
GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

# HERRAMIENTA DE VISUALIZACIÓN MULTIDIMENSIONAL BIG DATA DE TRAYECTOS DE LA PLATAFORMA DE BICI COMPARTIDA BICIMAD

Autor: Cayetano Valero Amores

Director: David Contreras Bárcena

Madrid

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

Herramienta de visualización multidimensional Big Data de trayectos de la plataforma de bici compartida BiciMad

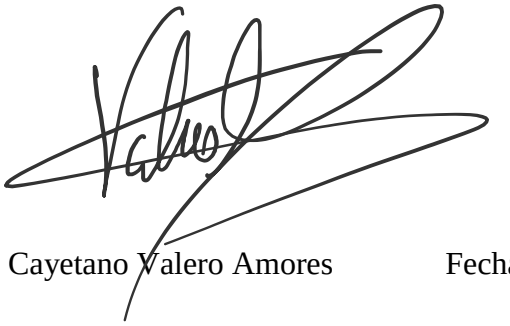
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

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

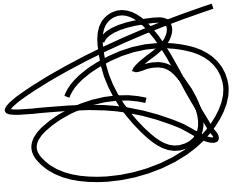
tomada de otros documentos está debidamente referenciada.



Fdo.: Cayetano Valero Amores Fecha: 17/ 06/ 2018

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO



Fdo.: David Contreras Bárcena Fecha: 17/ 06/ 2018

**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. Cayetano Valero Amores DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Herramienta de visualización multidimensional Big Data de trayectos de la plataforma de bici compartida BiciMad, 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.

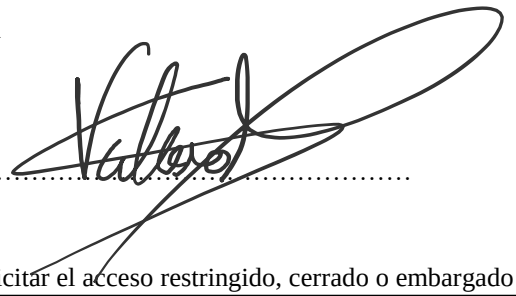
***6º. Fines y funcionamiento del Repositorio Institucional.***

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 17 de junio de 2018

**ACEPTA**



Fdo……

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

# HERRAMIENTA DE VISUALIZACIÓN MULTIDIMENSIONAL BIG DATA DE TRAYECTOS DE LA PLATAFORMA DE BICI COMPARTIDA BICIMAD

Autor: Cayetano Valero Amores

Director: David Contreras Bárcena

Madrid

# HERRAMIENTA DE VISUALIZACIÓN MULTIDIMENSIONAL BIG DATA DE TRAYECTOS DE LA PLATAFORMA DE BICI COMPARTIDA BICIMAD

**Autor: Valero Amores, Cayetano.**
Director: Contreras Bárcena, David.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

## RESUMEN DEL PROYECTO

El proyecto ha consistido en el desarrollo de una herramienta de visualización multidimensional basada en los datos de uso del servicio de alquiler de bicicletas eléctricas de Madrid con el objetivo de ofrecer una solución que ayude a visualizar y comprender los datos y resultados obtenidos a través de la interacción del usuario con diferentes parámetros.

**Palabras clave**: Big Data, Movilidad, Visualización, Multidimensional, Interactivo.

### 1. Introducción

Actualmente, la recolección y el procesado de volúmenes cada vez mayores de datos en gran cantidad de aspectos de la sociedad está requiriendo de técnicas de visualización y análisis más complejas debido entre otros a los problemas de escalabilidad que supone este nuevo escenario. Se necesitan técnicas adecuadas para una representación eficaz de los datos con los que se trabaja ya que ello permitirá comprender la información con la que trabajamos, comparar datos e identificar patrones y relaciones entre ellos [1].

Uno de los aspectos más importantes en las ciudades es la movilidad. Cómo nos desplazamos, por dónde lo hacemos y restricciones al tráfico en determinadas zonas son algunos de los factores primarios que influyen en la movilidad en una ciudad. Además, encontramos otros factores ajenos al control humano que también influyen en la movilidad, como son el clima o posibles accidentes y hechos que tienen consecuencias directas en la movilidad sin estar necesariamente relacionados con ella como son eventos de carácter cultural o deportivo, como eventos musicales o ferias o partidos de fútbol que concentran a miles de personas.

Gracias a las nuevas herramientas que ofrece el Big Data, la movilidad en la ciudad ha tomado una nueva dimensión. Este nuevo escenario ha cambiado tanto la forma de recolectar datos como su análisis y su representación, permitiendo estudiar tendencias y patrones que nos servirán para mejorar el servicio ofrecido a los ciudadanos y resolver problemas que ni si quiera se habían planteado.

Junto al uso de nuevas tecnologías, estamos experimentado un cambio en la forma en la que las personas se desplazan por la ciudad ya sea por trabajo o entretenimiento. En los últimos tiempos han aparecido nuevos servicios a disposición de los ciudadanos, como los novedosos alquileres de vehículos eléctricos sin conductor por cortos periodos de tiempo, como coches o motocicletas y servicios más consolidados como el alquiler de bicicletas eléctricas.

Desde hace 3 años, el Ayuntamiento de Madrid ha pasado a recoger rigurosamente datos sobre el uso de este servicio, incluyendo información sobre los trayectos que las personas realizan. Junto con los datos sobre trayectos de los alquileres de bicis, encontramos disponibles datos sobre accidentes en la ciudad de Madrid, con información detallada de la localización del suceso, día de la semana y fecha del accidente y tipos de vehículos implicados, estado de la calzada en el momento del accidente, etc.

En cuanto a estudios realizados sobre flujos de personas haciendo uso de técnicas Big Data encontramos dos que aportan resultados que servirán como punto de partida al proyecto. El primero de ellos centrado en el servicio de alquiler de bicicletas eléctricas de la ciudad de Madrid, ha sido desarrollado por el equipo de investigación de la Universidad Complutense de Madrid [2] y muestra 250 000 rutas en un mapa dinámico con el objetivo de estudiar el flujo de usuarios en la ciudad. En el segundo estudio, "Making Sense: An Innovative Data Visualization Application Utilized Via Mobile Platform", realizado por Jiayan Gu, Sebastian Mackin y Yongjun Zheng [3], se desarrollan visualizaciones multidimensionales sobre el flujo de pasajeros del metro de Shanghai atendiendo a distintos parámetros.

Ambos estudios presentan la idea de la visualización dinámica y multidimensional con diferentes parámetros que va a llevarse a cabo.

La motivación del proyecto es aprovechar todos los datos disponibles sobre trayectos de usuarios del servicio de alquiler de bicicletas eléctricas, accidentes en la ciudad de Madrid y otras fuentes de datos externas para desarrollar una herramienta de visualización en forma de aplicación web que pretende contribuir a mejorar la movilidad en la ciudad a través de la implementación de herramientas de análisis que permitan extraer conclusiones relevantes sobre el servicio y el uso que se hace de él.

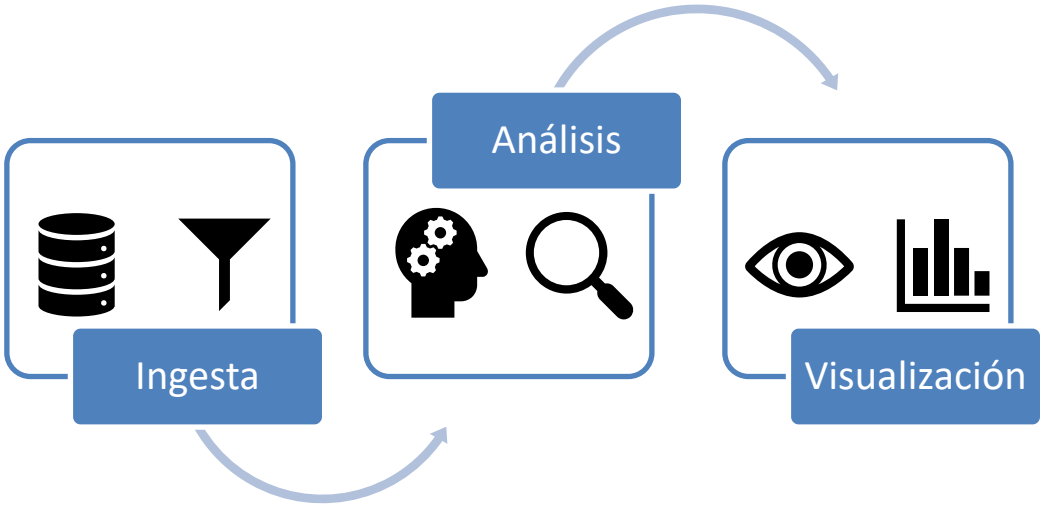
## 2. Definición del proyecto

El proyecto ha consistido en el desarrollo de una herramienta de visualización multidimensional dinámica basada en los datos de uso del servicio de alquiler de bicicletas eléctricas de Madrid. El trabajo se ha llevado a cabo en dos fases fundamentales. La primera de ellas centrada en un estudio de los datos de uso del servicio de alquiler disponibles, así como de otros conjuntos de datos relevantes, con el objetivo de comprobar cuales serán útiles para el desarrollo posterior, y una segunda fase centrada en la creación de la propia herramienta con un doble objetivo: la representación dinámica de los datos de uso del servicio de alquiler de bicicletas eléctricas (cobertura y distribución de las estaciones, visualización de los trayectos de los usuarios, etc.) y la integración de diferentes opciones que permitan filtrar los resultados y personalizar la visualización teniendo en cuenta los parámetros que un usuario de la herramienta seleccione.

## 3. Metodología

Antes de comenzar a describir el trabajo realizado, conviene destacar la estructura general del proyecto Big Data desarrollado.

*Figura a. Estructura general del proyecto Big Data*




Como se puede observar en la Figura a, el proyecto Big Data cuenta con tres fases fundamentales, cuyos resultados servirán como punto de partida a la siguiente fase:

- Fase de Ingesta de datos. Fase inicial del proyecto, cuyos objetivos son la obtención, el estudio y el filtrado de datos para su uso en el análisis.
- Fase de Análisis. Fase central del proyecto, donde se usarán los datos obtenidos en la ingesta para llevar a cabo el estudio y obtener las conclusiones.
- Fase de Visualización. Fase final del proyecto, que recogerá y adaptará el trabajo realizado en las fases posteriores para presentar la visualización y facilitar la comprensión de los datos y las conclusiones del análisis.

El proyecto representa la fase de visualización del desarrollo del proyecto Big Data, que ha sido resultado de la realización de tres proyectos independientes, llevados a cabo cada uno por un alumno y correspondientes a cada una de las fases.

Antes de comenzar con el trabajo individual en cada una de las fases, se realizó de forma conjunta una aproximación inicial a los *datasets*: Una vez se seleccionaron cuáles iban a ser relevantes para el proyecto, se llevó a cabo un estudio en R de los datos, con el fin de comprobar la estructura y el contenido de los conjuntos seleccionados. R permite una aproximación rápida y eficaz a los datos trabajando en un solo equipo, siempre que el tamaño del conjunto no sea muy grande. Como los datos de trayectos suponen un gran volumen de información si se tienen en cuenta los datos completos de los años 2017 y 2018 (JSON de 8 GB), ha sido necesario un cambio en la herramienta de trabajo para continuar con el estudio,

incorporando los datos de todos los años. Para el estudio completo de los datos se ha usado PySpark, librería de Python para el framework de procesamiento de datos Spark, apoyado en la potencia del clúster Big Data disponible en la Universidad. Una vez finalizado el estudio inicial del conjunto de los datos, cada uno de los alumnos se ha centrado en el proyecto individual que le corresponde.

Centrado el desarrollo en la parte de visualización, el primer paso ha sido el estudio de las herramientas disponibles para visualización dinámica de datos. De entre las disponibles, se seleccionó la librería de JavaScript D3, que permite la manipulación de datos añadiéndolos a elementos específicos dentro de una página web. Esto se consigue a través de la manipulación del *DOM* (estructura de datos donde están definidos los componentes de una página web), con diversas funciones. Es una herramienta compleja, pero que permite un gran nivel de personalización en las representaciones de datos y además ofrece herramientas para leer y cargar conjuntos de datos externos, que serán muy útiles para incorporar los resultados de las fases de ingesta y análisis para poder visualizarlos. El uso de esta herramienta ha requerido de un estudio exhaustivo de su funcionamiento, a través de libros [4] o la documentación de la propia librería [5].

Además de la librería D3, debido al carácter de los datos, se ha necesitado de otra herramienta que permita la manipulación y representación de datos geográficos de forma interactiva. Para ello se ha usado la librería de JavaScript Leaflet [6], que permite insertar mapas interactivos en aplicaciones web y construir capas de diferentes tipos para visualizarlas sobre ellos. Leaflet es una librería muy completa de trabajo con mapas que incorpora módulos para desarrollar diferentes componentes, como leyendas, cuadros de información, líneas o polígonos, que se ajustan muy bien al trabajo de visualización que se va a realizar.

Junto con Leaflet y D3, que cubren la parte de tratamiento y visualización de los datos, se han requerido dos herramientas más, para dar forma a la interfaz completa de la aplicación donde se integran los componentes anteriores y a la lógica del servidor que se encargará de la gestión interna de los conjuntos de datos y su paso a la aplicación. Para el desarrollo de la interfaz se ha usado el framework Bootstrap [7], que ofrece una gran cantidad de componentes y herramientas para trabajar con la distribución de elementos en una aplicación web. Para la programación del servidor se ha optado por el framework Flask [8], que permite programar en Python la lógica del servidor, que se encarga esencialmente en el proyecto de

proporcionar los archivos que contienen los datos de trayectos para poder leerlos desde la aplicación y visualizarlos. Además de la lógica del servidor, Flask proporciona la capacidad de incluir código Python a través de plantillas en la aplicación web, que ha sido útil para actualizar dinámicamente algunos de los contenidos de la aplicación, como las listas que hacen referencia a los días del año con datos, que se irán actualizando de forma automática a medida que se incluyan nuevos conjuntos de datos en el futuro.

