga un *mouseover*. El tooltip debe aparecer en una posición próxima al cursor del usuario cuando salta el evento, para lo cual volvemos a utilizar una funcionalidad de D3, d3.mouse, para obtener la posición absoluta del cursor y insertar el tooltip en esa posición. Con todo ya listo e implementado, la apariencia final del panel que contiene este Line Chart es la siguiente (5.8).

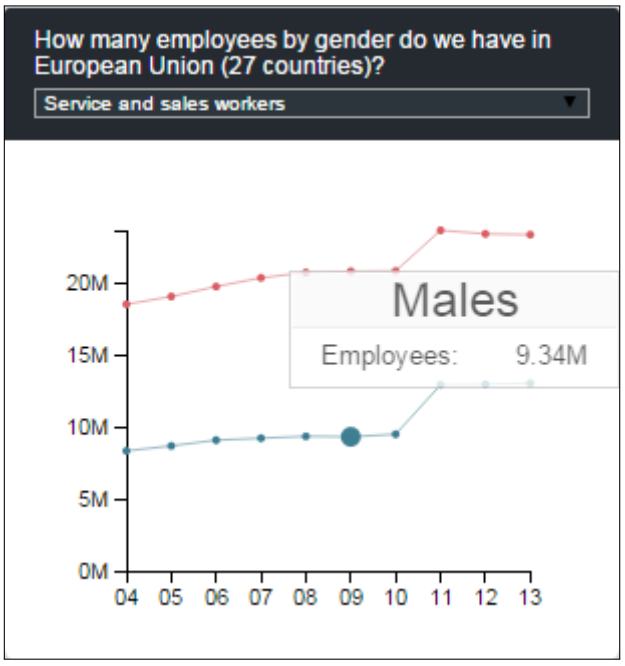

FIGURA 5.8: Gráfico Line Chart completado

5.3.1.3. StackedBarChart

Esta directiva creará una 100 % Stacked Bar Chart dentro de uno de los paneles de nuestro *DashBoard*. Esta representación estará formada por un conjunto de barras de tamaño fijo, más concretamente, creará tantas barras como valores de la variable temporal encontramos en el DataSet de empleo. Las barras estarán dispuestas en fila y divididas en dos segmentos: uno representará el porcentaje de empleo masculino y, el otro, el femenino. Tal y como ocurría en la directiva anterior, los datos asociados a esta visualización estarán sujetos al valor de las variables de interacción global. Además, esta gráfica también contará con la misma interacción local que definimos para la gráfica anterior: un selector HTML con la actividad laboral relacionada.

Como se puede intuir, los datos que observamos asociados a esta gráfica son los mismos que utilizamos para el *Line Chart*, pero la disposición y naturaleza propia del gráfico refuerza el enfoque comparativo de ambos conjuntos y cómo esta comparación evoluciona durante un intervalo concreto de tiempo. En contra, se hace ciertamente más complicado analizar las tendencias de cada uno de los conjuntos de forma independiente. Exactamente como ocurría hasta ahora, el segmento de barra asociada al género masculino volverá a representarse con el color azulado mientras que el femenino lo hará con el rosáceo.

La diferencia más notable que existe respecto a la directiva anterior, la encontramos en el diseño del contenido de la propia gráfica. Por un lado, los valores sobre los que trabajará esta representación dejan de ser cifras absolutas para dejar paso a los porcentajes, cuyo

valor se calcula respecto al total de empleados, representado con el 100 %. Únicamente contaremos con un eje, el horizontal, que servirá como referencia de la variable temporal, puesto que la utilidad del eje vertical en este tipo de gráfico desciende ostensiblemente. Cada una de las barras que conformarán la gráfica estará formada por dos rectángulos, cada uno de los cuales representa uno de los dos segmentos comentados previamente. Considerando que D3 toma como partida de referencia la esquina superior izquierda, el segmento que ocupará la zona superior de la barra será el primero que debamos dibujar. El segundo rectángulo toma como origen el punto en el que finaliza el rectángulo superior. En este tipo de gráfico resulta imprescindible que el orden de los segmentos de todas las barras sea idéntico, en caso contrario, estaríamos incurriendo en unos defectos de visualización considerables. Aquí (5.9) podemos observar el aspecto final que tiene el diseño del Stacked Bar Chart que hemos implementado.

Una vez definido el diseño del gráfico, es momento de centrarse en los detalles de éste. A diferencia del anterior, este gráfico no contará con los eventos y tooltips que generaban los elementos de la representación anterior, sino que utilizará una leyenda que ayudará al observador a identificar cada color con su conjunto correspondiente. Esta leyenda mantendrá la orientación de las barras del gráfico para acabar teniendo dos rectángulos dispuestos verticalmente uno encima del otro. Finalmente, exactamente igual que en la directiva anterior, se utiliza D3 para añadir las <option>del <select>de forma dinámica y se asigna el mismo evento y comportamiento resultante anterior. Finalmente, el panel definitivo tendrá el siguiente aspecto (5.9).

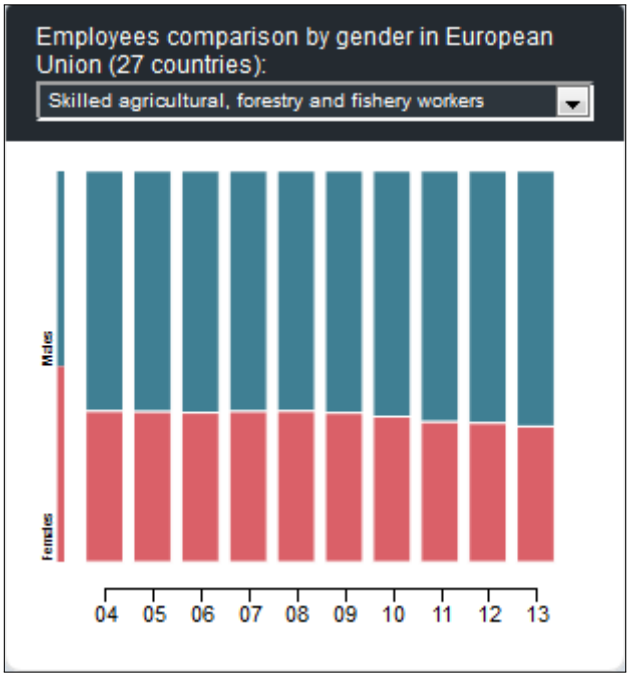

Figura 5.9: Gráfico Stacked Bar Chart completado

5.3.1.4. PieChart

Esta directiva dibujará, en uno de los paneles de nuestro *DashBoard*, uno de los gráficos mas populares que existen, un Pie Chart. Una vez asociados los datos encontrados en el DataSet de empleo que hemos utilizado en las dos directivas previas, nos servirá para fortalecer todavía más el análisis comparativo entre el número de empleados y empleadas. Además, en este gráfico se ha utilizado la variable temporal como herramienta de interacción en el selector HTML, con lo que se ha eliminado la actividad laboral que podíamos encontrar en la gráficas anteriores. En este caso, todos los datos que utiliza la gráfica cogen el total de las actividades por defecto.

Para crear este tipo de gráfico utilizamos el layout de pie chart que viene ya implementado por la librería D3 utilizando:

```
d3.layout.pie().value(datos);
```

Asignando a la propiedad value del layout los datos correctamente formateados, se crea la base del SVG que conformará nuestro Pie Chart. Además de esto, debemos construir los arcos que definen la circunferencia exterior e interior del gráfico, indicando su radio, y que otorga la forma deseada a nuestra visualización. El radio de la circunferencia interior debería ser cero siempre que no queramos convertir nuestro Pie Chart estándar en un Donut Pie Chart. Los datos que queramos asociar al layout deberán seguir la siguiente estructura:

```
[{'label':gender,
 'value':value,
 'percentage':percentage
 },
 {'label':gender,
 'value':value,
 'percentage':percentage
 }]
```

Para crear el Pie Chart se asocia como datos, a los arcos que definen la figura, el layout. Esto generará una visualización básica que podemos personalizar exactamente igual que hacemos con el resto de elementos SVG, asignamos los colores correspondientes a cada conjunto y perfilamos el contorno exterior. Como hemos mencionado anteriormente, esta vez la interacción local se realizará sobre la variable temporal y, al igual que las directivas anteriores que contenían un <select>, sus posibles opciones se añaden con D3 dinámicamente.

Existen diferentes maneras de indicar al usuario cómo interpretar la información que éste está visualizando, posiblemente la creación de una leyenda sea la más habitual. En nuestro caso, impera la idea de reducir al máximo posible la cantidad de leyendas de nuestro *DashBoard*. En nuestra aplicación, la interactividad con el usuario tiene un

peso importantísimo en las visualizaciones, con lo que se ha optado por transmitir esta información utilizando un tooltip al igual que pasaba en la directiva del Line Chart. El tooltip, que seguirá el mismo diseño estético que el anterior, nos indicará el porcentaje que representa, respecto al total, el conjunto sobre el que hagamos un evento *mouseover*. Así pues, el panel definitivo correspondiente a esta directiva tendrá el siguiente aspecto (5.10).

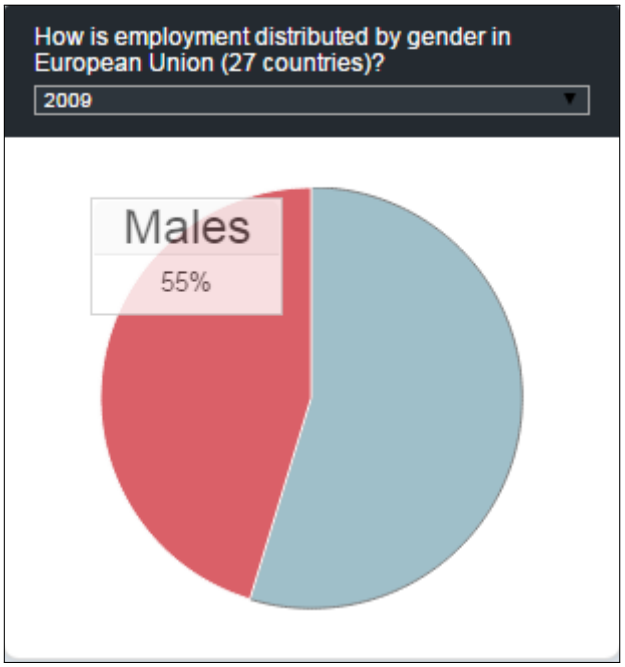

FIGURA 5.10: Gráfico Pie Chart completado

5.3.1.5. BubbleChartSex

Nos encontramos ante la única directiva que se utilizará más de una vez para generar gráficos independientes con el objetivo aprovechar el potencial del $scope de cada instancia de la directiva. Concretamente, se utilizará dos veces, cada una para crear un Bubble Chart con datos referentes a cada conjunto que estamos analizando, es decir, acabaremos teniendo un Bubble Chart para cada género. Los Bubble Chart generados utilizarán la variable temporal para crear transiciones, tal como hicimos para definir el mapa. Así pues, contaremos con un botón que iniciará las transiciones y podremos ver como la posición de cada una de las burbujas del gráfico va cambiando con el paso de los años. Además de la transición, uno de los puntos fuertes de los Bubble Chart radica en su capacidad de representar valores de tres variables: eje horizontal, vertical y diámetro del círculo.

