GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

VISUALIZACIÓN DESCRIPTIVA MEDIANTE GEOLOCALIZACIÓN DEL ANÁLISIS DE CONSUMOS ELÉCTRICOS POR UNA COMPAÑÍA ELÉCTRICA

Autor: Javier Caminos Colmenarejo

Directores: David Contreras Bárcena y Miguel Ángel Sanz Bobi

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Visualización descriptiva mediante geolocalización del análisis de consumos eléctricos por una compañía eléctrica

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2018/19 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Javier Caminos Colmenarejo Fecha: 20/ 05/ 2019

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: David Contreras Bárcena y Miguel Ángel Sanz Bobi Fecha: 20/ 05/ 2019

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Javier Caminos Colmenarejo DECLARA ser el titular de los derechos de propiedad intelectual de la obra: "Visualización descriptiva mediante geolocalización del análisis de consumos eléctricos por una compañía eléctrica", que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.
d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción

de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 20. de Mayo de 2019

ACEPTA

Fdo: Javier Caminos Colmenarejo

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

VISUALIZACIÓN DESCRIPTIVA MEDIANTE GEOLOCALIZACIÓN DEL ANÁLISIS DE CONSUMOS ELÉCTRICOS POR UNA COMPAÑÍA ELÉCTRICA

Autor: Javier Caminos Colmenarejo

Directores: David Contreras Bárcena y Miguel Ángel Sanz Bobi

Madrid

Agradecimientos

Esta sección es opcional

VISUALIZACIÓN DESCRIPTIVA MEDIANTE GEOLOCALIZACIÓN DEL ANÁLISIS DE CONSUMOS ELÉCTRICOS POR UNA COMPAÑÍA ELÉCTRICA

Autor: Caminos Colmenarejo, Javier.
Directores: Contreras Bárcena, David y Sanz Bobi, Miguel Ángel.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

Sobre un análisis avanzado de los consumos eléctricos, se ha desarrollado una potente herramienta de visualización descriptiva de la información y los resultados, mediante geolocalización. Para ello se ha realizado un previo procesamiento de los datos para obtener las tablas de datos necesarios para poder utilizarlas de la manera más óptima en un entorno web. Dicha presentación web está basada en un mapa de España donde se puede observar de manera dinámica, en función del día y la hora del día, los diferentes consumos de los municipios agrupados en provincias. Los consumos que se pueden visualizar incluyen la comparación del consumo residencial, consumo industrial y el consumo total de cada provincia.

Palabras clave: Consumos eléctricos, Apache Spark, Javascript, d3.js, Big Data, Visualización, Entorno web.

1. Introducción

La visualización de datos es de gran importancia en la actualidad ya que estos crecen a un ritmo muy elevado. Emplear un modo visual que presente la información de un gran volumen de datos permite ahorrar tiempo y costes, debido a que la información se puede visualizar de forma clara y sencilla, permitiendo una mejor toma de decisiones. Por otra parte, dicha visualización puede ser compartida con los distintos integrantes de una empresa a diferentes niveles o departamentos, sin necesidad de ser expertos en la materia del Big Data para comprender las visualizaciones. Resulta bastante sencillo para todos el poder interactuar con los datos para facilitar la interpretación y el análisis.

En definitiva, la visualización de datos pretende describir la presentación de la información abstracta en forma gráfica. Permitiendo detectar patrones y tendencias que de otra manera podrían pasar desapercibidos en informes o tablas de datos.

2. Definición del proyecto

La realización de este proyecto incluye las siguientes características u objetivos:

- Comprender y estudiar los datos disponibles sobre los consumos eléctricos de España. Esto incluye un filtrado tanto para eliminar los datos que no están relacionados con lo que se pretende visualizar, como una comprobación de que los datos son coherentes para evitar futuros errores al extraer conclusiones.
- Exportar el resultado del procesamiento de los datos para poder visulizarlos de manera efectiva en un entorno web.

- ❖ Desarrollar una dinámica y potente herramienta de visualización basada en un mapa de España donde se incluirán diferentes técnicas de representación de los datos.
- ❖ Proporcionar con dicha herramienta de visualización una manera eficiente de explorar los datos para poder tomar decisiones futuras sobre consumos eléctricos de manera que se beneficien las compañías eléctrcicas.

3. Descripción del modelo/sistema/herramienta

La finalidad del proyecto consiste en visualizar tanto los datos iniciales como los datos después de un análisis Big Data sobre ellos. Los datos se encuentran en el Clúster Big Data de ICAI, y el primer procesamiento de los datos se realizará en Jupyter Notebook dentro del propio servidor.

En primer lugar, se realizará la exploración y observación de los datos. Se hará un estudio de los mismos, es decir, se aplicarán técnicas de procesamiento como la transformación, limpiado, reducción e integración de los datos. Una vez procesados los datos se llevarán a cabo los filtrados pertinentes para excluir los parámetros que no son útiles para la visualización de los mismos. Esta tarea se finalizará generando distintos datasets en formato CSV que se exportarán al entorno de visualización.

El siguiente paso consiste en visualizar los datasets de manera coherente, potente y dinámica. Para ello, se representarán los datos en un entorno web, en concreto, utilizando tecnologías que involucran tanto HTML, Javascript y CSS. Las diferentes tecnologías que se podrían haber aplicado para la visualización fueron estudiadas al principio del desarrollo del proyecto, siendo la mencionada anteriormente la más adecuada. Se desarrollará una página web que consistirá de un mapa político de España al cual se añadirán SVGs, interacción con el usuario y dinamismo para observar la evolución de los consumos eléctricos a lo largo del tiempo de los datos. En cuanto al término SVG, se refiere a Scalable Vector Graphics, siendo abierto y basado en los gráficos XML, con ellos se pueden asociar datos a representaciones gráficas, [6].

La librería Javascript que se utilizará principalmente es d3.js (Data-Driven Documents), que es capaz de crear, a partir de unos datos, potentes visualizaciones dinámicas e interactivas en entornos web.

Ilustración 1 - Esquema del flujo del proyecto

4. Resultados

Los resultados del proyecto han sido satisfactorios ya que se ha desarrollado una herramienta de visualización web potente, dinámica e interactiva con el usuario, que era el principal objetivo.

El usuario será capaz de visualizar en un mapa de España los datos sobre consumos eléctricos en forma de círculos o rectángulos sobre el mapa, y en forma de mapa de calor. Las visualizaciones podrán ser elegidas por el usuario según el tipo de consumo (consumo total acumulado, consumo residencial acumulado, consumo industrial acumulado, consumo promedio total, consumo promedio residencial y consumo promedio industrial) y según la variable temporal (consumo absoluto cada día, consumo absoluto cada hora del día y consumo acumulado cada hora en un día), pudiendo escoger el rango de fechas que al usuario le interese.

Las siguientes imágenes muestran el consumo eléctrico en diferentes fechas en forma de mapa de calor y barra. La figura 2 muestra el consumo en un día y la figura 3 el consumo a una hora del día.

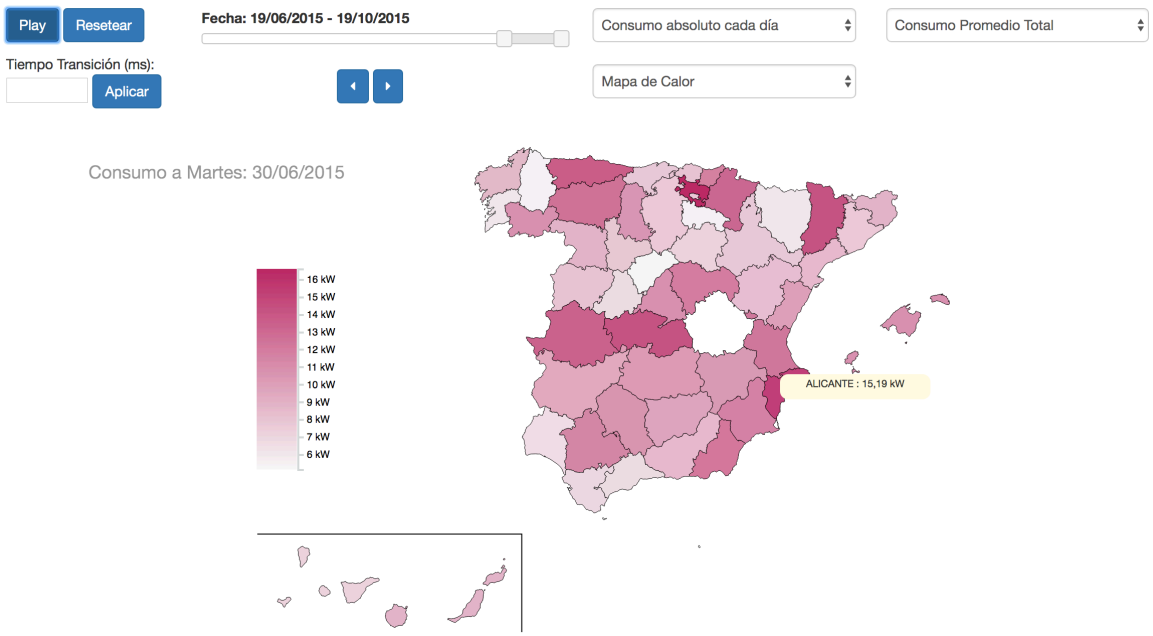

Ilustración 2 - Consumo promedio total el martes 30 de junio de 2015

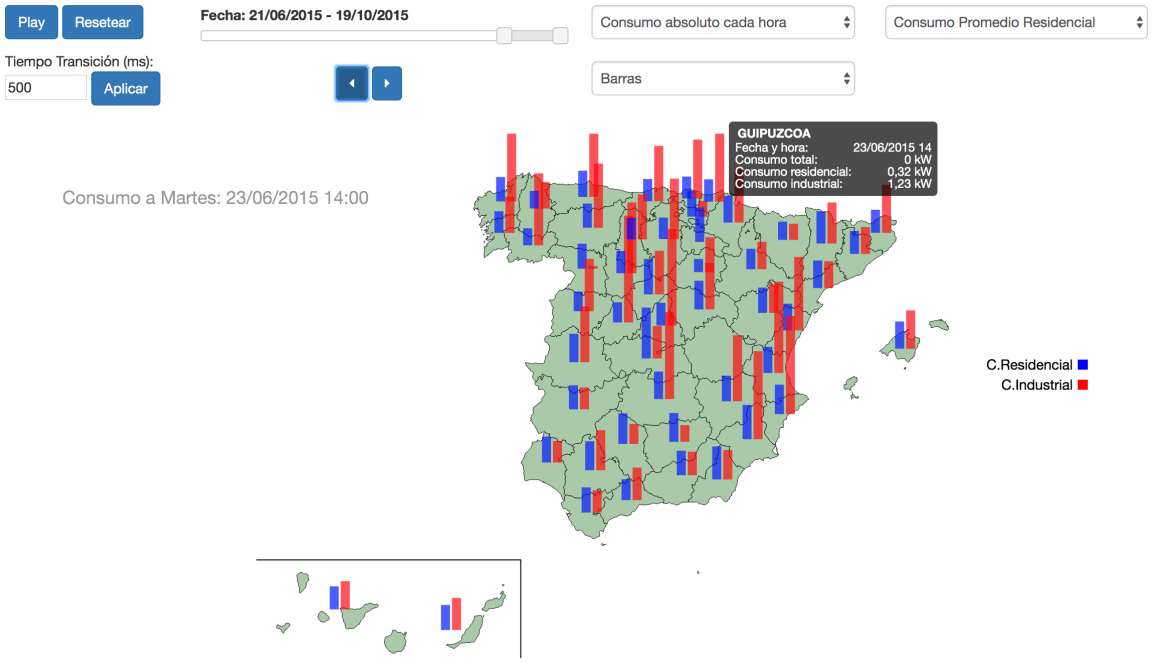

Ilustración 3 - Consumo promedio residencial e industrial el martes 23 de junio de 2015 a las 14:00

5. Conclusiones

Gracias a esta herramienta de visualización el usuario podrá visualizar los datos sobre consumos eléctricos en España de manera sencilla para encontrar patrones, encontrar elementos atípicos y ser capaz de entender los datos. Esto puede ser de utilidad de cara a la empresa ya que permite tomar decisiones de manera que se ahorre en tiempo y costes.

6. Referencias

[1] What is Scalable Vector Graphics (SVG)?, 2019, Techopedia, https://www.techopedia.com/definition/5239/scalable-vector-graphics-svg

DESCRIPTIVE VISUALIZATION BASED ON GEOLOCATION OF THE ANALYSIS OF ELECTRICITY CONSUMPTION BY AN ELECTRIC COMPANY

Author: Caminos Colmenarejo, Javier.
Supervisors: Contreras Bárcena, David y Sanz Bobi, Miguel Ángel.
Collaborating Entity: ICAI – Universidad Pontificia Comillas.

ABSTRACT

After an advanced analysis of electricity consumptions, a powerful descriptive visualization tool has been developed in order to display information and results, based on geolocation. To make it possible, a previous data processing was performed to obtain the data tables needed to use them in an optimal manner in a web environment. The web presentation is based on a Spain map where the user can observe, in a dynamic way, the consumption of the municipalities grouped by provinces in each day and time of the day. The consumptions available to display include the comparison between the residential consumption, industrial consumption and the total consumption in each province.

Keywords: Electricity consumption, Apache Spark, Javascript, d3.js, Big Data, Visualization, Web environment.

1. Introduction

Data visualization is really important nowadays because data is growing at a hectic pace. Using a visual method that represents a big volume of data allows us to save time and expenses, favoring better decisions. Furthermore, those visualizations can be shared between employees of a company from different departments or level of knowledge, without being experts in Big Data, and they can understand them. It is easier for everyone to interact with data to facilitate the interpretation and the analysis.

Ultimately, data visualization intends to describe the presentation of abstract information in a graphic format. Allowing us to detect patterns and trends that would go unnoticed in reports or data tables.

2. Project definition

The completion of this project include the following features or objectives:

- Understand and study the available data from the electricity consumption in Spain. This includes filtering the data to remove the data that doesn't contribute to the visualization and checking that the data is consistent to prevent future errors when drawing conclusions.

- Export the results obtained by preprocessing the data to be able to display them effectively in a web environment.

- ❖ Develop a powerful and dynamic visualization tool based on a Spain map where different representation of data techniques will be applied.

- ❖ Provide with the visualization tool an efficient manner to explore the data to draw future conclusions of the electricity consumptions, so electricity companies can beneficiate from it.

3. Description of the model/system/tool

The purpose of the project is to visualize both the initial data and the data after a Big Data analysis of them. The data can be found in ICAI´s Big Data Cluster and the first data processing will be done in Jupyter Notebook in the server.

First of all, an exploration and scan of the data will be performed. A study of them is needed, in other words, processing techniques such as transformation, cleaning, reduction and integration of the data will be applied. Once the preprocessing is done, filtering the data will be done to exclude the parameters that aren't useful for the visualization. This task will end up in generating multiple datasets in CSV format that will be exported to the web page.

The next step consists in display the different datasets in a coherent, interactive, dynamic and powerful manner. To make it happen, a web page will be developed, using technologies such as HTML, Javascript and CSS. Other technologies could have been applied as well, but a comparison between all of them was done at the beginning of the project, giving as a result the ones mentioned. A web page will be developed and it will include a political map of Spain where diverse SVG´s, user interaction and dynamism will be included to observe the evolution of the electrical consumptions among the time provided by the data. The term SVG refers to Scalable Vector Graphics, it is open source and based on the XML graphics, and it associates data to graphic representations, [6].

The Javascript library that will be used mainly is called d3.js (Data-Driven Documents), it is capable of creating, from some data, powerful visualizations with user interaction and dynamism in web environments.

Ilustración 4 - Diagram of the project flow

4. Results

The results of the project have been successful since a powerful, dynamic and interactive web visualization tool has been developed, which was the main objective.

The user is able to visualize in a map of Spain the electricity consumption in variuos ways, with circles or rectangles on top of the map and with a heatmap. The visualization can be chosen by the user in terms of the type of consumption (accumulated total consumption, accumulated residential consumption, accumulated industrial consumption, average total consumption, average residential consumption and average industrial consumption), in terms of the time variable (absolute consumption per day, absolute consumption per hour of the day, accumulated consumption per hour in one day), also being able to select the date the user is interested.

The following images show the electricity consumption in different dates in a heatmap and bars way. Figure 5 shows the consumption in one day and figure 6 the consumption in one hour of the day.

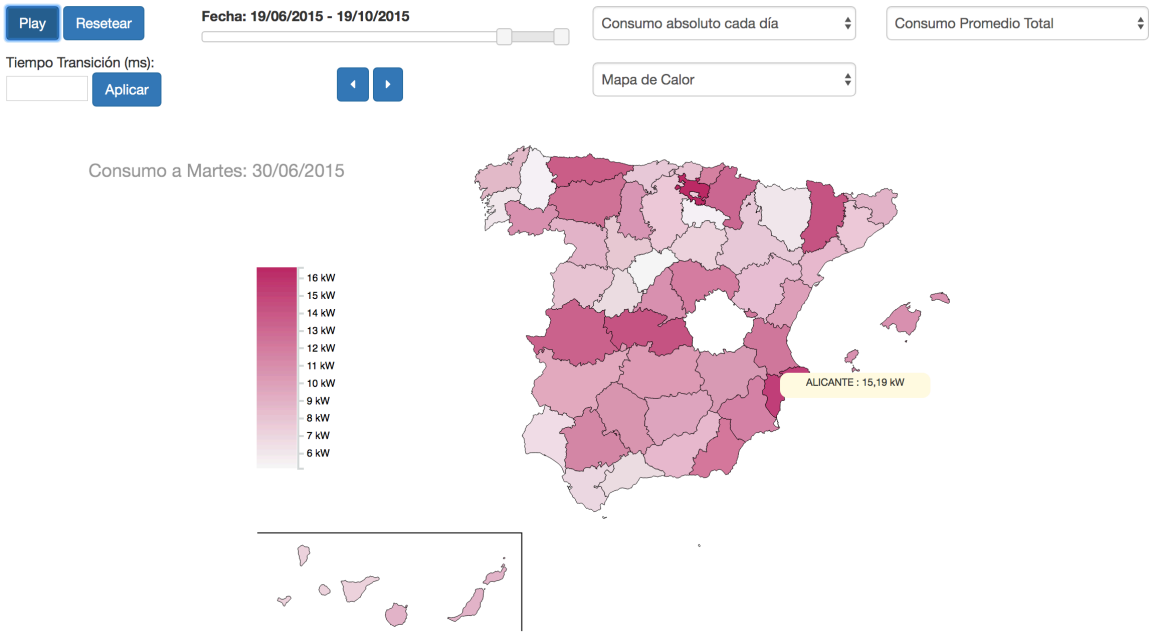

Ilustración 5 - Average total consumption on a Tuesday 30th of June 2015

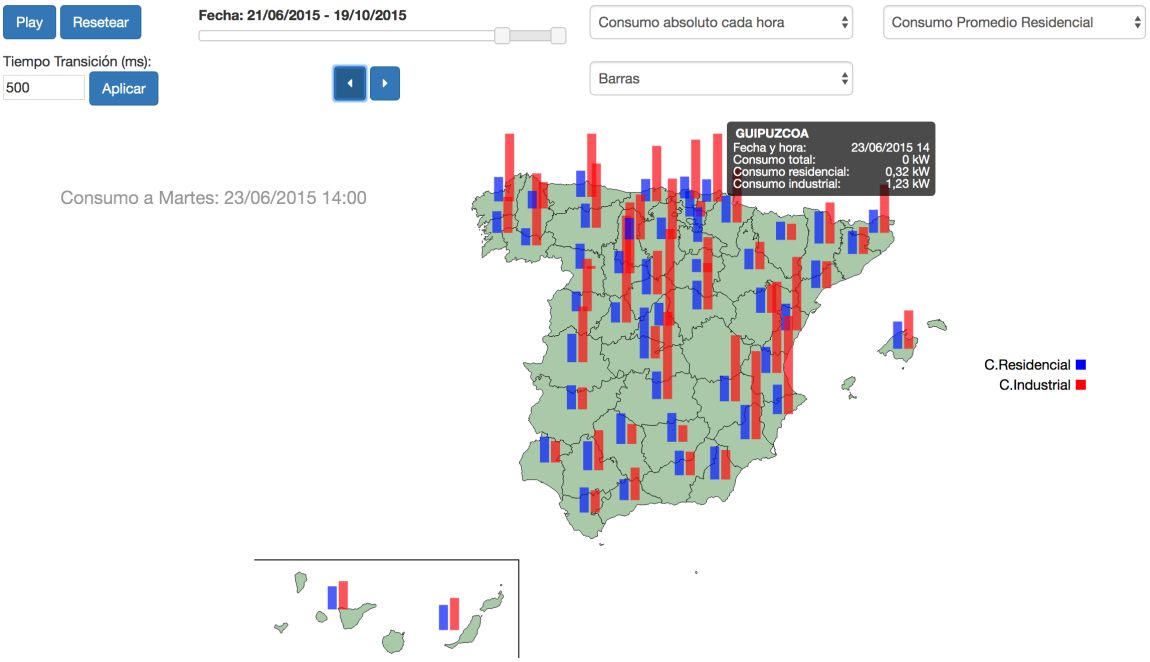

Ilustración 6 - Average industrial and residential consumption on a Tuesday 23rd of June 2015 at 14:00

5. Conclusions

Due to this visualization tool the user is able to display the electricity consumption data in Spain in a simple manner to find patterns, find outliers and understand easily the data. This could be very useful for a company because it allows you to make a choice or reach a decision quickly to save time and costs.

6. References

[1] What is Scalable Vector Graphics (SVG)?, 2019, Techopedia, https://www.techopedia.com/definition/5239/scalable-vector-graphics-svg

Índice de la memoria

Índice de figuras

Capítulo 1. INTRODUCCIÓN

Actualmente vivimos en una sociedad que ha evolucionado de un mundo analógico a un mundo digital. En esta sociedad de la información uno de los principales activos son los datos, los cuales se encuentran en cantidades ingentes y en continuo crecimiento. Esto provoca que los procesos entorno a estos datos también aumenten, no solo los relacionados con su organización, procesamiento o filtrado (Big Data), sino también en el campo de la infografía y de la visualización de los mismos. No se trata solo de utilizar tecnologías para analizarlos, sino de ser capaces de dotar de algún significado o sentido a esos datos para poder explicarlos a modo de historia.

La visualización de datos no tiene un origen novedoso. Desde la prehistoria el ser humano se ha preocupado por la visualización de datos con los dibujos de actividades cotidianas en las paredes de cuevas. La representación de datos ha evolucionado a lo largo de la historia pasando por los mapas de navegación, los primeros elementos 3D como vías de escape del plano, hasta las actuales gráficas de *networks*, [1]. Hace 20 años aproximadamente las empresas como las del sector eléctrico, medios de comunicación, entidades financieras o centros de investigación incorporaron gráficos de información impresos como manera de dar sentido y clarificar los datos. Luego estos impresos pasaron a las pantallas pero de manera estática. La novedad radica en el dinamismo de dicha información y la posibilidad de interacción por parte del usuario, todo ello propiciado por el desarrollo tecnológico de la informática y por supuesto, con la aparición de Internet.

Los datos cambian a gran velocidad, por lo que cualquier dato o información plasmado en una foto o gráfico estático puede pasar a estar desactualizado en cuestión de segundos. Como solución a este problema se propone la incorporación de una herramienta de visualización geotemporal, es decir, mostrar los datos sobre mapas y durante un periodo de tiempo concreto.

Los datos que se disponen para el desarrollo de la herramienta de visualización abarca un periodo de tiempo de casi dos años. Estos datos pertenecen al consumo eléctrico en España, por lo que una visualización geotemporal de los mismos es una manera óptima de hacerlo ya que los consumos están geolocalizados por municipios y tienen variables temporales que son el día y la hora del día. Con esta herramienta se pretenden dar sentido a un gran número de cifras (consumos) de manera sencilla para el usuario y que gracias a ella sea capaz de detectar anomalías, patrones y, en definitiva, sacar conclusiones reveladoras.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

A lo largo de la realización del proyecto se han utilizado una gran variedad de tecnologías. En un principio se exploraron diversas opciones para poder desarrollar la herramienta de visualización, entre las que destacan Tableau, StatPlantet, Shiny con el lenguaje de programación R y d3.js, librería de Javascript. Para ellas solo se realizó una comparación para ver cuál era la más apropiada. Por lo tanto, de las tecnologías verdaderamente usadas durante el proyecto caben destacar las siguientes.

Para empezar, relacionado con la etapa de observación y exploración de los datos, destacamos el uso del entorno de desarrollo Jupyter Notebook, que acompañado por el lenguaje de programación Python, ofrece una gran cantidad de posibilidades para realizar dicha tarea debido al gran número de librerías dedicadas a ello. De dichas librerías las más recurrentes son *pyspark*, que permite utilizar Spark mediante una API de Python; *numpy*, para agregar funcionalidades a vectores y matrices; y *pandas*, que ofrece estructuras de datos y operaciones para manipular y analizar datos.

En segundo lugar, para el desarrollo del entorno web, se ha optado por utilizar el lenguaje de programación Javascript, y en concreto, una librería llamada d3 (Data-Driven Documents), especializada en crear potentes visualizaciones. Aparte de esta librería, también podemos destacar otras como *jquery*, para facilitar la interacción con los elementos HTML; o *bootstrap*, para diseño y estructuración de la web. Este lenguaje de programación complementa que la página web es HTML, agregando estilo con el lenguaje de diseño gráfico CSS. Para implementar y escribir el código se ha utilizado el editor de código Sublime Text.