Una vez centradas todas las herramientas que se van a usar, se procede al desarrollo. El proceso de creación ha consistido en el desarrollo de una aplicación inicial básica, en la que se incluye la visualización de un mapa, las capas y el control de estas, a la que se han ido añadiendo las diferentes opciones a través de sucesivas iteraciones, cada una de ellas centrada en resolver un objetivo del proyecto. Así, después del desarrollo de la interfaz básica con el mapa, el siguiente paso ha sido la inclusión de los componentes centrados en la lectura e incorporación de los datos externos a la aplicación para proceder a su posterior visualización. Una vez se incorporaron todos los datos externos que se van a usar y se ha programado la lógica de funcionamiento interna, se procede a dar forma a la aplicación haciendo uso de Bootstrap. Para proporcionar el dinamismo y la interactividad en la visualización que se buscan en el proyecto, se ha optado por desarrollar un control fácil de usar, que permita una interacción sencilla con los conjuntos de datos. Este control está basado en un esquema similar a la reproducción de elementos multimedia, como audio o vídeo, en el que se selecciona el punto inicial de la reproducción y activando un botón se sucede la animación de los trayectos en el mapa. El paso final ha sido la integración de la aplicación con el servidor usando Flask, para permitir el acceso a los diferentes archivos de trayectos y la actualización dinámica de los componentes, que cambiarán conforme se añadan nuevos ficheros de datos.

## 4. Resultados

Del estudio inicial de los datos, se obtuvieron resultados sobre la estructura y el contenido de estos (Figura b), para ajustar el método de trabajo adecuado para ellos en las fases posteriores, además de construir diversas gráficas para facilitar la comprensión de los datos y estudiar el comportamiento general de los usuarios, como el tiempo que dedican a viajar (Figura c) o las veces que una misma persona hace uso del servicio en un día (Figura d).

*Figura b. Estructura de los datos de trayectos*

```
root
 |-- _id: struct (nullable = true)
 |    |-- $oid: string (nullable = true)
 |-- ageRange: long (nullable = true)
 |-- idplug_base: long (nullable = true)
 |-- idplug_station: long (nullable = true)
 |-- idunplug_base: long (nullable = true)
 |-- idunplug_station: long (nullable = true)
 |-- track: struct (nullable = true)
 |    |-- features: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- geometry: struct (nullable = true)
 |    |    |    |    |-- coordinates: array (nullable = true)
 |    |    |    |    |    |-- element: double (containsNull = true)
 |    |    |    |    |-- type: string (nullable = true)
 |    |    |    |-- properties: struct (nullable = true)
 |    |    |    |    |-- secondsfromstart: long (nullable = true)
 |    |    |    |    |-- speed: double (nullable = true)
 |    |    |    |    |-- var: string (nullable = true)
 |    |    |    |-- type: string (nullable = true)
 |    |-- type: string (nullable = true)
 |-- travel_time: long (nullable = true)
 |-- unplug_hourTime: struct (nullable = true)
 |    |-- $date: string (nullable = true)
 |-- user_day_code: string (nullable = true)
 |-- user_type: long (nullable = true)
 |-- zip_code: string (nullable = true
)
```

*Figura c. Estudio del tiempo de viaje*

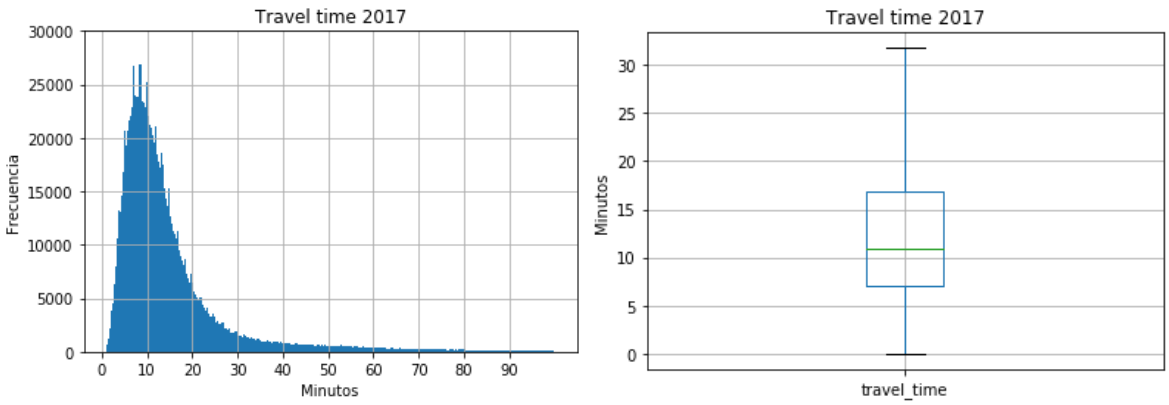
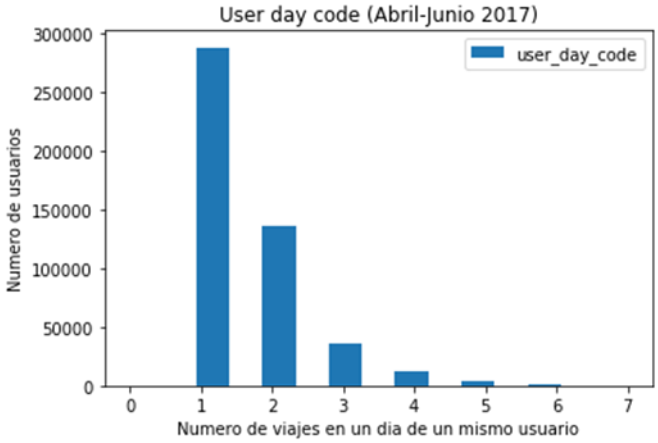


*Figura d. Estudio del código de usuario diario*




Centrando los resultados en la herramienta de visualización, como se puede observar en la vista general (Figura e), se ha optado por un diseño limpio y claro, en el que el mapa es el principal protagonista, y todas las opciones de filtrado, personalización y resultados se encuentran situadas en sus respectivas tarjetas temáticas, que hacen más fácil la visualización y el control por parte del usuario del contenido que desea ver en cada momento.

La aplicación cuenta con opciones de visualización separadas en dos grupos, que pueden seleccionarse en el selector de capas en la esquina superior derecha del mapa: por un lado, las opciones relativas al servicio, como la distribución de las estaciones (Figura f) y, por otro lado, las opciones de visualización de los trayectos realizados por los usuarios. Cuando se seleccione la visualización de trayectos, se mostrará una vista similar a la ofrecida en la Figura g, aportando información sobre el volumen de trayectos en el mapa en el margen temporal seleccionado, y una vista en forma de gráfico de los trayectos ocurridos por hora a lo largo del día seleccionado.

*Figura e. Vista inicial de la herramienta de visualización*

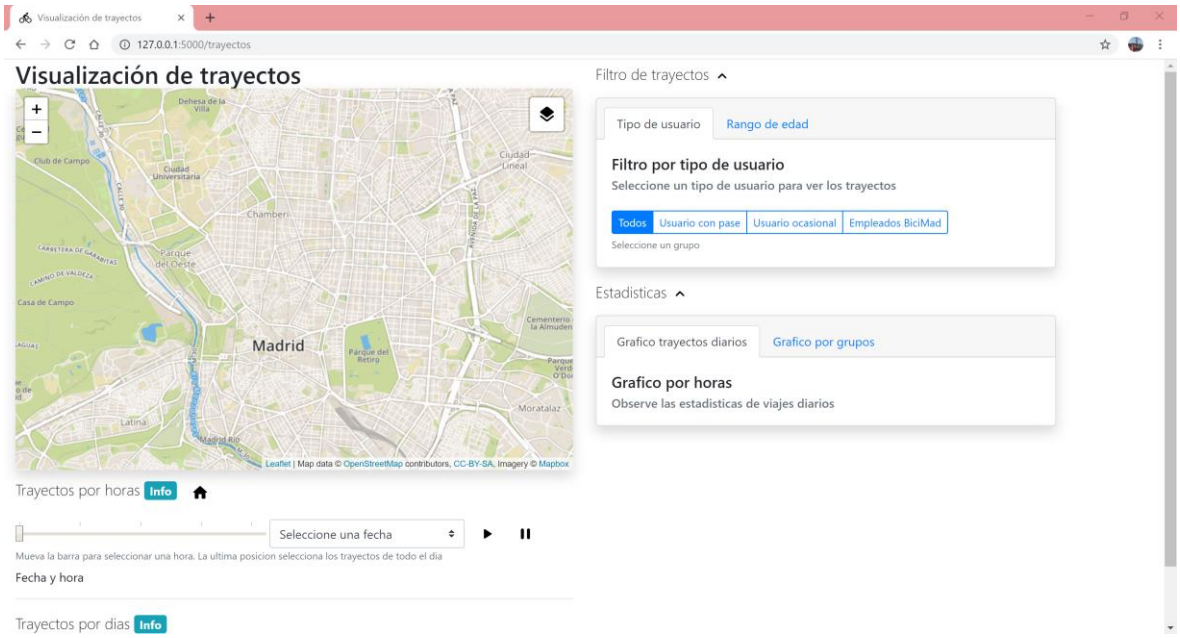



*Figura f. Vista de la cobertura del servicio*

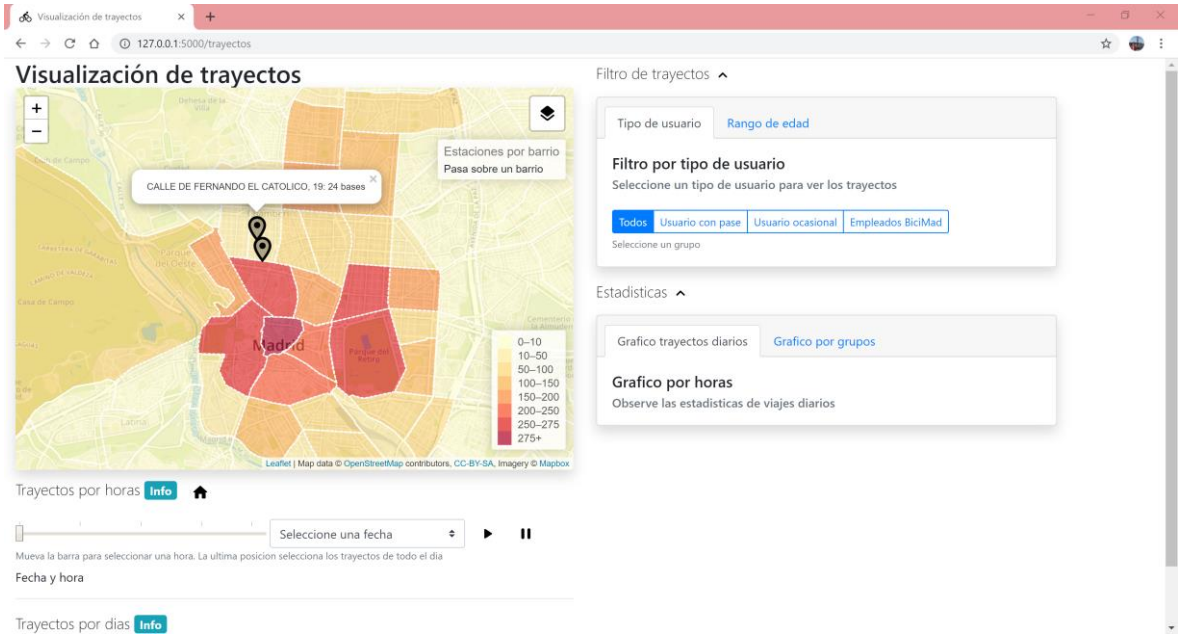
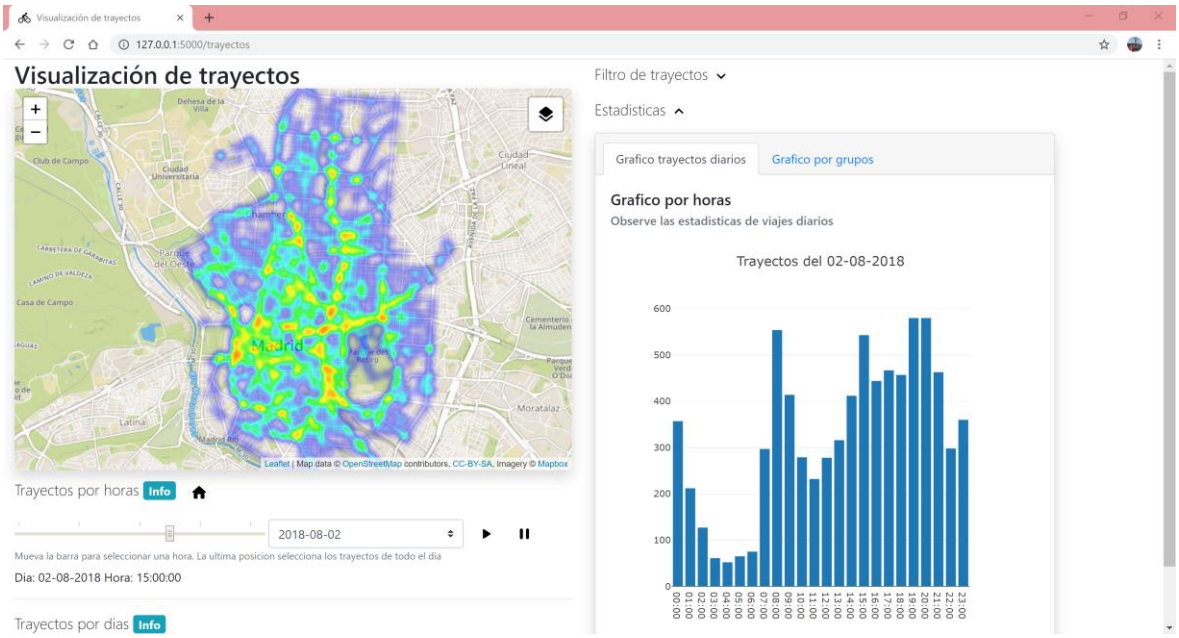


*Figura g. Vista de los trayectos*




## 5. Conclusiones

El estudio inicial de los datos ha permitido conocer la estructura de estos y adaptar la planificación y el desarrollo posteriores. Contando con una gran cantidad de herramientas de visualización en el mercado, ha sido importante el estudio llevado a cabo, gracias además a las conclusiones del estudio inicial de los datos, que han permitido centrar los resultados que se quieren obtener y seleccionar la herramienta adecuada para el desarrollo. Uno de los objetivos principales del proyecto, además de la visualización de los datos, era dotar a la herramienta de un carácter dinámico e interactivo, que permita observar la evolución en el tiempo de los datos. La inclusión de una interfaz de control clara y sencilla ha sido clave en este aspecto, proporcionando de forma rápida todos los controles necesarios, tanto de reproducción del contenido como su filtrado y clasificación.