Aprovecharemos esta característica para observar si existe correlación entre el salario que cobra de media cada conjunto y el PIB de todos los países que conforman la Unión Europea. La única variable de interacción que sugestionará los datos que representará el

gráfico es el nivel de educación, en adición a la ya mencionada variable temporal, puesto que en esta gráfica siempre tendremos todos los países, representados por burbujas, independientemente del contenido de la variable correspondiente en el $scope del controller. En definitiva, este gráfico estará formado por una serie de círculos cuyo diámetro vendrá definido por la población del país que represente, el eje horizontal nos indicará el salario medio existente en ese país mientras que el eje vertical hará lo propio con el PIB por habitante del país. A diferencia de las directivas creadas hasta ahora, este gráfico representará datos de múltiples DataSets, concretamente el de salario, el de demográficos y el de PIB. Al obtener los datos de distintas fuentes y almacenados en diferentes variables del $scope del controller, es bastante evidente la necesidad de organizar los datos necesarios para crear la representación.

Al leer datos de diferentes DataSets, podemos recibir dominios de datos comunes que no estén en sintonía. En este caso, como todos los datos harán uso de la variable temporal, lo primero que debemos hacer es utilizar un intervalo de tiempo común a todos ellos. En este tipo de gráficos también debemos considerar que la unidad de medida de las variables sea apta para analizar su correlación. Por ejemplo, no tiene demasiado sentido analizar el PIB en cifras globales y el salario, por persona, mientras que, si ambas unidades de medida se calculan por habitante, la cosa cambia.

Las transiciones en este gráfico funcionan de forma similar a las transiciones implementadas en la directiva map, pero a diferencia de ésta última, aquí no se define el color, que se mantiene constante para todas las burbujas, sino que se modifican los atributos de posición del centro (cx,cy) y el radio de los círculos. Si nos encontramos el caso en que representamos algún país que no tiene un dato concreto en un año, se utiliza interpolación para obtener una posible estimación.

En cuanto a eventos, asignaremos el habitual mouseover a cada uno de los círculos aplicando el ya recurrente cambio de opacidad, además de propagar, desde el centro del círculo, las lineas rectas que cortan con los ejes cartesianos de nuestra gráfica. Adicionalmente, este evento también originará la aparición de un tooltip que muestra, además del nombre del país relacionado al elemento en cuestión, las cifras exactas que determinan la posición de éste. Puesto que la gráfica está formada por múltiples círculos, la creación de una leyenda que ayude a identificar cada país está descartada y, de esta forma, se enfatiza la interacción del usuario con el elemento sobre el que recae su interés.

Finalmente, se ha aprovechado que controlamos todos los países de la Unión Europea para lograr que, mediante el evento *onclick*, el usuario sea capaz de modificar la variable de interacción global que determina el país actual con estas gráficas. Con esto, tenemos una alternativa al uso del mapa como herramienta de modificación de dicha variable. El resultado final de la representación lo podemos observar en (5.11).

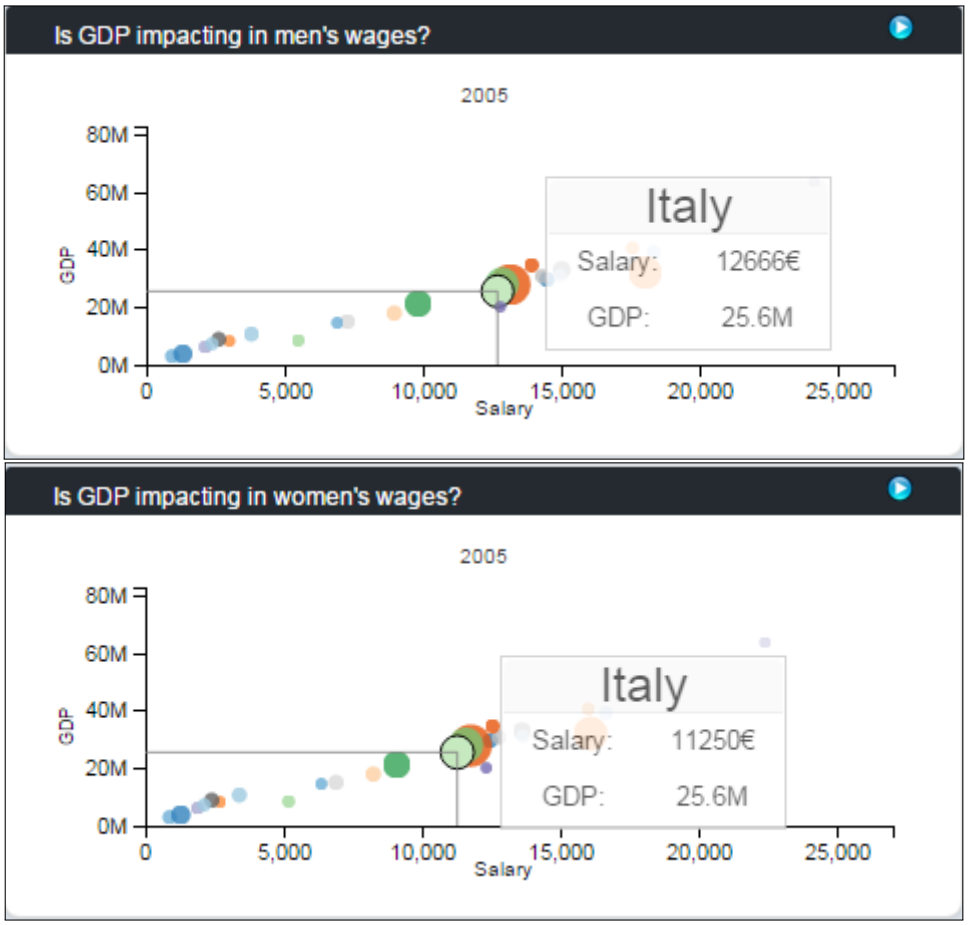

FIGURA 5.11: Gráficos Bubble Chart Sex completados

5.3.1.6. BubbleChartCompare

Nos encontramos ante otra directiva cuyo resultado será una gráfica Bubble Chart, pero a diferencia de la anterior, el contenido de los datos que representa es sustancialmente diferente, además de utilizarse una sola vez en nuestro *DashBoard*.

Este Bubble Chart servirá para analizar si el gasto en educación que tiene cada país influye en la diferencia del salario entre empleados y empleadas que tiene. Los datos para crear esta visualización los obtenemos tanto del DataSet de salarios como del de inversión, además de utilizar, por enésima vez, el de demográficos. Utilizaremos el eje horizontal de la gráfica para representar el porcentaje de la diferencia de salarios, junto a la inversión por habitante que realiza el país en temas de educación en el eje vertical. Además, volveremos a utilizar la población para determinar el radio de cada círculo. El punto interesante a analizar en esta visualización es la posición de los círculos respecto al centro del eje horizontal para clasificar al país en función de si son los hombres o bien las mujeres quienes tienen los salarios más altos. Para enfatizar esta diferencia, se marca claramente una linea vertical que representa la igualdad salarial y nos facilita la

identificación de estos grupos en función de la posición de los círculos respecto a esta linea.

Tal y como ocurría en la directiva anterior, esta gráfica dispondrá de transiciones (con su correspondiente botón de inicio) que nos darán una visión de la evolución de estos valores durante un intervalo de tiempo concreto. Así pues, también deberemos controlar que el intervalo de tiempo sea común en todos los DataSets que entran en escena en el diseño de está representación. Puesto que también tendremos siempre representados todos los países en esta gráfica, la variable de nivel de educación vuelve a ser la única que sugestiona externamente sus datos.

Evidentemente, tanto los eventos como sus comportamientos son exactamente los mismos que observamos en la directiva anterior. Tendremos otro tooltip con la información pertinente y esta gráfica supondrá la cuarta y última forma de modificar el país actual de la variable de interacción global de que dispondrá el usuario. El panel definitivo tendrá el siguiente aspecto (5.12).

Figura 5.12: Gráfico Bubble Chart Compare completado

5.3.1.7. BubbleChartReference

El tercer y último tipo de Bubble Chart que encontraremos en el *DashBoard*. Tanto la interpretación como su contenido presentan un cambio radical respecto a sus semejantes anteriores. Utilizaremos este Bubble Chart para observar el comportamiento de uno de los conjuntos que tratamos (empleados o empleadas) respecto al otro. La mayor diferencia que observamos respecto a los anteriores la tenemos en la cantidad de círculos que aparecen en la gráfica, puesto que únicamente aparece uno.

Este círculo, que se corresponderá con el país escogido por interacción global y el conjunto escogido por interacción local mediante un <select>, tendrá su posición en la gráfica determinada por el porcentaje de empleo que presenta ese género en ese país en el eje vertical y el salario medio en el eje horizontal. Además, para poder comparar este circulo con el conjunto complementario, el centro de la gráfica vendrá determinado por la media de los valores de este último. Para facilitar la lectura de la representación, se marcarán las lineas rectas que cortan en el centro de la gráfica y que, por lo tanto, tienen su origen en los puntos medios de ambos ejes. Más que analizar los datos del conjunto representado o la existencia de correlaciones, se pretende ver si la tendencia de los valores de ambos conjuntos a lo largo de los años es aproximarse o alejarse. Para identificar tanto los ejes de referencia como el círculo representado, se vuelven a utilizar los colores representativos de ambos conjuntos como en las directivas anteriores (azul, rosa). Los datos utilizados para representar esta gráfica se obtienen de los DataSets de salario, demográficos y empleo.

Este gráfico también utilizará las mismas transiciones sobre la variable temporal que utilizan los Bubble Chart anteriores. Tal como pasaba con las directivas previas, el nivel de educación escogido por interacción global determinará los datos que utilizará la representación. Tanto los eventos como los comportamientos de éstos son exactamente los mismos que encontramos en los Bubble Chart anteriores, a diferencia del onclick, que obviamente, ya no escogerá el país actual puesto que en esta gráfica no aparecen los países de la Unión Europea. El tooltip sigue mostrando las cifras exactas de los datos y el nombre del país al que hacen referencia. Las opciones disponibles del selector donde se elije el conjunto representado por la gráfica se generan dinámicamente utilizando D3, como viene siendo habitual.

Finalmente, se utiliza tooltipster para mostrar un tooltip que ofrece al usuario información extra sobre el gráfico y como debe éste interpretarlo al pulsar sobre el icono con el interrogante, tal y como ocurría en la directiva map. El resultado final del panel que contiene esta directiva luce de la siguiente manera (5.13).