2.1 JUPYTER NOTEBOOK

Figura 1- Jupyter Notebook logo, [2]

Como explicado al comienzo de este capítulo, el código relacionado con el estudio y transformación de los datos se ha desarrollado utilizando Jupyter Notebook. Jupyter Notebook es una aplicación web open-source que permite crear y compartir documentos que contienen código fuente, ecuaciones, visualizaciones y texto. Entre los usos más destacados están: transformación y limpieza de datos, simulaciones numéricas, modelos estadísticos, visualización de información y Machine Learning, [2]. Jupyter Notebook ofrece la posibilidad de programar en más de 40 lenguajes distintos, siendo el lenguaje de programación Python el utilizado para el proyecto.

Cada cuaderno tiene una interfaz muy sencilla, teniendo en la parte superior un menú y una barra de herramientas, y debajo ya se encuentran las celdas, en las que se estructura el cuaderno. En cada celda, podemos escribir las líneas de código que deseemos teniendo en cuenta que cada celda se puede ejecutar por separado y en el orden que el usuario prefiera. Por ello, se pueden dividir las celdas en operaciones distintas y así ejecutar las que el usuario

considere oportunas en cada momento. El resultado de la ejecución de cada celda aparecerá debajo de la misma.

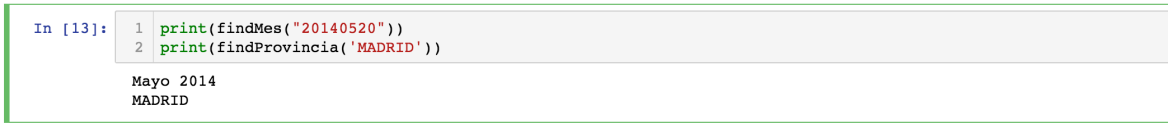

Figura 2- Ejemplo de celda ejecutada en Jupyter Notebook

2.2 SPARK

Apache Spark es un framework de computación en clúster open-source de propósito general y orientado a la velocidad. Proporciona APIs en Java, Scala, R y Python, que es el utilizado en este caso. Proporciona un motor optimizado que soporta la ejecución de grafos en general, suele ser utilizado para realizar consultas, análisis y transformaciones sobre los datos de manera rápida. Fue creado en 2009 en la universidad de California, en el AMPLab de Berkeley, [3].

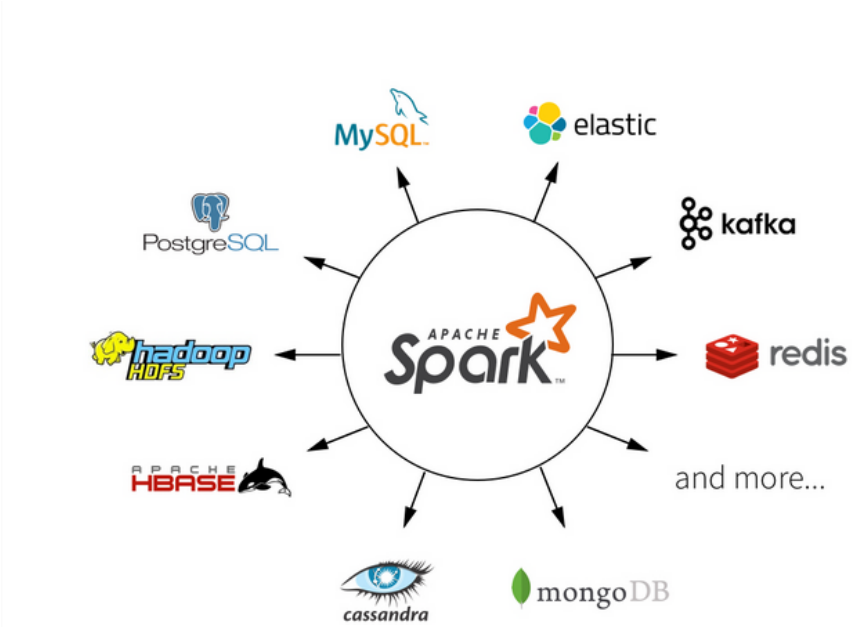

Figura 3- Apache Spark, [3]

Spark nos ofrece una gran cantidad de funcionalidades SQL, como la librería *pyspark.sql*, que serán utilizadas para la exploración de datos ya que es capaz de trabajar con cantidades grandes de datos (llegando incluso a petabytes) a una gran velocidad.

2.3 SUBLIME TEXT

Para desarrollar el código de la página web, se ha optado por utilizar Sublime Text. Sublime Text es un editor de texto y editor de código fuente escrito en C++ y Python para los plugins, [4]. En un principio, se desarrolló como una extensión de Vim (otro editor de texto) pero poco a poco fue creando una entidad propia, [4]. Es de descarga gratuita, pero no es software abierto y se debe obtener una licencia para su uso continuado. Aún así, la versión de evaluación es plenamente funcional y no tiene fecha de caducidad, [4].

Se ha escogido este editor de texto porque contiene muchas ventajas de cara a escribir y modificar código. Las más destacadas incluyen, [4]:

- Múltiples selecciones de código para renombrar variables de manera sencilla.
- Saltos rápidos a distintos archivos y funciones con la funcionalidad "GoTo Anything".
- Posee un potente rendimiento y está disponible tanto para Mac, Windows y Linux.
- Tiene un minimapa para una previsualización del código, muy útil para desplazarse sobre él.
- Se pueden realizar búsquedas dinámicas ya sean por archivos, directorios o proyectos individualmente o en conjunto.
- Soporta más de 43 lenguajes de programación, cada uno con un coloreado y sintaxis que hacen muy sencilla su lectura.

```
var margin = { left:80, right:100, top:50, bottom:100 },
    height = 500 - margin.top - margin.bottom,
    width = 800 - margin.left - margin.right;
```

Figura 4- Múltiple selección de código para renombrar variables

Para ejecutar el código se ha utilizado un servidor http en el puerto 8000 de mi localhost mediante el comando $python3 –m http.server. El resultado de este comando muestra los ficheros del directorio desde donde introducimos dicho comando. Navegando por ellos y con los métodos que posee este servidor (do_HEAD() y do_GET()) es capaz de ejecutar el código para presentarlo en modo web.

2.4 D3.JS (DATA-DRIVEN DOCUMENTS)

Figura 5- D3.js logo

D3.js es una librería de Javascript para manipular documentos basados en datos. D3 hace posible animar datos utilizando HTML, SVG y CSS. Su enfoque en estándares web hace que se utilicen los buscadores modernos en su máximo rendimiento, combinando potentes componentes de visualización y un enfoque data-driven para manipular el DOM (Document Object Model), [5]. Por ejemplo, de un vector de números se puede generar tanto una tabla como un gráfico de barras con transiciones e interacción. Se trata de manipular documentos basados en datos de manera eficiente. D3 es extremadamente rápido para grandes cantidades de datos y permite comportamientos dinámicos e interacción en las visualizaciones. Otra ventaja de esta librería es la gran comunidad detrás de ella, donde muchas dudas se pueden resolver de manera rápida.

Algunos ejemplos de potentes visualizaciones con esta librería son los siguientes:

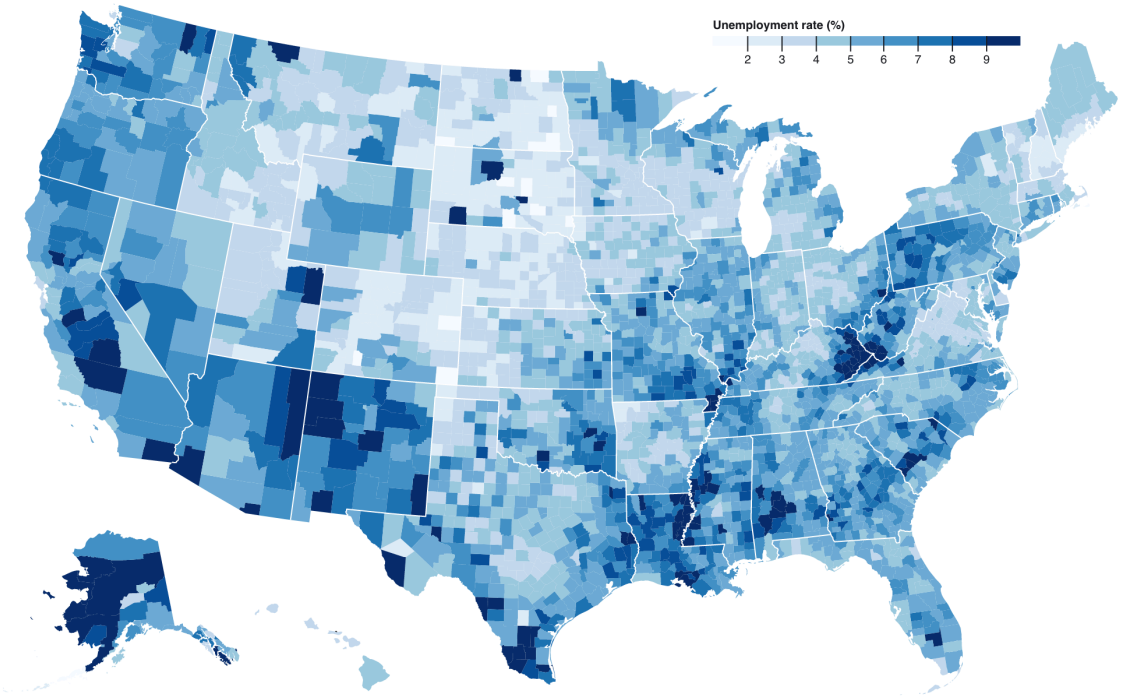

Figura 6 - Mapa de Estados Unidos con D3.js, [5]

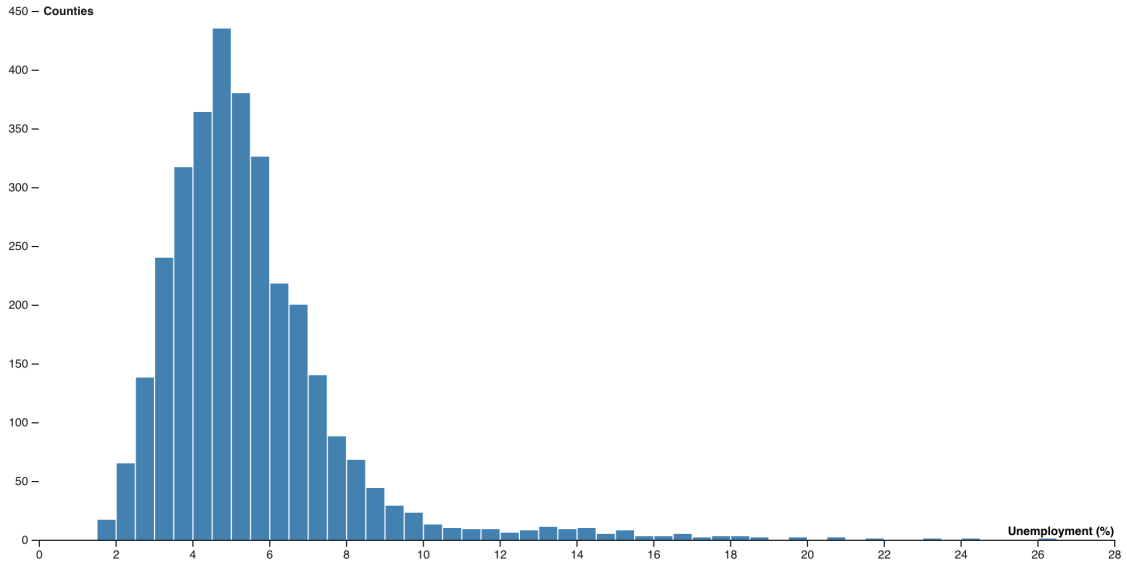

Figura 7 - Histograma con D3.js, [5]

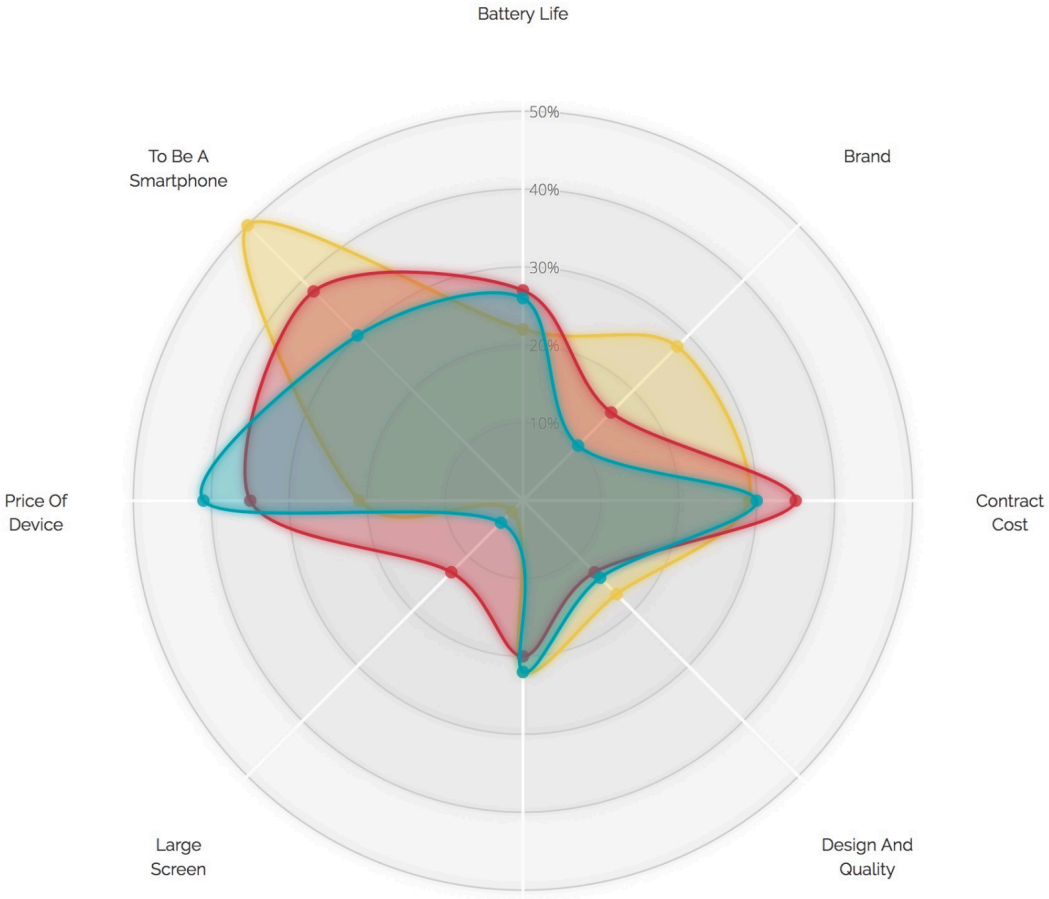

Figura 8 - Gráfico sobre la duración de la batería con D3.js, [5]

Capítulo 3. ESTADO DE LA CUESTIÓN

El objetivo principal de este proyecto consiste en visualizar de manera potente, dinámica, geolocalizada e interactiva los resultados de un análisis de datos sobre consumos eléctricos en España por una compañía eléctrica. Por lo tanto, en este capítulo se desarrollará la creciente digitalización del sector eléctrico, y la evolución de las herramientas de visualización de dicho sector.

3.1 DIGITALIZACIÓN DEL SECTOR ELÉCTRICO

Se entiende el suministro de energía eléctrica como la entrega de energía a través de las redes de transporte y distribución mediante contraprestación económica en las condiciones de regularidad y calidad que resulten exigibles, [6]. El grupo Red Eléctrica de España (representa un 93% del negocio total del grupo Red Eléctrica) se encarga del suministro de electricidad en España y cuenta con todos los activos de transporte y operación, los recursos humanos y los medios financieros vinculados a estas actividades, [7].

La misión del grupo Red Eléctrica de España es asegurar el correcto funcionamiento del sistema eléctrico español y garantizar en todo momento la continuidad y seguridad del suministro eléctrico, incorporando las nuevas tecnologías, [7]. Es reconocido a nivel mundial por ofrecer un servicio de máxima calidad, realizar una gestión ética y responsable, mantener un firme compromiso con el desarrollo sostenible y generar valor para todos sus grupos de interés, [7].

Hoy en día, vivimos en una sociedad orientada a la electrificación progresiva, propiciada por el cambio climático que da lugar a la progresiva descarbonización de la economía. Unido a esto, el desarrollo tecnológico está propiciando la introducción de material y equipamiento digital en un mundo que hace años era meramente analógico.

Según el grupo Red Eléctrica de España, esta digitalización del sector eléctrico se sustenta en cuatro maneras, [8]:

- El desarrollo de capacidades de aprovechamiento de la energía renovable disponible.
- El papel protagonista del cliente final.
- La optimización de la gestión de los activos.
- La optimización de los procesos operativos.

Este cambio de mentalidad pasa por identificar los datos como uno de los activos más importantes de una compañía y a su vez, el papel de los profesionales y de los clientes es mucho más relevante, ya que se comportan tanto como consumidores como productores, al tener acceso o producir datos que son muy valiosos. Para esta manipulación de datos y visualización de los mismos para cualquier tipo de usuario (responsable, profesional o cliente) es donde se centra el proyecto, en proporcionar dicha herramienta de visualización para el correcto entendimiento y procesamiento de los datos.

3.2 VISUALIZACIÓN DE DATOS EN EL SECTOR ELÉCTRICO

En el ámbito del sector eléctrico, las distintas empresas y compañías que participan en el mercado eléctrico poseen un gran volumen de datos y con información potencialmente muy valiosa, [8]. La visualización de estos datos, y el uso de herramientas que cumplen dicho cometido son fundamentales para agilizar el proceso de estudio de los datos, ahorrar tiempo y esfuerzos tanto a los expertos, que deben determinar de la manera más eficiente y precisa si el procesamiento de los datos se ajusta con lo esperado, como para cualquier otro tipo de implicado que pueda ver expresado de manera sencilla una gran cantidad de datos, que suelen ser inmensas tablas con filas y columnas con simples números.

Poder detectar patrones en el consumo de los usuarios es de vital importancia en el sector eléctrico para que la empresa pueda beneficiarse de ello, creando perfiles cada vez más específicos sobre sus clientes. Estas tomas de decisiones para modificar el comportamiento de la empresa de cara a los clientes son críticas ya que un error puede provocar la pérdida de

ganancias por parte de la empresa. El cerebro humano es capaz de detectar anomalías, patrones o comportamientos atípicos debido principalmente a la información recibida por la vista. Por lo que el uso de herramientas de visualización no solo simplifica y refuerza este tipo de actividades, sino que limita los riesgos en las tomas de decisiones.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

En un previo intento, las propuestas de visualización de datos se limitaron a gráficas estáticas sin capacidad de interacción por parte del usuario y sin estar geolocalizadas. Algunos ejemplos de estas gráficas incluyen gráficos de bloque, gráficos de barras o gráficos circulares, como se puede observar en las figuras Figura 9 - Gráfico circular estático y Figura 10. Estos gráficos por supuesto que aportan una gran cantidad de información de manera clara y concisa, pero si se pudieran combinar con una herramienta capaz de aportar dinamismo, interacción, geolocalización y atracción al usuario, facilitarían todas las tareas mencionadas en este apartado. El proyecto consiste en desarrollar dicha herramienta de visualización aplicada a consumos eléctricos en España.

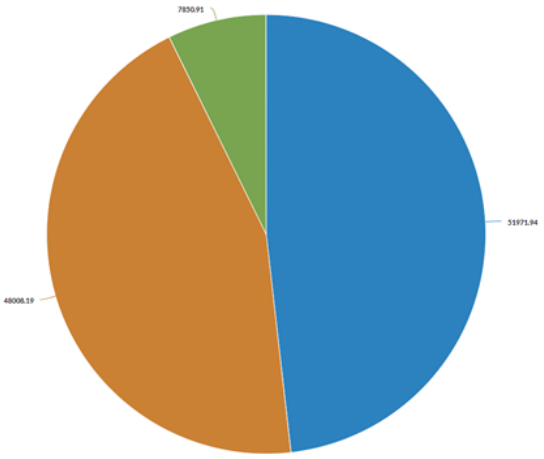

Figura 9 - Gráfico circular estático, [9]

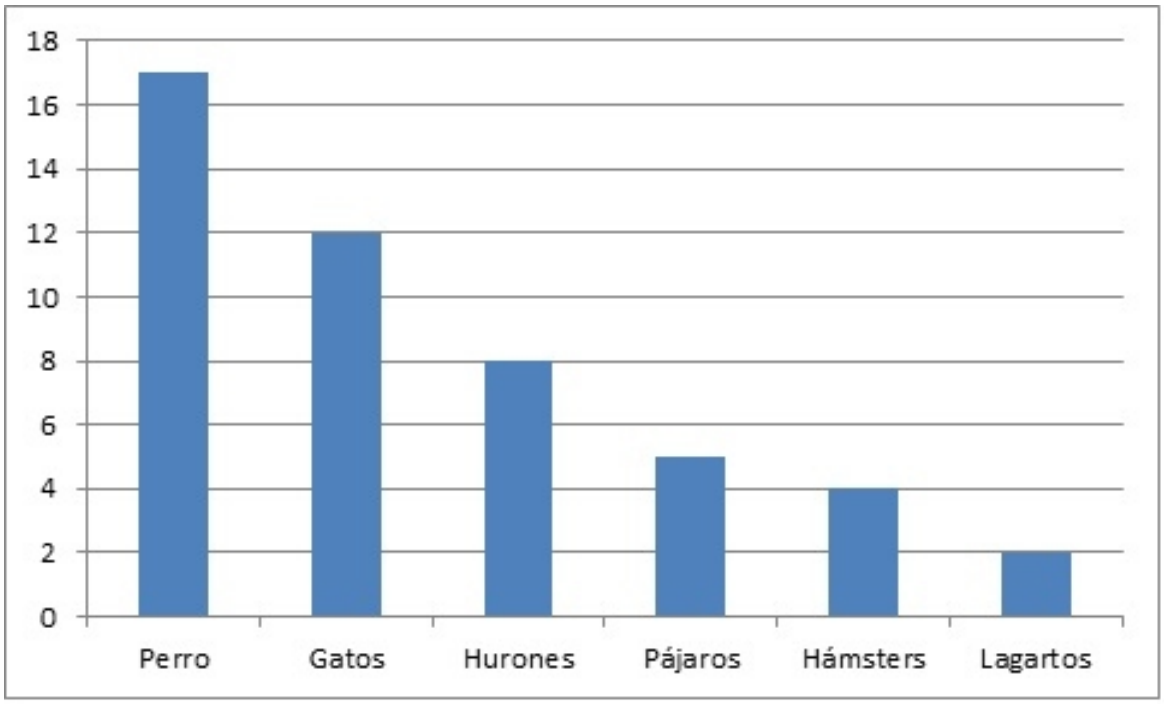

Figura 10 - Gráfico de barras estático, [10]

4.2 OBJETIVOS

1. **Comprender y analizar los datos disponibles de consumos eléctricos**

   Esto incluye realizar un estudio sobre cada una de las variables, encontrar posibles errores en los datos, ya que el tamaño del fichero es de 8 GB, y estructurar los datos de manera que sean adecuados para crear la herramienta de visualización.

2. **Representar los datos sobre consumos eléctricos y las relaciones que hay entre ellos**

   Se generarán distintos datasets a partir de los datos iniciales, los cuales estarán formados por transformaciones de variables, nuevas variables y eliminación de variables inútiles para el caso. Se representarán los consumos eléctricos en función de distintos factores, como el día de la semana, hora del día, el tipo de actividad

económica (industrial o residencial) o la provincia. Esto permitirá interpretar los datos de manera sencilla mediante un mapa geotemporal.

3. **Presentar las visualizaciones en un entorno web**

   Para lograr una visualización potente, dinámica e interactiva, en vez de representar los resultados en simples gráficas estáticas mediante Jupyter Notebook, se usarán tecnologías web avanzadas. Estas tecnologías ofrecen una mayor variedad de opciones de visualización que permiten a cualquier usuario interactuar con los datos indirectamente y en cualquier lugar al ser una aplicación web.

   Un objetivo complementario de este sería el escoger una tecnología adecuada para desarrollar la herramienta debido a la gran cantidad de opciones disponibles (D3.js, Tableau…).

4.3 METODOLOGÍA

Para comenzar, el primer paso sería realizar una comparación entre las tecnologías más potentes para llevar a cabo la visualización. Las características que se buscan es una tecnología capaz de crear visualizaciones geotemporales con gran cantidad de interacción y para un gran volumen de datos. Otra característica importante sería la posibilidad de representar la visualización en un entorno web para ofrecer disponibilidad al usuario en cualquier momento. No obstante, se estudiarán otros métodos que no impliquen tecnologías web. Entre las tecnologías más destacadas se encuentran Tableau, D3.js, StatPlanet o Shiny con R.

El siguiente paso sería la observación y exploración de los datos disponibles. Al tener una gran cantidad de datos (8 GB) hay que asegurarse que contienen errores como puede ser un consumo negativo o una fecha fuera del rango del resto. Se hará un estudio de los mismos, se aplicarán técnicas de procesamiento de datos para obtener las variables necesarias para

una correcta visualización de los consumos eléctricos en función de la provincia, el día de la semana y la hora del día. Estas técnicas de procesamiento incluyen filtrado de los datos, eliminación y creación de variables, y transformaciones para generar datasets que se exportarán al entorno de desarrollo de la herramienta de visualización. Esta fase del proyecto se realizará con Jupyter Notebook utilizando Python como lenguaje de programción.