## 6. Referencias


[1] A. Imawan, F. Putri y J. Kwon, «TiQ: A Timeline Query Processing System over Road Traffic Data,» de *IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, Chengdu, 2015.

[2] J. Huete, «La EMT mejorará su movilidad urbana gracias al Big Data de Moovit,» 30 Octubre 2018. [En línea]. Available: https://www.innovaspain.com/emt-movilidad-urbana-big-data-moovit/.

[3] H. Zhiyuan, Z. Liang, X. Ruihua y Z. Feng, «Application of big data visualization in passenger flow analysis of Shanghai Metro network,» de *2nd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, Singapur, 2017.

[4] S. Murray, Interactive Data Visualization for the Web. An Introduction to Designing with D3, Segunda ed., M. Foley, Ed., O'Reilly, 2017.

[5] M. Bostock, «D3 Documentation,» 22 Marzo 2018. [En línea]. Available: https://github.com/d3/d3/wiki.

[6] V. Agafonkin, «Leaflet,» [En línea]. Available: https://leafletjs.com/.

[7] Bootstrap Team, «Bootstrap,» [En línea]. Available: https://getbootstrap.com/.

[8] A. Ronacher, «Flask,» [En línea]. Available: http://flask.pocoo.org/.

# TOOL FOR MULTIDIMENSIONAL BIG DATA VISUALIZATION OF USER ROUTES FROM ELECTRIC BYCICLES SHARING SERVICE BICIMAD

**Author: Valero Amores, Cayetano.**
Supervisor: Contreras Bárcena, David.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

## ABSTRACT

The project consisted in the development of a multidimensional visualization tool based on the data of the users of the electric bicycle rental service in Madrid with the aim of offering a solution that helps to visualize and understand the data and results obtained through the user interaction with different parameters.

**Keywords**: Big Data, Mobility, Visualization, Multidimensional, Interactive.

### 1. Introduction

Currently, the collection and processing of increasing volumes of data in many aspects of society is requiring more complex visualization and analysis techniques due, among others, to the scalability problems that this new scenario entails. Appropriate techniques are needed for an effective representation of the data, since this will allow us to understand the information with which we work, compare data and identify patterns and relationships between them [1].

One of the most important aspects in cities is mobility. How we move, where we do it and restrictions on traffic in certain areas are some of the primary factors that influence mobility in a city. In addition, we find other factors beyond human control that also influence mobility, such as weather or possible accidents and events that have direct consequences on mobility without necessarily being related to it, such as cultural or sports events that concentrate thousands of people.

Thanks to the new tools offered by Big Data, mobility in the city, as well as many other aspects, take on a new dimension, both in the way of collecting data and in the way of analyzing and representing them, allowing us to study trends and patterns that will serve to improve the service offered to citizens, as well as discover and solve problems that had not even been raised.

Along with the use of new technologies, we are experiencing a change in the way people move around the city, whether for work or entertainment. In recent times, new services have been available to citizens, such as the rents of electric vehicles without a driver for short periods of time and more consolidated services such as the rental of electric bicycles.

For 3 years, the Madrid City Council has been collecting rigorously data on the use of this service, including information on the rides that people make. Along with the data of bike rentals, data of accidents in the city of Madrid are available, with detailed information on the location of the event, the day of the week and the date of the accident and the types of vehicles involved, the condition of the road at the time of the accident, etc.

As for studies conducted on the flow of people using Big Data techniques, we found, focused on the electric bicycle rental service in the city of Madrid, one developed by the research team of the Complutense University of Madrid [2] that shows 250,000 routes in a dynamic map with the aim of studying the flow of users in the city. In the study "Making Sense: An Innovative Data Visualization Application Utilized Via Mobile Platform" carried out by Jiayan Gu, Sebastian Mackin and Yongjun Zheng [3] multidimensional visualizations are developed to understand the flow of passengers of the Shanghai subway considering different parameters.

Both studies present solutions that can be a starting point for the project since they present the idea of multidimensional visualization with different parameters that will be carried out.

The motivation of the project is to take advantage of all available data on user trips of the electric bicycle rental service, accidents in the city of Madrid and other external data sources to develop a visualization tool in the form of a web application that aims to contribute to improve mobility in the city through the implementation of analysis tools that allow drawing relevant conclusions about the service and the use made of it.

## 2. Definition of the project

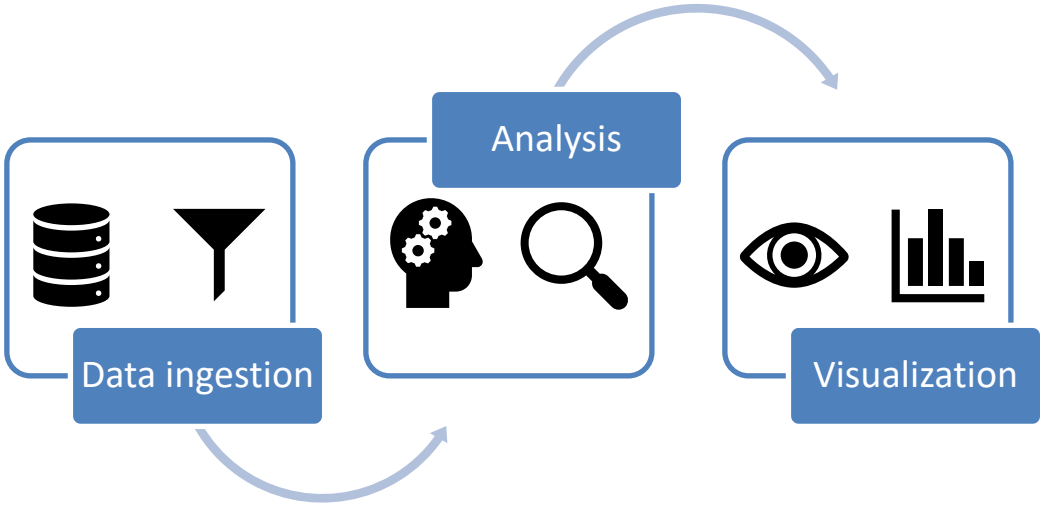
The project has consisted in the development of a dynamic multidimensional visualization tool based on the data provided by usage of the electric bicycle rental service in Madrid. The work has been carried out in two fundamental phases. The first of them focused on the study of the bicycle rental service usage data, as well as of other relevant data sets, with the objective of obtaining results about the structure and content of the data to be used in the subsequent development, and a second phase focused on the creation of the tool

itself with a double objective: the dynamic representation of the data of the electric bicycles rental service (coverage and distribution of the stations, visualization of the trips of the users, etc.) and the integration of different options that allow a user to filter the results and customize the visualization taking into account the parameters that a user of the tool selects.

## 3. Description of work done

Before describing the work done, it is important to highlight the general structure of a Big Data project.

*Figure a. General structure of the Big Data project*




As can be seen in Figure a, the Big Data project has three fundamental phases, whose results will serve as a starting point to the next phase:

- Phase of Data Ingestion. Initial phase of the project, whose objectives are obtaining, studying and filtering data for use it in the analysis.
- Phase of Analysis. Central phase of the project, where the data obtained in the ingestion will be used to carry out the study and obtain the conclusions.
- Phase of Visualization. Final phase of the project, which will collect and adapt the work done in the subsequent phases to present the visualization and facilitate the understanding of the data and the conclusions of the analysis.

The project represents the visualization phase of the development of the Big Data project, which has been the result of the development of three independent projects, each carried out by a student and corresponding to each of the phases.

Before beginning with the individual work in each of the phases, an initial approximation to the datasets was carried out jointly: Once the relevant datasets for the project were selected, a study of the data was carried out in R, in order to check the structure and content of the selected sets. R allows a fast and efficient approach to the data working in a single computer if the size of the set is not very large. As the service usage data represents a great volume of information if the complete data of the years 2017 and 2018 (JSON of 8 GB) are taken into account, it has been necessary a change in the working tool to continue with the study, incorporating the data of all years. For the complete study of the data, PySpark, a Python library for the Spark data processing framework, has been used, supported by the efficiency of the Big Data cluster available at the University. Once the initial study of the data set has been completed, each of the students has focused on the individual project that corresponds to them.

Focusing the development on the visualization part, the first step has been the study of the available tools for dynamic visualization of data. Among the available ones, the JavaScript library D3 was selected, which allows the manipulation of data by adding them to specific elements within a web page. This is achieved through the manipulation of the DOM (data structure where the components of a web page are defined), with various functions. It is a complex tool, but it allows a great level of customization in data representations and also offers tools to read and load external data sets, which will be very useful to incorporate the results of the phases of data ingestion and analysis to be able to visualize them. The use of this tool has required an exhaustive study of its functioning, through books [4] or the documentation of the library itself [5].

In addition to the D3 library, due to the nature of the data, another tool that allows the manipulation and representation of geographic data in an interactive way has been needed. For this the Leaflet JavaScript library [6] has been used, which allows inserting interactive maps in web applications and building layers of different types to visualize them on top of the map. Leaflet is a complete library for working with maps that includes modules to develop different components, such as legends, information tables, lines or polygons, which fit very well to the visualization work that is going to be done.

Together with Leaflet and D3, which cover the data processing and visualization part, two more tools have been required, to give shape to the complete interface of the application where the previous components are integrated and to the logic of the server that will be responsible for the internal management of the data sets and their passage to the application. For the development of the interface, the Bootstrap framework [7] has been used, which offers a large number of components and tools to work with the distribution of elements in a web application. For the programming of the server, the Flask framework [8] has been chosen, which allows programming the server logic in Python, which is essentially responsible for providing the files containing the bike rental service usage data in order to read them from the application and visualize them. In addition to the server logic, Flask provides the ability to include Python code through templates in the web application, which has been useful for dynamically updating some of the application's contents, such as the lists referring to the days of the year with data, which will be updated automatically as new datasets are included in the future.

Once all the tools that are going to be used are defined, we proceed to describe the development. The creation process consisted in the development of a basic initial application, which includes the visualization of a map, the layers and the control of these, to which the different options have been added through successive iterations, each of them focused on solving a project objective. Thus, after the development of the basic interface with the map, the next step has been the inclusion of the components centered on the reading and incorporation of the external data to the application to proceed with its visualization. Once all the external data that will be used has been incorporated and the internal operating logic has been programmed, the application is shaped using Bootstrap. To provide the dynamism and interactivity in the visualization that is sought in the project, we have opted to develop an easy-to-use control that allows straightforward interaction with the data sets. This control is based on a scheme like the reproduction of multimedia elements, such as audio or video, in which the starting point of the reproduction is selected and activating a button, the animation of the routes on the map takes place. The final step has been the integration of the application with the server using Flask, to allow access to the different files and the dynamic update of the components, which will change as new data files are added.

## 4. Results

From the initial study of the data, results were obtained on the structure and content of these (Figure b), to adjust the procedures suitable for them in later phases, in addition to various graphs to facilitate the understanding of the data and study the general behavior of users, such as the time they spend traveling (Figure c) or the times that the same person has used the service in one day (Figure d).