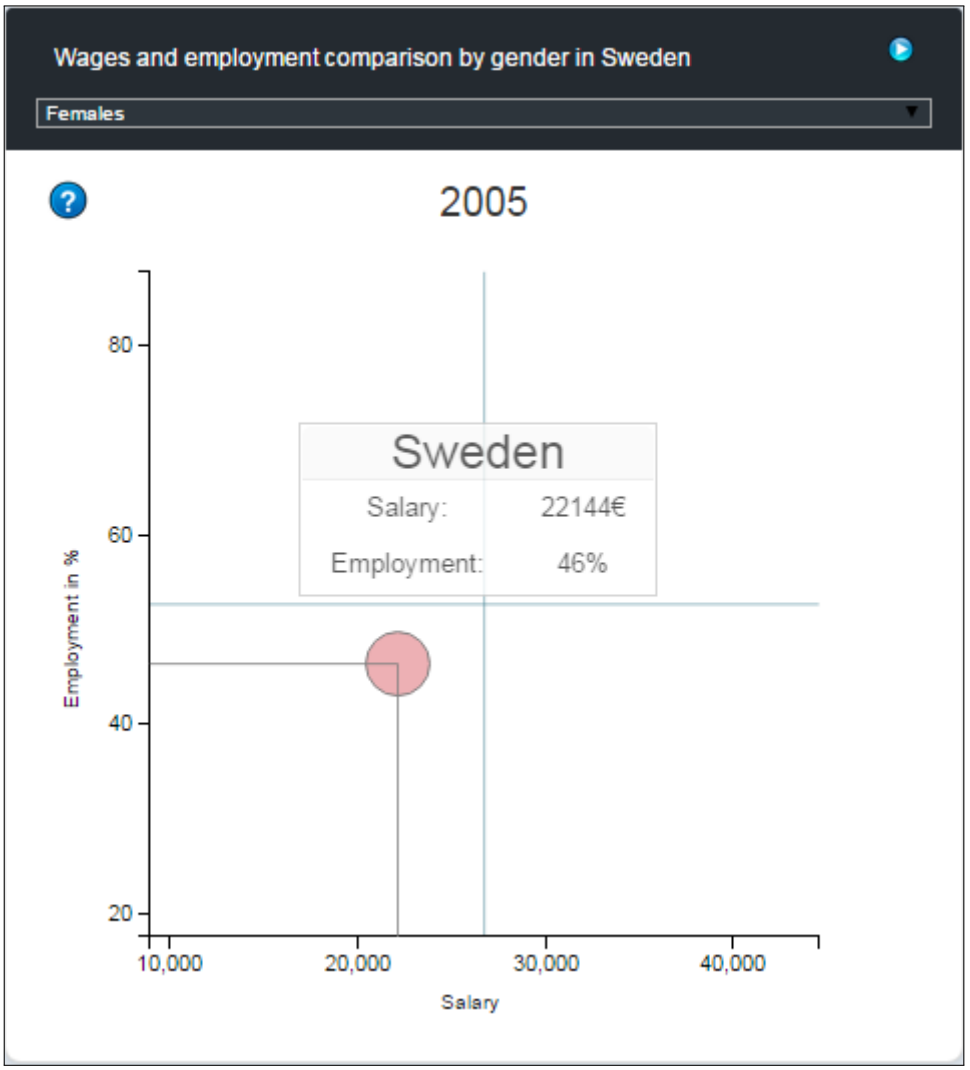

FIGURA 5.13: Gráfico Bubble Chart Reference completado

5.3.1.8. AreaChart

Esta directiva creará un Area Chart en el que se podrá observar la evolución tanto del PIB como de la inversión en educación del país escogido mediante interacción global durante un intervalo de tiempo. Se ha escogido esta gráfica puesto que ambas variables utilizan la misma unidad de medida que será representada por el eje vertical mientras que la variable temporal ocupará el eje horizontal. Además, este tipo de gráfico no solo nos permitirá observar ambas evoluciones, sino que también podremos ver si ambas son razonablemente similares. El Area Chart es una representación muy similar al Line Chart que hemos definido en una directiva anterior con la particularidad de que, al colorear al área formada por las lineas que definen los datos, podemos hacernos una idea más clara de la diferencia existente entre ambas lineas.

Los datos que hacen posible esta representación los obtenemos de los DataSets de inversión en educación y el de PIB. Puesto que la variable temporal viene representada

por un eje del gráfico y la variable de interacción global de nivel de educación tiene nula influencia sobre este gráfico al tratarse de datos económicos, la única variable que modifica el comportamiento de nuestro gráfico es el país actual. Esto también repercute en que este gráfico no disponga de transiciones ya que, a diferencia de los Bubble Chart, no son realmente útiles. Pese a que en este gráfico no estemos utilizando los conjuntos habituales sobre los que tratamos los datos (empleados y empleadas), se utilizarán los mismos colores con los que se representan por cuestión de diseño estético para diferenciar el PIB de la inversión en educación.

Aunque una leyenda sea lo más habitual en este tipo de gráficos y, ciertamente, sea la opción más idónea para indicar al usuario como interpretar esta gráfica, el tamaño del panel que la contendrá y el hecho de que nos encontremos ante una Stacked Area Chart nos ha empujado a volver a utilizar un tooltip como herramienta alternativa. En este caso, el tooltip contendrá el nombre al que hace referencia el área sobre la que realizamos el evento mouseover. Podemos observar el panel resultante aquí (5.14).

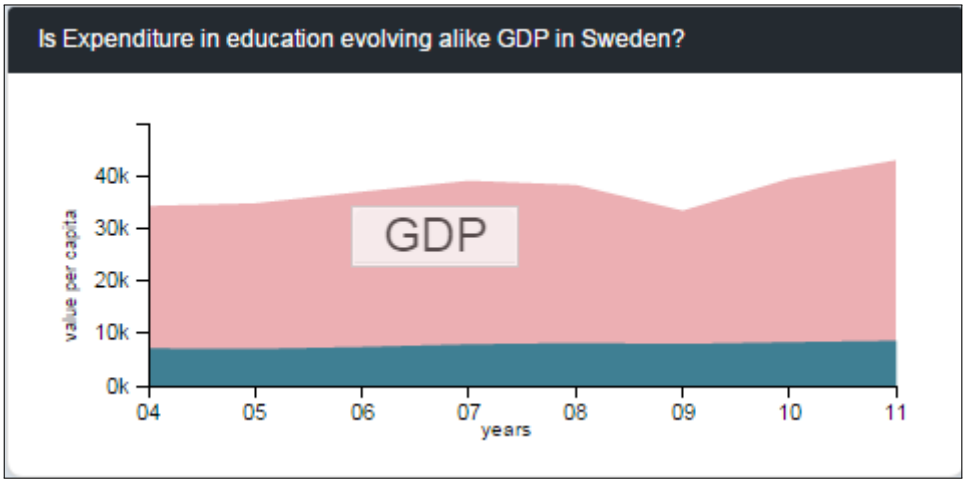

FIGURA 5.14: Gráfico Area Chart completado

5.3.1.9. LineBarChart

La siguiente directiva creará un Line Bar Chart, una gráfica que no es utilizada extensamente pero que nos permite representar tres variables, haciendo bastante hincapié en la comparativa de los diferentes BarChart que formarán parte de la gráfica. Así pues, esta visualización es el resultado de combinar un Bar Chart vertical junto a una Line Chart. Esta representación utiliza dos ejes verticales situados en el inicio y el final del eje horizontal. Para poder utilizar este tipo de gráfica es necesario que las variables representadas en los dos ejes verticales dependan de una variable conjunta que pueda ser representada por el eje horizontal. Así pues, utilizaremos esta gráfica para observar como varia la diferencia de empleo de hombres y mujeres (cada uno representado por una Bar Chart) junto a la inversión en educación que se representará con la Line Chart.

Esto nos permitirá ver si la tendencia que sufre la inversión en educación tiene influencia sobre los porcentajes de empleo que sufren ambos géneros.

Para poder realizar esta gráfica se utiliza la variable temporal para ser representada por el eje horizontal, con lo que no aparecen transiciones en esta gráfica. Serán pues, tanto el país actual como el nivel de educación, las variables de interacción encargadas de alterar el contenido de esta representación. Los datos asociados a esta gráfica se obtienen del DataSet de inversión en educación y del de empleo. Como viene siendo habitual, el color de cada barra se define por el conjunto que representa y se ha escogido el color amarillo para la linea. Este color, aparte de mantener la armonía con los colores que representan los géneros, destaca lo suficiente como para poder analizar su información.

Esta gráfica no utiliza leyenda, en gran parte, porque tanto los colores como los ejes ofrecen al usuario suficiente información como para que este pueda interpretarla de una forma bastante intuitiva. Una leyenda en este gráfico podría resultar en una sobrecarga de información innecesaria que acabaría siendo más de perjuicio que de ayuda. Pese a esto, se utiliza un tooltip que informará al usuario del valor concreto que tiene la barra sobre la que se origine un evento *mouseover*, en caso de que el usuario esté interesado en conocer porcentajes exactos. El resultado final del panel luce un aspecto similar al siguiente (5.15).

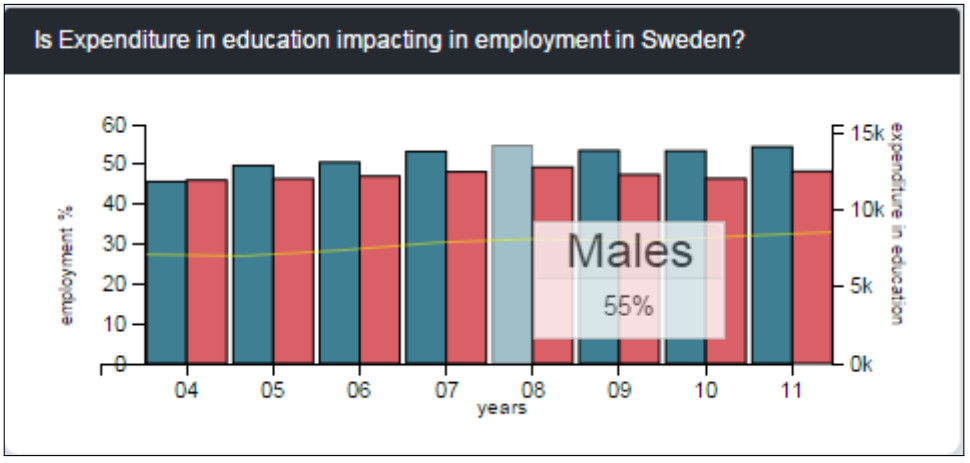

FIGURA 5.15: Gráfico Line Bar Chart completado

5.3.1.10. SunburstChart

Esta directiva se encargará de crear un Sunburst Chart. Nos encontramos ante una de las representaciones más complejas que pueden implementarse en D3.js. y para la que se ha adaptado el ejemplo creado por kerryrodden `http://bl.ocks.org/kerryrodden/477c1bfb081b783f80ad`. Está gráfica esta compuesta por una jerarquía construida sobre un layout de partición. A partir de los datos de los nodos con menos jerarquía de nuestra estructura de datos, va construyendo las jerarquías superiores considerando que la unión

de todas las categorías inferiores pertenecientes a la misma categoría superior equivale al valor de la jerarquía superior. Así pues, un paso clave a la hora de crear esta gráfica es estructurar los datos que se vincularán a la visualización de la forma correcta. Acto seguido, se definen los arcos que dan forma a la gráfica de manera similar a como hacíamos en el Pie Chart.