Una vez procesados los datos y teniendo los datasets disponibles, se procederá al desarrollo de la visualización. El objetivo de la herramienta será representar, mediante un mapa de España, los consumos eléctricos de manera geolocalizada por provincia. El mapa ofrecerá que los consumos sean dinámicos e interactivos, ofreciendo la posibilidad de visualizarlos a lo largo del tiempo, según el interés del usuario. Es decir, que se podrán observar los consumos en función del día de la semana o la hora del día y según el tipo de consumo: total, residencial o industrial.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

Se ha desarrollado un diagrama de Gantt para realizar una presentación temporal de las actividades a realizar en el proyecto, siguiendo el siguiente cronograma:

Nombre de la tarea	Fecha de inicio	Fecha final	Duración (dias)
Análisis de diferentes tecnologías de visualización	16/1/19	31/1/19	15
Exploración y estudio de los datos	1/2/19	15/2/19	14
Desarrollo de la aplicación web	16/2/19	1/5/19	74
Revisión proyecto y pulir detalles	2/5/19	15/5/19	13
Redactar memoria d	16/5/19	31/5/19	15

Figura 11 - Actividades a realizar

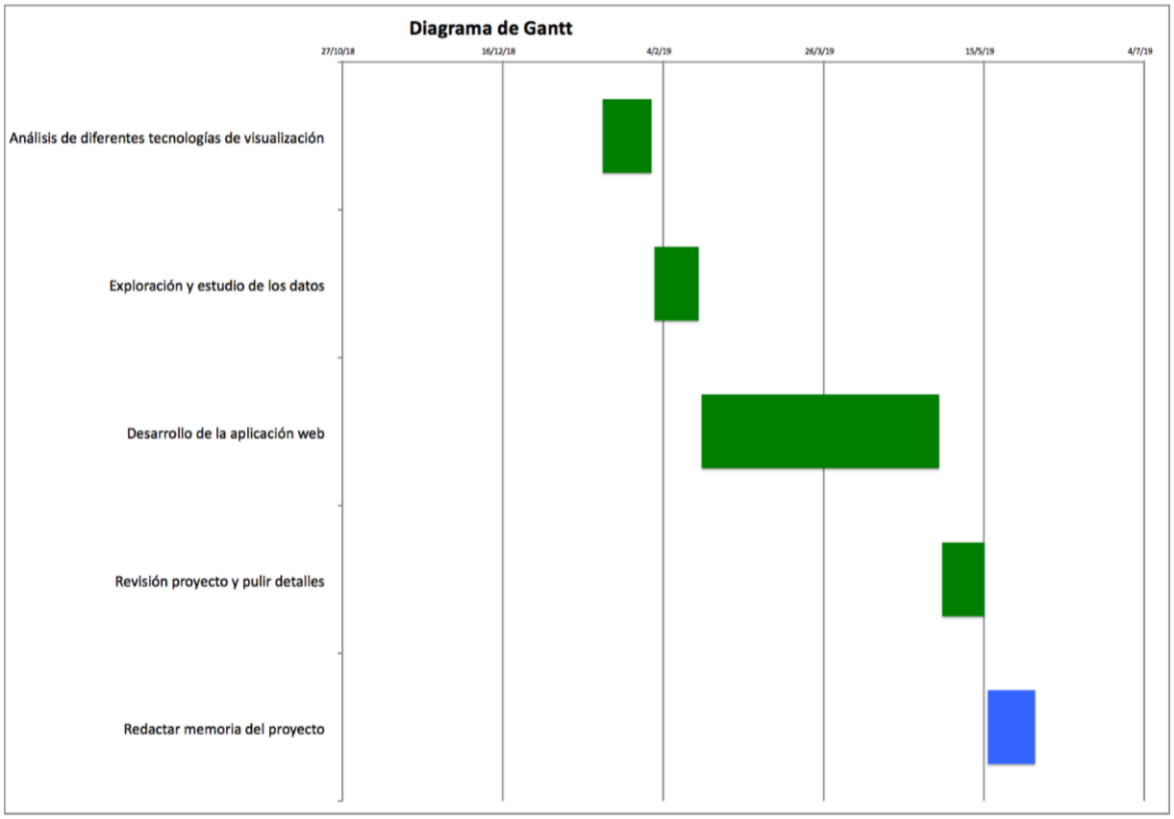

Figura 12 - Diagrama de Gantt

Las fechas de las distintas fases se podrían modificar debido a fallos técnicos o cualquier dificultad con el desarrollo o servidor. Otro aspecto a tener en cuenta es que el procesamiento de los datos puede ser una actividad recurrente a lo largo del desarrollo de la herramienta de visualización. Esto es debido a que ciertos datasets pueden no ser útiles para la plataforma o que se requieran nuevas formas de los datos, con lo que la actividad de exploración y estudio de los datos puede no darse por finalizada hasta que la visualización esté prácticamente desarrollada.

En la actividad de revisión del proyecto y pulir detalles lo que se pretende conseguir es un entorno amigable para el usuario, es decir, que el usuario se sienta cómodo al interactuar con la plataforma y pueda tomar decisiones de manera sencilla con solo un vistazo o un "click". En esta fase se tendrán en cuenta los colores usados, el tiempo de transición entre las distintas

visualizaciones y los tipos de elementos que van a representar los datos: círculos con diámetro variable, barras con distinta altura o un mapa de calor.

En cuanto a la estimación económica, al proyecto se le ha dedicado un total de 8 meses, de los cuales los últimos 6 meses han sido los de mayor concentración de trabajo. Se ha establecido comunicación con los directores del proyecto tanto a distancia como en persona, donde en persona la duración media de cada reunión ha sido de unos 45 minutos. Para desarrollar la web se ha tenido que aprender a programar con la librería de Javascript D3.js con un curso de precio 200$ y otras librerías mencionadas a lo largo del proyecto. Los primeros 4 meses se ha trabajado al 30% el proyecto mientras que los 4 últimos se ha trabajado al 90%.

Capítulo 5. ANÁLISIS DE TECNOLOGÍAS DE VISUALIZACIÓN

Para explicar el sistema final desarrollado se seguirán los mismos pasos que en la metodología del capítulo anterior, en la sección de Metodología. Por ello, comenzaremos explicando las diferentes tecnologías disponibles para desarrollar el proyecto y escogiendo la más adecuada (D3.js); la segunda parte se dedicará al procesamiento de los datos para extraer los distintos datasets, y para finalizar se describirá la herramienta de visualización final, los distintos componentes que la forman y la conclusión que se obtiene de ella.

Antes de comenzar con la implementación del proyecto en sí se ha tenido que realizar un estudio para comparar las tecnologías más apropiadas para desarrollar una potente herramienta de desarrollo. La tecnología que se busca para el proyecto tiene que ser capaz de generar una visualización geotemporal, que contenga a su vez dinamismo, interacción con el usuario y que pueda procesar una gran cantidad de datos en el menor tiempo posible.

En la actualidad hay una gran cantidad de opciones para intentar abordar el proyecto por lo que en este apartado se expondrán las más destacadas y, por supuesto, la utilizada en el proyecto. De cada tecnología se explicará su funcionamiento y las ventajas e inconvenientes de usarlas.

5.1 TABLEAU PUBLIC

Es la versión gratuita de Tableau, un software para visualización de datos que te permite crear gráficos interactivos, gráficos, mapas y cuadros de mando, [11]. El software guarda su trabajo en los servidores web públicos Tableau, como consecuencia, no se guarda en el equipo y su trabajo es accesible para cualquier persona para interactuar, descargar o modificar, [11]. No tiene un manejo tan sencillo como otras tecnologías, pero dedicándote

un poco de tiempo puedes lograr visualizaciones súper atractivas y también puedes combinar varios tipos de gráficos y maquetar tu propio dashboard.

Figura 13 - Tableau logo, [11]

Ventajas:

- Tiene un límite de hasta 10 millones de líneas por workbook.
- Se puede conectar a archivos locales con formatos de texto, a archivos de Google Sheets, Excel y Access. Puede conectarse también con archivos de software estadístico como SAS, SPSS y R.

Inconvenientes:

- Tanto los datos como las visualizaciones creadas en este entorno de desarrollo no son completamente privados, por lo que cualquier usuario puede acceder y modificar lo que vea conveniente.
- Relacionado con lo anterior, los archivos de visualización y datos del usuario se quedan guardados en la nube, no en el ordenador personal.

Como conclusión, podría haber sido una herramienta potente para el proyecto pero la interacción de cualquier usuario con el trabajo realizado y con los datos la hacen inviable, ya que los datos son de una compañía eléctrica proporcionados a la Universidad Pontificia de Comillas y deben permanecer a uso privado.

5.2 STATPLANET

StatPanet es una aplicación gratuita, que tiene la posibilidad de pagar para obtener diversas funcionalidades. Se utiliza para la creación de mapas interactivos muy personalizables, [12]. Este software también incluye la opción de gráficos interactivos e infografías, [12]. Se trata de una manera de visualizar los mapas que los colorea en función de la estadística que queramos desplegar. La aplicación ya contiene algunos datos (datos de la OMS, UNESCO…) con los que se puede interactuar.

Figura 14 - StatPlanet logo

Ventajas, [12]:

- Navegación sencilla para el usuario con efectos visuales potentes.
- Función de favoritos para comprar distintas estadísticas.
- Estadísticas fiables de fuentes oficiales.

Inconvenientes:

- Datos limitados y sin actualización automática.
- No se pueden exportar mapas como imágenes.
- Para poder introducir tus propios datos necesitas tener una versión del software de pago.

StatPlanet se presenta como una gran herramienta para visualizar datos de manera geotemporal pero el tener unos datos determinados y limitados con la versión gratuita impide que los datos sobre consumos eléctricos se puedan usar. Otra desventaja es que no hay posibilidad de crear una visualización dinámica de manera automática. Por estos motivos, el uso de este software queda descartado.

5.3 D3.JS (DATA-DRIVEN DOCUMENTS)

Esta tecnología fue explicada en el apartado D3.js (Data-Driven Documents) por lo que nos centraremos en ver sus ventajas e inconvenientes.

Ventajas:

- Se centra en los estándares web, dando toda la capacidad de los navegadores web, sin necesitar un framework que no sea de código abierto.
- Muy versátil para manipular datos desde el DOM.
- Manipulación de datos para generar visualizaciones complejas usando grandes cantidades de datos.

- Gran nivel de interacción y dinamismo en las visualizaciones

Inconvenientes:

- Complejidad a la hora de implementarlo o programarlo
- No funciona correctamente con navegadores antiguos (anteriores al IE8)

Esta opción de software de desarrollo es la utilizada en el proyecto por las ventajas comentadas anteriormente y por un aspecto clave que es el rendimiento. Es obvio que una aplicación que necesite de mucho tiempo para renderizar y mostrar los datos no es bien recibida por los usuarios, así que D3 ha sido pensado para que sea eficiente tanto en el cargado inicial como en las diversas transiciones, [5]. También se ha tenido en cuenta que se desarrolla en un navegador web, por lo que es necesario transmitir, cargar y procesar los datos en memoria, lo cual es costoso para otras aplicaciones. El hecho de ejecutarse en un navegador web permite la creación de visualizaciones sin necesidad de disponer de un complejo entorno de desarrollo.

Capítulo 6. PROCESAMIENTO DE LOS DATOS

Este capítulo se va a centrar en el análisis de los datos disponibles sobre consumos eléctricos en España. El principal objetivo es conocer las variables que vamos a necesitar para desarrollar la herramienta de visualización, ya sea eliminando variables que no son necesarias, introduciendo nuevas variables o transformándolas. Una vez las variables hayan sido procesadas, se procederá a generar los datasets necesarios para exportarlos al entorno de desarrollo web. El procesamiento de los datos se ha llevado a cabo utilizando Python como lenguaje de programación utilizando Jupyter Notebook.

Para comenzar, haremos una introducción a los datos disponibles seguido de una explicación de la arquitectura de desarrollo que ha hecho posible este procesamiento de los datos y, finalmente, el propio estudio y exploración de los datos.

6.1 CONSUMOS ELÉCTRICOS (DATOS)

Los datos totales disponibles sobre consumos eléctricos en España se encuentran en un fichero de 8GB. Estos datos están enmarcados en unas fechas fijas, desde el 12 de septiembre de 2013 hasta el 19 de octubre de 2019. A pesar de que los datos no son actuales, se utilizan como modelo para desarrollar la potente visualización y extraer conclusiones basadas en este marco temporal. Dicha herramienta de visualización podría ser aplicada a datos de cualquier fecha, por lo que cogemos estos datos disponibles para elaborarla. Aparte del fichero de 8 Gb, también se encuentra disponible otro fichero de menor tamaño (25 Mb) con el que se harán las pruebas iniciales para ahorrar tiempo de rendimiento y extraer las primeras conclusiones.

Cada fila o entrada de la base de datos que contiene el consumo eléctrico en España contiene las siguientes columnas o información:

- **DÍA**: fecha en la que se realizó el consumo siguiendo el formato Año/Mes/Día.
- **H1,H2,…,H25**: hora dentro del día correspondiente.
- **ACTIVA_H1, ACTIVA_H2,…, ACTIVA_H25**: energía activa consumida para cada hora medida en Wh (vatio hora). Se trata de la energía útil que los clientes absorben de la red y transforman en trabajo y/o calor en su casa.
- **REACTIVA_H1, REACTIVA_H2,…, REACTIVA_H25**: energía reactiva consumida para cada hora medida en VArh (voltiamperio reactivo hora). Se trata de un consumo suplementario que los clientes no pueden aprovechar. Actualmente la energía reactiva no se factura.
- **DE_MUNICIP**: municipio en el que se encuentra el cliente sobre el que se dan los datos de consumo horario. Referencia geográfica.
- **FECHA_ALTA_STRO**: fecha en la que dio de alta su suministro el cliente en cuestión con el siguiente formato Cuatrimestre/Año.
- **TARGET_TENENCIA_CUPS**: probabilidad de que en dicho municipio exista distribución de gas natural (no implica que el cliente tenga contratado servicio de gas natural).
- **IDENTIFICADOR**: código identificador que te permite individualizar los consumos por cliente.
- **CNAE**: (Clasificación Nacional Actividades Económicas) indica si el cliente es doméstico (T1) o no lo es (T2), que se considera industrial.
- **PRODUCTO**: tarifa/producto eléctrico que el cliente tiene contratado como servicio. Existen hasta 120 productos.
- **MERCADO**: indica si se trata de un cliente con tarifa regulada (M1) o un cliente con tarifa de Mercado Libre (M2).

Todas estas variables disponibles no serán útiles para el desarrollo del proyecto. Este proyecto se centra específicamente en las variables: DÍA, hora del día (H1,H2,…), consumo activo (ACTIVA_H1, ACTIVA_H2,…), municipio del cliente (DE_MUNICIP) y tipo de actividad económica (CNAE). Además, se crearán otras variables para cumplir los requisitos de la visualización, ya que al visualizar en mapa de España el consumo tiene que estar

agrupado por provincias, por lo que la variable provincia habrá que crearla. A continuación se muestran las cinco primeras filas del fichero completo.

	DIA	H1	ACTIVA_H1	REACTIVA_H1	H2	ACTIVA_H2	REACTIVA_H2	H3	ACTIVA_H3	REACTIVA_H3	...	ACTIVA_H25	REACTIVA_H25	DE_MUNICIP	I
0	20150222	1	1922	0	2	523	0	3	125	0	...	0	0	SANT BOI DE LLOBREGAT	
1	20150301	1	104	0	2	150	0	3	121	0	...	0	0	TORROX	
2	20150114	1	1892	0	2	1634	0	3	717	0	...	0	0	SANT BOI DE LLOBREGAT	
3	20150218	1	252	0	2	233	0	3	244	0	...	0	0	RUBI	
4	20150225	1	696	0	2	779	0	3	95	0	...	0	0	L'HOSPITALET DE LLOBREGAT	

Figura 15 - Ejemplo entradas del fichero parte 1

...	ACTIVA_H25	REACTIVA_H25	DE_MUNICIP	FECHA_ALTA_STRO	TARGET_TENENCIA_CUPS	IDENTIFICADOR	CNAE	PRODUCTO	MERCADO	ACTIVA_TOTAL
...	0	0	SANT BOI DE LLOBREGAT	Q21993	0.0000	18005	T1	P2	M1	28781
...	0	0	TORROX	Q21989	0.0000	60226	T1	P1	M1	4648
...	0	0	SANT BOI DE LLOBREGAT	Q31986	0.0000	77096	T1	P36	M2	13308
...	0	0	RUBI	Q22008	1.0000	49209	T1	P1	M1	7581
...	0	0	L'HOSPITALET DE LLOBREGAT	Q11977	0.5625	17750	T1	P1	M1	9354

Figura 16 - Ejemplo entradas del fichero parte 2

6.2 PLATAFORMA DE PROCESAMIENTO DE DATOS

Para el procesamiento de los datos se ha utilizado una determinada arquitectura la cuál hay que entender. Para comenzar, Apache Spark es un framework computacional, como ya se ha explicado en el capítulo 2, en el apartado Spark. Recapitulando, este framework computacional dispone de un motor de procesamiento de datos en memoria capaz de extraer, transformar, analizar y cargar datos,… Todo esto lo realiza sobre grandes volúmenes de datos mediante el uso de APIs para diversos lenguajes de programación, entre los que se encuentra Python, que será el utilizado en el proyecto, [3].

También ofrece la posibilidad de ejecutarse de diversas formas, ya sea en local, en la nube o en clústers. El clúster utilizado en el proyecto se ejecuta sobre Hadoop YARN.

Partiendo de unos datos iniciales, Spark crea Resilient Distributed Datasets(RDDs), que son la abstracción fundamental de esta aplicación, sobre los cuales se pueden realizar operaciones para obtener resultados con la opción de guardarlos, [13]. Este concepto surge debido a la ineficiencia de otras herramientas a la hora de ejecutar ciertos algoritmos iterativos y procesos de minería de datos. Mantener los datos en memoria hace que el rendimiento mejore y los procesos sean más eficientes.

Los dos tipos de operaciones o acciones que se pueden realizar son las siguientes, [14]:

- **Transformaciones**: al aplicarlas obtenemos un nuevo RDD diferente al anterior, obviamente basado en el anterior.
- **Acciones**: consiste en aplicar una operación a un RDD para obtener un valor como resultado.

Consecuentemente con las operaciones que se pueden realizar, los RDDs mediante la función *cache()* se almacenan los datos en memoria para que no sea necesario acceder a ellos en disco. Las figuras Figura 17 y Figura 18 muestran operación iterativa e interactiva sobre RDDs.

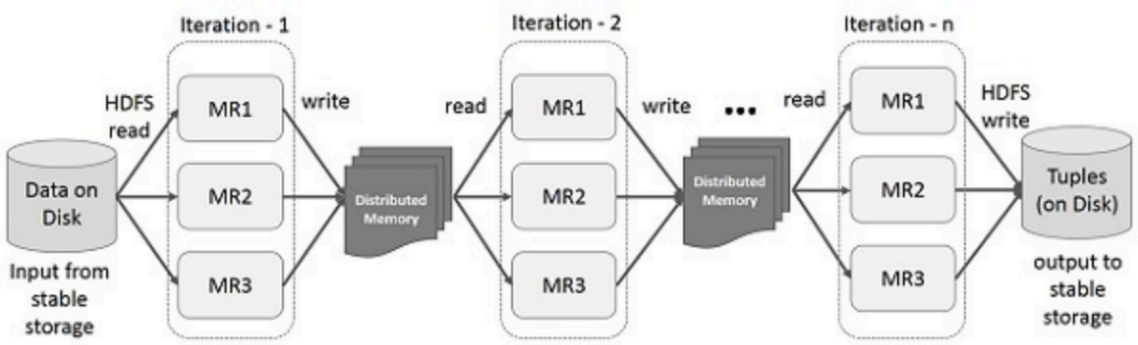

Figura 17 - Operación iterativa sobre Spark RDD, [15]

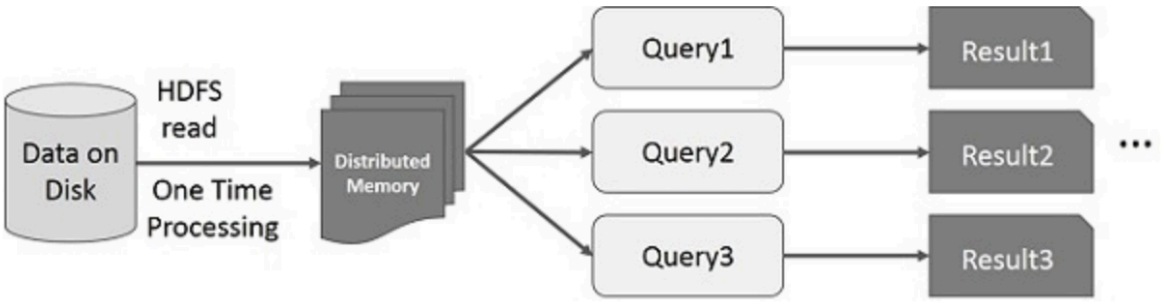

Figura 18 - Operación interactiva sobre Spark RDD, [16]

6.2.1 ARQUITECTURA

Spark emplea una arquitectura basada en la relación maestro/esclavo, [17]. Cada aplicación de Spark llama al módulo SparkContext, que representa la conexión al clúster de Spark y a través del cual se puede interactuar con las variables guardadas en el mismo, [17]. El contexto de Spark se comunica con el encargado de coordinar todos los procesos, el clúster Manager (master node). La función principal del clúster Manager es la distribución de los recursos entre todas las aplicaciones activas. Una vez conectado, Spark puede crear ejecutores, encargados de la computación y del almacenamiento de datos, en los nodos trabajadores (worker node) del clúster. A continuación, manda el código de la aplicación a los ejecutores. Finalmente, SparkContext manda tareas que serán procesadas por los ejecutores. La imagen de la Figura 19 refleja la arquitectura descrita.

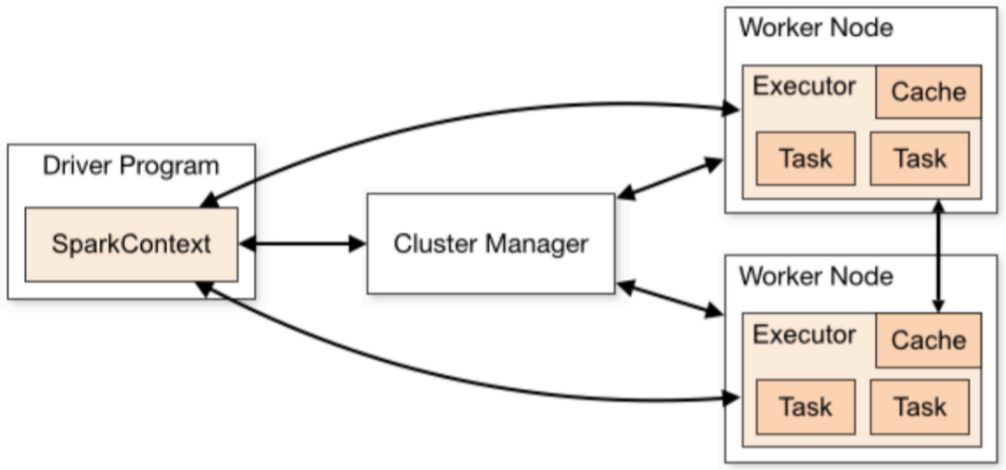

Figura 19 - Arquitectura Spark

Las ventajas más destacadas de la arquitectura Spark son las siguientes, [17]:

- Cada aplicación tiene sus propios ejecutores, lo que permite aislar una aplicación de otra.
- Spark soporta una gran cantidad de administradores de clústers.
- SparkContext debe ser accesible desde todos los nodos que forman el clúster, siendo preferible que formen parte de la misma LAN (Local Area Network).

El clúster empleado es este proyecto es el Clúster Big Data ICAI. Dicho clúster posee 1TB de memoria RAM, 96 TB de disco duro y 192 cores, todo ello repartido en ocho servidores Huawei. La arquitectura del clúster está compuesta por tres nodos master y cuatro nodos worker. Además, presenta un nodo edge. Este nodo funciona como una pasarela de salida y punto de conexión para los usuarios finales. En el esquema de la Figura 20, muestra la arquitectura seguida por el clúster empleado.

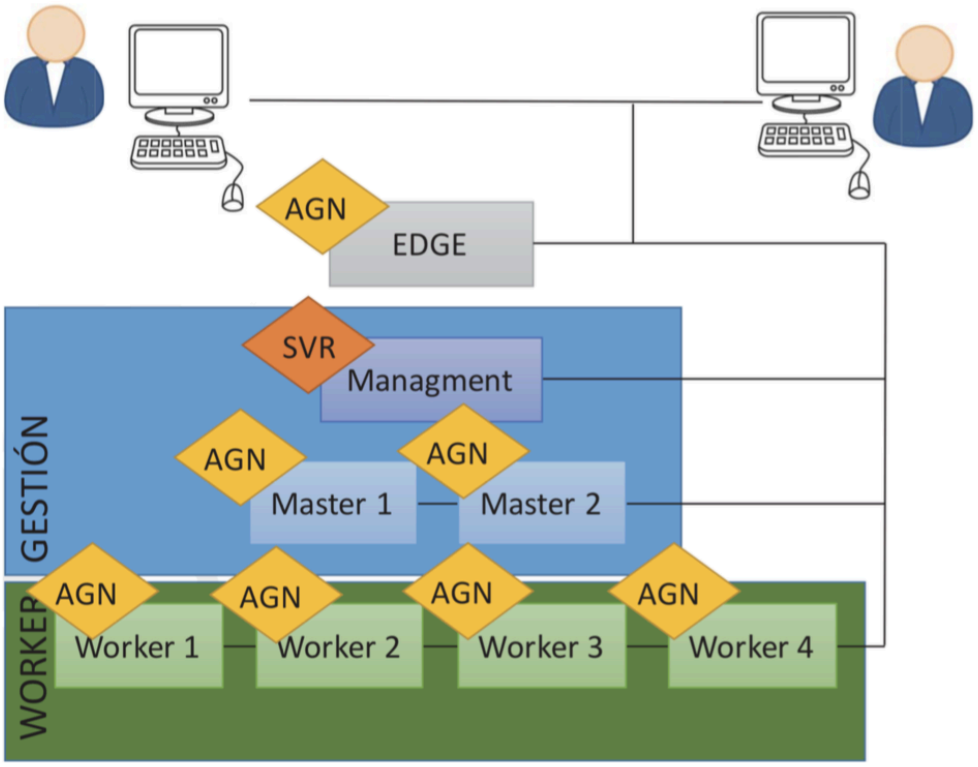

Figura 20 - Clúster Big Data ICAI

Con el fin de poder monitorizar las aplicaciones que se están ejecutando en el clúster, Hadoop ofrece una plataforma web dedicada a ello. A través de la misma, se puede acceder a información relacionada con la cantidad de recursos que se están empleado a nivel general y por cada aplicación.