*Figure b. Structure of the data*

```
root
 |-- _id: struct (nullable = true)
 |     |-- $oid: string (nullable = true)
 |-- ageRange: long (nullable = true)
 |-- idplug_base: long (nullable = true)
 |-- idplug_station: long (nullable = true)
 |-- idunplug_base: long (nullable = true)
 |-- idunplug_station: long (nullable = true)
 |-- track: struct (nullable = true)
 |     |-- features: array (nullable = true)
 |     |     |-- element: struct (containsNull = true)
 |     |     |     |-- geometry: struct (nullable = true)
 |     |     |     |     |-- coordinates: array (nullable = true)
 |     |     |     |     |     |-- element: double (containsNull = true)
 |     |     |     |     |-- type: string (nullable = true)
 |     |     |     |-- properties: struct (nullable = true)
 |     |     |     |     |-- secondsfromstart: long (nullable = true)
 |     |     |     |     |-- speed: double (nullable = true)
 |     |     |     |     |-- var: string (nullable = true)
 |     |     |     |-- type: string (nullable = true)
 |     |-- type: string (nullable = true)
 |-- travel_time: long (nullable = true)
 |-- unplug_hourTime: struct (nullable = true)
 |     |-- $date: string (nullable = true)
 |-- user_day_code: string (nullable = true)
 |-- user_type: long (nullable = true)
 |-- zip_code: string (nullable = true
)
```

*Figure c. Travel time study*

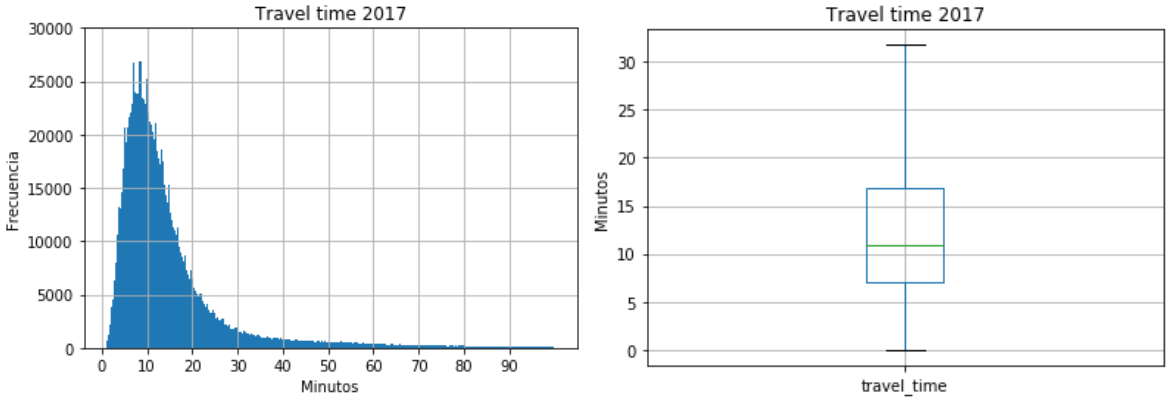
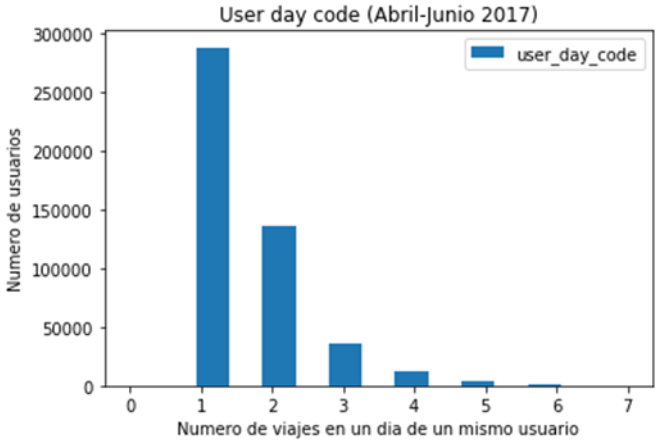


*Figure d. User day code study*




Focusing the results on the visualization tool, as can be seen in the overview (Figure e), we have opted for a clear design, in which the map is the main component, and all filtering options, personalization and results are located in their respective cards, which make easier the visualization and the control by the user of the content that he want to see at every moment.

The application has display options separated into two groups, which can be selected in the layer selector in the upper right corner of the map. The first group includes the options for viewing the service characteristics, like the coverage of the service (Figure f). In the other group, there are options available for viewing the trips made by users. When the trip view is selected, a display similar to the one shown in Figure g will be shown, providing information on the volume of trips on the map in the selected time frame, and a graphical view of the journeys that occur per hour throughout the selected day.

*Figure e. Initial view of the visualization tool*

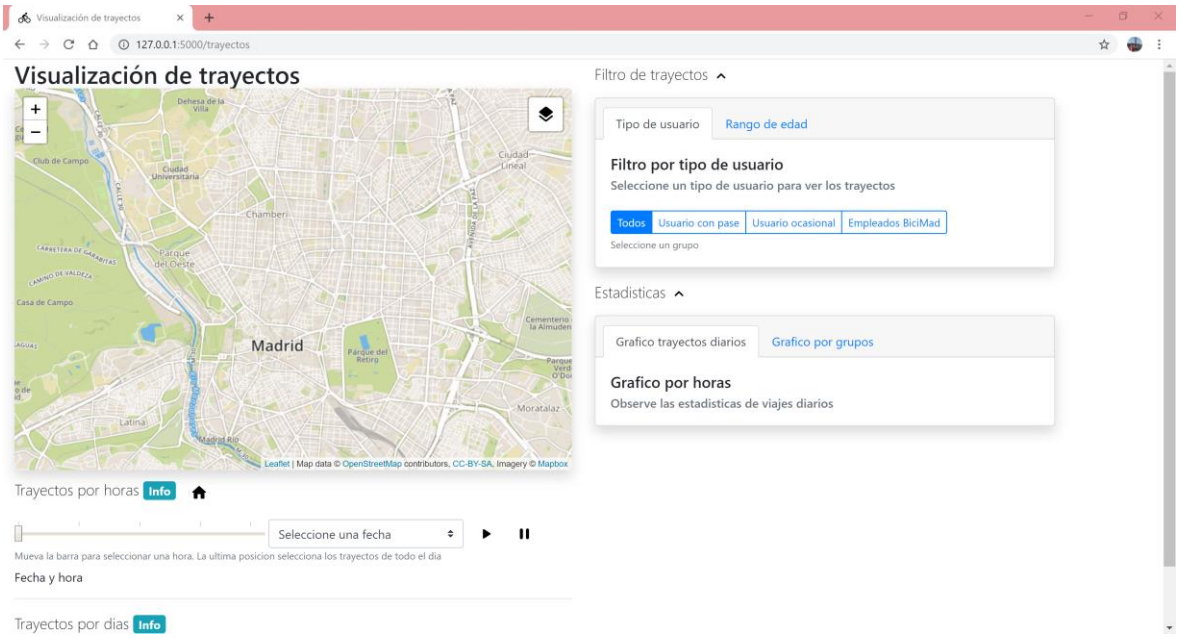



*Figure f. View of the coverage information disply*

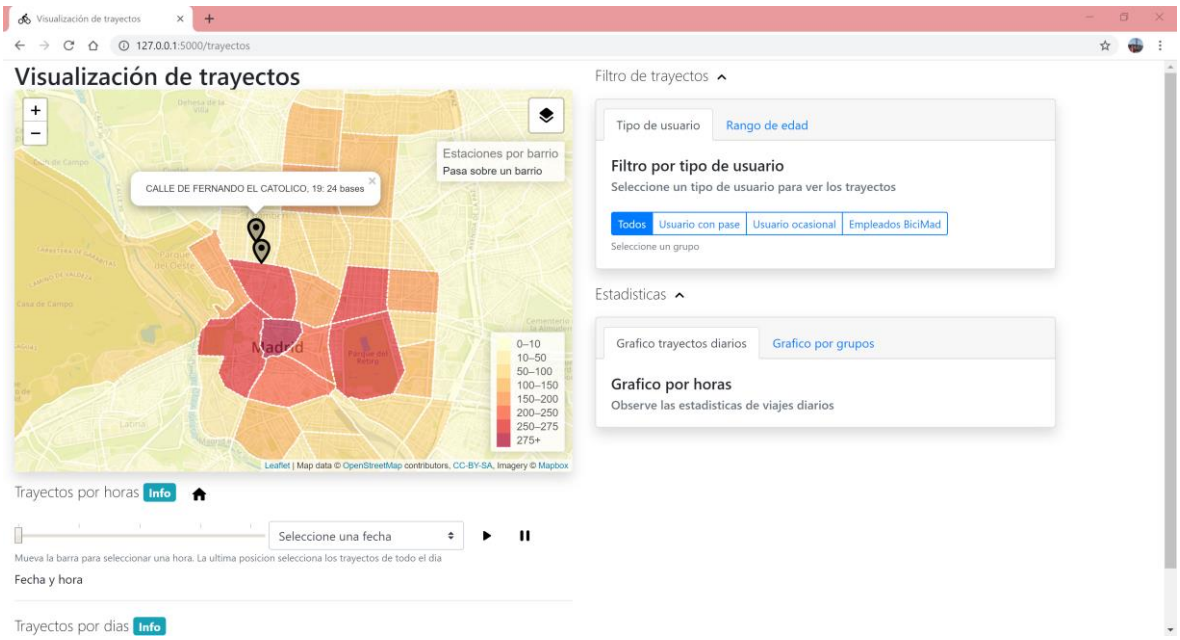
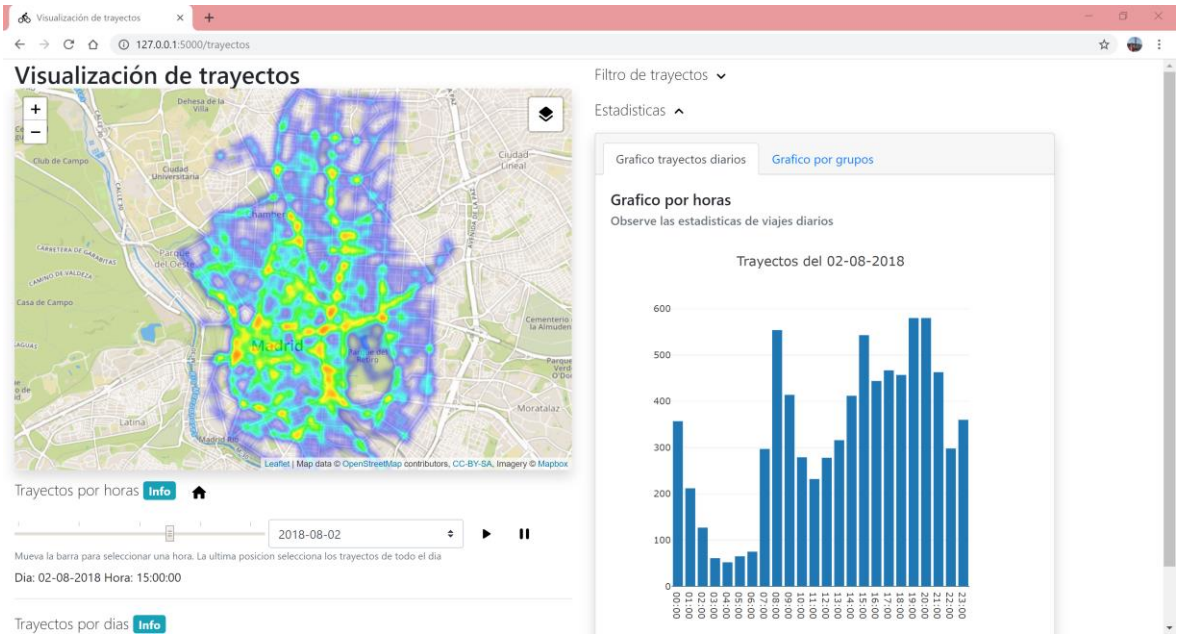


*Figure g. View of the trips information display*




## 5. Conclusion

The initial analysis of the data allowed to study the structure of the information available and adapt the subsequent planning and development. With a large number of visualization tools in the market, the study carried out to select appropriate tools has been important, thanks also to the conclusions of the initial study of the data, which have allowed us to focus the results we want to obtain and select the appropriate tool to the development. One of the main objectives of the project, in addition to the visualization of the data, was to provide an application with dynamic character and interactivity features that allows observing the evolution over time of the data. The inclusion of a easy to use control interface has been key in this aspect, providing all the necessary controls in a user-friendly menu, both for content playback, filtering and classification.

## 6. References


[1] A. Imawan, F. Putri y J. Kwon, «TiQ: A Timeline Query Processing System over Road Traffic Data,» de *IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, Chengdu, 2015.

[2] J. Huete, «La EMT mejorará su movilidad urbana gracias al Big Data de Moovit,» 30 Octubre 2018. [En línea]. Available: https://www.innovaspain.com/emt-movilidad-urbana-big-data-moovit/.

[3] H. Zhiyuan, Z. Liang, X. Ruihua y Z. Feng, «Application of big data visualization in passenger flow analysis of Shanghai Metro network,» de *2nd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, Singapur, 2017.

[4] S. Murray, Interactive Data Visualization for the Web. An Introduction to Designing with D3, Segunda ed., M. Foley, Ed., O'Reilly, 2017.

[5] M. Bostock, «D3 Documentation,» 22 Marzo 2018. [En línea]. Available: https://github.com/d3/d3/wiki.

[6] V. Agafonkin, «Leaflet,» [En línea]. Available: https://leafletjs.com/.

[7] Bootstrap Team, «Bootstrap,» [En línea]. Available: https://getbootstrap.com/.

[8] A. Ronacher, «Flask,» [En línea]. Available: http://flask.pocoo.org/.

# *Índice de la memoria*

# *Índice de figuras*

# Capítulo 1. INTRODUCCIÓN

Actualmente, la recolección y el procesado de volúmenes cada vez mayores de datos en gran cantidad de aspectos de la sociedad está requiriendo de técnicas de visualización y análisis más complejas debido entre otros a los problemas de escalabilidad que supone este nuevo escenario. Se necesitan técnicas adecuadas para una representación eficaz de los datos con los que se trabaja ya que ello permitirá comprender la información con la que trabajamos, comparar datos e identificar patrones y relaciones entre ellos [1]. Algunos de los mayores retos que surgen al hacer frente a desarrollos basados en grandes conjuntos de datos son, además de la escalabilidad mencionada anteriormente, junto a la obtención, análisis y visualización de los datos, la verificación de los mismos y el tratamiento con fuentes de datos muy diversas: Es difícil verificar en un gran conjunto de datos si la información que se recoge es correcta y tratar con datos de diferente naturaleza hace más complejo el proceso de adaptación de los desarrollos.