Un efecto más que interesante del Sunburst implementado, es la capacidad de zoom jerárquico, es decir, la capacidad de modificar la gráfica al hacer onclick sobre una de sus secciones para que éste se convierta en el nodo de mayor jerarquía de la estructura de datos. Esto nos permite obtener visualizaciones más concretas de las jerarquías inferiores del Sunburst.

En nuestro caso, utilizaremos todo el potencial del DataSet de empleo para obtener toda la información posible sobre la que interactuar que nos ofrece este DataSet. Las jerarquías que definiremos en la estructura de datos que vincularemos al gráfico son de mayor a menor: total, nivel de estudios, actividad laboral y género. A efectos prácticos, el resultado de está estructuración jerárquica nos lleva a interpretar el gráfico de esta manera: nos indica la aportación masculina y femenina que existe sobre cada actividad laboral en función del nivel de estudios. Para representar el género se vuelven a utilizar los colores representativos tradicionales mientras que para el resto de nodos se aplican colores categóricos de D3. Para poder crear este gráfico, únicamente los nodos inferiores deben contener valores y cada nodo tendrá una lista con los nodos de la jerarquía inferior que derivan de él. El contenido de esta gráfica viene definido por el país actual definido en el $scope del controller, así como el año seleccionado por el usuario con la variable de interacción local en el selector HTML del panel.

La creación de un tooltip para este tipo de gráfico es necesario puesto que, a simple vista, únicamente se puede interpretar secciones de diferentes colores que derivan en múltiples secciones de jerarquía inferior. En vez de escribir el nombre al que representa cada sección dentro de ella misma, se usa un tooltip que lo indica al hacer *mouseover* sobre ésta, además de mostrar al usuario su valor total, o lo que es lo mismo, la suma de los valores de sus nodos descendientes. Al igual que ocurría en otras directivas vistas previamente, también mostrará un tooltip al pulsar sobre el icono con símbolo de interrogante que nos indicará como manipular e interpretar esta visualización. El panel resultante es el siguiente (5.16).

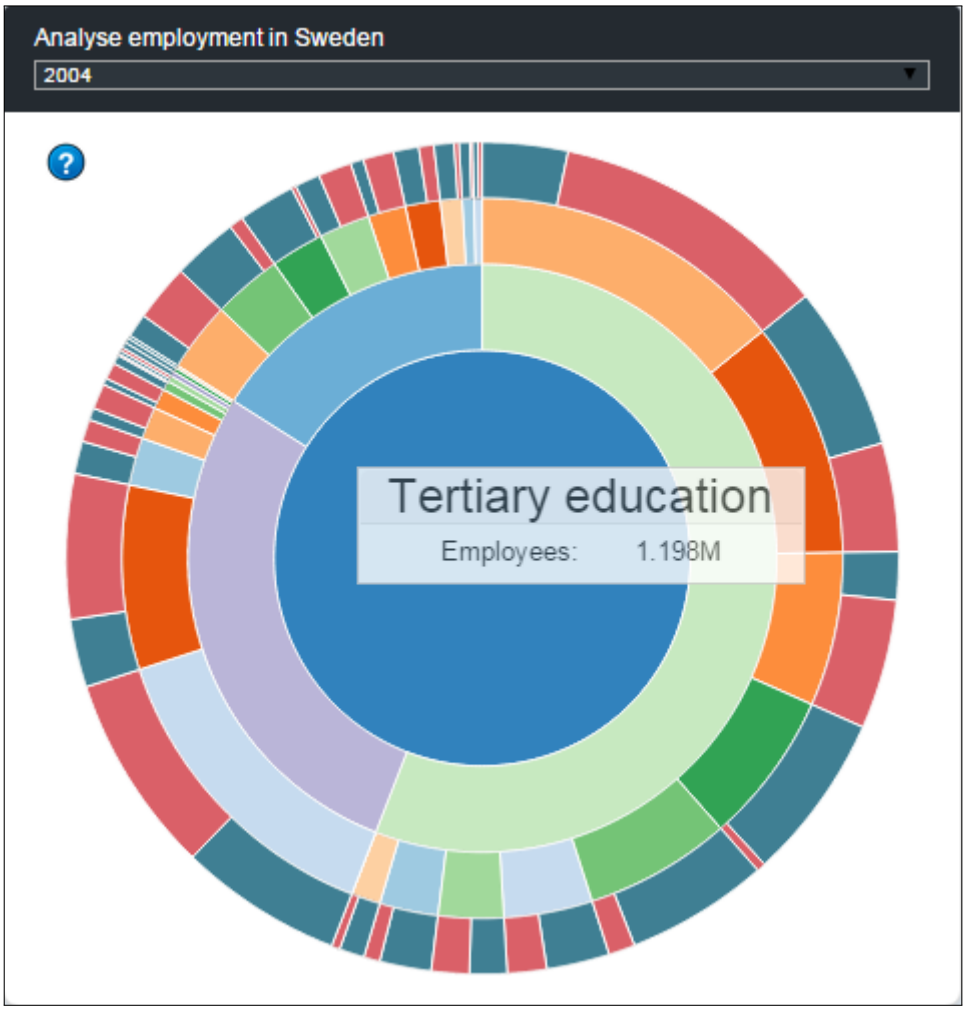

FIGURA 5.16: Gráfico Sunburst Chart completado

5.3.1.11. ChordDiagram

Junto al Sunburst, esta directiva creará otro de los gráficos más complejos que es capaz de crear D3.js. Tal y como pasaba con el anterior, se adaptará el ejemplo creado por mbostock http://bl.ocks.org/mbostock/4062006 para acabar obteniendo un Diagrama de Cuerdas que represente los datos con los que estamos trabajando. Consiste en una serie de conjuntos sobre los que se analiza la aportación de cada uno de ellos al resto. Esto se realiza gracias al layout de chord proporcionado por D3 junto a una estructuración matricial de los datos que queremos representar. Tanto en filas como en columnas tendremos los diferentes conjuntos donde cada valor indicará la aportación de la fila sobre la columna. En nuestro caso, vamos a utilizar esto para representar cuanto aporta cada actividad laboral al total de empleados y empleadas. Así pues el género se relacionará con las actividades y viceversa, pero los géneros no se relacionaran entre sí y las actividades laborales tampoco. Los arcos que definen la relación tendrán el mismo color de la actividad a la que pertenecen y el resto utilizará colores categóricos de D3.

El orden es un factor decisivo para este tipo de gráfico, ya que tanto las interrelaciones como el posicionamiento que siguen los propios conjuntos en el circulo exterior deben seguir un orden descendente para que el gráfico mantenga armonía y no dificulte su interpretación. Para identificar los conjuntos que forman el contorno exterior del círculo, introducimos una serie de labels dentro de éstos. Además, como suele ser habitual, se implementará un tooltip al hacer *mouseover* sobre los conjuntos exteriores. Este tooltip muestra el conjunto sobre el que se ha lanzado el evento así como las cifras de empleados y empleadas (si es una actividad) o el total de empleados de ese género sumando todas las actividades (si es un género).

Además de generar el tooltip, al activar el evento de mouseover, la opacidad de las interrelaciones no pertenecientes al conjunto en cuestión desaparece, con lo que únicamente se mantienen a la vista los enlaces de este conjunto. Tanto el país actual como el nivel de educación influyen en el contenido de la gráfica, además, tal y como pasaba con el sunburst, el usuario puede seleccionar el valor de la variable temporal mediante un <select>. El panel resultante tiene el siguiente aspecto (5.17).

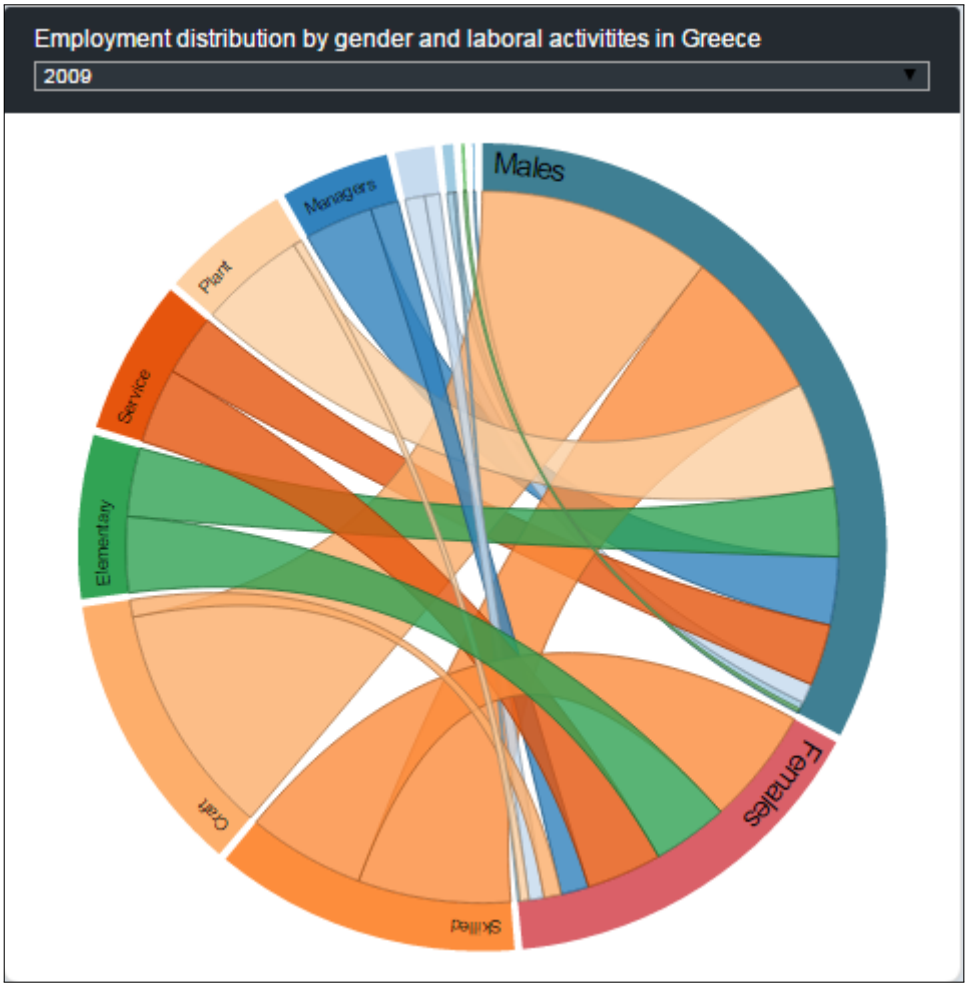

FIGURA 5.17: Gráfico Chord Diagram completado

5.3.2. DashBoard

Una vez implementadas todas las directivas con los gráficos que queremos mostrar en nuestro *DashBoard*, es momento de crearlo. Para eso vamos a utilizar la directiva ng-include proporcionada por angular que nos permite cargar en un contenedor el contenido de otro HTML. El uso de esta directiva nos permite tener definidos en ficheros adicionales la estructura que seguirá nuestro *DashBoard*, los templates, para poder trabajar con ellos de forma independiente. Así pues, nuestra aplicación contará con tres templates: uno se encargará de definir el código HTML que creará el navbar mientras que los otros dos crearan la organización de los paneles con sus respectivas directivas.