6.3 ANÁLISIS Y PROCESAMIENTO DE LOS DATOS

Sabiendo el funcionamiento de la plataforma de desarrollo y el contenido original de los datos, nos centramos en el propio proceso de análisis de los datos con el fin de obtener las variables que necesitamos a partir de los datos disponibles, y así generar diversos ficheros o datasets que serán utilizados en el entorno web.

Esta sección se ha llevado a cabo utilizando Jupyter Notebook, creando notebooks y código dentro de ellos que han hecho posible obtener los resultados esperados. En el capítulo de descripción de tecnologías (2), en el apartado Jupyter Notebook se ha hecho una introducción a esta tecnología. Un ejemplo del funcionamiento de celdas, menú de herramientas y los notebooks ejecutándose ("*running"*) se muestra en las figuras Figura 21 y Figura 22. En la segunda figura se puede observar las aplicaciones o notebooks que están activos, es decir, que se están ejecutando, en este caso, sobre el clúster Big Data de la universidad. Cuando no se esté usando el notebook se debe hacer "shutdown" del mismo para no consumir recursos de manera innecesaria y permitir a otras personas su uso.

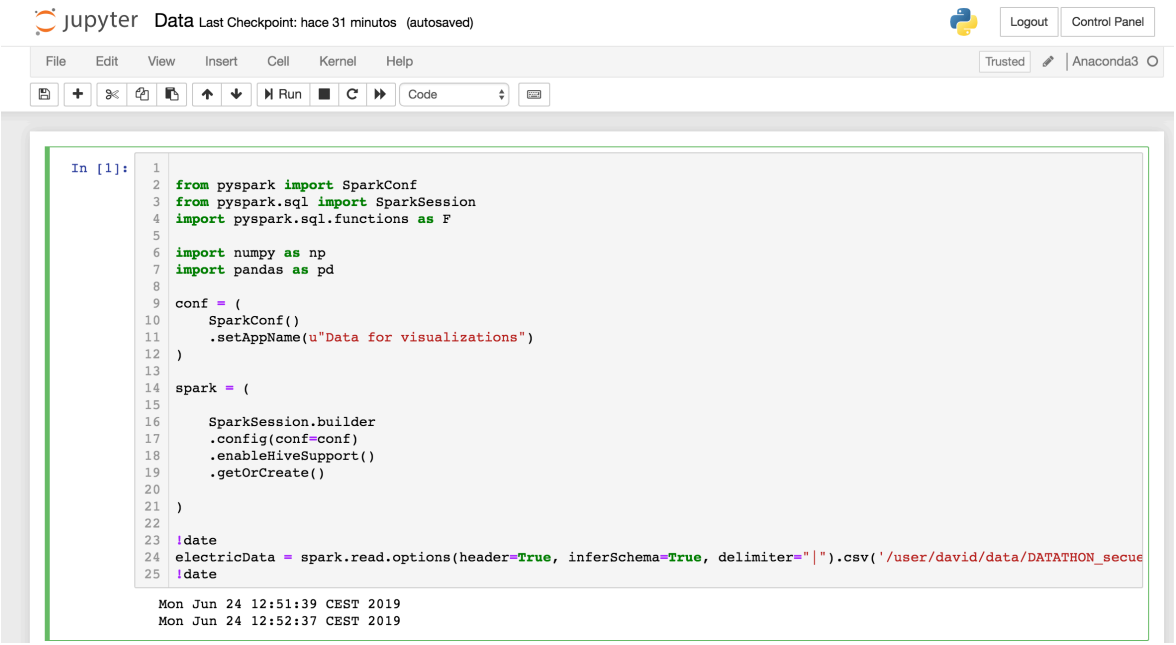

Figura 21 - Ejemplo de celda y menú de herramientas de Jupyter Notebook

Figura 22 - Ejemplo del menú running en Jupyter Notebook

El lenguaje de programación del código empleado en los notebooks es Python (versión 3.0). Se han utilizado diversas librerías Python, donde las más destacadas son:

- **PySpark**: se trata de incorporar las funcionalidades de Apache Spark al lenguaje de programación Python mediante una API. Se realizó una breve introducción a Apache

Spark en el capítulo 2, apartado Spark, y en el siguiente apartado se hará una descripción de la librería en mayor profundidad.

- **Numpy**: es una de las librerías más importantes en el uso de Machine Learning. Es utilizada porque proporciona potentes estructuras de datos (matrices y matrices multidimensionales) sobre las cuales se realizan operaciones eficientes, [18].
- **Pandas**: es una librería dedicada al análisis de datos que cuenta con las estructuras necesarias para limpiar los datos y convertirlos en aptos para procesarlos. Podría considerarse como complementaria a la librería Numpy, [19].

PySpark

Apache Spark es una de las herramientas más utilizadas en el ámbito del Big Data para trabajar con grandes volúmenes de datos. Aparte, Python es un lenguaje de programación ampliamente utilizado por analistas de datos y, en general, por expertos del sector IT (Information Technology). Con lo que tener la capacidad de utilizar ambas al mismo tiempo sería un gran avance en esta materia. Apache Spark creó la herramienta PySpark, que es básicamente una API para Python que permite integrar y trabajar con RDDs utilizando código Python. En el código desarrollado lo primero que se hace con los RDDs es transformarlos en DataFrames. Los DataFrames permiten tener los datos estructurados (al contrario que los RDDs que están desestructurados), y cada columna con el nombre de la variable correspondiente, siendo mucho más sencillo y accesible la consulta, modificación o transformación de los mismos, [20].

Para realizar operaciones con los RDDs tenemos dos tipos de funcionalidades: Acciones y Transformaciones. Las operaciones más utilizadas a lo largo del proyecto son las siguientes, [20]:

- **Acciones**
  - count(): nos devuelve el número total de elementos en un RDD.
  - collect(): transforma la colección distribuida en un array en memoria.

- **Transformaciones**
  - filter(function): selecciona un subconjunto de elementos del RDD que cumplen una determinada expresión booleana.
  - distinct(): devuelve los elementos del RDD que son distintos.

6.3.1 DESCRIPCIÓN DEL CÓDIGO

Para el procesamiento de los datos se ha creado un notebook de Jupyter llamado "Data.ipynb". A lo largo de esta sección se irá describiendo el código utilizado para finalizar con la creación de los datasets. También se harán ciertos apuntes sobre el rendimiento de las funciones utilizadas.

En primer lugar, antes de ponernos a escribir código necesitamos importar las librerías correspondientes. De la librería PySpark se utilizarán las funciones relacionadas con SQL y con la configuración de Spark. Aparte, también utilizaremos las librerías *numpy* y *pandas*.

In [1]:
```
1
2  from pyspark import SparkConf
3  from pyspark.sql import SparkSession
4  import pyspark.sql.functions as F
5
6  import numpy as np
7  import pandas as pd
8
```

Figura 23 - Librerías utilizadas en el notebook

El siguiente paso es establecer la configuración de la aplicación Spark. Se utiliza para definir los parámetros Spark como parejas clave-valor, [20]. En el proyecto el único parámetro definido es el nombre de la aplicación, "Data for visualizations". Junto a la configuración, hay que iniciar una *SparkSession* que es el objeto principal a partir de la cual se desglosa

toda la funcionalidad de Apache Spark. Se distingue de *SparkContext* en que sirve para trabajar con SQL, los DataFrame y DataSet, mientras que el anterior sirve para RDDs. Para la sesión Spark también hay que fijar unos parámetros. En este proyecto se han utilizado los parámetros de configuración mencionados, se ha habilitado el Apache Hive para incorporar un interfaz SQL en la base de datos y el parámetro *getOrCreate()* para crear una nueva sesión solo en caso de que no haya una ya creada. Esta configuración se muestra en la Figura 24.

```
8
9  conf = (
10     SparkConf()
11     .setAppName(u"Data for visualizations")
12 )
13
14 spark = (
15
16     SparkSession.builder
17     .config(conf=conf)
18     .enableHiveSupport()
19     .getOrCreate()
20
21 )
22
```

Figura 24 - Configuración de la sesión Spark

Tras configurar la sesión Spark, procedemos a leer el fichero de los datos de consumos eléctricos sobre los que vamos a trabajar. La Figura 25 muestra dicha lectura.

```
22
23 !date
24 electricData = spark.read.options(header=True, inferSchema=True, delimiter="|").csv('/user/david/data/DATATHON_secue
25 !date
```

```
Mon Jun 24 12:51:39 CEST 2019
Mon Jun 24 12:52:37 CEST 2019
```

Figura 25 - Lectura del fichero de datos

Al estar la base de datos en formato CSV, se realiza la lectura siguiendo esos parámetros y se aplica la función *cache()* para almacenar los datos en memoria en vez de en disco. En un primer intento se leyó el fichero de menor tamaño (25 Mb) y el tiempo de procesamiento era de unos pocos segundos. En cambio, el fichero de 8GB, como se puede observar en la figura anterior, tarda en leerse alrededor de un minuto.

Para el estudio de los datos solo tenemos en cuenta el consumo activo y no el reactivo por lo que las comprobaciones de errores en los datos se centrarán en esta variable. Para ello hemos de comprobar que todos los valores de consumo son mayores de cero. Con la función *filter(function)* obtenemos un nuevo RDD que no contiene consumos negativos, como se observa en la Figura 26.

```
In [2]: 1 electricData = electricData.filter((F.col('ACTIVA_H1') > 0) & (F.col('ACTIVA_H2') > 0) & (F.col('ACTIVA_H3') > 0) &
        2
```

Figura 26 - Filtrado para eliminar consumos negativos

Una vez filtrados los posibles errores en el consumo por hora, necesitamos crear una nueva variable que englobe el consumo diario, es decir, una suma del consumo de cada hora para cada entrada del RDD. Esta variable es creada para poder visualizarla en la web, así se podrá visualizar tanto el consumo diario como el consumo a cada hora del día. La creación de esta nueva variable ha sido posible gracias a la función *withColumn(newColName, colExpr)*, que permite crear una nueva columna aplicando una transformación sobre el RDD. Después mostramos las cinco primeras entradas del RDD pero en formato DataFrame de la librería *pandas*. Este proceso se muestra en la Figura 27, donde se ve como se ha añadido la columna de consumo diario (ACTIVA_TOTAL).

```
= electricData.withColumn("ACTIVA_TOTAL", sum(electricData[x] for x in electricData.columns if x.startswith("ACTIVA_
.limit(5).toPandas()
```

Out[2]:

...	ACTIVA_H25	REACTIVA_H25	DE_MUNICIP	FECHA_ALTA_STRO	TARGET_TENENCIA_CUPS	IDENTIFICADOR	CNAE	PRODUCTO	MERCADO	ACTIVA_TOTAL
...	0	0	SANT BOI DE LLOBREGAT	Q21993	0.0000	18005	T1	P2	M1	28781
...	0	0	TORROX	Q21989	0.0000	60226	T1	P1	M1	4648
...	0	0	SANT BOI DE LLOBREGAT	Q31986	0.0000	77096	T1	P36	M2	13308
...	0	0	RUBI	Q22008	1.0000	49209	T1	P1	M1	7581
...	0	0	L'HOSPITALET DE LLOBREGAT	Q11977	0.5625	17750	T1	P1	M1	9354

Figura 27 - Consumo diario (ACTIVA_TOTAL)

En la anterior figura todavía se muestran variables que no son útiles para la herramienta de visualización. Por ello, el siguiente paso es eliminar las variables que no nos interesan. En el apartado Consumos Eléctricos (Datos) se explican las variables en las que se va a centrar el proyecto. Para proceder con la transformación, utilizamos la función *drop(colName)* y nos quedamos con un nuevo RDD con las variables útiles, como se observa en la Figura 28.

```
In [3]: iva_total = tabla_con_activa_total.drop("FECHA_ALTA_STRO", "TARGET_TENENCIA_CUPS", "IDENTIFICADOR", "REACTIVA_H1", "
```

Figura 28 - Eliminación de variables sin utilidad

A continuación, ya que en la web se va a desplegar un mapa de España con las provincias delimitadas, necesitamos agrupar los municipios (variable DE_MUNICIP) en provincias para más tarde englobar los consumos de dichos municipios en sus respectivas provincias. Lo primero es listar todos los municipios existentes mediante la función *distinct()*, después creamos un *array* para cada provincia donde incluimos el nombre de cada municipio perteneciente a la provincia. El proceso se muestra en la Figura 29.

```
In [7]:
pd.options.display.max_rows = 450
tabla_def.select(F.col("DE_MUNICIP")).distinct().toPandas()
```

```
In [8]:
lacoruña = ["A CORUÑA", "FISTERRA", "MONFERO", "CARBALLO", "OLEIROS", "ARTEIXO", "SADA", "TARRIO", "CULLEREDO", "CA
cantabria = ["SANTANDER", "CAMPOO DE ENMEDIO", "CILLORIGO DE LIEBANA", "TAMA", "BAREYO", "CASTAÑEDA", "POLANCO", "C
huelva = ["VALVERDE DEL CAMINO", "ALJARAQUE", "HUELVA", "GIBRALEON", "PUEBLA DE GUZMAN", "LEPE", "PUNTA UMBRIA", "L
guipuzcoa = ["DONOSTIA-SAN SEBASTIAN", "TOLOSA", "ANOETA", "IBARRA", "HERNIALDE", "LASARTE-ORIA"]
salamanca = ["VILLARES DE LA REINA", "TARDAGUILA", "SALAMANCA", "SAN CRISTOBAL DE LA CUESTA", "LA TALA", "GOMECELLO
castellon = ["BURRIANA", "CASTELLON", "LA LLOSA", "PEÑISCOLA", "CASTELLON DE LA PLANA", "VILLARREAL/VILA-REAL", "AL
baleares = ["INCA", "SA POBLA", "ALCUDIA", "ANDRATX", "POLLENÇA", "SELVA", "SANTA MARGALIDA", "MAÓ", "SANT LLUIS",
santa_cruz_tenerife = ["LA GUANCHA", "EL PINAR", "ARONA", "PUERTO DE LA CRUZ", "S/C DE TENERIFE", "EL PINAR DE EL H
ciudad_real = ["SOCUELLAMOS", "DAIMIEL", "CAMPO DE CRIPTANA", "TORRALBA DE CALATRAVA", "MIGUELTURRA", "CIUDAD REAL",
alicante = ["BENIDORM", "ELS POBLETS", "DENIA", "JAVEA/XABIA", "JAVEA", "EL VERGER", "TORREVIEJA", "VERGER (EL)", "
alava = ["VITORIA-GASTEIZ"]
gerona = ["L´ESCALA", "FIGUERES", "ROSES", "SALT", "GIRONA", "L?ESCALA", "SANTA COLOMA DE FARNERS", "LLORET DE MAR",
madrid = ["DAGANZO DE ARRIBA", "RIVAS-VACIAMADRID", "VILLALBILLA", "CAMARMA DE ESTERUELAS", "MEJORADA DEL CAMPO", "
guadalajara = ["GALAPAGOS", "TORREJON DEL REY", "HERRERIAS", "LA HUERCE", "LA TOBA", "UCEDA", "GUADALAJARA", "SIGÜE
toledo = ["SESEÑA", "MIGUEL ESTEBAN", "ILLESCAS", "UGENA", "YELES", "YUNCOS", "ALAMEDA DE LA SAGRA", "AÑOVER DE TAJ
zaragoza = ["SADABA", "CUARTE DE HUERVA", "ALMONACID DE LA SIERRA", "TAUSTE", "VILLANUEVA DE GALLEGO", "VILLANUEVA
barcelona = ["MONTGAT", "BADIA DEL VALLES", "SITGES", "SANTA COLOMA DE GRAMENET", "IGUALADA", "L'HOSPITALET DE LLOB
las_palmas = ["AGUIMES", "MOGAN", "SAN BARTOLOME DE TIRAJANA", "PAJARA", "TELDE", "ANTIGUA", "SANTA LUCIA DE TIRAJA
burgos = ["MELGAR DE FERNAMENTAL", "SOTRESGUDO", "BURGOS", "BARRIO DE BRICIA", "VILLADIEGO", "HORNILLOS DEL CAMINO",
teruel = ["VALVERDE", "CALACEITE", "CRETAS", "CUERVO (EL)", "ALCAÑIZ", "TERUEL", "CALANDA", "SARRION", "MAZALEON"]
granada = ["CHADIX", "GRANADA", "GRANADA", "MOCLIN", "RUBION", "LOJA", "PURULLENA", "MORRIL", "PORTUGOS", "ALBOLOTE
```

Figura 29 - Creación de arrays de provincias con los municipios

Con estos *arrays* de provincias ya podemos hacernos una idea de cómo están distribuidos los datos. Por ejemplo, Cantabria, Barcelona, Madrid y Palencia son las provincias con mayor número de municipios registrados, mientras que otras provincias solo tienen un municipio registrado, como es el caso de Ávila o Álava, o ninguno como Cuenca. Hay que tener esto en cuenta ya que a la hora de visualizar sus consumos, tanto total como promedio, el número de municipios y los clientes de dichos municipios pueden causar "outliers" o resultados no esperados.

Para introducir la variable provincia en cada una de las filas del RDD se ha creado una función (*findProvincia(municipio)*) que teniendo el municipio como argumento devuelve la provincia en la que se encuentra. Con esta función y *withColumn(newColName, colExpr)* creamos la variable, como se presenta en las figuras Figura 30 y Figura 31. Para comprobar que hemos agrupado todos los municipios de los datos, si alguno no perteneciera a ningún *array* se introduciría la provincia "MISS". Por lo tanto, contamos el número de provincias "MISS" mediante la función *count()* y obtenemos un cero, así que todos los municipios han sido agrupados correctamente.

```
In [9]:
def findProvincia(municipio):
    if municipio in lacoruña:
        return "LA CORUNA"
    elif municipio in huelva:
        return "HUELVA"
    elif municipio in cantabria:
        return "CANTABRIA"
    elif municipio in guipuzcoa:
        return "GUIPUZCOA"
    elif municipio in salamanca:
        return "SALAMANCA"
    elif municipio in castellon:
        return "CASTELLON"
    elif municipio in baleares:
        return "ISLAS BALEARES"
    elif municipio in santa_cruz_tenerife:
        return "SANTA CRUZ"
    elif municipio in ciudad_real:
        return "CIUDAD REAL"
    elif municipio in alicante:
        return "ALICANTE"
    elif municipio in alava:
        return "ALAVA"
    elif municipio in gerona:
        return "GERONA"
    elif municipio in madrid:
        return "MADRID"
    elif municipio in guadalajara:
        return "GUADALAJARA"
    elif municipio in toledo:
        return "TOLEDO"
    elif municipio in zaragoza:
        return "ZARAGOZA"
    elif municipio in barcelona:
        return "BARCELONA"
```

Figura 30 - Función findProvince()

```
In [14]:
findProvinciaUDF = F.udf(lambda z: findProvincia(z))
tabla_def1 = tabla_def.withColumn("PROVINCIA", findProvinciaUDF(F.col('DE_MUNICIP')))
print(tabla_def1.filter(F.col("PROVINCIA") == "MISS").count())
tabla_def1.select(F.col("PROVINCIA")).distinct().toPandas()
```

Figura 31 - Creación de la variable Provincia

Con la variable provincia, ya podemos agrupar los consumos en función del día, la provincia y el tipo de actividad económica (residencial o industrial). El objetivo es mostrar el consumo total, residencial e industrial, tanto como una suma del consumo de cada cliente en una provincia cualquier día, como un promedio del consumo de los mismos. Por lo tanto, con la función groupBy() agrupamos los datos en función del día y la provincia (consumo total) y en función del día, provincia y actividad económica (consumo residencial e industrial). Esa

agrupación la realizamos como una suma directa de los consumos (función *sum()*) y como una media de los consumos (función *avg()*). La Figura 32 muestra dichas agrupaciones.

```
tabla_def2 = tabla_def1.groupBy("PROVINCIA", "DIA", "CNAE").sum("ACTIVA_H1", "ACTIVA_H2", "ACTIVA_H3", "ACTIVA_H4",
tabla_def4 = tabla_def1.groupBy("PROVINCIA", "DIA").sum("ACTIVA_H1", "ACTIVA_H2", "ACTIVA_H3", "ACTIVA_H4", "ACTIVA

tabla_def5 = tabla_def1.groupBy("PROVINCIA", "DIA").avg("ACTIVA_H1", "ACTIVA_H2", "ACTIVA_H3", "ACTIVA_H4", "ACTIVA
tabla_def6 = tabla_def1.groupBy("PROVINCIA", "DIA", "CNAE").avg("ACTIVA_H1", "ACTIVA_H2", "ACTIVA_H3", "ACTIVA_H4",
```

Figura 32 - Nuevos RDDs agrupados

Aunque esta sección se ha llevado a cabo principalmente antes del desarrollo de la herramienta de visualización, una vez comenzado su diseño se ha tenido que recurrir de nuevo a esta tarea para introducir por cada entrada de los nuevos RDDs las coordenadas para colocar cada dato en la provincia correspondiente del mapa de España. Por ello se han creado dos funciones: *addCentroidx(provincia)* y *addCentroidy(provincia)*. Estas funciones introducen tanto la coordenada X como la coordenada Y, en píxeles, del centro de cada provincia en el mapa de España de la web. El proceso es similar al de añadir la variable provincia, pero en este caso añadiremos dos: Cx y Cy, correspondiendo a cada coordenada respectivamente. Las figuras Figura 33, Figura 34 y Figura 35 representan este proceso.