Uno de los aspectos más importantes en las ciudades es la movilidad. Cómo nos desplazamos, por dónde lo hacemos y restricciones al tráfico en determinadas zonas son algunos de los factores primarios que influyen en la movilidad en una ciudad. Además, encontramos otros factores ajenos al control humano que también influyen en la movilidad, como son el clima o posibles accidentes y hechos que tienen consecuencias directas en la movilidad sin estar necesariamente relacionados con ella como son eventos de carácter cultural o deportivo, como eventos musicales o ferias o partidos de fútbol que concentran a miles de personas.

Gracias a las nuevas herramientas que ofrece el Big Data, la movilidad en la ciudad ha tomado una nueva dimensión. Este nuevo escenario ha cambiado tanto la forma de recolectar datos como su análisis y su representación, permitiendo estudiar tendencias y patrones que nos servirán para mejorar el servicio ofrecido a los ciudadanos [2] y resolver problemas que ni si quiera se habían planteado, que tomarán especial importancia con la implantación en el futuro del 5G y las “ciudades inteligentes”.

Junto al uso de nuevas tecnologías, estamos experimentado un cambio en la forma en la que las personas se desplazan por la ciudad ya sea por trabajo o entretenimiento. En los últimos tiempos han aparecido nuevos servicios a disposición de los ciudadanos, como los novedosos alquileres de vehículos eléctricos sin conductor por cortos periodos de tiempo, como coches o motocicletas y servicios más consolidados como el alquiler de bicicletas eléctricas.

Desde hace 3 años, el Ayuntamiento de Madrid ha pasado a recoger rigurosamente datos sobre el uso de este servicio, incluyendo información sobre los trayectos que las personas realizan. Junto con los datos sobre trayectos de los alquileres de bicis, encontramos disponibles datos sobre accidentes en la ciudad de Madrid, con información detallada de la localización del suceso, día de la semana y fecha del accidente y tipos de vehículos implicados, estado de la calzada en el momento del accidente, etc.

La motivación del proyecto es aprovechar todos los datos disponibles sobre trayectos de usuarios del servicio de alquiler de bicicletas eléctricas, accidentes en la ciudad de Madrid y otras fuentes de datos externas para desarrollar una herramienta de visualización en forma de aplicación web que pretende contribuir a mejorar la movilidad en la ciudad a través de la implementación de herramientas de análisis que permitan extraer conclusiones relevantes sobre el servicio, el uso que se hace de él y el impacto que tienen en el servicio los factores externos descritos anteriormente.

# Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

A continuación, se describen las herramientas específicas usadas en el proyecto, por orden cronológico de inclusión en el desarrollo.

- Spark. Framework destinado al procesamiento de datos a gran escala con especial enfoque en lograr tiempos de ejecución reducidos que ofrece una alternativa al paradigma de computación MapReduce. En el proyecto se hace uso de la librería de Python para Spark (PySpark).
- D3 [3]. Librería construida con JavaScript cuyo propósito es facilitar la interacción del desarrollador con el DOM (Document Object Model) de una página web, donde se almacena la información sobre los componentes de esta. Entre todas las utilidades incluidas en la librería, tienen especial relevancia en el proyecto la capacidad de añadir datos externos a una página web, a través de funciones que permiten importar y leer conjuntos de datos en diversos formatos, y la capacidad de vincular los datos leídos a elementos concretos de la página, para darles forma posteriormente y poder visualizarlos.
- Leaflet [4]. Librería para trabajar con mapas interactivos en aplicaciones web, que facilita tanto la lógica de trabajo interna como una hoja de estilos propia para los componentes. Leaflet proporciona gran cantidad de funcionalidades importantes a la hora de incluir mapas en la aplicación, entre las más destacadas:
    - Soporte de capas de diferente naturaleza sobre el mapa, como polígonos o imágenes.
    - Funciones de control de la interacción del usuario con el mapa, como *zoom* o eventos, ya sean generales o propios de la librería.
    - Capacidad de extensión, personalización e inclusión de funcionalidades nuevas a través de *plugins* externos.

- Bootstrap [5]. Conjunto de librerías para el diseño de aplicaciones web que se ocupa del *front-end*. Incluye plantillas de diseño para diferentes elementos, además de funciones adicionales a través del uso de extensiones de JavaScript. Esta herramienta facilita tanto la distribución del contenido en la aplicación como la personalización de los componentes.
- Flask [6]. Framework para la creación de un servidor usando Python. Flask ofrece la posibilidad de introducir código Python en una página web a través de diversas plantillas, que serán procesadas por el servidor antes de ser enviadas al cliente.

# Capítulo 3. ESTADO DE LA CUESTIÓN

En cuanto a estudios realizados sobre flujos de personas haciendo uso de técnicas de visualización dinámicas, encontramos, centrado en el servicio de alquiler de bicicletas eléctricas de la ciudad de Madrid, el desarrollado por el equipo de investigación de la Universidad Complutense de Madrid [7] que muestra 250 000 rutas en un mapa dinámico (Ilustración 1 e Ilustración 2) con el objetivo de estudiar el flujo de usuarios en la ciudad, representando con diferentes colores la concentración de trayectos en las calles de Madrid.

*Ilustración 1. Visualización de los trayectos [7]*

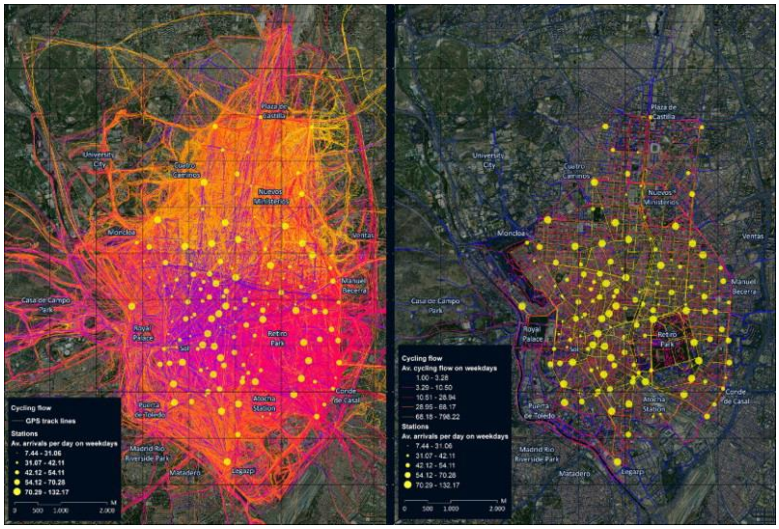


*Ilustración 2. Visualización de trayectos a lo largo de un día [7]*


La visualización dinámica aplicada en este estudio servirá como punto de partida del análisis de técnicas de visualización para la representación dinámica de los trayectos, que además se dotará en el proyecto de un carácter interactivo.

En cuanto a técnicas de visualización multidimensionales, en el estudio "Making Sense: An Innovative Data Visualization Application Utilized Via Mobile Platform" realizado por Jiayan Gu, Sebastian Mackin y Yongjun Zheng [8] se desarrollan visualizaciones sobre el flujo de pasajeros del metro de Shanghai atendiendo a distintos parámetros, como el volumen de trayectos en función de la hora del día y la distancia de los recorridos hechos en esas horas (Ilustración 3), o el flujo de entrada y salida de pasajeros en el metro de Shanghai en las diferentes estaciones (Ilustración 4).

*Ilustración 3. Volumen de trayectos en el metro de Shanghai por horas [8]*

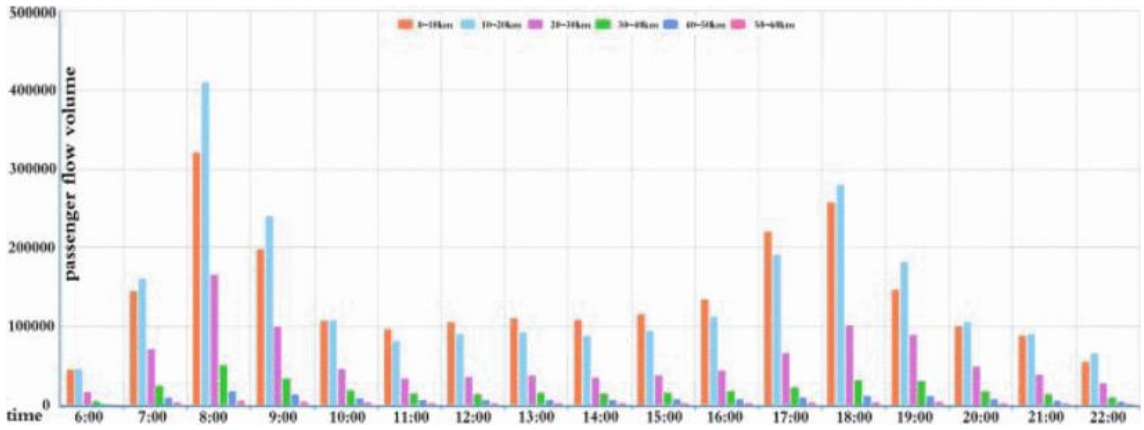
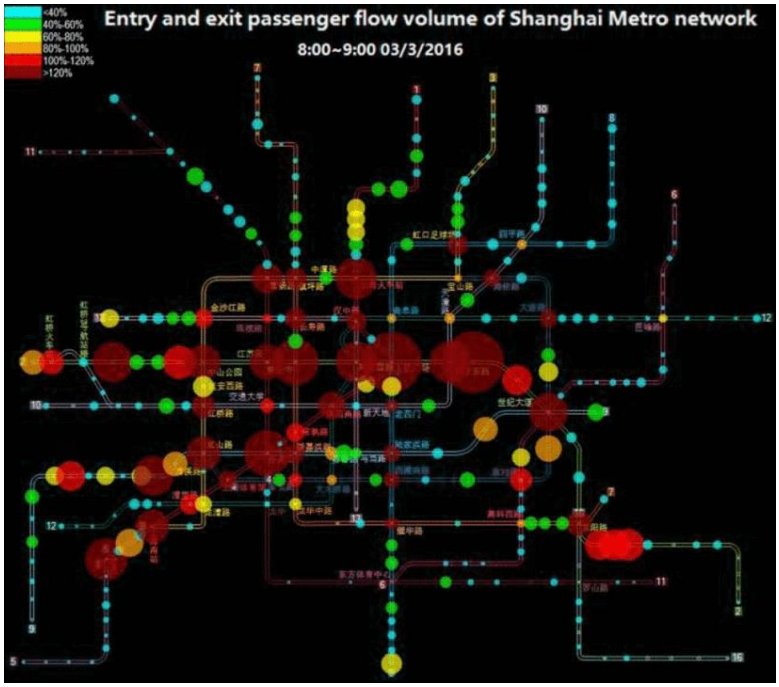



*Ilustración 4. Flujo de entrada y salida de pasajeros en el metro de Shanghai [8]*



# Capítulo 4. DEFINICIÓN DEL TRABAJO

## *4.1 JUSTIFICACIÓN*

Como se ha descrito en el apartado anterior, existen soluciones que han aportado ideas de visualización de trayectos y flujos similares, sin embargo, en este proyecto se busca un enfoque diferente, tratando de ofrecer además de una visualización multidimensional que ayude a comprender los datos, una herramienta dinámica e interactiva en la que el usuario pueda, seleccionando los parámetros, vistas y clasificaciones que desee en cada momento, ajustar la visualización a sus necesidades lo máximo posible a través de una interacción y combinación total entre los diferentes parámetros que se ofrecen. En el proyecto se busca ofrecer una solución que aporte una visualización eficaz a un conjunto de datos complejo gracias a la introducción de elementos que permitan al usuario controlar el contenido.

## *4.2 OBJETIVOS*

- Desarrollo de un estudio estadístico de los conjuntos de datos.
- Estudio y evaluación de herramientas de visualización.
- Desarrollo de una interfaz interactiva y dinámica de visualización multidimensional para datos masivos usando el framework D3 y Leaflet, incluyendo:
    - Visualización de trayectos y flujos de usuarios del servicio.
    - Visualización del estado de las estaciones por número de bicis.
    - Visualización de la cobertura del servicio.

## 4.3 METODOLOGÍA

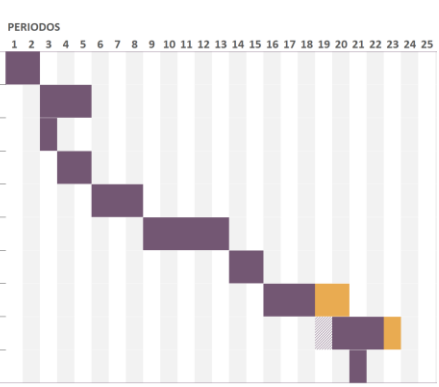
La metodología de trabajo para el desarrollo de la herramienta de visualización se ha planteado como un proceso iterativo para cada uno de los objetivos. En cada iteración se ha planificado el trabajo para el cumplimiento del objetivo, se ha desarrollado el diseño, probado y verificado, para continuar en la siguiente iteración con la planificación del siguiente objetivo, repitiendo este proceso con cada una de las metas hasta alcanzar el desarrollo final.

## 4.4 PLANIFICACIÓN

A continuación, se muestra un cronograma con la programación de las tareas del proyecto (Ilustración 5), en el que se indica la semana esperada en la que se inicie la tarea, su duración esperada, la semana real de inicio y la duración real de la tarea. Cada uno de los periodos corresponde a una semana, siendo el periodo uno la primera semana del año. Las tareas que se han extendido más periodos de lo inicialmente previsto tienen representados estos periodos en color amarillo. Las tareas que han comenzado en un periodo posterior al previsto tienen representados estos periodos con líneas diagonales.