Del *navigation bar* cabe destacar el uso de la directiva ng-click, definida por angular, que asocia, al evento onClick, la ejecución de una función que cambiará el valor del nivel de educación del $scope del controller por el elemento del navbar seleccionado.

Los otros dos templates definen dos combinaciones de paneles y gráficas diferentes. Uno de ellos define la página que vemos al inicial la aplicación y que estructura el *DashBoard* de la siguiente manera (5.18). Por otro lado, al pulsar sobre uno de los países integrantes del mapa y, por consiguiente, modificar el país actual en el $scope del controller, todo el contenido se elimina mediante JavaScript y se sustituye por la estructura definida en un segundo template que únicamente muestra aquellas gráficas que dan información específica del país seleccionado. El resultado, una vez seleccionado un país en el mapa, es el siguiente (5.19).

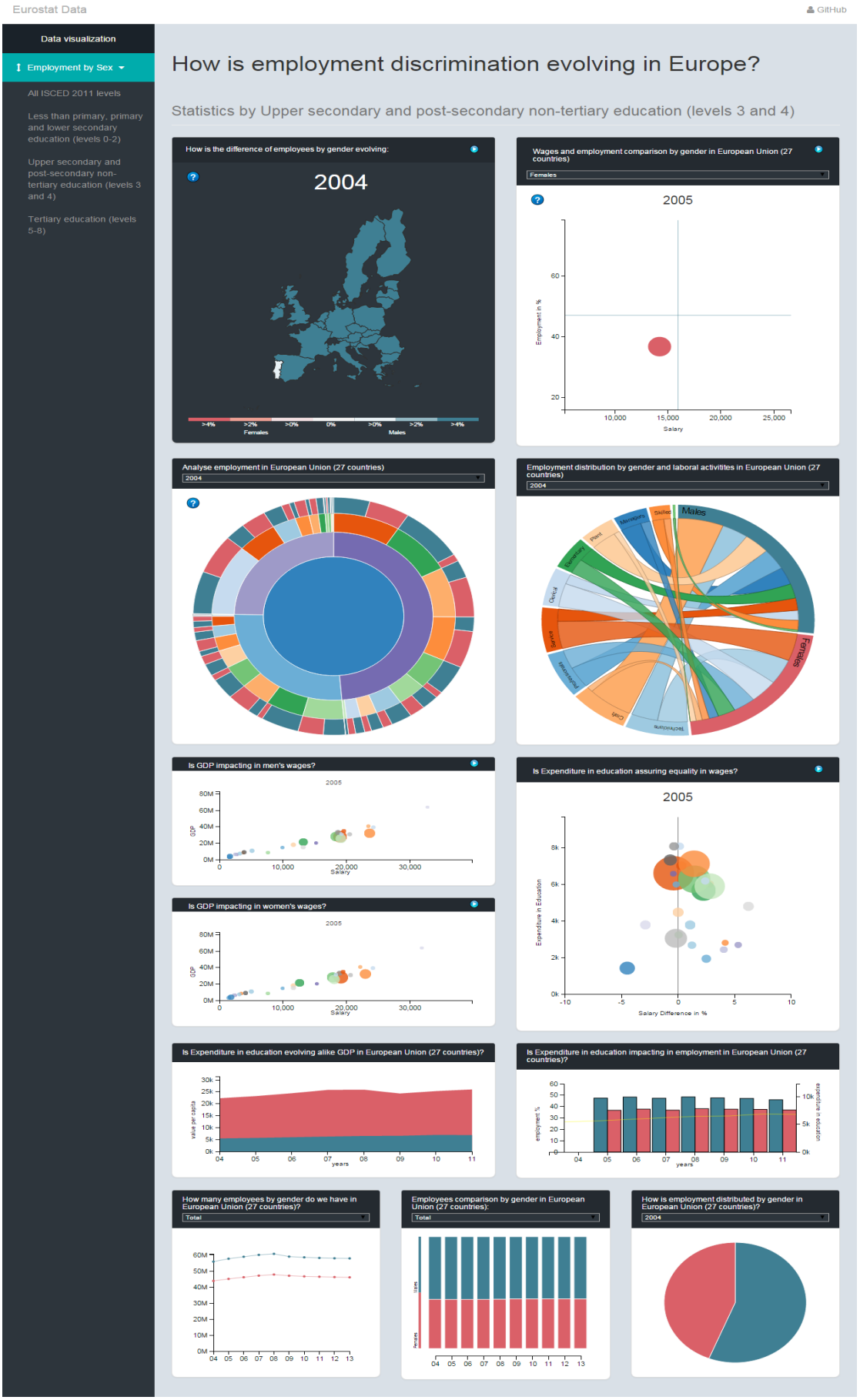

FIGURA 5.18: Vista DashBoard inicial

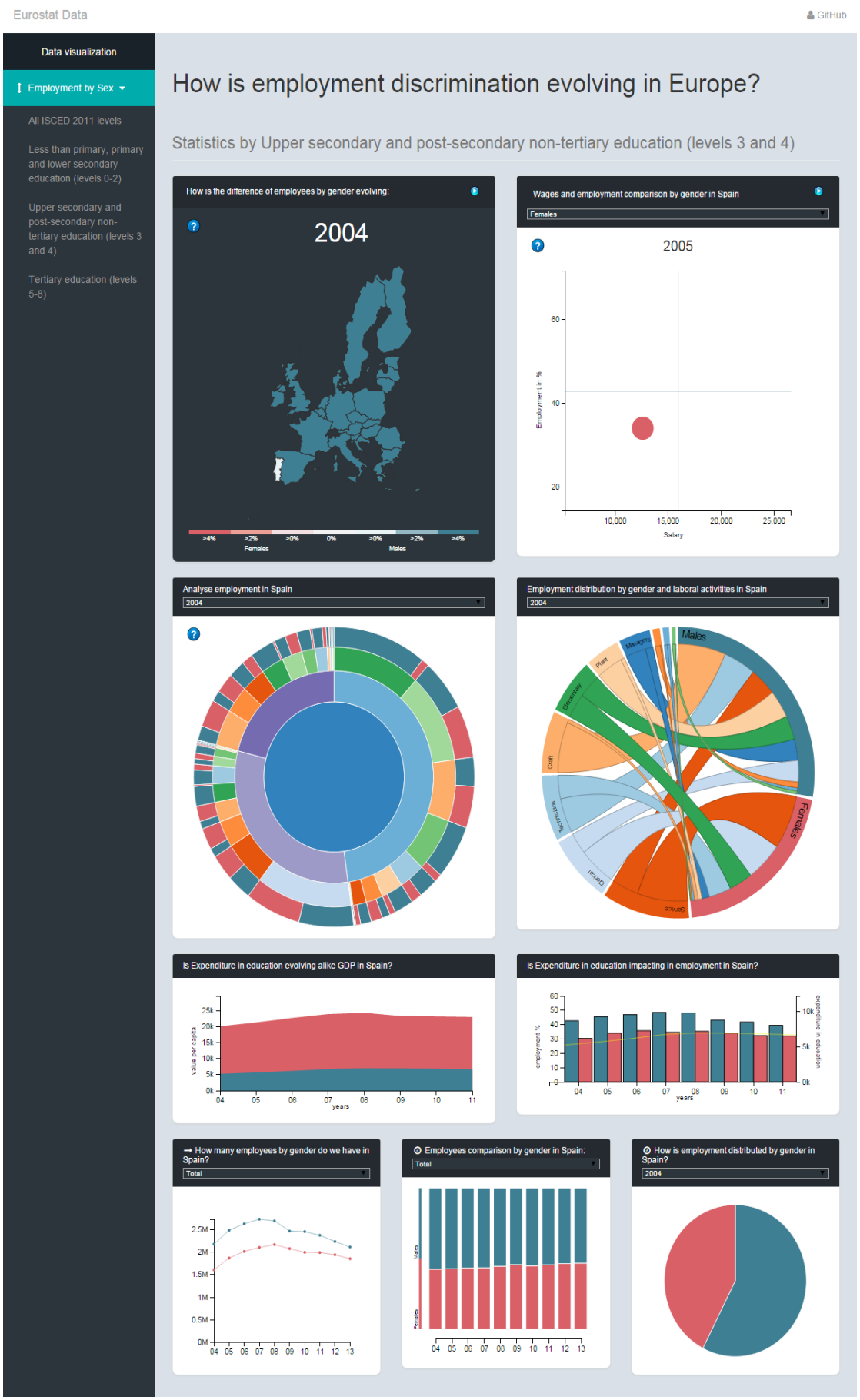

FIGURA 5.19: Vista DashBoard reestructurado

Capítulo 6

Resultados

Una vez implementado el DashBoard y aplicados los conocimientos de *data visualization* para crear las gráficas que lo forman, es momento de analizar los resultados obtenidos gracias a estas representaciones y lo que nos ofrece el *DashBoard* que acabamos de crear. Además, también debemos corroborar si estas gráficas van a ser capaces de ayudarnos a responder las cuestiones que nos hemos planteado al inicio del proyecto.

A nivel general, hemos logrado crear un *DashBoard* como el que nos habíamos planteado, con distintas visualizaciones sobre diversos DataSets enfocados en un tema en concreto. Estas gráficas contienen un fuerte componente interactivo con el usuario que no solo es capaz de observar los datos que éstas le están exponiendo, sino que el propio usuario puede manipularlas a su antojo con cierta libertad. Además, hemos logrado añadir transiciones a algunas de ellas, lo que convierte algunas gráficas en semi-reproducciones de alta calidad estética que permiten al usuario observar los cambios que se están produciendo de la forma más natural posible. En cuanto al detalle de los datos representados, considero que es acertado, ya que permite definir múltiples variables en las gráficas para especificar con cierta precisión el contenido que se desea visualizar. Finalmente, se ha logrado una interconexión global entre todas las gráficas permitiendo que muchas de ellas se vayan modificando en función de ciertas variables de una forma óptima gracias a la inestimable ayuda de Angular.js.

Además de analizar los datos en el marco europeo, también nos pararemos un poco para observar particularmente a España y ver si tiene un comportamiento similar al resto de Europa. Intentaremos analizar como influyen los diferentes niveles de educación en las gráficas y ver si nos aporta información interesante.

6.1. Map

Cogiendo como referencia cualquier nivel de educación, es aplastante observar como en prácticamente toda Europa,la cantidad de hombres trabajando siempre se mantiene superior desde el 2004 hasta el 2012, generalmente por encima del 4 %. Únicamente en tres países del este (Estonia, Lituania i Letonia) se puede observar como este hecho se invierte durante algún periodo de tiempo concreto. El siguiente paso es observar el comportamiento que experimentará el mapa al centrar nuestra atención en un nivel de educación concreto.