In [10]:

```
def addCentroidx(provincia):
    if provincia == "LA CORUNA":
        return 365.320615635561
    elif provincia == "TOLEDO":
        return 510.56751032840947
    elif provincia == "TERUEL":
        return 626.6089747273927
    elif provincia == "TARRAGONA":
        return 681.6138189035776
    elif provincia == "SORIA":
        return 562.6930897812628
    elif provincia == "SEVILLA":
        return 454.1855675659901
    elif provincia == "SEGOVIA":
        return 512.7334889993974
    elif provincia == "CANTABRIA":
        return 512.0287741713955
    elif provincia == "SALAMANCA":
        return 442.70323399643365
    elif provincia == "PONTEVEDRA":
        return 363.5930605166956
    elif provincia == "PALENCIA":
        return 495.6632296832606
    elif provincia == "ASTURIAS":
        return 447.2308270444209
    elif provincia == "ORENSE":
        return 392.30604700343474
    elif provincia == "NAVARRA":
        return 592.1580135495063
    elif provincia == "MURCIA":
        return 611.1308243735267
    elif provincia == "MALAGA":
        return 490.4420032242093
```

Figura 33 - Función addCentroidx(provincia)

In [11]:

```
def addCentroidy(provincia):
    if provincia == "LA CORUNA":
        return 37.428013071083434
    elif provincia == "TOLEDO":
        return 193.77511782188995
    elif provincia == "TERUEL":
        return 148.39142466096826
    elif provincia == "TARRAGONA":
        return 123.6644693133128
    elif provincia == "SORIA":
        return 107.83380732009813
    elif provincia == "SEVILLA":
        return 305.10622711106083
    elif provincia == "SEGOVIA":
        return 129.9885037006659
    elif provincia == "CANTABRIA":
        return 37.34084185873941
    elif provincia == "SALAMANCA":
        return 146.8148282718496
    elif provincia == "PONTEVEDRA":
        return 68.77927824646693
    elif provincia == "PALENCIA":
        return 75.31584974360142
    elif provincia == "ASTURIAS":
        return 32.9771371339625
    elif provincia == "ORENSE":
        return 81.1814415035977
    elif provincia == "NAVARRA":
        return 58.509178333230636
    elif provincia == "MURCIA":
        return 274.2050062426311
    elif provincia == "MALAGA":
        return 334.8837097736757
```

Figura 34 - Función addCentroidy(provincia)

```
addCentroidxUDF = F.udf(lambda z: addCentroidx(z))
tabla_def2 = tabla_def2.withColumn("Cx", addCentroidxUDF(F.col('PROVINCIA')))
print(tabla_def2.filter(F.col("Cx")==0).count())

addCentroidyUDF = F.udf(lambda z: addCentroidy(z))
tabla_def2_with_centroids = tabla_def2.withColumn("Cy", addCentroidyUDF(F.col('PROVINCIA')))
print(tabla_def2_with_centroids.filter(F.col("Cx")==0).count())
tabla_def2_with_centroids.filter(F.col("DIA") == "20141124").toPandas()
```

Figura 35 - Añadiendo las variables Cx y Cy

El consumo eléctrico de los datos disponibles se encuentra en Wh (vatios hora) y, normalmente, los contratos de electricidad se realizan kWh (kilovatios hora), así que hay que pasar los datos a kWh. Con las agrupaciones anteriores las nuevas columnas del RDD se crean como "sum(ACTIVA_H1)" o "avg(ACTIVA_H1)" que no coinciden con los datos iniciales. Para solucionar estos dos problemas, primero se ha añadido una nueva columna por cada consumo, llamada ACTIVA_H1, ACTIVA_H2,… , con el consumo en kWh (dividiendo el consumo actual entre mil), y se han eliminado las columnas anteriores con el prefijo sum o avg. La Figura 36 presenta estos cambios y la Figura 37 muestra un ejemplo de cómo queda el RDD final para el 24 de noviembre de 2014.

```
tabla_def4_with_centroids = tabla_def4_with_centroids.withColumn("ACTIVA_H2", F.col("avg(ACTIVA_H2)")/1000)
tabla_def4_with_centroids = tabla_def4_with_centroids.drop("avg(ACTIVA_H2)")

tabla_def2_with_centroids = tabla_def2_with_centroids.withColumn("ACTIVA_H2", F.col("sum(ACTIVA_H2)")/1000)
tabla_def2_with_centroids = tabla_def2_with_centroids.drop("sum(ACTIVA_H2)")
```

Figura 36 - Consumos a kWh y nuevos nombres de columna

Out[17]:

	PROVINCIA	DIA	CNAE	Cx	Cy	ACTIVA_H1	ACTIVA_H2	ACTIVA_H3	ACTIVA_H4	ACTIVA_H5	...	ACTIVA_H16
0	LAS PALMAS	20141124	T2	310.3452200871819	464.21907629873556	279.865	252.888	246.162	240.674	239.136	...	257.206
1	GUADALAJARA	20141124	T2	563.1020489590896	144.97992675357645	14.842	12.180	9.971	9.740	9.372	...	19.513
2	TOLEDO	20141124	T2	510.56751032840947	193.77511782188995	4.440	2.573	2.150	2.052	1.676	...	3.203
3	ALICANTE	20141124	T2	643.3646317700382	249.25080290432558	99.582	91.063	81.381	76.025	74.560	...	159.242
4	ZARAGOZA	20141124	T1	614.9281160401472	105.00075540530281	425.549	310.698	271.508	255.992	257.886	...	506.070
5	ALAVA	20141124	T2	556.0165220641754	52.57561563565324	2.861	2.592	2.355	2.314	2.319	...	7.148
6	ALBACETE	20141124	T2	590.519126075333	236.4657881551595	89.409	82.057	80.509	75.495	75.270	...	138.525
7	ASTURIAS	20141124	T1	447.2308270444209	32.9771371339625	13.923	9.552	9.491	7.588	6.011	...	8.945
8	PALENCIA	20141124	T2	495.6632296832606	75.31584974360142	40.621	36.775	30.058	29.962	30.171	...	42.420
9	ISLAS BALEARES	20141124	T1	763.3343899549734	184.13373367666114	563.467	438.336	384.853	358.802	354.598	...	732.118

Figura 37 - Ejemplo de los datos finales

Teniendo los datos perfectamente modelados y procesados, el último paso es generar los datasets en formato CSV mediante la función *to_csv()* del DataFrame de la librería pandas. La Figura 38 muestra dicha función. En total se han obtenido cuatro datasets para exportarlos al entorno de desarrollo web.

```
4
5 tabla_def2_with_centroids.toPandas().to_csv('/home/btorreiro/Javier/data_total_provincias_horas_dia.csv', index=Fal
6
```

Figura 38 - Método para obtener los datasets en formato CSV

Los cuatro datasets obtenidos son:

- "data_total_provincias_horas_dia.csv", consumo total sumado agrupado por provincia, día y actividad económica.
- "consumo_total_provincias_horas_dia.csv", consumo total sumado agrupado por provincia y día.
- "consumo_promedio_provincias_conCNAE", consumo promedio agrupado por provincia, día y actividad económica.
- "consumo_promedio_provincias_sinCNAE", consumo promedio agrupado por provincia y día.

Para continuar con el desarrollo del proyecto, el siguiente paso ya concierne al sistema de visualización desarrollado que utilizará estos ficheros de datos, que será el contenido del próximo capítulo.

Capítulo 7. SISTEMA/MODELO DESARROLLADO

En este capítulo se va a desarrollar cómo se ha implementado la herramienta de visualización. El objetivo del proyecto es crear una visualización geotemporal de los consumos eléctricos en España que incluya dinamismo, interactividad con el usuario y nos permita extraer conclusiones rápidas sobre esos datos.

La tecnología escogida para desarrollar el proyecto ha sido la librería de Javascript D3.js (versión 5.0). En el capítulo 2, apartado D3.js (Data-Driven Documents) se realiza una introducción a dicha librería. La combinación de HTML, CSS y Javascript ha hecho posible la creación de la potente herramienta de visualización. Aparte de la librería D3.js, se han utilizado otras librerías que se describen a continuación:

- **jQuery**: librería Javascript que facilita la manipulación de elementos HTML, gestión de los eventos de la web, animaciones y Ajax con una simple API que funciona en multitud de navegadores web, [21].
- **Bootstrap**: librería Javascript que consiste en un framework que simplifica el proceso de creación de diseños web, estructurando la página web, [22].
- **D3-composite-projections**: librería Javascript que incluye un conjunto de proyecciones para mostrar países con todos los territorios juntos. Es decir, en nuestro caso se ha utilizado para incorporar las islas Canarias al mapa, [23].
- **TopoJSON**: librería Javascript que se utiliza en conjunto con D3.js. Sirve para crear proyecciones mediante la transformación de objetos TopoJSON en GeoJSON. En el proyecto ha sido utilizado para proyectar el mapa de España delimitado por provincias, [24].
- **FontAwesome**: librería Javascript para implementar iconos y logos en una página web, [25].

```
<script src="js/jquery.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/jquery-ui.min.js"></script>
<script src="js/d3.min.js"></script>
<script src="js/d3-composite-projections.min.js"></script>
<script src="js/d3.min.js"></script>
<script src="js/d3-tip.js"></script>
<script src="js/main.js"></script>
<script src="https://d3js.org/topojson.v1.min.js"></script>
<script src="https://kit.fontawesome.com/1d240e8faf.js"></script>
```

Figura 39 - Librerías utilizadas

A continuación nos centramos en aspectos internos de la librería D3.js para tener una idea general de la potencia de esta tecnología.

D3.js (Data-Driven Documents)

En cuanto al funcionamiento interno, D3.js tiene características que hacen posible la creación de una herramienta potente, dinámica e interactiva. Los aspectos más destacados son, [5]:

- **Operadores**: se pueden aplicar operaciones como establecer atributos, propiedades o contenido de elementos seleccionados.
- **Selección**: ofrece la posibilidad de seleccionar uno o varios elementos mediante nombre, identificador, clase o atributos entre otros.
- **Eventos**: la librería proporciona métodos y herramientas para supervisar la interacción del usuario mediante teclado o ratón.
- **Datos**: con la función *data()* podemos relacionar los datos con elementos que forman la visualización (círculos, barras…). Los datos pueden ser filtrados y modificados según las necesidades de cada sección. La manera que tiene D3.js de vincular los datos a elementos de la visualización hace posible el dinamismo de los mismos. Mediante la función *enter()* los elementos son introducidos para luego ser actualizados mediante un *update()* o eliminados con *exit()*.

- Además de estas propiedades, D3 incorpora: configuraciones a modo de plantilla (layouts) para generar visualizaciones a partir de ellas, uso de transiciones animadas y el eficiente rendimiento para visualizar los datos.

7.1 DESCRIPCIÓN DEL CÓDIGO

La implementación del código se ha llevado a cabo mediante el editor de texto Sublime Text (versión 2). Para ejecutar el código se ha utilizado un servidor http proporcionado por Python en el propio ordenador (localhost). A lo largo de la sección se explica cuál ha sido el camino seguido para llevar a cabo los objetivos del proyecto.

Para comenzar hay que cargar los distintos datasets, que son los cuatro extraidos del notebook de Jupyter más uno llamado "spain.json" que contiene las coordenadas para dibujar el mapa de España y los identificadores de cada provincia. Para ello la versión 5 de la librería D3 introduce las "promises" que permite primero cargar todos los datasets para luego, una vez cargados con éxito, realizar cualquier operación sobre ellos con la función *then()*. Esto supone un gran avance respecto a las anteriores funciones llamadas "callbacks" que carecen de, por ejemplo, del encadenamiento explicado. En la Figura 40 se observa la carga de los datos. Los datos cargados se transforman del formato original a arrays de objetos Javascript para permitir su posterior uso.

```
var promises = [
    d3.json("data/spain.json"),
    d3.csv("data/data_total_provincias_horas_dia.csv"),
    d3.csv("data/consumo_total_provincias_horas_dia.csv"),
    d3.csv("data/consumo_promedio_provincias_conCNAE.csv"),
    d3.csv("data/consumo_promedio_provincias_sinCNAE.csv")
]

Promise.all(promises).then(function(allData){
    spain = allData[0];
    consumos = allData[1];
    consumo_total_por_hora = allData[2],
    consumos_promedio = allData[3],
    consumos_promedio_total = allData[4];
```

Figura 40 - Carga de datos con "promises"

Al cargar los datos, todas las variables se encuentran como variables "string" o de texto, por lo que las variables que deberían ser un valor numérico son transformadas como enteros. La variable DÍA es transformada en objeto de fecha Javascript mediante la función *d3.timeParse()*, que al introducir un formato de fecha (en este caso Año/Mes/Día) la transforma en fecha Javascript. La Figura 41 muestra esta manipulación de los datos cargados.

```
consumos.forEach(function(d){
    d.ACTIVA_H1 = +d.ACTIVA_H1;d.ACTIVA_H2 = +d.ACTIVA_H2;d.ACTIVA_H3 = +d.ACTIVA_H3;d.ACTIVA_H4 =
    d.ACTIVA_H10 = +d.ACTIVA_H10;d.ACTIVA_H11 = +d.ACTIVA_H11;d.ACTIVA_H12 = +d.ACTIVA_H12;d.ACTIV
    d.ACTIVA_H20 = +d.ACTIVA_H20;d.ACTIVA_H21 = +d.ACTIVA_H21;d.ACTIVA_H22 = +d.ACTIVA_H22;d.ACTIV
    d.ACTIVA_TOTAL = +d.ACTIVA_TOTAL;
    d.Cx = +d.Cx;
    d.Cy = +d.Cy;
    d.DIA = parseTime(d.DIA);
})
```

Figura 41 - Cambio en el formato de las variables

Una vez los datos están preparados, el siguiente paso consiste en crear la parte de geolocalización con el mapa de España. El funcionamiento de D3 para crear visualizaciones consiste en crear una plantilla, que es el elemento HTML SVG, sobre la cual se introducen grupos que introducen la visualización. Para el mapa se ha utilizado la plantilla que muestra la Figura 42.

```
var svg = d3.select("#chart-area")
.append("svg")
.attr("width", 1040)
.attr("height", 550)
var g = svg.append("g")
    .attr("transform", "translate(" + margin.left +
        ", " + 30 + ")");
```

Figura 42 - SVG y grupo para el mapa de España

Antes de proceder a dibujar el mapa, tenemos que definir dos parámetros. Uno sería el tipo de proyección que tendrá el mapa, que se ha escogido el tipo *d3.geoConicConformalSpain()* de la librería D3-composite-projections. El otro sería el camino a seguir para dibujar el mapa, con la función *d3.geoPath()* generamos el camino geográfico en función de la proyección anteriormente definida. La Figura 43 muestra estos parámetros.

```
var projection = d3.geoConicConformalSpain();

var path = d3.geoPath()
    .projection(projection);
```

Figura 43 - Proyección y generador geográfico

Ahora ya estamos en condiciones de pasar al dibujo de las provincias. Para ello utilizamos la función *feature()* de la librería TopoJSON para escoger las provincias del fichero spain.json. Después generamos cada una de las provincias (llamadas regiones en el código) mediante la variable path (ver figura 44) e introduciendo en dicha variable cada uno de los datos referidos a cada provincia. A su vez definimos unos atributos como el color de la provincia, y el color y tamaño de las fronteras de cada provincia. Además, tenemos que crear la separación entre las islas Canarias y la península Ibérica. Esto se lleva a cabo con la función *getCompositionBorders()* de la variable projection (ver figura 44). La Figura 44 contiene este proceso y la Figura 45 muestra el mapa de España final.

```
es = topojson.feature(spain, spain.objects.provinces);

g.selectAll(".region")
    .data(es.features)
    .enter()
    .append("path")
    .attr("d", path)
    .attr("class","region")
    .style("fill", "#aca")
    .style("stroke", "#000")
    .style("stroke-width", "0.5px")


g.append("path")
    .style("fill","none")
    .style("stroke","#000")
    .attr("d", projection.getCompositionBorders());
```

Figura 44 - Código para dibujar el mapa

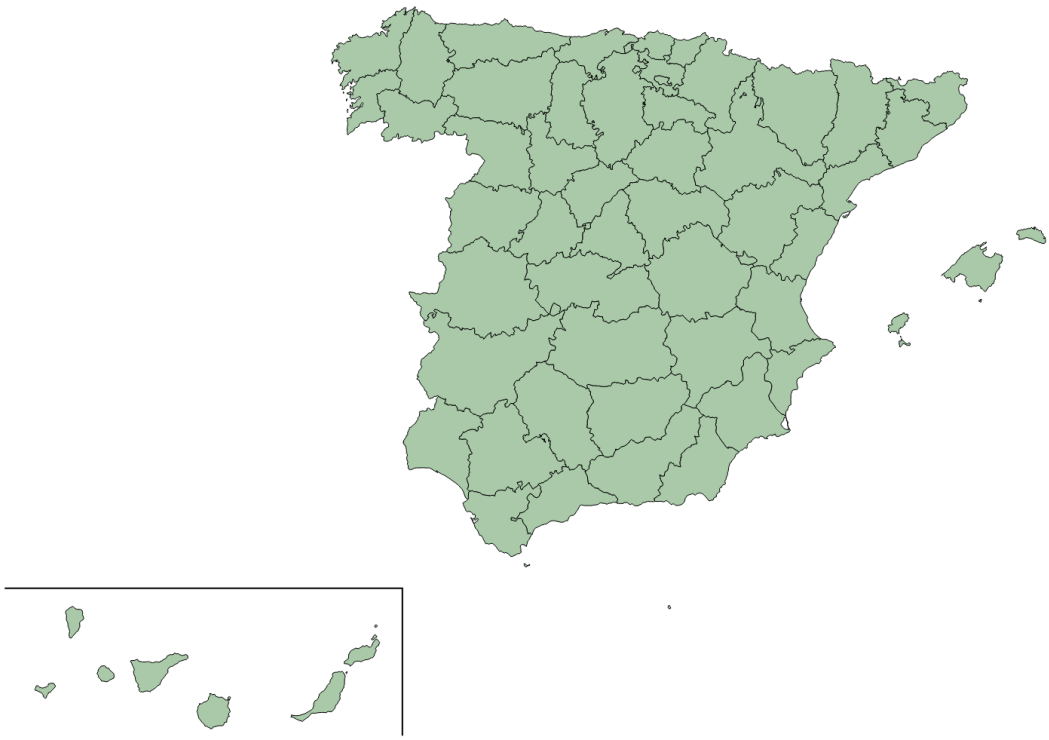

Figura 45 - Mapa de España en la web

7.1.1 REPRESENTACIÓN DE LOS DATOS

Para visualizar los consumos eléctricos se han utilizado tres maneras distintas: mediante círculos con centro en cada provincia, barras en cada provincia y en forma de mapa de calor.

Para cada uno de ellos el proceso es muy similar. Primero, se filtran los datos en función del día que se quiera visualizar, después en función del tipo de consumo y de la representación temporal (día u hora) se escogen los datos adecuados y se fijan los dominios del tamaño de los círculos, barras y rango de colores. Acto seguido, se modifica una leyenda de texto donde nos indica el instante temporal de la visualización y, finalmente, se representan los datos en sus distintas formas.

CÍRCULOS

Mediante la función *update()* representamos los datos en forma de círculos sobre el mapa de España.

Para comenzar, se filtran los datos en función del día, se filtran los cuatro datasets disponibles como se muestra en la Figura 46. Hay que tener en cuenta que en el proyecto tenemos una variable temporal con un inicio predeterminado en el 12 de septiembre de 2013 (fecha de inicio de los datos) y un final en el 19 de octubre de 2015. La evolución de la variable temporal será explicada en el apartado de dinamismo en este capítulo.

```
data2 = consumos.filter(function(d){
        return d.DIA.getTime() == parseTime(formatTime(new Date(time))).getTime();
});

data_fil = consumo_total_por_hora.filter(function(d){
        return d.DIA.getTime() == parseTime(formatTime(new Date(time))).getTime();
});

data_fil_prom = consumos_promedio_total.filter(function(d){
    return d.DIA.getTime() == parseTime(formatTime(new Date(time))).getTime();
});

data2_prom = consumos_promedio.filter(function(d){
    return d.DIA.getTime() == parseTime(formatTime(new Date(time))).getTime();
});
```

Figura 46 - Filtrado de datos en función del día

Ahora hay que distinguir entre qué tipo de consumo se quiere visualizar. Las posibilidades son: consumo total, consumo total residencial, consumo total industrial, consumo promedio total, consumo promedio residencial y consumo promedio industrial. Para cada uno de ellos se hace una distinción entre la variable temporal: consumo absoluto hora a hora, consumo absoluto día a día y, solo en este tipo de representación, consumo acumulado hora a hora. Dentro de cada opción se modifica el dominio del área de los círculos, de consumo 0 al consumo máximo de los datos filtrados con la ayuda de la función de D3 *d3.max()*, que a partir de unos datos obtiene el máximo de la variable en la que estamos interesados. En el caso de consumo absoluto hora a hora tenemos que fijar el dominio en función de la hora que se esté visualizando en el momento, ya que aparte de la variable temporal tenemos una variable hora que corresponde con la hora del consumo que se quiere visualizar. Un ejemplo de esta división de consumos se muestra en la Figura 47, que corresponde al caso de consumo promedio total. La variable "scale_x" será comentada en el apartado comparación

```
}else if(consumo_type == "consumo_promedio_total"){
    area.range([0, 500*Math.PI]);
    if(increment == 0 && time_period.hora_a_hora == 1){

        if(hour == 0){ area.domain([0, d3.max(data_fil_prom, function(d){ return d.ACTIVA_H1; })]); sca

    }else if(increment == 0 && time_period.dia_a_dia == 1){

        area.domain([0, d3.max(data_fil_prom, function(d){ return d.ACTIVA_TOTAL;})]);
        scale_x.domain([0, d3.max(data_fil_prom, function(d){ return d.ACTIVA_TOTAL;})]);

    }else{
        area.domain([0, d3.max(data_fil_prom, function(d){ return d.ACTIVA_TOTAL;})]);
        scale_x.domain([0, d3.max(data_fil_prom, function(d){ return d.ACTIVA_TOTAL;})]);
    }
```

Figura 47 - Ejemplo de fijación de dominios para un tipo de consumo

Cada vez que se llame a la función *update()* va a suponer que la variable temporal ha cambiado, así que hay que modificar la leyenda de texto que muestra el momento del consumo, mostrando la hora si el consumo se muestra por horas y el día de la semana. Con el objeto Date obtenemos el día de la semana en función de un número y con la función *timeFormat()*, al especificar un formato de fecha, transforma una fecha Javascript al formato especificado. Es la variable opuesta a *timeParse()*. Esta modificación de la fecha es común

tanto a la representación por barras como por mapa de calor. La Figura 48 muestra está actualización de la fecha.

```
var d = new Date(time);
var day_week = d.getDay();

switch(day_week){

    case 0:
        if(time_period.dia_a_dia == 1){
            timeLabel.text("Consumo a Domingo: " + formatTime3(new Date(time)));
        }else {
            timeLabel.text("Consumo a Domingo: " + formatTime2(new Date(time)) + ":00");
        }
        break;
    case 1:
        if(time_period.dia_a_dia == 1){
            timeLabel.text("Consumo a Lunes: " + formatTime3(new Date(time)));
        }else {
            timeLabel.text("Consumo a Lunes: " + formatTime2(new Date(time)) + ":00");
        }
        break;
    case 2:
        if(time_period.dia_a_dia == 1){
            timeLabel.text("Consumo a Martes: " + formatTime3(new Date(time)));
        }else {
            timeLabel.text("Consumo a Martes: " + formatTime2(new Date(time)) + ":00");
        }
        break;
    case 3:
        if(time_period.dia_a_dia == 1){
            timeLabel.text("Consumo a Miércoles: " + formatTime3(new Date(time)));
        }else {
            timeLabel.text("Consumo a Miércoles: " + formatTime2(new Date(time)) + ":00");
        }
        break;
    case 4:
```

Figura 48 - Actualización de la leyenda con la fecha

Al cambiar de una representación de datos a otra, necesitamos cambiar la anterior forma por la nueva que se quiere visualizar, por lo que hay que eliminar las anteriores formas (en este caso barras) mediante la función remove() y devolver el color original a las provincias en caso de haber usado el mapa de calor. A continuación (Figura 49) se representa esta transición.

```
if(svg_type_var.circulos == 1 && rects != undefined){
    rects.remove();
    d3.selectAll(".region")
        .style("fill", "#aca");
}else if(svg_type_var.circulos == 1){
    d3.selectAll(".region")
        .style("fill", "#aca");
}
```

Figura 49 - Transición para eliminar las formas anteriores

Una vez tenemos el mapa preparado para incorporar los círculos procedemos a ello. Mediante la potente herramienta de selección de D3 (d3.selectAll()) seleccionamos todos los círculos que tenemos en la web y aplicamos a esos círculos los datos filtrados en función del tipo de consumo a mostrar. Si estamos llamando a la función *update()* después de haber mostrado círculos tenemos que eliminarlos con la función *exit()* y *remove()*, para dar paso a los nuevos círculos. Para añadir los nuevos círculos utilizamos la función *enter()*, y fijamos los atributos y elementos de interacción para cada círculo, que habrá un círculo por cada provincia. Como elementos de interacción se mostrará una leyenda con el consumo seleccionado cada vez que se haga la acción "mouseover" y se dejará de mostrar al retirar el ratón. Al hacer "click" en los círculos se desplegará debajo del mapa una gráfica con dicho consumo, esta función llamada *addChart()* será explicada junto con la variable "scale_x" más adelante. Al introducir los nuevos círculos se hará mediante una transición para que sea progresivo y quede mejor visualmente. Como atributos de los círculos tenemos la opacidad, para que los bordes de los círculos permanezcan visibles en todo momento incluso si se solapan, el color de relleno que dependerá del consumo: color negro para el consumo total, azul para el consumo residencial y rojo para el consumo industrial. Las coordenadas tanto del eje X como Y se encuentran en cada entrada de los datos, en las variables Cx y Cy. El elemento más importante de los círculos consiste en el radio, variable para cada consumo, así que se ha utilizado para ello una función, llamada *circleRadius()*, que recibe como

argumento una entrada de los datos y devuelve el radio utilizando la variable area anteriormente definida. En la Figura 50 se muestra la creación de los círculos.

```
if(consumo_type == "consumo_total" && increment == 0){

    circles = g.selectAll("circle").data(data_fil, function(d){
        return d.PROVINCIA;
    });
}else if(consumo_type == "consumo_promedio_total"){

    circles = g.selectAll("circle").data(data_fil_prom, function(d){
        return d.PROVINCIA;
    })
}else if(consumo_type == "consumo_promedio_residencial" || consumo_type == "consumo_promedio_industrial"){

    circles = g.selectAll("circle").data(data2_prom, function(d){
        return d.PROVINCIA;
    })

}else{

     circles = g.selectAll("circle").data(data2, function(d){
        return d.PROVINCIA;
    });

}


circles.exit()
    .attr("class", "exit")
    .remove();

circles.enter()
    .append("circle")
    .attr("class", "enter")
    .on("mouseover", tip.show)
    .on("mouseout", tip.hide)
    .on("click", function(d){ addChart(d); })
    .merge(circles)
    .transition(t)
        .attr("opacity", 0.7)
        .attr("fill", function(d){ return circle_color;})
        .attr("cy", function(d){ return d.Cy; })
        .attr("cx", function(d){ return d.Cx; })
        .attr("r", function(d){ return circleRadius(d)});
```

Figura 50 - Implementación de los círculos en el mapa

La función circleRadius(d) está formada por un switch-case en el que en función de la hora que se vaya a mostrar escogemos la variable adecuada (ACTIVA_H1, ACTIVA_H2,…) y la introducimos en una variable a modo de mapa donde cada provincia es un array de tres valores: consumo residencial, consumo industrial y consumo total. El caso especial en es en la hora 0, que si el usuario quiere mostrar los datos en función del día de la semana, la hora se fija a 0 y se escoge la variable ACTIVA_TOTAL. Por lo que en este caso hay que hacer

la distinción tanto entre el tipo de consumo como de la variable temporal. La función devuelve el radio del círculo en píxeles, tras aplicar una serie de operaciones para que el tamaño de los círculos sean adecuados. Esta función se muestra en la Figura 51, en concreto el caso en el que la hora es 0.

```
function circleRadius(data){

    switch(hour){
        case 0:
            if((consumo_type == "consumo_total" || consumo_type == "consumo_promedio_total") && (increment =
                if(time_period.hora_a_hora == 1){
                    provincias[data.PROVINCIA.replace(/\s+/g, '')][2] = data.ACTIVA_H1;
                    return Math.sqrt(area(provincias[data.PROVINCIA.replace(/\s+/g, '')][2]) / Math.PI)
                }else if(time_period.dia_a_dia == 1){
                    provincias[data.PROVINCIA.replace(/\s+/g, '')][2] = data.ACTIVA_TOTAL;
                    return Math.sqrt(area(provincias[data.PROVINCIA.replace(/\s+/g, '')][2]) / Math.PI)
                }
            }else{
                if (data.CNAE == "T1"){
                    if(increment == 0 && time_period.hora_a_hora == 1){
                        provincias[data.PROVINCIA.replace(/\s+/g, '')][0] = data.ACTIVA_H1;
                    }else if(increment == 0 && time_period.dia_a_dia == 1){
                        provincias[data.PROVINCIA.replace(/\s+/g, '')][0] = data.ACTIVA_TOTAL;
                    }else{
                        provincias[data.PROVINCIA.replace(/\s+/g, '')][0] = provincias[data.PROVINCIA.replac
                    }
                    if (consumo_type == "consumo_total" || consumo_type == "consumo_promedio_total"){
                        provincias[data.PROVINCIA.replace(/\s+/g, '')][2] = provincias[data.PROVINCIA.replac
                        return Math.sqrt(area(provincias[data.PROVINCIA.replace(/\s+/g, '')][2]) / Math.PI)
                    }else{
                        return Math.sqrt(area(provincias[data.PROVINCIA.replace(/\s+/g, '')][0]) / Math.PI)
                    }

                }else if(data.CNAE == "T2"){
                    if(increment == 0 && time_period.hora_a_hora == 1){
                        provincias[data.PROVINCIA.replace(/\s+/g, '')][1] = data.ACTIVA_H1;
                    }else if(increment == 0 && time_period.dia_a_dia == 1){
                        provincias[data.PROVINCIA.replace(/\s+/g, '')][1] = data.ACTIVA_TOTAL;
                    }else{
                        provincias[data.PROVINCIA.replace(/\s+/g, '')][1] = provincias[data.PROVINCIA.replac
                    }
                    if (consumo_type == "consumo_total" || consumo_type == "consumo_promedio_total"){
                        provincias[data.PROVINCIA.replace(/\s+/g, '')][2] = provincias[data.PROVINCIA.replac
                        return Math.sqrt(area(provincias[data.PROVINCIA.replace(/\s+/g, '')][2]) / Math.PI)
                    }else{
                        return Math.sqrt(area(provincias[data.PROVINCIA.replace(/\s+/g, '')][1]) / Math.PI)
                    }
```

Figura 51 - Función para devolver el radio del círculo, caso de hora 0

Como resultado de aplicar la función *update()* obtenemos los siguientes mapas de España para las variables consumo promedio total, consumo promedio residencial y consumo promedio industrial en función de la hora del día. El resto de resultados de otras variables se presentarán en el capítulo de Análisis de Resultados.