*Ilustración 5. Planificación del proyecto*

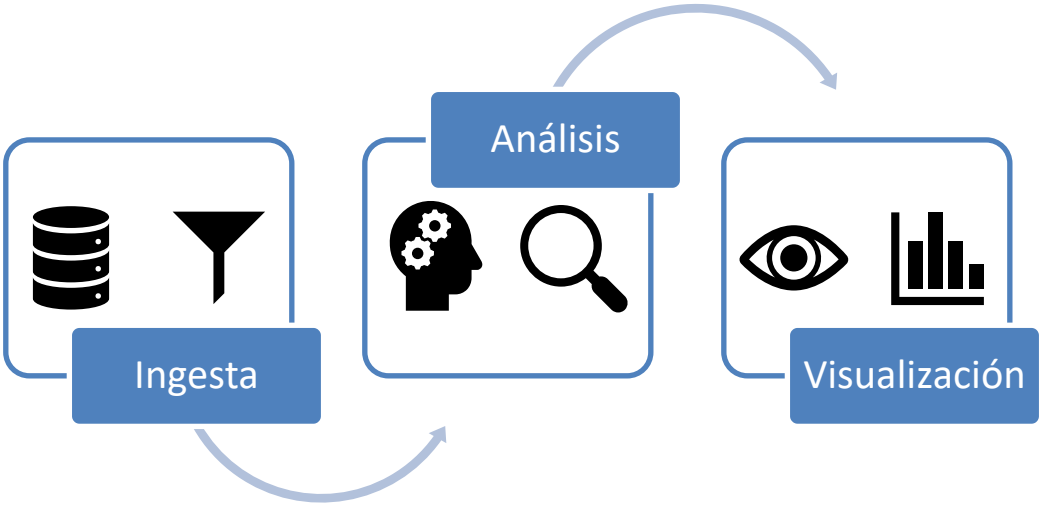
| ACTIVIDAD | INICIO DEL PLAN | DURACIÓN DEL PLAN | INICIO REAL | DURACIÓN REAL | PORCENTAJE COMPLETADO |
|---|---|---|---|---|---|
| Aproximación al conjunto de datos de trayectos con R | 1 | 2 | 1 | 2 | 100% |
| Estudio completo del conjunto de datos de trayectos con PySpark | 3 | 3 | 3 | 3 | 100% |
| Aproximación al conjunto de datos de accidentes con Excel | 3 | 1 | 3 | 1 | 100% |
| Estudio completo del conjunto de datos de accidentes con PySpark | 4 | 2 | 4 | 2 | 100% |
| Estudio del framework D3 y Leaflet como herramientas de visualización | 6 | 3 | 6 | 3 | 100% |
| Desarrollo de la herramienta | 9 | 5 | 9 | 5 | 100% |
| Estudio del framework Bootstrap | 14 | 2 | 14 | 2 | 100% |
| Actualización de interfaz inicial a Bootstrap y desarrollo final | 16 | 3 | 16 | 5 | 100% |
| Documentación del trabajo realizado | 19 | 4 | 20 | 4 | 100% |
| Implementación de la aplicación web en el servidor | 21 | 1 | 21 | 1 | 100% |



# Capítulo 5. DESCRIPCIÓN DEL TRABAJO REALIZADO

Antes de comenzar a describir el trabajo realizado, conviene destacar la estructura general del proyecto Big Data que se va a desarrollar.

*Ilustración 6. Estructura general del proyecto Big Data*




Como se puede observar en la Ilustración 6, un proyecto Big Data cuenta con tres fases fundamentales, cuyos resultados servirán como punto de partida a la siguiente fase:

1. Fase de Ingesta de datos. Fase inicial del proyecto, cuyos objetivos son la obtención, el estudio y el filtrado de datos para su uso en el análisis.
2. Fase de Análisis. Fase central del proyecto, donde se usarán los datos obtenidos en la ingesta para llevar a cabo el estudio y obtener las conclusiones.
3. Fase de Visualización. Fase final del proyecto, que recogerá y adaptará el trabajo realizado en las fases posteriores para presentar la visualización y facilitar la comprensión de los datos y las conclusiones del análisis.

El trabajo representa la fase de visualización del proyecto Big Data, que ha sido resultado de la realización de tres proyectos independientes, llevados a cabo cada uno por un alumno y correspondientes a cada una de las fases.

Antes de comenzar con el trabajo de visualización, se realizó de forma conjunta una aproximación inicial a los *datasets*, que se describe en el apartado 5.1, como punto de partida para el trabajo en cada uno de los proyectos. En el apartado 5.2 se detallará el trabajo de desarrollo de la herramienta de visualización.

## *5.1 APROXIMACIÓN A LOS CONJUNTOS DE DATOS Y ANÁLISIS ESTADÍSTICO GENERAL*

El objetivo de esta fase es doble. Por un lado, analizar los conjuntos de datos disponibles para verificar que cumplen los requisitos necesarios para ser utilizados como punto de partida en el proyecto y cumplir con las metas propuestas, y por otro realizar un estudio estadístico general de los datos, que permita entenderlos y obtener conclusiones.

Los datos deben tener consistencia y ser suficientemente amplios como para que llevar a cabo un proceso de visualización complejo realmente ayude a entenderlos.

Como se ha indicado anteriormente, el primer paso ha sido la aproximación al conjunto de datos, con el fin de conocer su estructura y contenido. Los conjuntos principales que se han usado son los datos de trayectos hechos por los usuarios del servicio de alquiler de bicicletas eléctricas, los datos que contienen la descripción de las estaciones de carga del servicio y los datos de accidentes de tráfico de la ciudad de Madrid.

Las herramientas utilizadas para realizar el estudio de los datos cambian en función de la naturaleza de estos. Los datos de trayectos se almacenan en un archivo en formato JSON, divididos por meses, los datos de las estaciones de carga se encuentran en un archivo CSV con información sobre la localización y características de cada estación y los datos de accidentes se guardan en un CSV que contiene la información de todos los accidentes ocurridos durante un año, donde se indica la localización del accidente, el tipo de vehículo o vehículos implicados, el estado de la vía en el momento del accidente, etc.

A continuación, se detallará el proceso seguido de análisis para cada uno de los conjuntos.

### 5.1.1 ESTUDIO DE LOS DATOS DE TRAYECTOS

Los datos de trayectos suponen el mayor volumen de información del proyecto. Hay datos disponibles para los años 2017 y 2018, siendo un total aproximado de 8 GB de información. Como se ha descrito anteriormente, los datos de trayectos están divididos por meses y mantienen su estructura en todos los conjuntos. Aprovechando esta situación y con la idea de realizar una aproximación inicial al conjunto de datos, se comienza a trabajar con los datos de un mes de trayectos usando el lenguaje R, que ha servido para conocer la estructura que tienen los datos. Con el fin de continuar el estudio de los datos durante todo el año, se hace palpable la necesidad de cambiar de herramienta de trabajo. R ha permitido una aproximación eficaz a los datos, pero a medida que crece el volumen de información, los tiempos de ejecución hacen inviable continuar trabajando en un solo equipo.

El siguiente paso por tanto es hacer uso de la potencia del clúster Big Data de la Universidad programando en PySpark. Todos los datos de trayectos disponibles se guardan en HDFS para hacerlos accesibles al clúster y gracias a la distribución de tareas entre los nodos, se consiguen unos tiempos de ejecución más ajustados para unos conjuntos de datos mucho más grandes. PySpark ofrece una librerías y funciones para el análisis y visualización de datos que han facilitado el estudio de la información.

Una vez realizado este análisis obtenemos conclusiones importantes sobre la estructura de los datos que sentarán la base del desarrollo de visualización posterior.

La estructura de los datos de trayectos es la siguiente:

```
root
 |-- _id: struct (nullable = true)
 |    |-- $oid: string (nullable = true)
 |-- ageRange: long (nullable = true)
 |-- idplug_base: long (nullable = true)
 |-- idplug_station: long (nullable = true)
 |-- idunplug_base: long (nullable = true)
 |-- idunplug_station: long (nullable = true)
 |-- track: struct (nullable = true)
 |    |-- features: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- geometry: struct (nullable = true)
 |    |    |    |    |-- coordinates: array (nullable = true)
 |    |    |    |    |    |-- element: double (containsNull = true)
 |    |    |    |    |-- type: string (nullable = true)
 |    |    |    |-- properties: struct (nullable = true)
 |    |    |    |    |-- secondsfromstart: long (nullable = true)
 |    |    |    |    |-- speed: double (nullable = true)
 |    |    |    |    |-- var: string (nullable = true)
 |    |    |    |-- type: string (nullable = true)
 |    |-- type: string (nullable = true)
 |-- travel_time: long (nullable = true)
 |-- unplug_hourTime: struct (nullable = true)
 |    |-- $date: string (nullable = true)
 |-- user_day_code: string (nullable = true)
 |-- user_type: long (nullable = true)
 |-- zip_code: string (nullable = true
)
```

- Id. Identificador único del trayecto
- ageRange. Rango de edad del usuario que ha realizado el trayecto, dividido en:
    - 0. Edad sin especificar.
    - 1. Edad entre 0 y 16 años.
    - 2. Edad entre 17 y 18 años.
    - 3. Edad entre 19 y 26 años.
    - 4. Edad entre 27 y 40 años.
    - 5. Edad entre 41 y 65 años.
    - 6. Edad superior a 66 años.
- Idplug_base. Identificador de la base dentro de la estación en la que se recogió la bicicleta.
- Idplug_station. Identificador de la estación en la que el usuario recoge la bicicleta.

- Idunplug_base. Identificador de la base dentro de la estación en la que el usuario devuelve la bicicleta.
- Idunplug_station. Identificador de la estación en la que el usuario devuelve la bicicleta.
- Track. Detalles del recorrido del usuario (si existe)
- Travel_time. Tiempo de viaje empleado en realizar el trayecto en segundos, desde que se recoge la bicicleta hasta que se devuelve.
- Unplug_hourTime. Hora en la que el usuario recoge la bicicleta.
- User_day_code. Identificador único para un usuario en un mismo día.
- User_type. Tipo de usuario.
    - 0. Tipo sin especificar.
    - 1. Usuario con pase anual.
    - 2. Usuario ocasional.
    - 3. Trabajador de la empresa.
- Zip_code. Código postal del usuario que realiza el trayecto.

### 5.1.2 ESTUDIO DE LOS DATOS DE ESTACIONES

La información con las estaciones se encuentra en un archivo en formato CSV con 173 entradas en las que se detallan los siguientes campos para cada estación:

- Número. Identificador de la estación
- Fecha de alta. Fecha en la que se incorpora la estación al servicio
- Distrito.
- Barrio.
- Calle.
- Número de plazas. Capacidad total de la estación en número de bases.
- Localización de la estación con coordenadas.

Como no es un archivo muy pesado, se ha preferido el estudio directamente usando Excel. Una vez estudiados los datos, se comprueban que cumplen satisfactoriamente con las necesidades del proyecto, incorporando información relevante para la visualización por localización, por barrio o por número de bases de cada una de las estaciones.

### 5.1.3 ESTUDIO DE LOS DATOS DE ACCIDENTES

Los registros de accidentes se encuentran en archivos divididos por años en formato CSV. Al igual que con el conjunto de datos sobre estaciones, la dimensión contenida de los archivos hace posible su estudio en Excel. El objetivo principal del estudio en esta etapa es conocer la distribución de los accidentes atendiendo a los siguientes parámetros incluidos en los archivos descriptivos de los accidentes para comprobar si aportan valor a la posterior fase de análisis:

- Localización del accidente (Ilustración 7).
- Distrito en el que ocurrió el accidente (Ilustración 8).
- Hora y día de la semana (Ilustración 9 e Ilustración 10).