Al seleccionar el nivel más bajo, el cambio respecto al total se observa instantáneamente. Pese a que en la mayoría de países los hombres siguen dominando el mercado laboral, han aparecido algunos en los que la situación se ha invertido de forma clara. Principalmente, nos encontramos ante países del este de Europa (Romanía, Hungría, Eslovaquia, República Checa) y, sorprendentemente, también Reino Unido. Al iniciar la reproducción de las transiciones, observamos que, con el paso de los años, en los países en los que los hombres dominaban se mantiene esa situación y en los países en que esto no ocurría, la tendencia general es ir adoptando esta situación hasta el punto en el que en el año 2012, únicamente Austria, República Checa y Eslovaquia ven como las mujeres se imponen a los hombres.

Si nos centramos en el nivel de educación medio, aquel que cuenta con más representación en la Unión Europea, observamos como el mapa es demoledor, desde 2004 hasta 2012 se mantiene siempre por encima el número de empleados masculinos en la totalidad de los países Europeos (a excepción de Portugal) y no solo eso, sino que además, en su mayoría, la tendencia no es propensa al equilibrio sino a mantener esta diferencia.

Si por consiguiente escogemos visualizar el nivel de educación mas alto, observamos una situación bastante diferente respecto a las anteriores. Podemos observar como prácticamente existe una división perfecta entre los países en que predominan las mujeres y en los que predominan los hombres. Es interesante observar como, por ejemplo, en todos los países nórdicos existen más empleadas. En contrapartida, en el centro de Europa observamos como son los hombres los que todavía se imponen a las mujeres en este sentido. Pero el factor más sorprende lo encontramos al iniciar la reproducción y abandonar el año 2004, ya que la tendencia en prácticamente todos los países Europeos se inclina hacia un dominio aplastante del género femenino, hasta el punto en que en el año 2012, únicamente mantienen más presencia laboral masculina cuatro países centroeuropeos (Alemania, Holanda, República Checa y Austria) y Grecia.

6.2. Line Chart

Analizar los datos relacionados con todas las actividades laborales sería prácticamente imposible así que únicamente analizaremos aquellas que tengan comportamientos que nos resulten interesantes. Observaremos todos los niveles de educación puesto que para analizar fuertemente cada uno de los niveles de educación posibles, probablemente sea más útil e interesante el gráfico Sunburst.

Primero empezaremos observando a Europa en su conjunto. Observamos como la diferencia de empleados se mantiene de forma constante con el paso de los años. Si comenzamos a curiosear por las diferentes actividades laborales, observamos que algunos comportamientos y tendencias son realmente interesantes.

Si observamos a los profesionales, vemos como en el año 2004, el número de hombres trabajando en esta actividad era superior al de las mujeres pero que lentamente esta tendencia se va invirtiendo. El cambio definitivo lo observamos en el año 2011 en el que las mujeres superan definitivamente a los hombres en esta actividad y, no solo eso, sino que la diferencia a favor de las mujeres va en aumento año tras año. Otra actividad laboral que resulta interesante es la de los técnicos. Esta actividad se mantiene con preferencia femenina de forma constante hasta el año 2011, en el que el número de empleadas se desploma de forma significativa hasta el punto de que son los hombres los que predominan en la actividad. La última actividad con un comportamiento curioso la observamos en ocupaciones elementales. Esta actividad se caracteriza hasta el año 2008 por ser la más equilibrada de todas, pero a partir de ese año, pese a que el número de mujeres va lentamente en ascenso, los hombres se desploman hasta tal punto de que la cantidad de trabajadores en 2013 es inferior a la que teníamos en 2004, con lo que la diferencia respecto a las mujeres aumentó de forma significativa.

Si centramos nuestro análisis en el estado español, lo primero que observamos es que el número de trabajadores y trabajadoras va creciendo lentamente hasta el año 2007/2008, a partir de este año, el empleo global va cayendo significativamente, coincidiendo con el inicio de la crisis económica. Es interesante como el número de mujeres trabajando se mantiene más o menos constante a pesar de todo y es el género masculino el que se desploma de forma considerable. Mirando el comportamiento específico de las distintas actividades laborales, vemos como nos encontramos con casos bastante interesantes. En los profesionales, por ejemplo, vemos como, pese a que en el año 2004 existía una igualdad prácticamente perfecta, la diferencia entre ambos géneros ha ido creciendo de forma constante a lo largo de los años y no solo eso, sino que además esta actividad no ha visto reducido su número de empleados con la llegada de la crisis, sino que ha aumentado. También es bastante curioso el caso de los técnicos, pese a que los hombres siguen el patrón que habíamos visto en el total, a partir del año 2010 se produce un descenso brutal de las mujeres, que en términos globales, habían aguantado mejor la crisis. Con los servicios observamos como nos encontramos ante una actividad en la que

la diferencia entre hombres y mujeres se ha mantenido constante y, no solo eso, sino que ambos sectores han aumentado el número de trabajadores sin cesar desde el año 2004. En agricultura, actividad forestal y pesca, el descenso desde el año 2004 ha sido constante para ambos sexos, aunque de forma más pronunciada en el caso de los hombres. En cuanto a manufactura, observamos como el desplome a partir del año 2008 en el número de empleados masculinos roza lo catastrófico. Las mujeres, por su parte, sostienen su poca participación en esta actividad. Para finalizar, destacar el comportamiento que observamos en la actividad de ocupaciones elementales. Estamos ante una actividad que en el año 2004 gozaba de igualdad y que, pese a que el número de empleadas se ha mantenido, en el caso de los hombres se ha producido tal descenso de empleados a partir del 2008 que ha provocado que prácticamente el número de mujeres doble al de los hombres.

6.3. Stacked Bar Chart & Pie Chart

Interesante para analizar que actividades laborales presentan proporciones de empleo drásticamente diferentes que en el total de estas. Observando el Pie Chart a nivel global Europeo, vemos que la proporción de empleados se decanta a favor de los hombres por un 56 % en el año 2004 para acabar con un 54 % en el 2013. Verdaderamente, la distribución laboral no evoluciona excesivamente con el paso de los años. Así pues, podemos mirar la Stacked Bar Chart para analizar que sectores presentan distribuciones que se desvían de estos números. Incluso podemos clasificar las actividades laborales en tres grupos:

- Masculinas: Mánager, Agricultura/pesca/forestales, Manufactura, Operadores de maquinaria, Fuerzas armadas.
- Femeninas: Apoyo clerical, Servicios.
- Adaptativas: Profesionales, Técnicos, Ocupaciones elementales.

Observamos como todavía existen ciertos sectores que son considerados 'trabajos masculinos' y otros que son considerados 'trabajos femeninos' con diferencias de empleo devastadoras entre ambos géneros. La situación de empleo ideal se daría si todas las actividades lograsen clasificarse en el tercer grupo. La aparición de los dos primeros grupos se debe mayoritariamente a dos motivos: por un lado, la prioridad de contratación que tienen las empresas y, por otro lado, la ideología, fuertemente arraigada en la sociedad, de que existen trabajos que no pueden realizarse independientemente del género.

Si centramos nuestra visión en España, lo primero que observamos es que la desigualdad laboral en número de empleados presenta un considerable mayor nivel inicial, alcanzando la cifra del 61 % a favor de los hombres en el año 2004. Lo que es más extraordinario

es la velocidad a la que se están equilibrando estos números con el paso de los años, hasta llegar a un 54 % (la misma cifra que presenta Europa) en el año 2013. A nivel de clasificación de actividades, éstas siguen una clasificación similar a la Europea, con la diferencia que durante los últimos años, las ocupaciones elementales cada vez están decantándose más a convertirse en 'trabajo femenino'.

6.4. Line Bar Chart

Observamos como no se puede establecer de forma concisa que exista influencia de la inversión en educación sobre el porcentaje de empleo de ambos géneros independientemente del nivel de educación seleccionado. Lo que sí podemos observar es que el porcentaje de empleo masculino es superior en términos generales al femenino tanto en Europa como en España. Pese a todo, observamos como a partir del año 2008 se produce un bajón considerable en el número de trabajadores masculinos. Como el número de trabajadoras se mantiene, las diferencias en este aspecto entre ambos géneros se han reducido considerablemente.

6.5. Area Chart

Este gráfico nos muestra cómo, en términos generales, sí que existe cierta similitud entre las tendencias de evolución a lo largo de los años del PIB de un país y su inversión en educación. Podemos ver como en Europa la inversión en educación se ha mantenido de una forma más constante que el PIB, que ha sufrido pendientes mucho más pronunciadas. Los resultados en España son muy similares, ya que la inversión sigue destacando por tener un valor constante a lo largo de los años. Quizás la diferencia más importante la encontramos en que el PIB también ha sufrido cambios menos significativos comparándolo con Europa, con lo que la similitud entre ambas lineas es incluso mayor que la observada en Europa.

6.6. Bubble Chart Sex

Si hiciésemos una encuesta preguntando si los países más ricos también ofrecen salarios más altos, la lógica llevaría a la mayoría a responder afirmativamente. Pues bien, la idea de estos dos gráficos es demostrar que esta afirmación en correcta dejando la lógica a un lado y basándonos en puros datos.

Podemos determinar que, efectivamente, existe una correlación entre el PIB del país y los salarios de los empleados. Aquellos países con un PIB más alto, por lo general,

también ofrecen a los trabajadores salarios más elevados, tanto a los hombres como a las mujeres.

Centrémonos primero en el género masculino:

Si miramos todos los niveles de educación, únicamente Chipre parece que se desmarca considerablemente de esta afirmación puesto que ofrece salarios bastante altos considerando su PIB. Luxemburgo, con un PIB por habitante realmente muy por encima del resto, pese a que también ofrece salarios relativamente altos, podría ofrecerlos más elevados si consideramos únicamente su PIB.

Si miramos específicamente los niveles de educación más bajos, también observamos esta correlación. Luxemburgo sigue el mismo comportamiento anterior y es Francia quien se desmarca un poco, ofreciendo salarios altos.

Con el nivel de educación medio, todos los países observan como los ingresos son más altos si los comparamos con el nivel inferior, cosa que parece bastante lógica. Es este caso podemos ver como todos los países cumplen la afirmación anterior, exceptuando Reino Unido hasta el año 2008, puesto que sus salarios, tal como pasaba con Francia i Chipre anteriormente, eran ligeramente superiores.

Para finalizar, es momento de observar los niveles de educación más altos. Curiosamente, nos encontramos ante el caso que presenta países que más se desmarcan. Hasta el año 2008, tanto Portugal como Chipre ofrecían unos salarios realmente altos. Suecia y Luxemburgo, por contra, podríamos considerar que siempre han ofrecido unos salarios algo más bajos que los que podrían ofrecen a sus empleados.

Si miramos el género femenino:

Con todos los niveles de educación, observamos un comportamiento similar en todos las países al género masculino.

Si cogemos los niveles de educación más bajos, la diferencia más notable radica en que Francia no se desmarca de la afirmación planteada previamente.

Si nos centramos en los niveles de educación medio, Luxemburgo podría ofrecer salarios más altos mientras que los de Chipre están ligeramente por encima de lo que su PIB podría indicar.

Finalmente, si escogemos los niveles de educación superiores, observamos como Portugal y Chipre ofrecen salarios relativamente altos y, por contra, Suecia ve incrementado su PIB a partir del año 2010, pero hasta el 2013 los salarios no se corresponden.