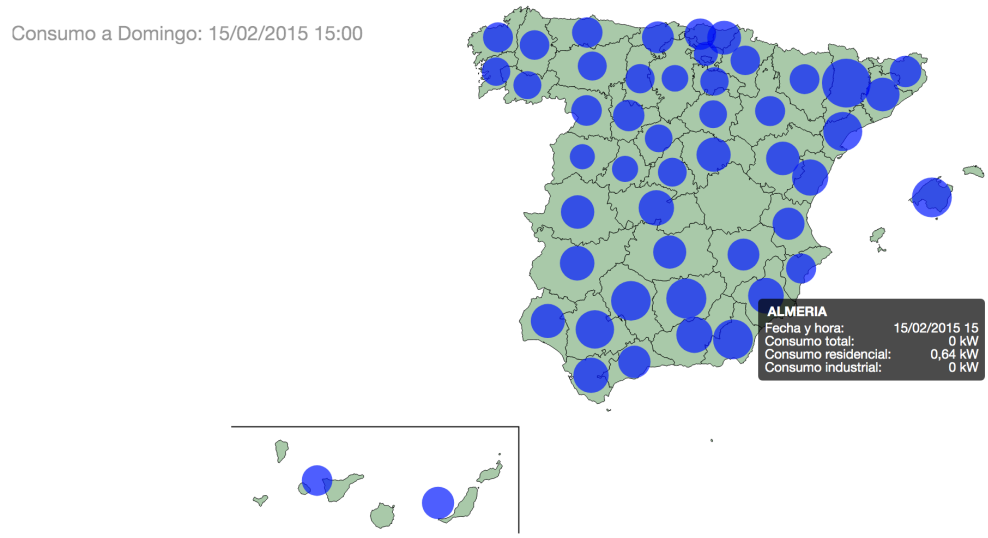

Figura 52 - Consumo promedio residencial el domingo 15 de febrero de 2015 a las 15:00

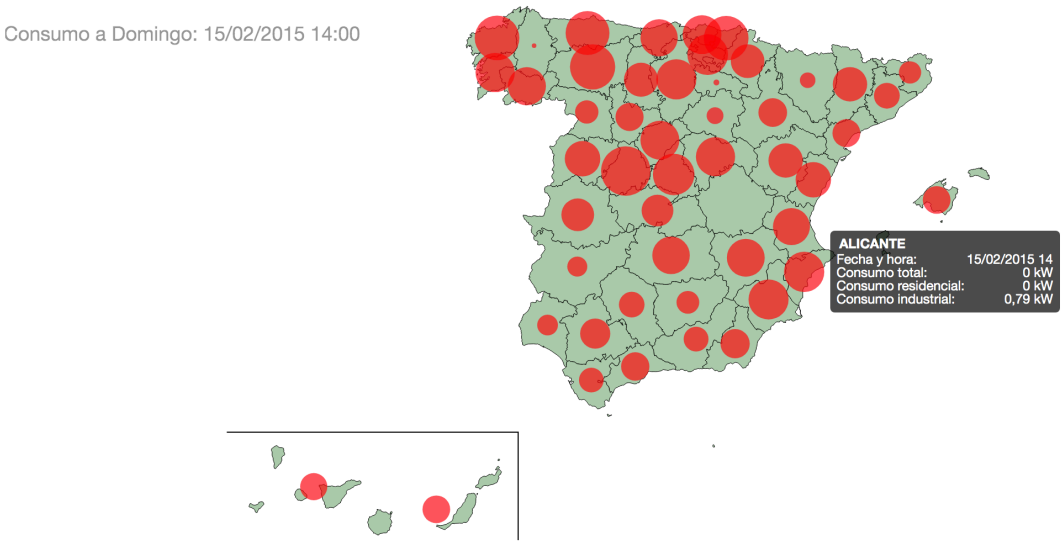

Figura 53 - Consumo promedio industrial el domingo 15 de febrero de 2015 a las 14:00

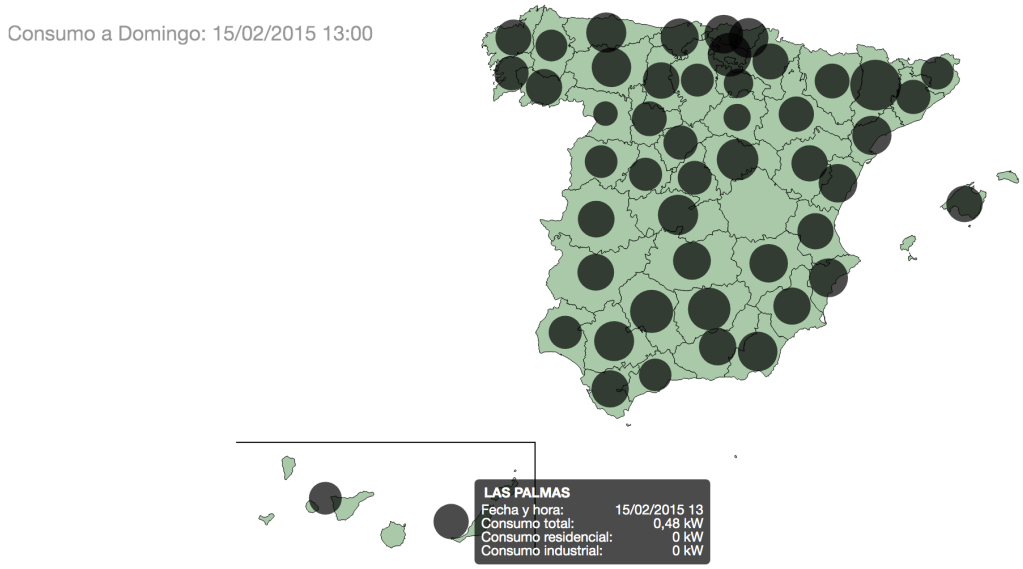

Figura 54 - Consumo promedio industrial el domingo 15 de febrero de 2015 a las 13:00

BARRAS

Mediante la función *update2()* representamos los datos en forma de barras sobre el mapa de España.

El procedimiento de filtrado de los datos es similar al de la representación mediante círculos, con la diferencia de que el dominio que se fija es el de los rectángulos (variable yTOT) como se muestra en la siguiente figura.

```
if(consumo_type == "consumo_total" || consumo_type == "consumo_residencial" || consumo_type == "cor

    if(increment == 0 && time_period.hora_a_hora == 1){

        if(hour == 0){ yTOT.domain([0, d3.max(data_fil, function(d){ return d.ACTIVA_H1; })]); scal

    }else if(increment == 0 && time_period.dia_a_dia == 1){

        yTOT.domain([0, d3.max(data_fil, function(d){ return d.ACTIVA_TOTAL;})]);
        scale_x.domain([0, d3.max(data_fil, function(d){ return d.ACTIVA_TOTAL;})]);
    }else if(increment == 0 && time_period.mes_a_mes == 1){

    }else{
        yTOT.domain([0, d3.max(data_fil, function(d){ return d.ACTIVA_TOTAL;})]);
        scale_x.domain([0, d3.max(data_fil, function(d){ return d.ACTIVA_TOTAL;})]);
    }
```

Figura 55 - Ejemplo de fijación del dominio de los rectángulos

Al igual que con círculos, la leyenda con el texto del momento de visualización del consumo eléctrico se actualiza de la misma manera y también se eliminan los círculos en caso de que el usuario cambie de una representación con círculos a barras.

A continuación nos centramos en el proceso de incorporar los rectángulos al mapa. Para empezar, al igual que con círculos, fijamos los datos a dichos rectángulos en función del consumo que se quiere representar. Acto seguido se eliminan los rectángulos anteriores que se quieran actualizar y se introducen los nuevos rectángulos. Como elementos de interacción mantenemos la leyenda haciendo "mouseover" y desapareciendo al hacer "mouseout" y la función *addChart()* al hacer "click". En este caso los atributos son diferentes a los de los círculos. La coordenada Y se define como la variable Cy en función del dominio del rango de tamaño del rectángulo para que éstos crezcan de manera vertical y hacia arriba. La coordenada X solo se mantiene como la variable Cx si el consumo es el total. En cambio, si el consumo es residencial o industrial se van a mostrar dos barras, una por cada tipo de consumo, para poder comparar ambos consumos al mismo tiempo sobre el mapa, por lo que la coordenada X será diferente para cada tipo de consumo. La anchura de los rectángulos es fija, limitada a 9 píxeles para evitar solapes. El color de los rectángulos sigue el mismo patrón que en los círculos: color negro para el consumo total, azul para el consumo residencial y rojo para el consumo industrial. La altura de los rectángulos se realiza mediante la función *rectangleHeigth()*, que tiene como argumento una entrada de los datos y devuelve la altura en píxeles. La función consiste en un "switch-case" con la hora como parámetro, y en cada caso se escoge la altura en función del tipo de consumo, y se introduce cada consumo en la variable de cada provincia (variable en forma de mapa). Al igual que con los círculos, el caso de hora 0 hace referencia también al consumo día a día. Las siguientes figuras (Figura 56 y Figura 57) muestran el proceso de creación de los rectángulos.

```
rects.enter()
.append("rect")
.attr("class", "enter")
//.attr("id", function(d){return d.PROVINCIA;})
.on("mouseover", tip.show)
.on("mouseout", tip.hide)
.on("click", function(d){ addChart(d); })
.attr("y", function(d){  return d.Cy;  })
.attr("opacity", 0.7)
.merge(rects)
.transition(t)
    .attr("x", function(d){
        if(consumo_type == "consumo_total" || consumo_type == "consumo_promedio_total"){
            return d.Cx;
        }else if(d.CNAE == "T1"){
            return d.Cx - 5.5;
        }else{
            return d.Cx + 5.5;
        }
    })
    .attr("height", function(d){ return rectangleHeight(d); })
    .attr("y", function(d){
        if(consumo_type == "consumo_total" || consumo_type == "consumo_promedio_total"){
            yTOT.range([d.Cy, d.Cy-50]);
            return yTOT(provincias[d.PROVINCIA.replace(/\s+/g, '')][2]);
        }else if(d.CNAE == "T1"){
            yTOT.range([d.Cy, d.Cy-50]);
            return yTOT(provincias[d.PROVINCIA.replace(/\s+/g, '')][0]);
        }else{
            yTOT.range([d.Cy, d.Cy-50]);
            return yTOT(provincias[d.PROVINCIA.replace(/\s+/g, '')][1]);
        }
    })
    .attr("width", 9)
    .attr("fill", function(d){
        if(consumo_type == "consumo_total" || consumo_type == "consumo_promedio_total"){
            return "black";
        }else if(d.CNAE == "T1"){
            return "blue";
        }else{
            return "red";
        }
    });
```

Figura 56 - Creación de rectángulos

```
switch(hour){
    case 0:
        if(consumo_type == "consumo_total" && increment == 0 && time_period.hora_a_hora == 1){
            provincias[data.PROVINCIA.replace(/\s+/g, '')][2] = data.ACTIVA_H1;
            return (data.Cy - yTOT(provincias[data.PROVINCIA.replace(/\s+/g, '')][2]));
        }else if(consumo_type == "consumo_total" && increment == 0 && time_period.dia_a_dia == 1){
            provincias[data.PROVINCIA.replace(/\s+/g, '')][2] = data.ACTIVA_TOTAL;
            return (data.Cy - yTOT(provincias[data.PROVINCIA.replace(/\s+/g, '')][2]));
        }else if(consumo_type == "consumo_promedio_total" && increment == 0 && time_period.hora_a_hora =
            provincias[data.PROVINCIA.replace(/\s+/g, '')][2] = data.ACTIVA_H1;
            return (data.Cy - yTOT(provincias[data.PROVINCIA.replace(/\s+/g, '')][2]));
        }else if(consumo_type == "consumo_promedio_total" && increment == 0 && time_period.dia_a_dia ==
            provincias[data.PROVINCIA.replace(/\s+/g, '')][2] = data.ACTIVA_TOTAL;
            return (data.Cy - yTOT(provincias[data.PROVINCIA.replace(/\s+/g, '')][2]));
        }else{
            if(consumo_type == "consumo_total" || consumo_type == "consumo_promedio_total"){
                provincias[data.PROVINCIA.replace(/\s+/g, '')][2] = provincias[data.PROVINCIA.replace(/\
                return (data.Cy - yTOT(provincias[data.PROVINCIA.replace(/\s+/g, '')][2]));
            }else if (data.CNAE == "T1"){
                if(increment == 0 && time_period.hora_a_hora == 1){
                    provincias[data.PROVINCIA.replace(/\s+/g, '')][0] = data.ACTIVA_H1;
                    return (data.Cy - yTOT(provincias[data.PROVINCIA.replace(/\s+/g, '')][0]));
                }else if(increment == 0 && time_period.dia_a_dia == 1){
                    provincias[data.PROVINCIA.replace(/\s+/g, '')][0] = data.ACTIVA_TOTAL;
                    return (data.Cy - yTOT(provincias[data.PROVINCIA.replace(/\s+/g, '')][0]));
                }else{
                    provincias[data.PROVINCIA.replace(/\s+/g, '')][0] = provincias[data.PROVINCIA.replac
                }   return (data.Cy - yTOT(provincias[data.PROVINCIA.replace(/\s+/g, '')][0]));
            }else{
                if(increment == 0 && time_period.hora_a_hora == 1){
                    provincias[data.PROVINCIA.replace(/\s+/g, '')][1] = data.ACTIVA_H1;
                    return (data.Cy - yTOT(provincias[data.PROVINCIA.replace(/\s+/g, '')][1]));
                }else if(increment == 0 && time_period.dia_a_dia == 1){
                    console.log(data.ACTIVA_TOTAL);
                    provincias[data.PROVINCIA.replace(/\s+/g, '')][1] = data.ACTIVA_TOTAL;
                    return (data.Cy - yTOT(provincias[data.PROVINCIA.replace(/\s+/g, '')][1]));
                }else{
                    provincias[data.PROVINCIA.replace(/\s+/g, '')][1] = provincias[data.PROVINCIA.replac
                }   return (data.Cy - yTOT(provincias[data.PROVINCIA.replace(/\s+/g, '')][1]));
            }
        }
        break;
```

Figura 57 - Caso de hora 0 para la función rectangleHeight()

En el caso de visualizar el consumo residencial o industrial, ya sea en total acumulado o en promedio, se despliegan dos barras por provincia por lo que es necesaria una leyenda para distinguir ambos tipos de consumo. Las siguientes figuras (Figura 58 y Figura 59) presentan algunos ejemplos de resultados representando los consumos eléctricos en formato de barras:

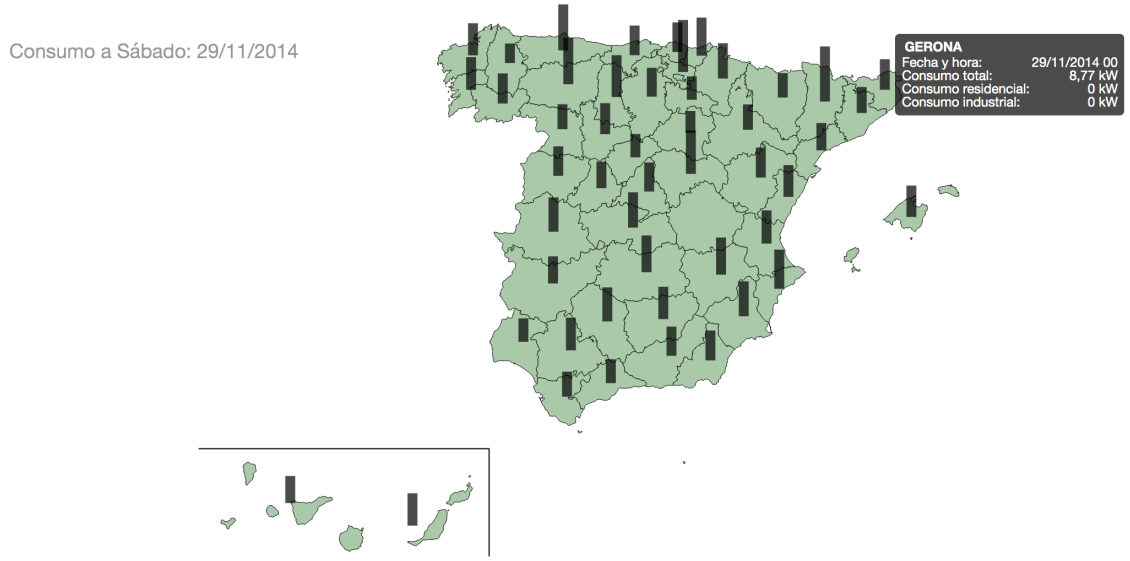

Figura 58 - Consumo promedio total el sábado 29 de noviembre de 2014

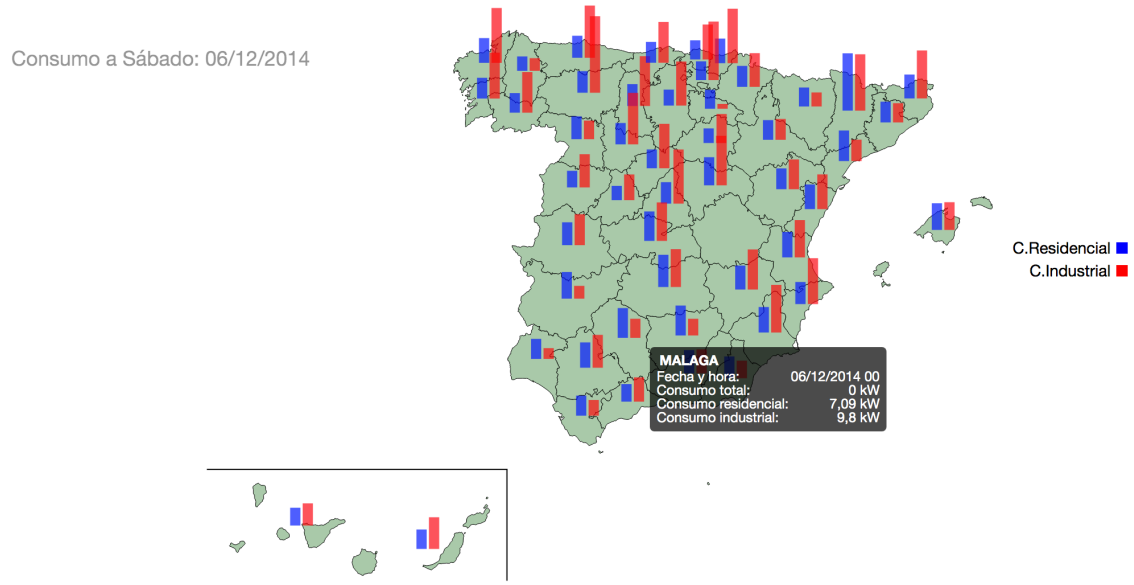

Figura 59 - Consumo promedio residencial e industrial el sábado 6 de diciembre de 2014

MAPA DE CALOR

Mediante la función *updateChropleth()* representamos los datos en forma de mapa de calor sobre el mapa de España.

Como en las anteriores representaciones de los datos, el primer paso es filtrar los datos en función del día que se quiera representar, y en función del tipo de consumo fijamos los dominios de dos variables. Una de ellas es el rango de colores del mapa de calor, cambiando de manera lineal desde el blanco (#f9f9f9) hasta un granate intenso (#bc2a66), y su dominio se establece entre el mínimo valor de los datos (función d3.min()) y el máximo valor (función d3.max()). La otra variable va a ser la leyenda a modo de escala para hacernos una idea del consumo mostrado en función del color. La Figura 60 muestra los parámetros del dominio para el caso de consumo total.

```
if(consumo_type == "consumo_total"){
    if(time_period.hora_a_hora == 1){
        if(hour == 0){ color_choropleth.domain([d3.min(data_fil, function(d){ return d.ACTIVA_H1; }), d3
    }else if(time_period.dia_a_dia == 1){
        var dato_min = d3.min(data_fil, function(d){ return d.ACTIVA_TOTAL; }),
            dato_max = d3.max(data_fil, function(d){ return d.ACTIVA_TOTAL;});
        color_choropleth.domain([dato_min,dato_max]);
        y.domain([dato_min, dato_max]);
        scale_x.domain([0, dato_max]);
    }
```

Figura 60 - Dominio de las variables para el mapa de calor

Continuamos con la actualización de la leyenda con la fecha del consumo y la eliminación de círculos o rectángulos si se ha cambiado al mapa de calor después de representar los datos de alguna de estas dos maneras.

En este tipo de visualización no se va a incorporar ninguna forma para visualizar los datos, sino que se van a utilizar las propias provincias del mapa, llamadas regiones en el entorno web, para acceder a cada una de ellas y aplicar el color correspondiente.

Con la herramienta de selección, escogemos cada una de las provincias y fijamos tanto el atributo del color como los elementos de interacción. Como elementos de interacción

mantenemos la idea de una leyenda al hacer "mouseover", en este caso distinta a las de las anteriores formas, y también conservamos la función *addChart()* al hacer "click". El color de cada provincia es calculado aplicando la función de color (*color_choropleth()*) sobre el consumo deseado, que se obtiene con la función *coloredProvince()*.

La función *coloredProvince()* tiene como argumento la información sobre dicha provincia, proporcionada por la librería TopoJSON y el archivo "spain.json", y devuelve el consumo apropiado. Para ello, primero obtenemos la provincia en cuestión, que sería el "id" del argumento y, en función del tipo de consumo, obtenemos la entrada de los datos sobre consumos eléctricos. Luego, dependiendo de la variable temporal que se haya seleccionado, se escoge la variable adecuada sobre esos datos.