*Ilustración 7. Mapa de calor con la localización de los accidentes*

*Ilustración 8. Número de accidentes por distritos en 2018*

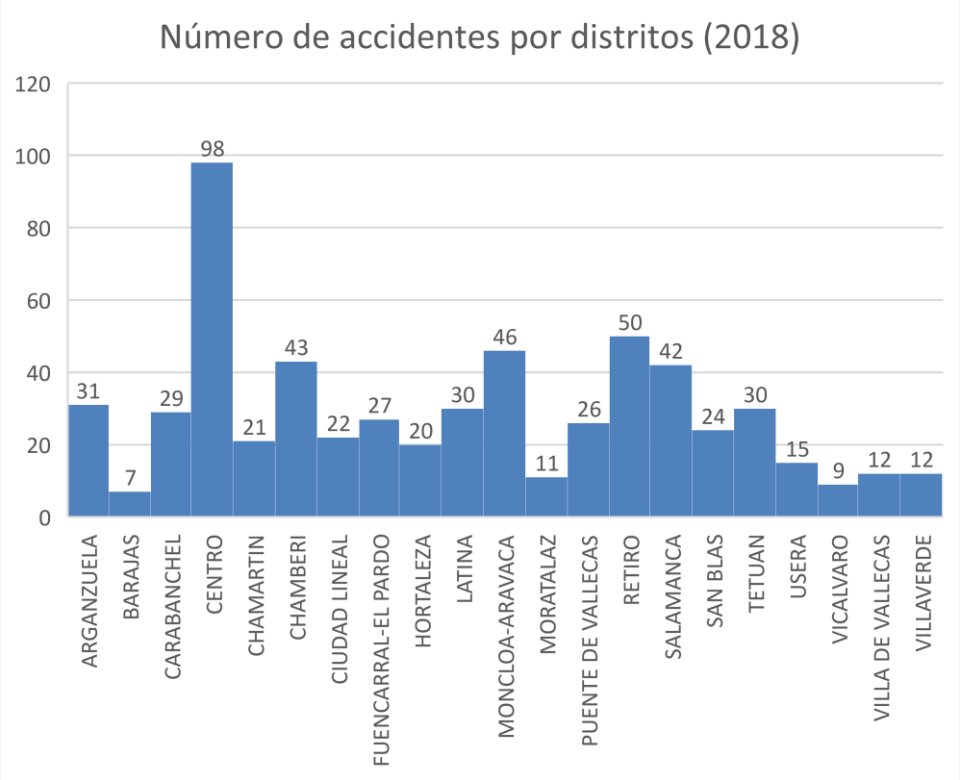



*Ilustración 9. Número de accidentes por días de la semana en 2018*

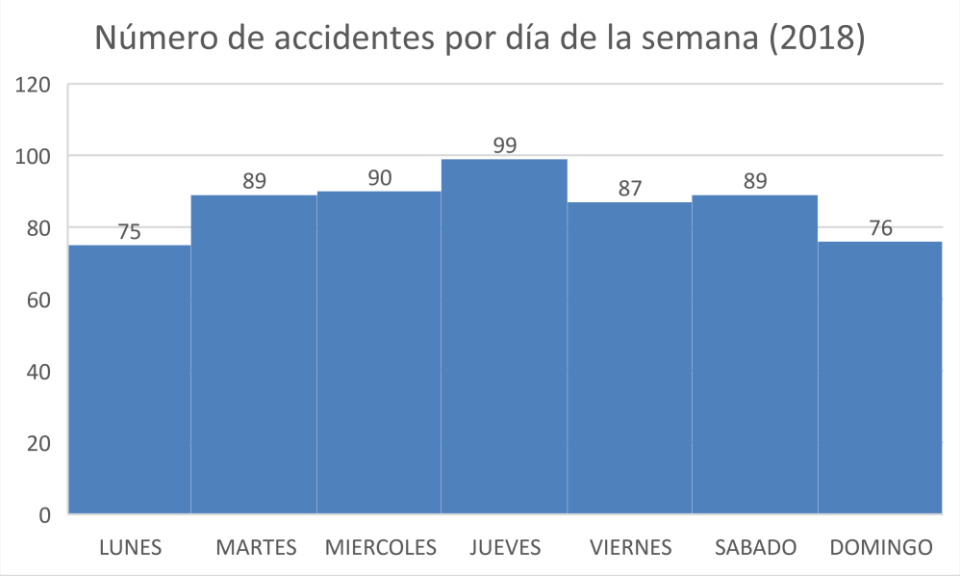
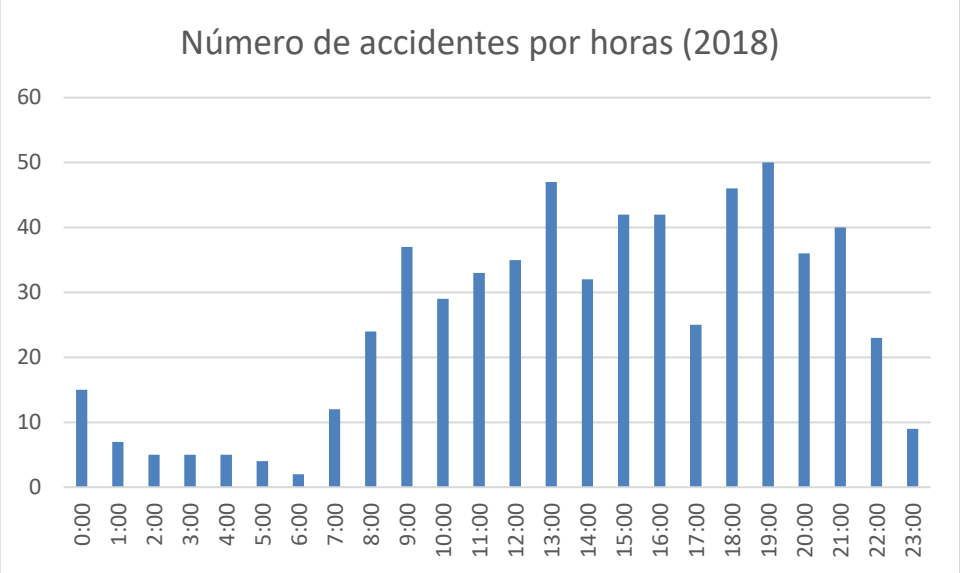


*Ilustración 10. Número de accidentes por horas en 2018*




Como se puede observar en las ilustraciones anteriores, hay parámetros que influyen en los accidentes de tráfico claramente, como la localización, el distrito o la hora del día, y otros como el día de la semana que no aportan información suficiente al estudio, siendo el número de accidentes muy similar entre días de la semana.

## *5.2 DESARROLLO DE LA HERRAMIENTA DE VISUALIZACIÓN*

Una vez finalizada la fase inicial de acercamiento y análisis de los conjuntos de datos, se comienza con el desarrollo de la herramienta de visualización. El proceso de desarrollo se ha dividido en fases de carácter incremental, comenzado por la introducción de las características centradas en los mapas y la visualización de datos, continuando con la integración de una interfaz de control para la aplicación y finalizando con la implementación de un servidor que se encargue de la lógica interna para la obtención de los archivos de datos que la aplicación usará para la visualización.

### 5.2.1 DESARROLLO DE LA VISUALIZACIÓN

La primera parte ha tenido como objetivo la creación de un mapa en la aplicación web y la integración de los conjuntos de datos para su visualización. En esta parte se pueden distinguir dos conjuntos de visualizaciones:

- Por un lado, el relativo a toda la información sobre la cobertura del servicio de alquiler de bicicletas, que ha requerido de la información sobre las estaciones de carga (Estudio de los datos de estaciones), así como de archivos adicionales con información sobre los barrios de la ciudad de Madrid en formato JSON para la clasificación de los barrios de la ciudad por la cobertura del servicio.
- Por otro lado, el conjunto de visualizaciones de los trayectos realizados, en el que se ha usado como fuente fundamental los archivos de trayectos en formato JSON.

A continuación, cada uno de los subapartados se centrará en explicar el proceso seguido para la obtención de las visualizaciones descritas en los puntos anteriores. Junto a la descripción escrita del proceso de desarrollo se incluye el código de la aplicación para cada apartado. Hay partes de código redundantes, como la obtención de los datos, que sirve a varias visualizaciones a la vez. Para reflejar correctamente el nuevo código introducido, se mostrará en `negrita` el código correspondiente a cada nueva visualización introducida en el caso de que se incluya código anteriormente expuesto.

#### *5.2.1.1 Visualización de la cobertura del servicio de alquiler de bicicletas*

Se han realizado 3 visualizaciones diferentes sobre el mapa para representar la cobertura del servicio:

- Visualización con marcadores de la localización de las estaciones.
- Visualización por barrios en una escala de color de la cobertura del servicio.
- Visualización con mapa de calor de la influencia de las estaciones en una zona según el número de bases de cada estación.

Para crear las visualizaciones se han necesitado dos conjuntos de datos: la información de las estaciones (5.1.2) y un archivo con los barrios de la ciudad de Madrid en formato JSON.

El primer paso antes de leer e insertar los datos para crear las visualizaciones ha sido desarrollar una interfaz inicial para la aplicación formada por un mapa, las capas y el control de estas, para ir añadiendo cada una de las visualizaciones a medida que se desarrollen.

HTML

```
<div class="…" id="map" style="…"></div>
```

JavaScript

```
//Creamos el mapa
var map = L.map('map', {
      center: [40.428, -3.70],
      zoom: 12.5,
});

//Creamos las capas base
var mapbox = L.tileLayer('…', {
      maxZoom: 18,
      attribution: '…', id: 'mapbox.streets'});

var openstreetmap = L.tileLayer(…}', {foo: 'bar', attribution: '…'});

var light = L.tileLayer('…', {foo: 'bar', attribution: '…'});

//Capa base predeterminada para el mapa
map.addLayer(mapbox);

//Creamos las capas superpuestas donde se añadirá el contenido
var barrios = L.layerGroup();
var ptsInteres = L.layerGroup();
var estaciones = L.layerGroup();
var estacionesBarrio = L.layerGroup();
//Mapa de calor de las estaciones
var heatMapEstaciones = L.heatLayer([], {
  radius: 25,
  minOpacity: 0.40
});

//Agrupamos las capas base del mapa
var baseMaps = {
  "Mapbox": mapbox,
  "Open Street Maps": openstreetmap,
  "Light": light
};
```

```
//Agrupamos las capas superpuestas del mapa
var groupedOverlays = {
  'Cobertura del servicio': {
    'Barrios': barrios,
    'Puntos de Interes': ptsInteres,
    'Estaciones': estaciones,
    'Estaciones por barrio': estacionesBarrio,
      'Mapa de calor': heatMapEstaciones
      }
    };

//Creamos y añadimos el control de capas al mapa
L.control.groupedLayers(baseMaps, groupedOverlays).addTo(map);
```

A continuación, se tratará el proceso de creación de cada una de las visualizaciones.

#### 5.2.1.1.1 Visualización con marcadores de la localización de las estaciones

Para esta visualización ha sido necesario recurrir a los siguientes campos de los datos de las estaciones (5.1.2):

- Latitud y longitud.
- Dirección completa de la estación.
- Número de bases.

Una vez se tienen el mapa y las capas donde se almacenará la información de cada una de las visualizaciones, para poder seleccionar por separado cada una de ellas, se pasa a la lectura de los archivos de datos de las estaciones, usando la librería D3.

D3 permite una carga asíncrona de los datos, es decir, cuando se ejecute la función para cargar los datos del archivo y operar con ellos, al mismo tiempo el resto de los elementos de la aplicación web se seguirán cargando y en caso de error de lectura de los datos, la aplicación web se habrá cargado en su totalidad, exceptuando la parte relativa a los datos externos. Este comportamiento vendrá bien para aislar posibles errores y ofrecer una interacción con el usuario más natural, posibilitando la opción de introducir mensajes de alerta en caso de que se produzca un error en la lectura de los archivos con D3.

Otro de los aspectos a destacar de D3 es que las variables que se modifiquen dentro de una función no reflejarán los cambios fuera de la misma. Se debe tener en cuenta este comportamiento a la hora de programar las funciones D3.

Con estos conceptos establecidos se puede continuar con la descripción del proceso de visualización, siguiendo con la lectura de los datos de las estaciones en D3. Del análisis previo de los datos de las estaciones, se observó que no se trata de un archivo CSV al uso, si no que cada uno de los valores está separado de los demás por punto y coma en vez de por coma. La librería D3 incorpora funciones de lectura de archivos con los formatos más comunes, siendo uno de ellos el formato CSV, pero debido al problema indicado anteriormente, hay que recurrir a una función de lectura de datos con formato personalizado, para indicar a D3 que los valores se encuentran separados por punto y coma. Una vez establecido el proceso de lectura de los datos se guarda la información leída en una variable en formato *array*, para hacerla accesible a lo largo de la función y poder trabajar con la información que el archivo contiene.

```
//Variable para guardar los datos de las estaciones
var data;

//Informacion de las estaciones
d3.text("{{ url_for('static', filename='data/bases_bicimad_corregido_v3.csv')
}}", function(error, raw){
  //Cada uno de los elementos esta separado por un ;. Necesitamos especificar el
elemento separador a D3 ya que no se trata de un CSV al uso
  var dsv = d3.dsvFormat(';');
  //Realizamos el parsing personalizado de los datos y los guardamos
  data = dsv.parse(raw);
```

Estando disponible la información de las bases en una variable, el siguiente paso es visualizar esa información. Para ello, se hace uso de un bucle para acceder a cada una de las estaciones, para ir obteniendo los campos descritos al comienzo del apartado. Cada una de las estaciones se representará haciendo uso de un marcador de Leaflet, un componente que recibe una latitud y una longitud de un punto en el mapa y sitúa un icono en dicho punto. Además de la información de localización, a cada marcador se le asigna un cuadro de texto

desplegable cuando se hace *click* en él, en el que se incluye la dirección completa de la estación y el número de bases con las que cuenta.

```
  //Variables en la que guardaremos los datos de localizacion de cada estacion
  var datosDireccion;
  var direccion;

  for(var i = 0; i < data.length; i++){
    //Separamos y ordenamos la direccion de las estaciones
    datosDireccion = data[i].Direccion.split(",");
    if (datosDireccion.length == 4) {
      direccion = datosDireccion[1].trim() + " " + datosDireccion[2].trim() + " "
+ datosDireccion[0].trim() + ", " + datosDireccion[3];
    } else {
      direccion = datosDireccion[1].trim() + " " + datosDireccion[0].trim() + ",
" + datosDireccion[2];
    }

    //Añadimos un marcador para las estaciones
    L.marker([parseFloat(data[i].Latitud), parseFloat(data[i].Longitud)], {
              riseOnHover: true,
              bounceOnAdd: true,
              bounceOnAddOptions: {duration: 500, height: 75}
            }).bindPopup("" + direccion + ": " + data[i].NumPlazas + "
bases").addTo(estaciones);
  }
});
```

#### 5.2.1.1.2 Visualización con mapa de calor de la cobertura del servicio

El esquema de trabajo para esta visualización se ha basado en el mismo procedimiento que en la anterior, leyendo los datos de las estaciones con D3 desde el archivo externo y guardando la información en una variable de tipo *array*. Para representar los datos en forma de mapa de calor se ha recurrido a un *plugin* para Leaflet diseñado para conseguir este tipo de gráficos. Es un *plugin* con un tamaño muy reducido que permite un renderizado rápido de los mapas, característica especialmente importante para la posterior visualización de los trayectos, que cuentan con un volumen mucho mayor de información.