6.7. Bubble Chart Compare

Considerando que la igualdad laboral pasa por que los salarios percibidos por ambos géneros sean similares, gracias a este gráfico podremos valorar la veracidad existente en que a más inversión destinen los países a la educación, estas diferencias salariales heredadas de antaño desaparecerán progresivamente. Si consideramos todos los niveles de educación, podemos ver que en absolutamente todos los países de la unión europea, los salarios masculinos están por encima. Además, un dato preocupante es que la mayoría de estas diferencias son superiores al 5 %, independientemente de la inversión en educación del país. Así pues podemos asegurar que no existe correlación entre la diferencia de salarios y la inversión. Esto queda en evidencia al observar que los dos países con más inversión por estudiante (Austria y Dinamarca) tienen valores de desigualdad cercanos al 5 %. También es curioso observar como el país con menos inversión (Romania) es el país en el que esta diferencia alcanza los valores más bajos, concretamente un 1.59 %. Con el paso de los años, a pesar de que la tendencia de la inversión de los países es aumentar, las diferencias a nivel salarial se mantienen constantes.

Al centrar nuestro enfoque sobre los niveles de educación más bajos, observamos como muchísimos países acercan las diferencias salariales, llevando inclusive al género femenino a tener salarios superiores en algunos países. En este caso, la tendencia con el paso de los años es oscilar alrededor de la igualdad, encontrándonos con países que de año en año van variando su posición respecto a ésta. Cabe destacar que Romanía y Croacia son los dos únicos países en los que las mujeres con este nivel de educación siempre han tenido salarios superiores desde el 2004.

Al analizar el nivel de educación medio, vemos que los resultados son muy similares a los obtenidos con los niveles más bajos. Similar al caso anterior, los países se caracterizan por oscilar alrededor de la linea de igualdad con el paso de los años. En este caso son Romanía y Alemania los países en los que el género femenino siempre ha tenido salarios superiores desde el 2004.

Sorprende el resultado obtenido al seleccionar el nivel de educación más alto puesto que en este rango salarial en prácticamente todos los países europeos los salarios masculinos son bastante superiores, con países por encima del 6 % de diferencia. Es demoledor observar como, a diferencia de los casos anteriores, ningún país ha mantenido los salarios femeninos por encima con el paso de los años. Más desolador si cabe es ver como únicamente Austria y Dinamarca han tenido en algún año salarios femeninos por encima de los masculinos.

Queda bastante en evidencia que la inversión en educación influye más bien poco en las diferencias salariales existentes entre géneros en los países que conforman la Unión Europea. Además, también hemos podido concluir que los trabajos que requieren más nivel de educación no se caracterizan por ofrecer salarios similares a ambos géneros.

6.8. Bubble Chart Reference

El diseño de esta gráfica nos permite definir cuatro cuadrantes que definen la posición laboral de la mujer en cuanto a salario y porcentaje de empleo respecto al género masculino:

- Superior izquierda. Porcentajes de empleo mayor pero con menos retribución salarial.
- Superior derecha. Porcentajes de empleo mayor con mejor retribución salarial.
- Inferior izquierda. Porcentajes de empleo menor y con menos retribución salarial.
- Inferior derecha. Porcentajes de empleo menor con mejor retribución salarial.

Primero analizaremos el comportamiento de Europa en general. Si consideramos todos los niveles de educación, vemos como el género femenino se mantiene en el cuadrante inferior izquierdo durante todos los años. Si seleccionamos los niveles inferiores de educación, observamos como pese a que inicialmente se sitúan en el cuadrante izquierdo, con el paso de los años el salario se va equilibrando hasta llegar a cotas bastante similares, inclusive alcanza el cuadrante inferior derecho durante un par de años. Respecto al nivel de educación medio, observamos un comportamiento similar al anterior. En este caso alcanza antes el cuadrante inferior derecho y lo mantiene durante un intervalo de tiempo más prolongado. Finalmente, si escogemos centrarnos sobre el nivel educativo superior, vemos como el género femenino no abandona el cuadrante inferior izquierdo en todo el intervalo de tiempo de las transiciones.

Si miramos el comportamiento del estado español, vemos como con todos los niveles de educación, pese a que la posición inicial es el cuadrante inferior izquierdo, desde el 2008 hasta el 2011, las mujeres se sitúan en el cuadrante inferior derecho antes de volver al inferior izquierdo durante el 2012. Si especificamos el nivel educativo inferior, observamos como el comportamiento es muy similar al total, pero a diferencia de este, una vez alcanza el cuadrante inferior derecho no lo abandona. Con el nivel de educación medio, el comportamiento es prácticamente el mismo que observamos al seleccionar todos los niveles de educación. Finalmente, al seleccionar el nivel de educación superior, vemos un comportamiento similar al seleccionar el inferior de todos. La diferencia mayor radica en que se sitúa a niveles más cercanos de igualdad en número de trabajadores.

6.9. Chord Diagram & Sunburst Chart

Gracias a estos dos gráficos podemos observar la aportación de cada una de las actividades laborales al total de empleados por género y ver como evoluciona con el transcurrir

de los años, además de observar la aportación de cada uno de los niveles de educación al total.

Si analizamos los datos en Europa:

Con todos los niveles de educación, observamos que en el año 2004 no existe una actividad que aporte excesivamente por encima del resto, sino que predomina una división bastante equitativa. Aún así, las actividades que aportan más empleados son, por este orden, Técnicos, Manufactura, Profesionales y Servicios. La actividad de manufactura es ciertamente interesante porque es la actividad que mas empleados masculinos aporta, pero en contra, es la tercera con menos empleadas. Los servicios viven la misma situación pero esta vez en sentido inverso. Los operadores de maquinaria, pese a que en global no tienen una aportación al número de empleados muy grande, tienen una situación parecida: muchos trabajadores y bastantes menos trabajadoras.

Con el paso de los años, la actividad de servicios y técnicos ven incrementada su aportación de forma significativa. Observamos como en el año 2013, pese a que las actividades que aportan más empleados son las mismas que en año 2004, el orden de estas varia: Profesionales, Servicios, Técnicos y Manufactura.

Si consideramos únicamente el nivel de educación más bajo, en el año 2004, las actividades con más empleados son: Ocupaciones elementales, Servicios, Manufactura y Operadores de máquinas. Curiosamente, estas actividades no dividen equitativamente su aportación a los empleados por género, sino que apreciamos bastante diferencia. Si nos fijamos como varían estas actividades con el transcurso de los años, observamos que las actividades con más empleados son exactamente las mismas y que la mayor diferencia radica en las Ocupaciones elementales, puesto que se reducen drásticamente la diferencia entre géneros.

Si tenemos en cuenta el nivel de educación medio, vemos que las actividades que más aportan al número de empleados global en el año 2003 son: Técnicos, Manufactura, Servicios y Apoyo clerical. Las mayores diferencias entre géneros ocurren en la actividad de Servicios (ampliamente dominado por mujeres) y Manufactura (ampliamente dominado por hombres). Finalmente, al pasar los años, las actividades con más aportación se mantienen pero con diferente orden: Servicios, Técnicos, Manufactura y Apoyo clerical.

Para acabar, si miramos el nivel de educación más alto, Profesionales y Técnicos aportan más del 50 % de los empleados con este nivel de educación en el año 2003. Un dato curioso es que la aportación a ambos géneros de las diferentes actividades es bastante equitativa en este caso. Con el paso de los años, tanto las actividades como su aportación no sufren demasiados cambios.

Un dato interesante que podemos ver es que, con el paso de los años, la cantidad de trabajadores con el nivel de educación inferior se va reduciendo y es el nivel de educación

más alto el que va creciendo. El nivel de educación medio se mantiene y vemos como aporta aproximadamente el 50 % de los empleados en la Unión Europea.

Si analizamos los datos de España:

La mayor diferencia respecto a lo observado previamente en el conjunto Europeo, si cogemos todos los niveles de educación, radica en las actividades con más empleados en el año 2004: Manufactura, Ocupaciones elementales, Servicios y Profesionales. El orden de estas actividades varia si observamos años posteriores, como por ejemplo en el 2013: Servicios, Profesionales, Ocupaciones elementales y Manufactura.

Si consideramos únicamente el nivel de educación más bajo, en el año 2004, Manufactura, Ocupaciones elementales y Servicios tienen el 50 % de los empleados. Con el paso de los años, Servicios y Ocupaciones elementales prácticamente aportan el 50 % de los empleados. Destaca el crecimiento enorme que experimenta la actividad de Servicios y como su aportación a ambos géneros se va equilibrando más y más cada año que pasa.

Teniendo en cuenta el nivel de educación medio, volvemos a observar que no existe una aportación abrumadora de ciertos sectores en el año 2004 y que las actividades más comunes son: Servicios, Apoyo Clerical, Técnicos y Manufactura. Con el transcurso de los años volvemos a observar un crecimiento brutal de los Servicios y como las Ocupaciones elementales le ganan la partida a Manufactura consiguiendo más aportación.

El comportamiento si escogemos el nivel de educación más alto es muy similar al que hemos visto en Europa, consiguiendo Técnicos y Profesionales más del 50 % de los empleados durante todos los años.

En España observamos datos sorprendentes: en el año 2004, la cantidad de empleados con estudios de nivel inferior era superior al resto y, a diferencia de Europa, los trabajadores con un nivel de educación media ocupan el último peldaño en aportación. Con el paso de los años, el nivel de educación superior se erige como el más común entre los empleados, con cifras muy similares al nivel de educación inferior.

6.10. Planteamientos

Es momento de utilizar los datos extraídos de cada una de las gráficas para intentar responder las cuestiones planteadas durante la introducción:

- **¿Han alcanzado las mujeres a los hombres en cuanto a número de trabajadores?**

  Evidentemente, pese a que tanto Europa como España van por buen camino, de momento las mujeres no han alcanzado las cifras de empleo sobre las que se mueve el género masculino.

- **¿Influye el PIB en los salarios de los empleados?**

  Rotundamente sí, observamos una correlación más que evidente entre el PIB de un país y los salarios que tanto hombres como mujeres perciben. Pese a esto, es verdad que existe algún país cuyos salarios podrían ser superiores o inferiores si consideramos únicamente su PIB y la tendencia del resto de países europeos.

- **¿La inversión en educación garantiza el equilibrio de salarios entre hombres y mujeres?**

  Ciertamente, no. Independientemente del nivel de educación que estamos considerando, cuyo impacto sobre la diferencia de salarios es fuerte, no observamos que la inversión en educación ejerza ningún tipo de influencia sobre los salarios. Es más, algunos de los países con más igualdad son, precisamente, los que tienen menos inversión en educación.

- **¿En qué actividades existe más diferencia de empleados y como han ido evolucionando éstas con el paso de los años?**

  Las tres actividades en las que observamos más diferencia de empleados en función del género son: Manufactura, Servicios y Operadores de máquinas. Tanto la primera como la tercera se caracterizan por tener más hombres mientras que la segunda lo hace por tener muchísimas más mujeres trabajando en ella.