En las figuras Figura 61 y Figura 62 se observa el código de la modificación de las provincias y la función *coloredProvince()*.

```
d3.selectAll(".region")
    .on("mouseover", function(d){
        if(svg_type_var.calor == 1){
            div.transition().duration(300)
            .style("opacity", 1)
            if(consumo_type == "consumo_total" || consumo_type == "consumo_promedio_total"){
                div.text(d.id + " : " + Number(provincias[d.id.replace(/\s+/g, '')][2]).toLocaleString("e
                .style("left", (d3.event.pageX) + "px")
                .style("top", (d3.event.pageY -30) + "px");
            }else if(consumo_type == "consumo_residencial" || consumo_type == "consumo_promedio_residenci
                div.text(d.id + " : " + Number(provincias[d.id.replace(/\s+/g, '')][0]).toLocaleString("e
                .style("left", (d3.event.pageX) + "px")
                .style("top", (d3.event.pageY -30) + "px");
            }else{
                div.text(d.id + " : " + Number(provincias[d.id.replace(/\s+/g, '')][1]).toLocaleString("e
                .style("left", (d3.event.pageX) + "px")
                .style("top", (d3.event.pageY -30) + "px");
            }
        }
    })
    .on("mouseout", function(){
        if(svg_type_var.calor == 1){
            div.transition().duration(300)
            .style("opacity", 0);
        }
    })
    .on("click", function(d){
        if(svg_type_var.calor == 1){
            addChart(d);
        }
    })
    .transition(t)
        .style("fill", function(d){
            return color_choropleth(coloredProvince(d));
        });
```

Figura 61 - Generación del mapa de calor cambiando cada provincia

```
}else if(consumo_type == "consumo_promedio_residencial" || consumo_type == "consumo_promedio_industrial"){
    for(i=0; i<data2_prom.length; i++){
        if(data2_prom[i]["PROVINCIA"] == provincia){
            data_row = data2_prom[i];
        }
    }
    if(data_row.length == 0){
        return 0;
    }
}


switch(hour){
    case 0:
        if(consumo_type == "consumo_total" || consumo_type == "consumo_promedio_total"){
            if(time_period.hora_a_hora == 1){
                provincias[provincia.replace(/\s+/g, '')][2] = data_row.ACTIVA_H1;
                return data_row.ACTIVA_H1;
            }else if(time_period.dia_a_dia == 1){
                provincias[provincia.replace(/\s+/g, '')][2] = data_row.ACTIVA_TOTAL;
                return data_row.ACTIVA_TOTAL;
            }
        }else if(consumo_type == "consumo_residencial" || consumo_type == "consumo_promedio_residencial"){
            if(time_period.hora_a_hora == 1){
                provincias[provincia.replace(/\s+/g, '')][0] = data_row.ACTIVA_H1;
                return data_row.ACTIVA_H1;
            }else if(time_period.dia_a_dia == 1){
                provincias[provincia.replace(/\s+/g, '')][0] = data_row.ACTIVA_TOTAL;
                return data_row.ACTIVA_TOTAL;
            }
        }else if(consumo_type == "consumo_industrial" || consumo_type == "consumo_promedio_industrial"){
            if(time_period.hora_a_hora == 1){
                provincias[provincia.replace(/\s+/g, '')][1] = data_row.ACTIVA_H1;
                return data_row.ACTIVA_H1;
            }else if(time_period.dia_a_dia == 1){
                provincias[provincia.replace(/\s+/g, '')][1] = data_row.ACTIVA_TOTAL;
                return data_row.ACTIVA_TOTAL;
            }
        }
        break;
    case 1:
```

Figura 62 - Caso de hora 0 en la función coloredProvince()

Una vez modificados los colores de las provincias, mostramos la leyenda con la escala de colores y el consumo que corresponde a cada segmento de color para hacernos una idea del consumo de cada provincia, en kW.

Como resultado, obtenemos los siguientes ejemplos de mapa de calor (Figura 63 y Figura 64):

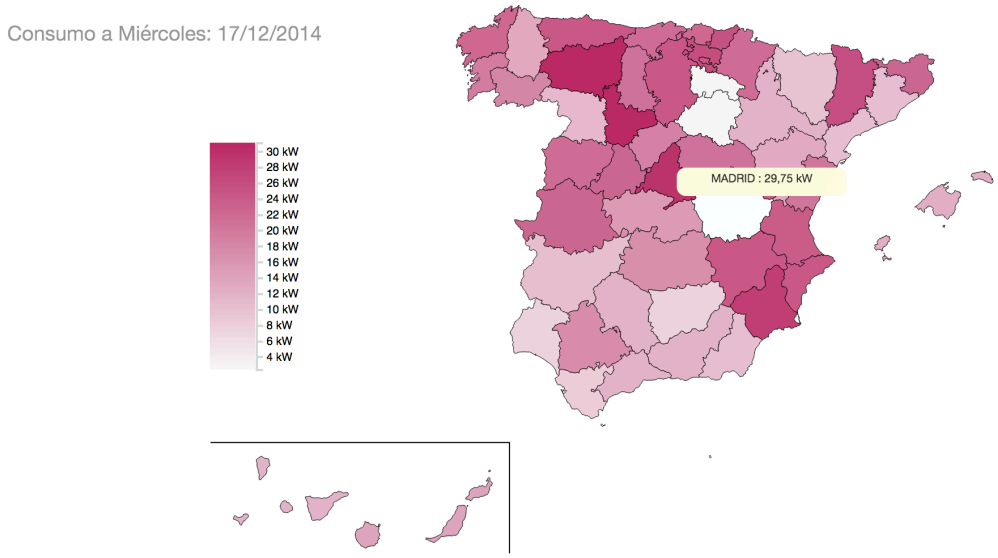

Figura 63 - Consumo promedio industrial el 17 de diciembre de 2014

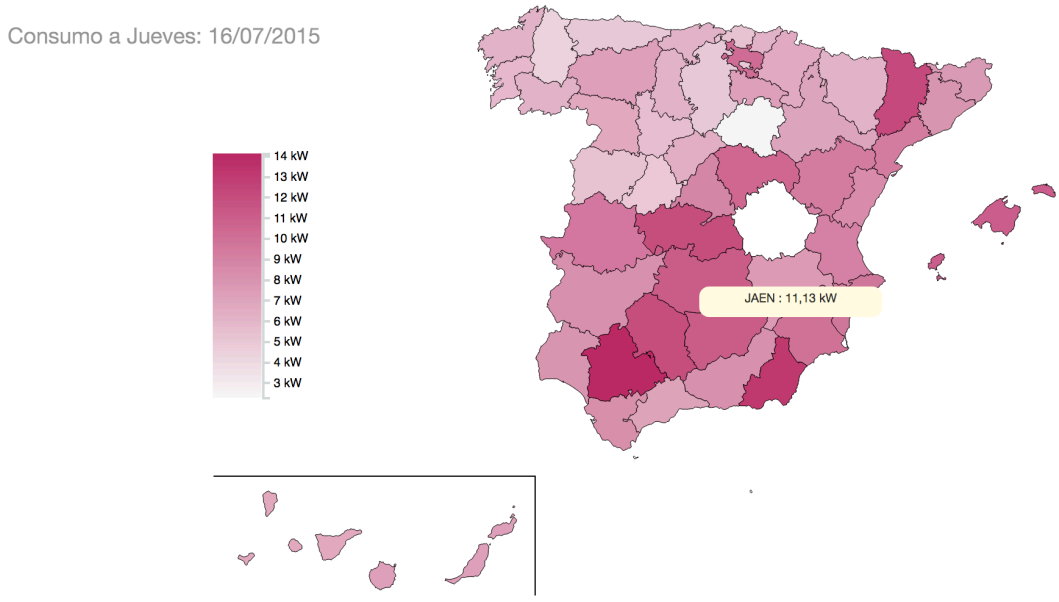

Figura 64 - Consumo promedio residencial el 16 de julio de 2015

7.1.2 DINAMISMO E INTERACCIÓN

Una de los objetivos del proyecto es aportar una visión dinámica e interactiva en la visualización de los datos. El dinamismo que se ha implementado en el proyecto está relacionado con la variable temporal, tanto del día de la semana como la hora del día. En cuanto a interactividad, el usuario tiene a su elección elegir el tipo de consumo, la variable temporal (día u hora del día) y la forma de representar los datos (círculos, barras o mapa de calor).

Para comenzar, el carácter dinámico se ha implementado gracias a una característica de Javascript llamada intervalos, donde con la función *setInterval()*, especificando otra función y un tiempo en milisegundos, se accede a dicha función cuando se llega a ese tiempo de manera iterativa. Para terminar el intervalo se utiliza la función *clearInterval(intervalo)*. Se ha utilizado un botón de "play" para comenzar un intervalo (iniciado cada 100ms) y para terminarlo al darle otra vez al botón, en este caso estará en modo pausa. Cada 100ms se llama a la función *step()* que actualiza la variable temporal "time" y hora, ya sea añadiendo un día o una hora en milisegundos. Si se llega al límite temporal de los datos se vuelve al inicio de manera automática. Esta función, una vez modificada la variable temporal realiza una llamada a las distintas funciones para representar los datos: *update()* para círculos, *update2()* para barras y *updateChropleth()* para mapa de calor. Las figuras Figura 65 y Figura 66 muestran la función del botón de "play" y la función *step()*. Para volver al estado temporal inicial en cualquier momento, se ha implementado un botón para resetear que inicializa la variable temporal.

```
$("#play-button")
    .on("click", function(){
        var button = $(this);
        if (button.text() == "Play"){
            button.text("Pause");
            interval = setInterval(step, transition_step);
            data_chart = [];
            chart_visible = 0;
            xAxis.attr("visibility", "hidden");
            yAxis.attr("visibility", "hidden");
            if(element != undefined && rectangles!= undefined){
                element.remove();
                rectangles.remove();
                legendRow.remove();
                legendRow2.remove();
            }
        }
        else {
            button.text("Play");
            clearInterval(interval);
        }
    })
```

Figura 65 - Funcionalidad del botón "play"

```
function step(){
    // At the end of our data, loop back
    //console.log(hour);
    console.log(hour);

    if(time < timeLimit){
        if(time_period.mes_a_mes == 1){
            time = time;
        }else if(time_period.dia_a_dia == 1){
           time = time + 86400000;
            for(var key in provincias){
                provincias[key] = [0,0,0];
            }
        }else{
            time = time + 3600000;
            hour++;
            if(hour > 23){
                hour = 0;
                for(var key in provincias){
                    provincias[key] = [0,0,0];
                }
            }
        }

    }else{
        time = timeInit;
        hour = 0;
        for(var key in provincias){
                provincias[key] = [0,0,0];
        }
    }

    if(svg_type_var.circulos == 1){
        update();
    }else if(svg_type_var.barras == 1){
        update2();
    }else if(svg_type_var.calor == 1){
        updateChropleth();
    }
}
```

Figura 66 - Función step()

Como mencionado, el tiempo de transición entre actualizaciones del mapa es de 100ms. El usuario puede modificar este tiempo de transición introduciendo el tiempo en milisegundos en una caja de texto, que contiene el conveniente control de errores para el usuario.

Para controlar la variable temporal de los datos, es decir, que el usuario pueda poner un intervalo de fechas distinto al inicio y final de los datos. Por ejemplo, el usuario puede escoger una semana en concreto de los datos y observar el comportamiento de los datos durante esa semana dando al botón de "play". Para implementarlo se ha optado por un slider de jQuery con inicio en el principio de los datos y final en el fin de los datos disponibles. Si el usuario quiere ver un día exacto o compararlo con otro sin que comience el intervalo del botón "play", con dos botones debajo del slider se podrá avanzar y retrasar la variable temporal para ver los datos un día antes o después, o una hora antes o después. La siguiente figura (Figura 67) muestra el slider junto con los botones para avanzar o volver.

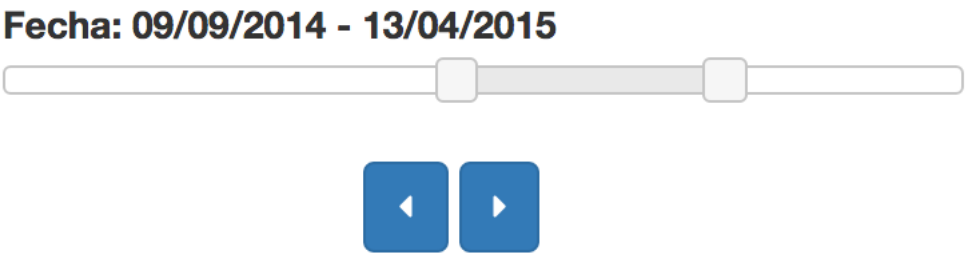

Figura 67 - Slider y botones para modificar la variable temporal de los datos

El tipo de consumo, la forma de representar los datos y la variable temporal se controlan mediante tres menús que se despliegan con las distintas opciones.

Tipos de consumo disponibles:

- Consumo total: se refiere el consumo total acumulado de cada provincia
- Consumo residencial: se refiere al consumo total residencial acumulado de cada provincia.

- Consumo industrial: se refiere al consumo total industrial acumulado de cada provincia.
- Consumo promedio total: referido al consumo promedio total de cada provincia
- Consumo promedio residencial: referido al consumo promedio residencial de cada provincia.
- Consumo promedio industrial: referido al consumo promedio industrial de cada provincia.

Formas de representar los datos:

- Círculos
- Barras o rectángulos
- Mapa de calor

Variables temporales:

- Consumo absoluto cada día
- Consumo absoluto cada hora del día
- Consumo acumulado cada hora del día

El menú completo para la interacción del usuario se muestra en la Figura 68, y el mapa estaría justo debajo:

Figura 68 - Menú de interacción con el usuario

Otro elemento de interacción con el usuario introducido es la capacidad para comparar consumos eléctricos de las provincias. En el mapa se puede hacer mouseover sobre las provincias para ver el consumo de dicha provincia. Pero con esta leyenda no nos vale si queremos comparar el consumo de varias provincias. Para poder compararlos se ha implementado una función en cada forma de representar los datos de manera que al hacer "click" sobre ellas se despliega debajo del mapa un gráfico de barras donde se van añadiendo los consumos de cada provincia que hacemos "click".

Esta función tiene de nombre *addChart()* y genera un gráfico de barras con el consumo que se esté visualizando. En el eje X se mide el consumo eléctrico en kW y en el eje Y está la variable categórica de la provincia. Para el caso del mapa con rectángulos, si la opción de consumo residencial, consumo industrial, consumo promedio residencial o consumo promedio industrial está seleccionada, por cada "click" sobre la provincia se desplegarán en el gráfico dos rectángulos por provincia. El gráfico de barras también tiene la leyenda correspondiente para indicar el consumo que se está visualizando. Las figuras Figura 69 y Figura 70 muestran el gráfico de barras tras haber hecho "click" en las provincias que aparecen.

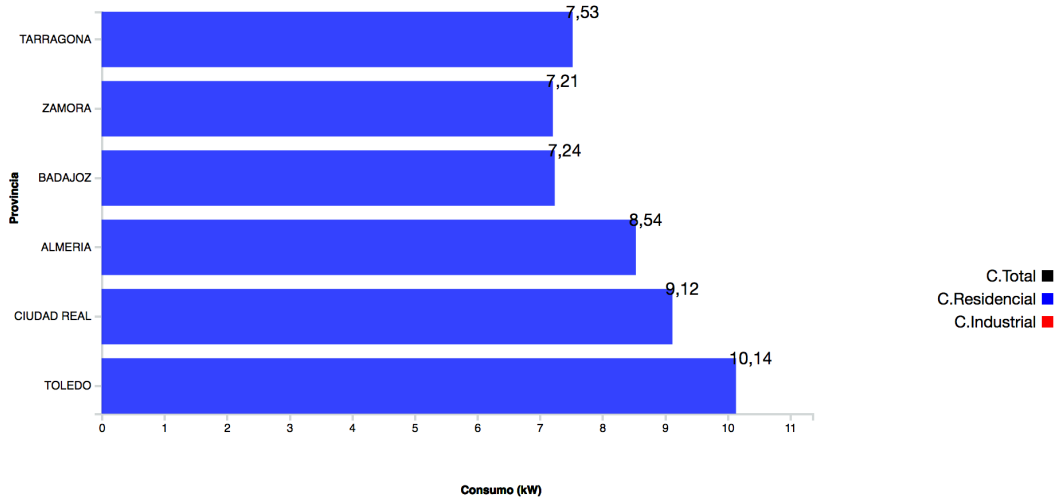

Figura 69 - Consumo residencial en dichas provincias

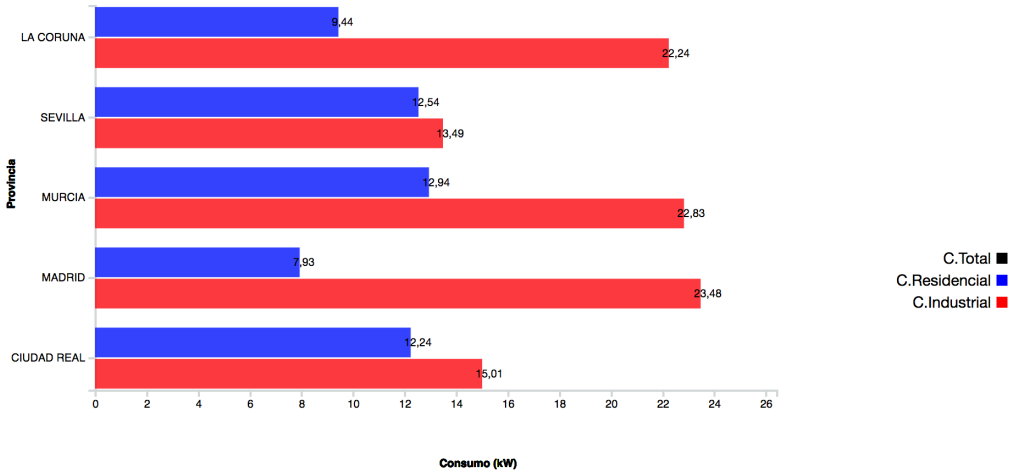

Figura 70 - Ejemplo del caso del mapa con rectángulos

Capítulo 8. ANÁLISIS DE RESULTADOS

Con la potente herramienta de visualización creada, este capítulo se va a centrar en extraer conclusiones de los datos y analizarlas mediante algunos ejemplos. El objetivo ahora es identificar patrones y dar explicaciones rápidas sobre lo que nos ofrecen los datos en el mapa, cosa que sería imposible de hacer con los datos aislados.

Para comenzar, nada más cargar por primera vez la página web se observan datos solo en la provincia de La Coruña (Figura 71), esto es debido a que hasta el 15 de mayo de 2014 solo se poseen datos sobre el consumo industrial de esta provincia. A partir de esta fecha se van incorporando poco a poco datos del resto de las provincias.

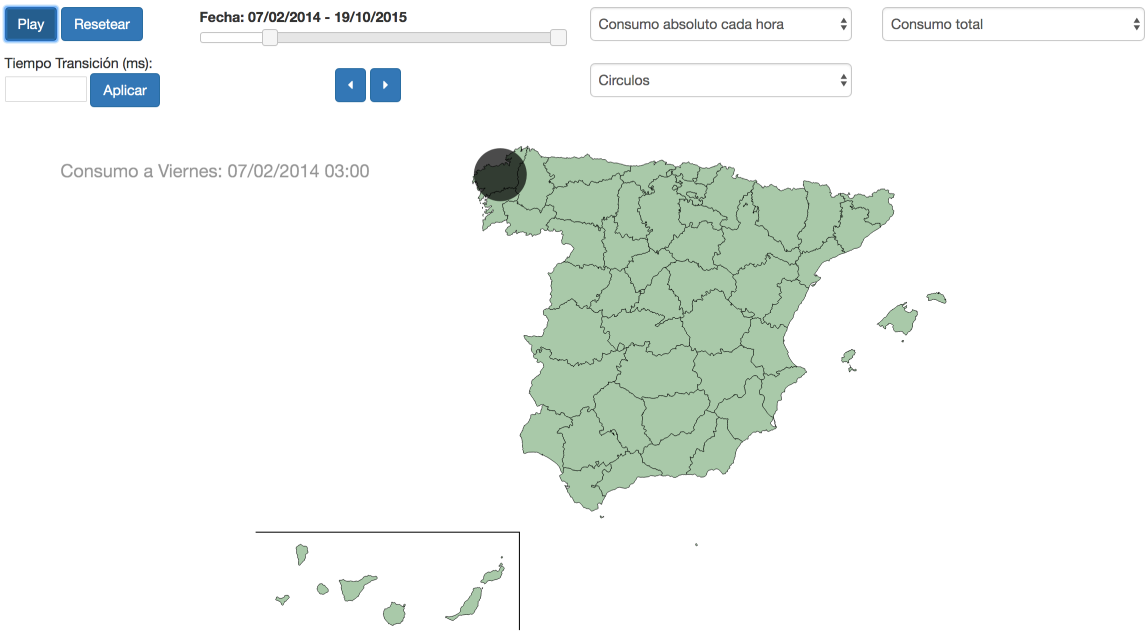

Figura 71 - Consumo solo en La Coruña

Otro aspecto a destacar es que no se poseen datos de la provincia de Cuenca, por lo que no se representarán sus datos sobre el mapa, como se irá viendo a lo largo de las figuras de esta sección.

Tras interactuar en las distintas fechas se ha llegado a la conclusión de que de los datos de consumo residencial recogidos en la provincia de Lérida forman un "outlier" ya que de los pocos datos que hay todos poseen un consumo demasiado elevado en comparación con otras provincias que deberían tener un consumo parecido. A lo largo del capítulo se irán observando estos datos atípicos.

Por estos motivos de ausencia de datos en determinadas fechas, el análisis de los resultados va a estar relacionado con el invierno de 2014-2015 y el verano de 2015 como ejemplos de resultados con datos completos. Se utilizarán tanto círculos, barras y mapa de calor para obtener dichos análisis, ya que cada uno puede presentar unas ventajas visuales en función de los datos.

INVIERNO 2014-2015 (21 de diciembre de 2104 – 21 de marzo de 2015)

En cuanto al consumo promedio total diario, a lo largo del invierno se observa un mayor consumo (promediando consumo residencial e industrial) en el norte peninsular, destacando Álava, que en el resto del mapa, como se puede observar en la siguiente figura:

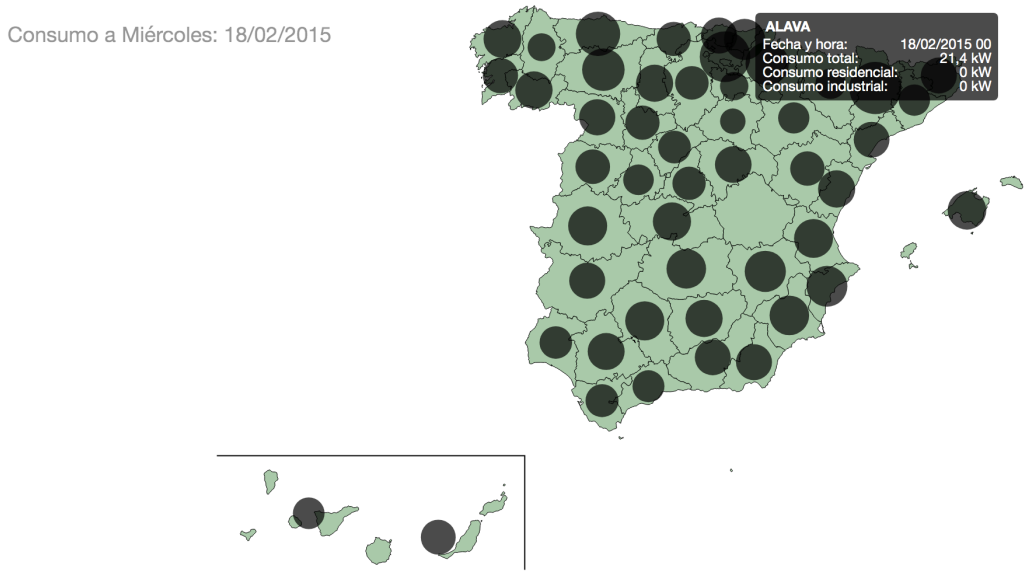

Figura 72 - Consumo promedio total un día en invierno

Para averiguar cuál es el motivo de esta afirmación, mostramos el consumo promedio residencial e industrial de cada provincia. En cuanto al consumo promedio residencial, en general, el consumo eléctrico es ligeramente superior en el sur de España y en las islas Baleares que en el resto de España, a excepción de Lérida que es el "outlier" comentado anteriormente. En cambio, el consumo promedio industrial es mucho mayor en el norte peninsular que en el sur, destacando también un consumo elevado en la Comunidad Valenciana, Murcia y Albacete. Las figuras Figura 73 y Figura 74 muestran ambos consumos.

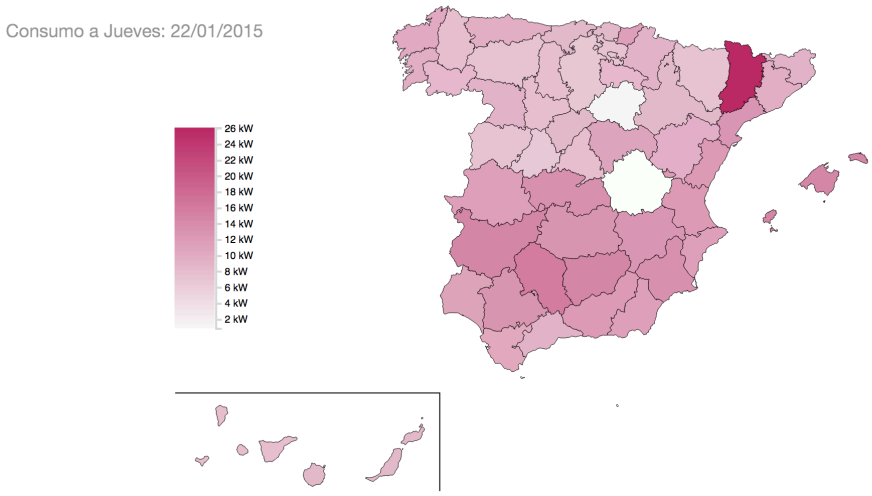

Figura 73 - Consumo promedio residencial un día de invierno

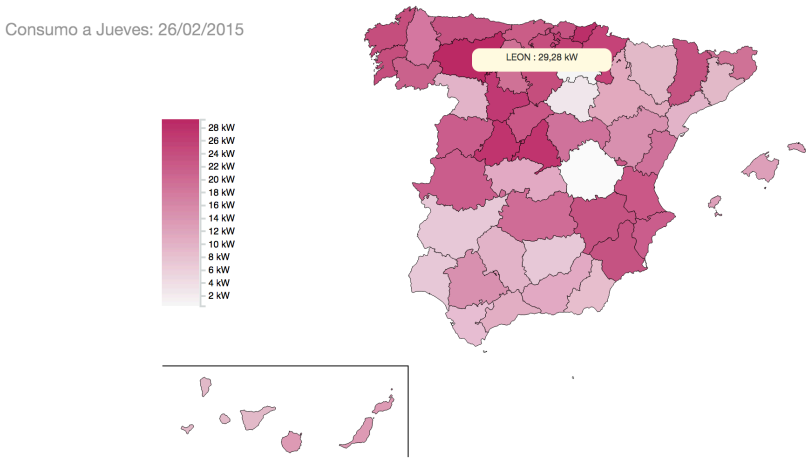

Figura 74 - Consumo promedio industrial un día de invierno

Para obtener una comparación directa entre el consumo promedio residencial e industrial acudimos a la representación mediante barras. La Figura 75 muestra dicha comparación. Con esta visualización se confirma que la razón por la que el consumo promedio total es mayor en el norte peninsular se debe a a la cantidad de industrias activas.