  A nivel Europeo, la actividad de Manufactura pasa de ser la segunda actividad con más empleados, a ser la cuarta, principalmente porque ve reducido tanto su nombre de trabajadores como el de trabajadoras. Mirando Servicios vemos como sufre un comportamiento inverso al de Manufactura. Pasa de ser la cuarta actividad a ser la segunda más concurrida. Además, ha visto aumentado, en cifras muy similares, su número de empleados masculinos y femeninos. Finalmente, vemos como Operadores de máquinas siempre es la tercera actividad con menos empleados y, con el paso de los años, se va reduciendo su aportación progresivamente.

- **¿Qué actividades han sufrido un mayor descenso de empleados durante los últimos años?¿Qué género lo ha sufrido más?**

  A nivel general, ha sido el género masculino el que ha visto reducido de manera más notoria su número de trabajadores durante los últimos años, concretamente a partir del 2008/2009. Las actividades que más han sufrido este bajón son: Mánagers, Técnicos, Agricultura/Forestal y Pesca, Manufactura, Operadores de máquina y Ocupaciones Elementales.

  La actividad de Mánager ha experimentado una fuerte bajada en el año 2010 en ambos géneros. Los técnicos han observado un descenso en el año 2011 de sus empleadas, mientras que el número de empleados se mantuvo. En Agricultura/-Forestal y Pesca, tanto hombres como mujeres han visto un desplome continuo en ambos géneros. Manufactura sufrió un descenso de empleados masculinos en el año

2008, mientras que las empleadas se mantuvieron en cifras similares. Los Operadores de máquina tienen un comportamiento muy similar al de manufactura. Para finalizar, las Ocupaciones elementales ven reducida desde el año 2009 su cantidad de trabajadores masculinos, mientras que su cifra de empleadas, se mantiene.

- **¿Como se distribuye la actividad laboral en función de los estudios alcanzados? En las actividades que requieren más educación... ¿existe más equilibrio entre el número de empleados y de empleadas?**

  Si consideramos el nivel de estudio más bajo, las actividades laborales en Europa se distribuyen de la siguiente manera: Ocupaciones elementales (21 %), Manufactura (19 %), Servicios (15 %), Operadores de máquina (13 %), Agricultura/Forestal y Pesca (10 %), Apoyo clerical (8 %), Mánagers (6 %), Técnicos (5 %), Profesionales (2 %) y Fuerzas armadas (1 %).

  Si consideramos el nivel de estudio medio, las actividades laborales en Europa se distribuyen de la siguiente manera: Técnicos (18 %), Manufactura (17 %), Servicios (16 %), Apoyo clerical (14 %), Operadores de máquina (10 %), Ocupaciones elementales (8 %), Mánagers (7 %), Agricultura/Forestal y Pesca (5 %), Profesionales (4 %) y Fuerzas armadas (1 %).

  Si consideramos el nivel de estudio más alto, las actividades laborales en Europa se distribuyen de la siguiente manera: Profesionales (44 %), Técnicos (20 %), Mánagers (12 %), Apoyo clerical (8 %), Servicios (5 %), Manufactura (4 %), Ocupaciones elementales (3 %), Operadores de máquina (2 %), Agricultura/Forestal y Pesca (1 %) y Fuerzas armadas (1 %).

  Considerando las actividades de Profesionales y Técnicos como aquellas que requieren más nivel de educación, sí que podemos apreciar como nos encontramos ante dos actividades cuyo número de empleados y de empleadas no tienen diferencias abismales, sino que realmente están más cercanas al 50 % que la mayoría del resto.

- **¿Las tendencias en algunos sectores se han ido invirtiendo con el paso de los años?**

  Existen dos actividades cuya tendencia ha sufrido un cambio considerable a partir del año 2010. Nos encontramos ante las dos actividades con más trabajadores con un alto nivel de educación: Técnicos y Profesionales.

  Observamos como los Profesionales, hasta el 2010, pese a ser una actividad con bastante equilibrio de género, siempre había tenido más hombres que mujeres trabajando en ella. Con la llegada de este año, observamos cómo esta situación pega un cambio radical en el que las mujeres son la que están por delante y, no sólo eso, sino que además todo parece indicar que esta diferencia se irá ampliando más y más con el transcurrir de los años.

A los técnicos les ocurre, justamente, la situación inversa. Nos encontramos frente a una actividad que, pese a que también a gozado de bastante equilibrio hasta el año 2010, siempre ha contado con más mujeres en sus filas. Pues bien, a partir de ese año, esta situación se invierte, estableciéndose un dominio del género masculino. A diferencia de la actividad anterior, no parece indicar que esta brecha siga siendo propensa a aumentar, sino a estrecharse con el paso de los años.

Capítulo 7

Conclusiones

La aparición del campo de *data visualization* viene estrechamente ligada con el auge de *data science*. Es evidente la gran aportación que aporta la visualización de datos al *data science*: frente a un gráfico bien diseñado somos capaces de extraer información de una manera más intuitiva, en cambio, si lo que tenemos delante es una simple tabla, seguramente nos cueste muchísimo más trabajo organizar los datos. Además, una representación visual permite analizar ciertos comportamientos que de otra manera serían mucho más difíciles de percibir como correlaciones o contribuciones a un conjunto. Uno de los mayores beneficios que aportan las representaciones visuales es la capacidad de combinar diferentes DataSets y lograr que esto no suponga una dificultad añadida a la hora de extraer la información. Probablemente, su uso más extendido sea mostrar una serie de resultados obtenidos mediante un análisis previo.

La visualización de datos interactiva va un paso más allá e involucra al espectador en la creación del propio gráfico para lograr experiencias personalizadas. Ya no solo se ve el gráfico como medio de intercambio de información entre creador y observador sino que permite al usuario definir los datos que quiere observar y manipularlo como crea conveniente. Pese a todo, también es cierto que se trata de un área que todavía se encuentra en expansión y que muchísimos aspectos que engloba están sujetos a una fuerte componente subjetiva. Durante el desarrollo de este proyecto, he podido observar cómo influye el diseño del gráfico en la forma en la que el observador interpreta los datos que tiene delante, con la parte positiva y negativa que esto conlleva. Mientras que por un lado es una muy interesante opción para mostrar ciertos resultados analíticos, también es peligrosa la sugestión que puede llegar a realizar sobre el observador.

A nivel de tecnologías e implementación, cabe destacar la aportación del framework Angular.js al proyecto. Gracias a éste y a su estructura jerárquica de $scopes, la creación de un sistema de variables compartidas para todos los elementos que forman parte del *DashBoard* se ha simplificado de sobremanera. Si bien es cierto que algunas funcionalidades que aporta podrían haber sido reemplazas por soluciones de complejidad similar,

puedo concluir que la incorporación ha resultado beneficiosa para la creación de la aplicación SPA (Simple Page Application) en su conjunto. La modularización que aporta, junto a la posibilidad de un diseño MVC, ha resultado de muchísima ayuda, no solo para estructurar el código JavaScript, sino también para controlar el flujo de ejecución de los diferentes procedimientos de la aplicación. Otro punto interesante de Angular.js son las directivas, ya que nos permiten configurar secciones de nuestra página independientes del resto y asignarles comportamientos ante su compilación. Esto provoca que podamos diseñar conjuntos de directivas que actúen como un 'framework' utilizable por terceros siempre y cuando se realicen las importaciones adecuadas.

D3.js es, desde mi punto de vista, la mejor librería existente para trabajar con SVG en programación web. Su mayor virtud radica en la simplicidad de dibujar elementos sencillos en este formato, tales como rectángulos y círculos. Además, ofrece un conjunto de funciones predefinidas que simplifican aspectos básicos que deben tener los gráficos como ejes y escalas. Pese a esto, crear composiciones complejas requiere un nivel de conocimiento bastante amplio de la librería que no es fácil obtener. Se trata de una librería bastante popular que cuenta con una importante aportación por parte de la comunidad, con lo que es fácil de encontrar por la red multitud de gráficos complejos a modo de ejemplo. Para ayudar en la creación de estos gráficos de complejidad superior, D3.js implementa layouts predefinidos que, pese a que son útiles, al carecer de cierta flexibilidad, hacen que algunas estructuras sean demasiado difíciles de representar. Para finalizar, quiero comentar que la implementación y el uso de las transiciones son verdaderamente útiles pese a que algunos aspectos de éstas se podrían mejorar, como por ejemplo la generación de transiciones en bucle o las transiciones sobre agrupaciones. Uno de los mayores problemas existentes para gestionar las transiciones radica en la carencia de la agrupación de las transiciones que actúan sobre elementos ya agrupados. En definitiva, creo que nos encontramos frente a una librería muy útil pese a que tenga algunas funcionalidades complejas, motivo por el cual se han creado librerías basadas en ésta simplifican estos aspectos, como por ejemplo: NVD3, DC.js, Dimple, xCharts, etc.

7.1. Trabajo Futuro

En el desarrollo de este proyecto nos hemos centrado principalmente en el campo de la visualización de datos, pero *data science* nos permite efectuar múltiples acciones diferentes sobre estos datos para potenciar los conocimientos que queremos representar. Así pues, para concluir el trabajo, me gustaría comentar algunas de las posibles vías de evolución que podrían darse en etapas futuras para complementar este proyecto:

- Aprovechar los datos estructurados de las gráficas creadas para calcular posibles predicciones sobre los países de la Unión Europea.

- Implementar un clasificador para identificar grupos de países con comportamientos similares.

A nivel tecnológico, algunos cambios que podrían introducirse que considero interesantes:

- Implementar los cálculos anteriores en un servidor programado en JS utilizando la tecnología Node.js.
- Desarrollar un webservice para el acceso y filtrado de los datos de Eurostat.
- Implementar el entorno web en lenguaje HTML5.
- Desarrollar, a partir de las directivas, una librería de visualización basada en D3/Angular.

Bibliografía

[1] Keim, D.A.: Information visualization and visual data mining, `http://ieeexplore.ieee.org/servlet/opac?punumber=2945`, 2002.

[2] Shneiderman, B.: The eyes have it: a task by data type taxonomy for information visualizations,
IEEE Symposium on Visual Languages, 1(1):336-343, 1996.

[3] Nick Qi Zhu: Data Visualization with D3.js Cookbook, *Packt Publishing Ltd.*, 2013.

[4] Schroeder, W. & Martin, K. & Lorensen, B.: The Visualization Toolkit,
VTK Community, (3):1-5/379-398, 2004.

[5] http://www.datavizcatalogue.com/index.html

[6] http://datavisualization.ch/

[7] http://www.dashingd3js.com/table-of-contents

[8] http://d3js.org/

[9] https://angularjs.org/

[10] http://www.codecademy.com/es/learn/learn-angularjs

[11] https://keen.io/blog/101269629091/charts-on-grids-responsive-dashboard-templates-with?s=twc1

[12] http://homes.esat.kuleuven.be/ bioiuser/blog/from-interactive-to-interactivated-visualizations/

[13] http://www.visualnews.com/tag/data-visualization-101/

[14] http://www.bloomberg.com/dataview/2014-04-17/how-americans-die.html

[15] http://animateddata.co.uk/articles/d3/whatisd3/

[16] http://ec.europa.eu/eurostat

[17] https://en.wikipedia.org/

[18] http://startbootstrap.com/