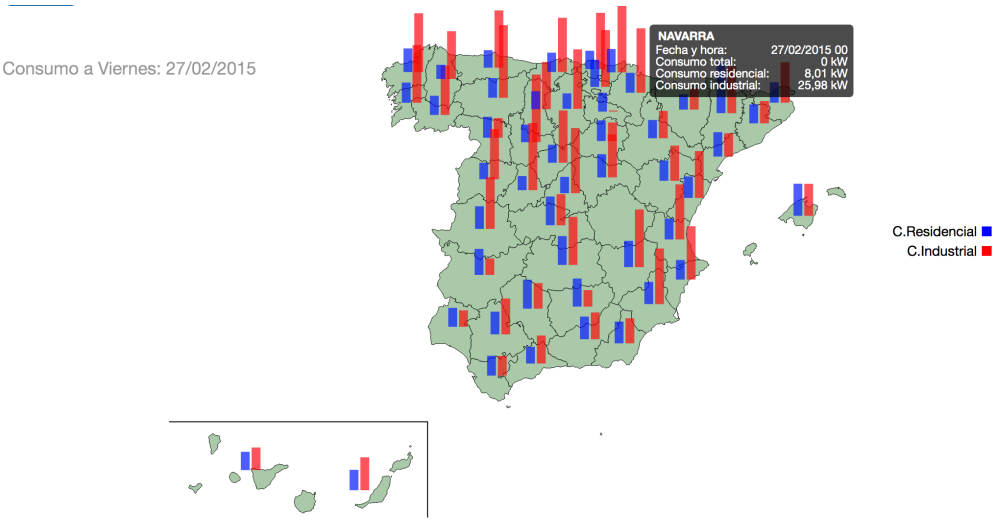

Figura 75 - Comparación de ambos consumos en un día de invierno

VERANO 2015 (21 de junio de 2015 – 23 de septiembre de 2015)

Vamos a seguir el mismo planteamiento que en invierno, es decir, comenzamos analizando el consumo promedio total y acto seguido, el motivo de dichas conclusiones con el consumo promedio residencial y el consumo promedio industrial.

El consumo promedio total se presenta bastante similar cerca de las costas y es superior al interior de la península, con un consumo pronunciado en la costa mediterránea, destacando las provincias de Alicante, Murcia, Almería o las islas Baleares. Esto tiene sentido ya que es una época de vacaciones y la gente del interior peninsular tiende a viajar a la costa, como se muestra en la siguiente figura:

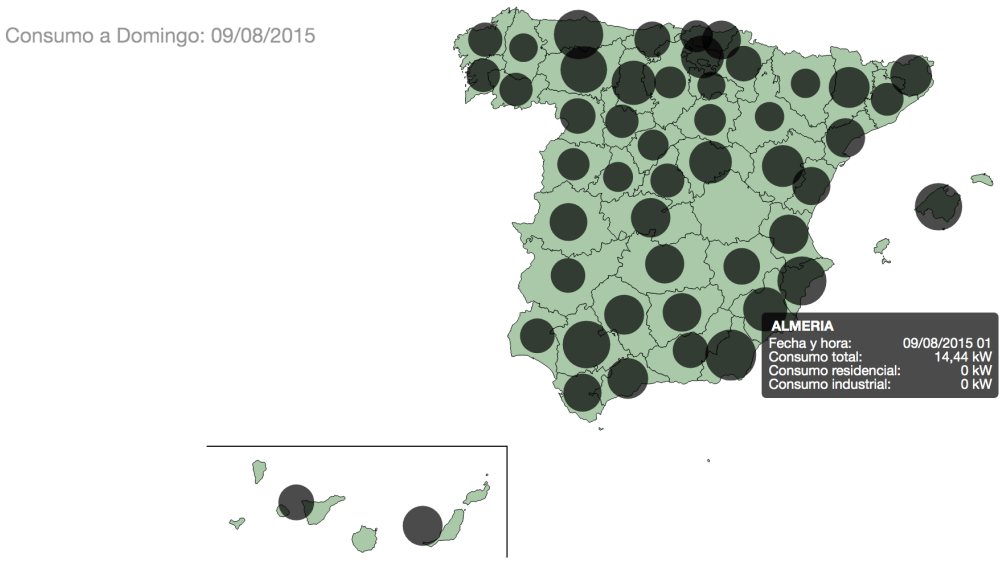

Figura 76 - Consumo promedio total un domingo de agosto

Para desglosar el motivo de este consumo promedio total, vamos a observar el consumo promedio residencial y el consumo promedio industrial.

El consumo promedio residencial a lo largo del verano, por lo general, es mayor en el sur peninsular y en la costa mediterránea que en el norte. Dentro del sur peninsular es menor en las zonas costeras ya que disponen de la brisa marina y no hacen tanto uso del aire acondicionado como en las zonas más interiores como Sevilla, Córdoba o Jaén. También destaca la zona sureste de la península, Almería, Murcia o Alicante, que suele ser una zona escogida para las vacaciones por mucha gente, por lo que el consumo residencial es mayor. El mismo razonamiento siguen las islas Baleares. La Figura 77 señala estas afirmaciones.

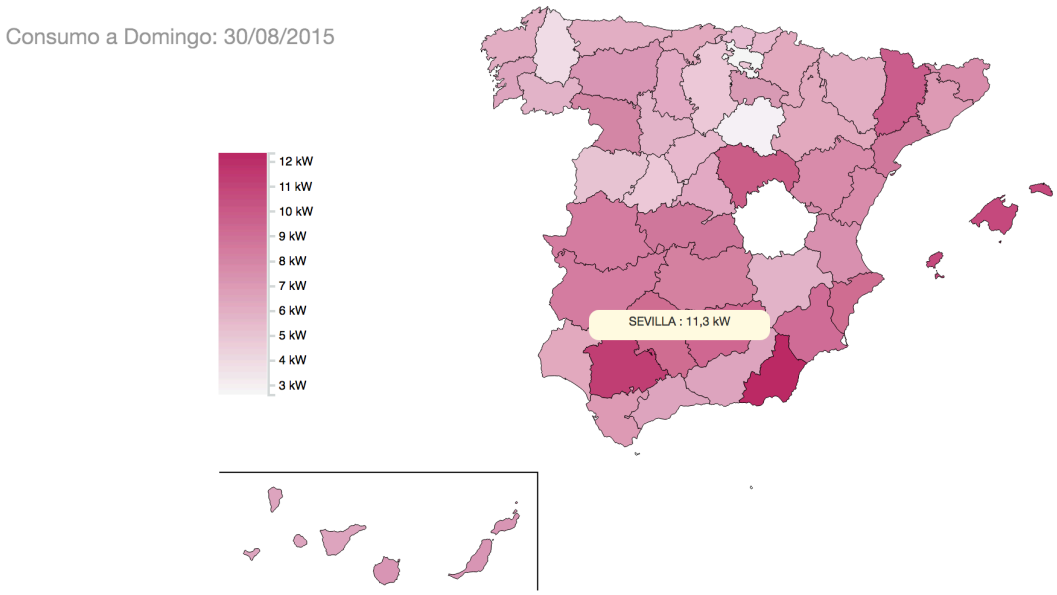

Figura 77 - Consumo promedio residencial un domingo de agosto

El consumo promedio industrial en verano sigue una dinámica parecida a la del invierno, con un mayor consumo en el noroeste y sureste peninsular. En la comunidad valenciana puede ser debido a las grandes empresas hoteleras y en el norte hay predominio de las industrias. La Figura 78 muestra estos resultados.

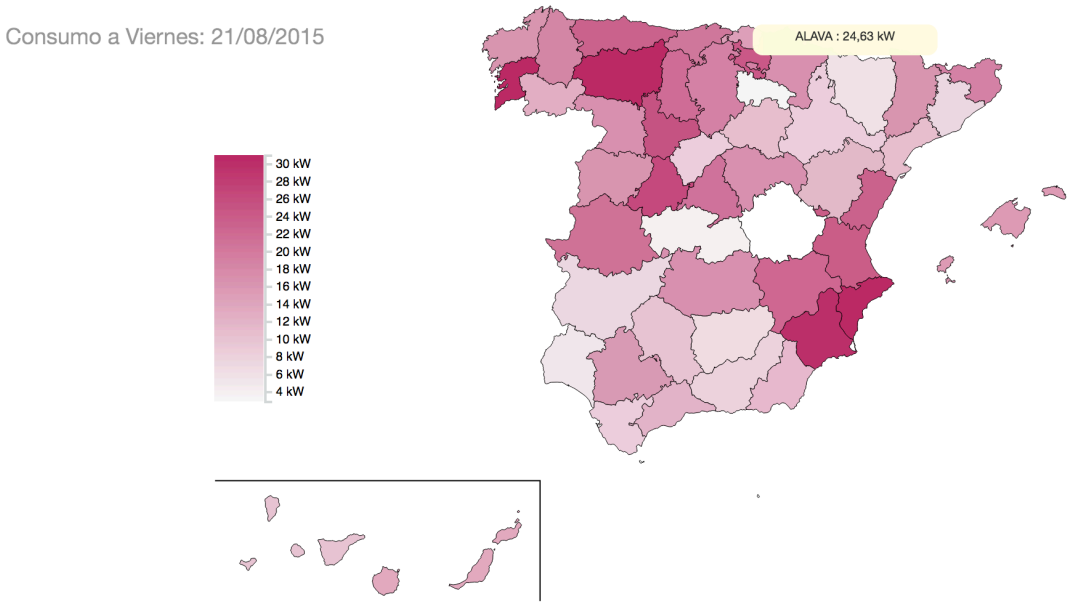

Figura 78 - Consumo promedio industrial un viernes de agosto

La diferencia entre el consumo en invierno y en verano radica en el elevado consumo residencial por parte del sur peninsular debido a las vacaciones y el uso del aire acondicionado por las elevadas temperaturas. De aquí que el consumo promedio total sea más equilibrado que en invierno.

Para asegurarnos vamos a observar el mapa con diagrama de barras para ver la comparación directa entre ambos consumos. La siguiente figura (Figura 79) muestra la comparación. Se puede observar como el consumo promedio residencial en el sur de la península y en la costa mediterránea es superior al del norte, mientras que el consumo industrial predomina en el norte y en la costa valenciana.

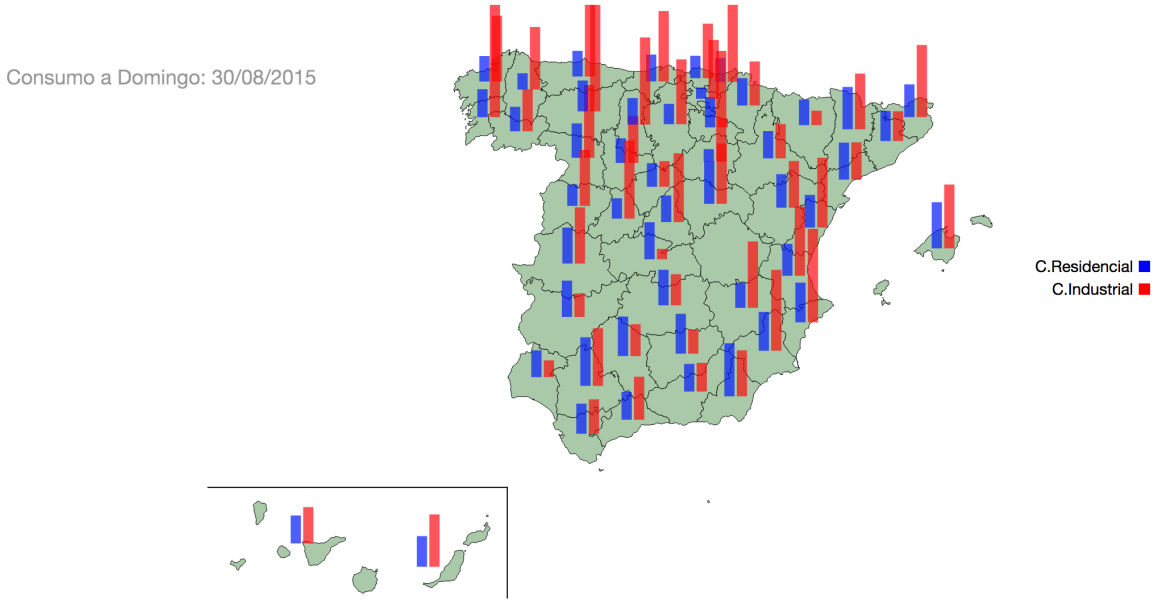

Figura 79 - Comparación de consumos un domingo de agosto

CONSUMO DURANTE EL DÍA

Mediante el diagrama de barras en el mapa podemos ver que el consumo eléctrico durante el día varía de gran manera. Durante la mañana y la tarde los consumos tanto residencial como industrial son más elevados que durante la noche como es de esperar. En las siguientes figuras (Figura 80, Figura 81 y Figura 82, Figura 83) se observa el cambio durante el día y la noche. Para ver el cambio de manera exacta acudimos al gráfico de barras para comparar el consumo de distintas provincias en horas diferentes.

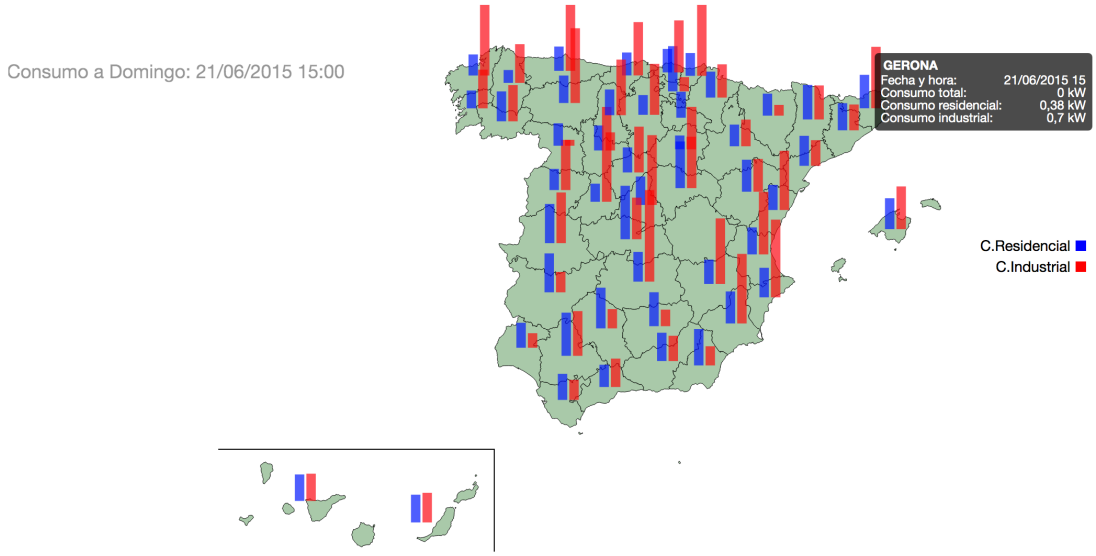

Figura 80 - Consumos industrial y residencial a las 15:00 un domingo de agosto

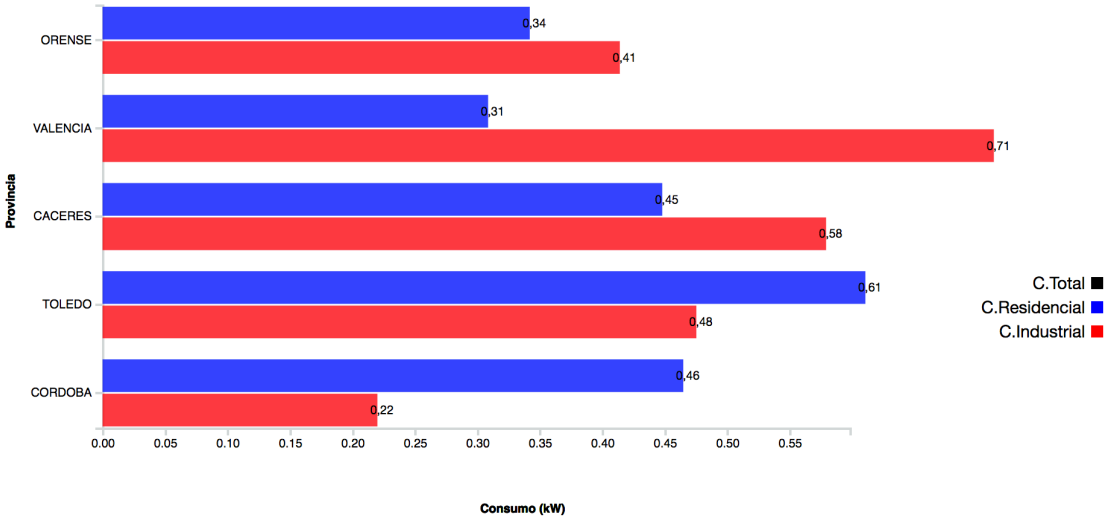

Figura 81 - Consumos industrial y residencial de algunas provincias a las 15:00 un domingo de agosto

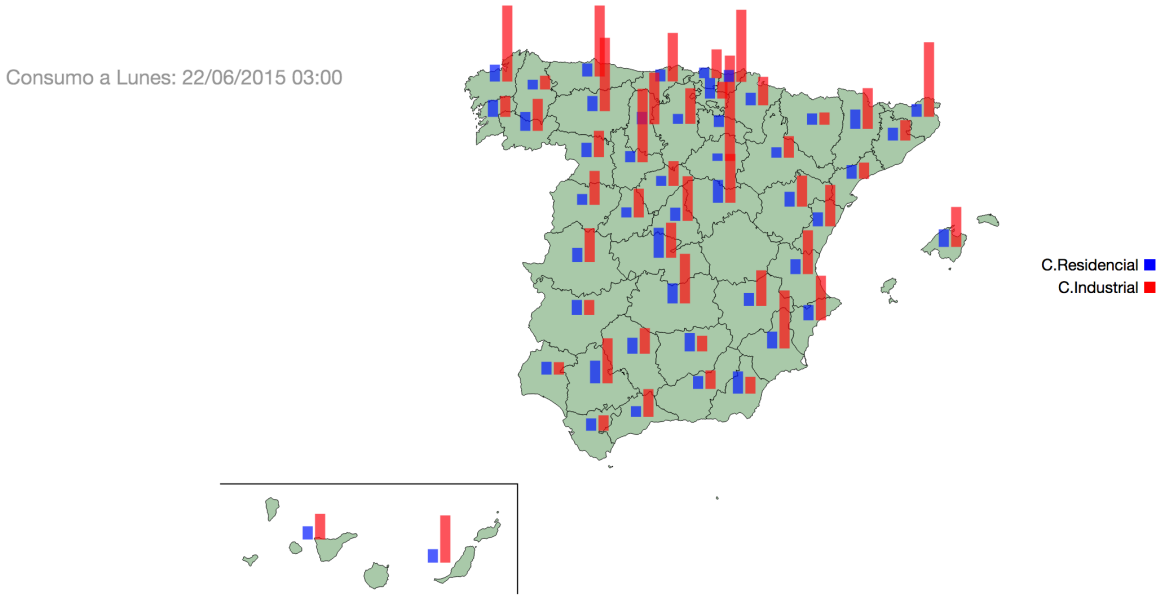

Figura 82 - Consumos industrial y residencial a las 3:00 un lunes de agosto

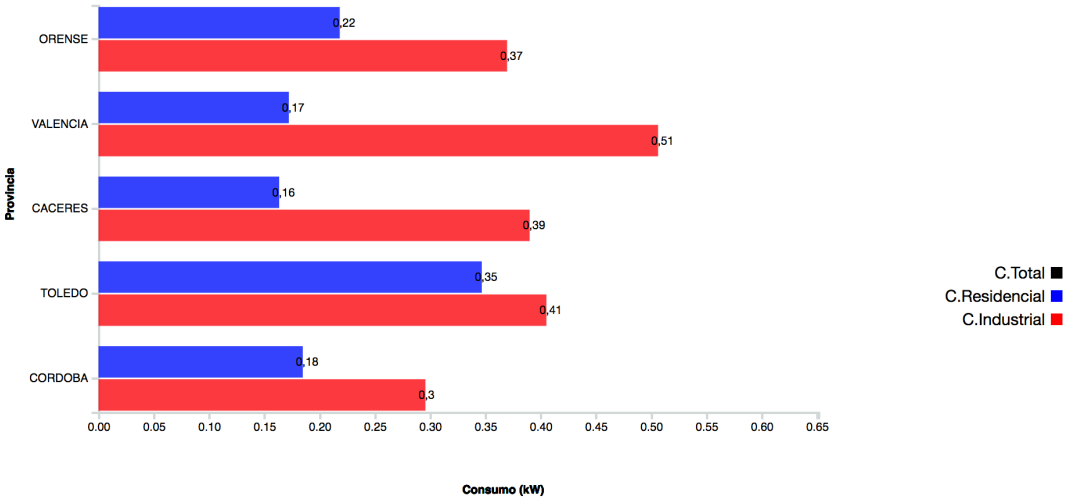

Figura 83 - Consumos industrial y residencial de algunas provincias a las 3:00 un lunes de agosto

Capítulo 9. CONCLUSIONES Y TRABAJOS FUTUROS

Se ha desarrollado una potente herramienta de visualización para representar los consumos eléctricos de España. Las características u objetivos del proyecto que se han llevado a cabo eran los indicados, siendo tecnología de visualización un mapa de España geotemporal, con dinamismo e interacción con el usuario. Al ser un entorno web el usuario puede visualizar los datos en cualquier lugar y momento. A su vez se ofrecen distintas maneras de visualizar los datos para obtener distintas perspectivas y poder tanto comparar como tomar decisiones de manera rápida y eficaz.

Esta página web ha sido posible gracias a la tecnología utilizada (D3.js) y al análisis previo de los datos mediante Jupyter Notebook para extraer los datasets necesarios.

Se ha evolucionado respecto a un anterior intento donde la visualización era muy pobre, sin dinamismo ni interacción con el usuario, al ser simples gráficas estáticas. Para un futuro se pretenden incluir en la herramienta de visualización los patrones obtenidos mediante algoritmos de detección de los mismos como algoritmos de clustering.

Capítulo 10. BIBLIOGRAFÍA

[1] Visualización de datos: definición, tecnologías y herramientas, iniciativa aporta, Gobierno de España, noviembre 2016, https://datos.gob.es/sites/default/files/doc/file/informe_herramientas_visualizacion.pdf

[2] 2019 Project Jupyter, The Jupyter Notebook, 12 de Abril 2019, https://jupyter.org

[3] The Apache Software Foundation, Apache Spark, https://spark.apache.org

[4] Sublime HQ Pty Ltd, Sublime Text, https://www.sublimetext.com

[5] Mike Bostock, D3 Data-Driven Documents, https://d3js.org

[6] Ministerio de Energía, Turismo y Agenda Digital, "Energía Eléctrica", https://energia.gob.es/electricidad/Paginas/Index.aspx

[7] Red Eléctrica de España, "Misión y Visión", https://www.ree.es/es/conocenos/ree-en-2-minutos/mision-y-vision

[8] Red Eléctrica de España , "Transformación Digital del Sector Eléctrico", https://www.ree.es/sites/default/files/Transformacion_Digital_Sector_Electrico.pdf

[9] Gráfico circular, HighBond, 27 de junio de 2019, https://help.highbond.com/helpdocs/results/current/user-guide/es/Content/visualizations/interpretations/charts/pie_chart.html

[10] Gráfica de barras, UTCV Calidad en el Mantenimiento, 2004, https://sites.google.com/site/utcvcalidadenelmantenimiento/estadistica-y-probabilidad/2-1-3-graficas/2-1-3-1-grafica-de-barras

[11] Tableau public, 2019 Tableau Software, https://public.tableau.com/en-us/s/

[12] StatPlanet | StatSilk, 2019, StatSilk, https://www.statsilk.com/software/statplanet

[13] Qué son los RDD, OpenWebinars, 3 de julio 2018, https://openwebinars.net/blog/que-son-los-rdd/

[14] Apache Spark: qué es y cómo funciona, Geeky Theory, 2019, https://geekytheory.com/apache-spark-que-es-y-como-funciona

[15] Iterative operations with Spark, ResearchGate, https://www.researchgate.net/figure/Iterative-operations-on-Spark-RDD_fig11_326572328

[16] Interactive operations with Spark, ResearchGate, https://www.researchgate.net/figure/Interactive-operations-on-Spark-RDD_fig12_326572328

[17] Jacek Laskowski, https://jaceklaskowski.gitbooks.io/mastering-apache-spark/spark-overview.html

[18] Numpy, Numpy.org, https://www.numpy.org/

[19] Python Data Analysys Library, pandas, https://pandas.pydata.org/

[20] PySpark master documentation, module code, Apache Spark, https://spark.apache.org/docs/2.3.0/api/python/_modules/pyspark/conf.html

[21] The jQuery Foundation, jQuery 2019, "What is jQuery?", https://jquery.com

[22] Bootstrap, https://getbootstrap.com/

[23] D3-composite-projections, https://geoexamples.com/d3-composite-projections/

[24] TopoJSON, andrewharvey and mbostock, https://github.com/topojson/topojson

[25] Font Awesome, https://fontawesome.com/