Teniendo en una variable la información de las estaciones, para crear el mapa de calor se tendrán que ir añadiendo los puntos a dicho mapa, identificados por su latitud, longitud e intensidad, que determinarán la posición del punto en el mapa (latitud y longitud) y el color (intensidad).

El paso siguiente es crear una escala en D3, que permita transformar el número de bases de cada estación en un número entre 0 y 1, que es el valor de intensidad que recibirá cada punto cuando se añada al mapa de calor. Para crear la escala se hace corresponder el número máximo de bases por estación con el 1 y el número mínimo con el 0. Así, cuando se introduzca un valor en la función de escala siempre devolverá un valor entre 0 y 1, relativo al número de bases introducido, funcionando como una transformación lineal. Para crear el mapa de calor, por tanto, se necesita ir recorriendo con un bucle el *array* de estaciones, extrayendo la latitud, la longitud y el número de bases de cada una, para transformar su número de bases siguiendo el procedimiento indicado anteriormente y añadir el punto al mapa de calor. Una vez se tienen todos los puntos guardados en el mapa de calor, este se añade a una capa en el mapa, para que cuando un usuario la invoque, se represente el mapa de calor.

```
//Informacion de las estaciones
d3.text("{{ url_for('static', filename='data/bases_bicimad_corregido_v3.csv')
}}", function(error, raw){
  //Cada uno de los elementos esta separado por un ;. Necesitamos especificar el
elemento separador a D3 ya que no se trata de un CSV al uso
  var dsv = d3.dsvFormat(';');
  //Realizamos el parsing personalizado de los datos y los guardamos
  data = dsv.parse(raw);

  //Escala de valores para el mapa de calor
  var heatScale = d3.scaleLinear().domain([d3.min(data, function(d){ return
d.NumPlazas; }), d3.max(data, function(d){ return d.NumPlazas; })])
.range([0, 1]);
  //Variable en la que guardaremos los datos de localizacion de la estacion
  var datosDireccion;
  var direccion;

  for(var i = 0; i < data.length; i++){
    //Separamos y ordenamos la direccion de las estaciones
    datosDireccion = data[i].Direccion.split(",");
    if (datosDireccion.length == 4) {
      direccion = datosDireccion[1].trim() + " " + datosDireccion[2].trim() + " "
+ datosDireccion[0].trim() + ", " + datosDireccion[3];
    } else {
      direccion = datosDireccion[1].trim() + " " + datosDireccion[0].trim() + ",
" + datosDireccion[2];
    }
```

```
    //Añadimos un marcador para las estaciones
    L.marker([parseFloat(data[i].Latitud), parseFloat(data[i].Longitud)], {
                riseOnHover: true,
                bounceOnAdd: true,
                bounceOnAddOptions: {duration: 500, height: 75}
            }).bindPopup("" + direccion + ": " + data[i].NumPlazas + "
bases").addTo(estaciones);

    //Añadimos los puntos al mapa de calor teniendo en cuanta para la intensidad
el numero de bases de cada estacion
    heatMapEstaciones.addLatLng([parseFloat(data[i].Latitud),
parseFloat(data[i].Longitud), heatScale(data[i].NumPlazas)])
.addTo(calorEstaciones);
    }
});
```

#### 5.2.1.1.3 Visualización por barrios de la cobertura del servicio

Esta visualización requiere, además del archivo con la descripción de las estaciones de carga, información adicional sobre los barrios de Madrid. Esta información es necesaria para realizar la distinción por barrios de la cobertura del servicio. Dicha información se encuentra en un archivo en formato GeoJSON, que define cada uno de los barrios como un objeto con las siguientes propiedades:

- Name. Nombre del barrio.
- Value. Numero de bases de carga en el barrio.
- Geometry.
    - Type. Definición de la forma geométrica. En este caso un polígono.
    - Coordinates. Coordenadas que unidas forman los límites del polígono.

Una vez descrita la estructura del archivo adicional que se va a usar, la visualización consistirá en definir los polígonos que representarán a los barrios en el mapa con un color más claro u oscuro dependiendo del número de bases de carga que haya en el barrio. Antes de representar los barrios en el mapa, por tanto, hay que definir distintos rangos de clasificación por colores atendiendo al número de bases. Junto a la capa con la representación de los barrios, se ha optado por introducir tres componentes adicionales: un cuadro de información que se actualizará dinámicamente mostrando el nombre del barrio sobre el que el usuario ponga el cursor y que aporte información adicional sobre el número de bases, una leyenda que acompañe al mapa para mostrar los rangos de clasificación por colores de los

barrios, y un menú contextual que se activará cuando el usuario haga *click* derecho en el barrio, en el que se le mostrará la opción de visualizar las estaciones sólo en ese barrio. Definidos todos los componentes, se procede a la explicación de su desarrollo.

*Desarrollo del cuadro informativo y la leyenda*

Leaflet contiene funciones propias para la creación de estos componentes. El proceso seguido en cada uno de ellos es el mismo. Se crea el componente, definiendo el contenido y se establece su comportamiento cuando se añada al mapa y se actualice, en caso de que sea dinámico.

```
//Variable para guardar el cuadro informativo
var info = L.control({position: 'topright'});

//Comportamiento del cuadro informativo cuando se añade al mapa
info.onAdd = function (map) {
  this._div = L.DomUtil.create('div', 'info');
  this.update();
  return this._div;
};

// Comportamiento del cuadro informativo cuando se actualiza el contenido
info.update = function (props) {
  this._div.innerHTML = '<h6>Estaciones por barrio</h6>' +  (props ?'<b>' +
props.name + '</b><br />' + props.value + ' estaciones' : 'Pasa sobre un
barrio');
};

//Variable para guardar la leyenda
var legend = L.control({position: 'bottomright'});

//Comportamiento de la leyenda cuando se añade al mapa
legend.onAdd = function (map) {
  var div = L.DomUtil.create('div', 'info legend'),
    grades = [0, 10, 50, 100, 150, 200, 250, 275],
    labels = [];

  //Recorre los intervalos definidos asignando el color correspondiente al
elemento
  for (var i = 0; i < grades.length; i++) {
    div.innerHTML += '<i style="background:' + getColor(grades[i] + 1) + '"></i>
' + grades[i] + (grades[i + 1] ? '&ndash;' + grades[i + 1] + '<br>' : '+');
  }
  return div;
};
```

Además de estas funciones, se ha añadido la posibilidad de que cuando se oculte la capa de cobertura por barrios, la leyenda y el cuadro informativo también desaparezcan, para aumentar el espacio disponible en el mapa cuando estos componentes no son de utilidad y se esté visualizando otro contenido.

```
//Cuando no vemos la capa de los barrios, la leyenda y la informacion desaparecen
map.on('overlayadd', function(eventLayer) {
  //Si seleccionamos la capa con la informacion de los barrios la leyenda se
activara
  if (eventLayer.name.localeCompare(nombreCapaBarrios) == 0) {
    legend.addTo(this);
    info.addTo(this);
  }
});

map.on('overlayremove', function(eventLayer) {
  //Si desactivamos la capa con informacion de los barrios, la leyenda se
desactiva
  if (eventLayer.name.localeCompare(nombreCapaBarrios) == 0) {
    this.removeControl(legend);
    this.removeControl(info);
  }
});
```

*Desarrollo de la visualización de los barrios y el menú contextual*

Antes de visualizar los barrios tal como se explicó en el apartado anterior, hacen falta dos funciones. Una función que reciba el número de bases en un barrio y que devuelva un color y una función que proporcione las características visuales a cada polígono.

```
function getColor(d) {
  return d > 275  ? '#b10026' :
         d > 250  ? '#e31a1c' :
         d > 200  ? '#fc4e2a' :
         d > 150  ? '#fd8d3c' :
         d > 100  ? '#feb24c' :
         d > 50   ? '#fed976' :
         d > 10   ? '#ffeda0' :
                    '#ffffcc';
};
```

```
function style(feature) {
  return {
    fillColor: getColor(feature.properties.value),
    weight: 2,
    opacity: 1,
    color: 'white',
    dashArray: '3',
    fillOpacity: 0.6,
  };
}
```

Con estas funciones desarrolladas, se procede a añadir los barrios al mapa. El siguiente paso es igual que en los casos anteriores. Con las funciones de D3 correspondientes al formato del archivo, se leen ambos y se guardan en dos variables diferentes. La variable que contiene la información de los barrios tiene que ser convertida a un objeto GeoJSON de Leaflet para poder incorporar la capa al mapa. Leaflet ofrece una funcionalidad para transformar a un GeoJSON la información leída de una variable. Además, esta funcionalidad permite realizar una acción sobre cada elemento a medida que se van leyendo, que será útil para aplicar a cada barrio el color correspondiente al número de bases que tenga antes de añadirlo al mapa.

```
//Variable para recorrer las estaciones seleccionadas en un barrio
var j = 0;

//Barrios de Madrid. Cargamos los datos y los añadimos a la capa
d3.json("{{ url_for('static', filename='data/madrid_corregido.json') }}",
function(json) {
  //Informacion de las estaciones
  d3.text("{{ url_for('static', filename='data/bases_bicimad_corregido_v3.csv')
}}", function(error, raw){
    //Cada uno de los elementos esta separado por un ;. Necesitamos especificar
el elemento separador a D3 ya que no se trata de un CSV al uso
    var dsv = d3.dsvFormat(';');
    //Realizamos el parsing personalizado de los datos y los guardamos
    data = dsv.parse(raw);

    (…)

    var geojson;

    geojson = L.geoJSON(json, {
      style: style,
      onEachFeature: onEachFeature
    }).addTo(barrios);
```

Para crear el menú contextual sobre cada barrio, hay que especificar el comportamiento de la capa cuando se haga *click* derecho sobre ella, a través de la edición de sus propiedades. Cuando un usuario haga *click* derecho sobre un barrio, se invocará a una función, que se encargará de crear un mensaje emergente que contendrá los botones para ver o eliminar la vista de las estaciones en el barrio. Cuando aparezca el mensaje emergente, se crearán ambos botones, asociados cada uno a una función que se encargará de establecer los marcadores para las estaciones en ese barrio o eliminarlos, recorriendo la variable donde se guardan los datos de las estaciones y filtrando por el nombre del barrio en el que el usuario hizo *click* derecho.

```
    //Funcion que define el comportamiento de la capa cuando el usuario
interactua con ella, pasando el raton por encima, haciendo click izquierdo o
derecho
    function onEachFeature(feature, layer) {
      layer.on({
        mouseover: highlightFeature,
        mouseout: resetHighlight,
        click: zoomToFeature,
        contextmenu: displayMenu
      });
    }

    //Funcion que define el comportamiento de la capa cuando pasamos el raton por
encima
    function highlightFeature(e) {
      var layer = e.target;
      layer.setStyle({
        weight: 3,
        color: '#666',
        dashArray: '',
        fillOpacity: 0.7
      });

      if (!L.Browser.ie && !L.Browser.opera && !L.Browser.edge) {
        layer.bringToFront();
      }
       info.update(layer.feature.properties);
    }

    //Funcion que define el comportamiento del click izquierdo, en este caso
asignado a un zoom a la capa
    function zoomToFeature(e) {
      map.fitBounds(e.target.getBounds());
    }
```

```
    //Funcion que devuelve la capa a su estado original cuando el raton sale de
ella
    function resetHighlight(e) {
      geojson.resetStyle(e.target);
      info.update();
    }

    //Funcion que incorpora el comportamiento del menu desplegable que se activa
haciendo click derecho en la capa
    function displayMenu(e) {
      var popupMenu = L.popup({
        maxWidth: 180,
        maxHeight: 200,
      });
      var container = L.DomUtil.create('div');
      var botonInsertarEstaciones = crearBoton('<img src"…" style="…"></img>' +
'<span class="align-middle">Mostrar estaciones</span>', container,
e.target.feature.properties.name);
      var botonEliminarEstaciones = crearBoton('<img src="…" style="…"></img>'
+'<span class="align-middle">Eliminar estaciones</span>', container,
e.target.feature.properties.name);

      popupMenu
        .setLatLng(e.latlng)
        .setContent(container)
        .openOn(map);

      //Funcion que define el la incorporacion de las estaciones al mapa
      botonInsertarEstaciones.onclick = function () {
        var numEstaciones = 0;
        var datosBarrio;
        for(var i = 0; i < data.length; i++){
          //Separamos y ordenamos la direccion de las estaciones
          datosDireccion = data[i].Direccion.split(",");
          if (datosDireccion.length == 4) {
            direccion = datosDireccion[1].trim() + " " + datosDireccion[2].trim()
+ " " + datosDireccion[0].trim() + ", " + datosDireccion[3];
          } else {
            direccion = datosDireccion[1].trim() + " " + datosDireccion[0].trim()
+ ", " + datosDireccion[2];
          }

          //Nombre del barrio en el que se encuentra la estacion
          datosBarrio = data[i].Barrio.substring(6, data[i].Barrio.length);
          //Comparamos el barrio de la estacion con el nombre del barrio en el
que se pulso el boton
          if (datosBarrio.toLowerCase().localeCompare(this.value.toLowerCase())
== 0) {
            //Añadimos un marcador para las estaciones dentro del barrio
            marcadores[j] = L.marker([parseFloat(data[i].Latitud),
parseFloat(data[i].Longitud)], {
              icon: iconoMarcadores,
              zIndexOffset: 1000,
```

```
              barrio: datosBarrio,
              riseOnHover: true,
              bounceOnAdd: true,
              bounceOnAddOptions: {duration: 500, height: 75}
            }).bindPopup("" + direccion + ": " + data[i].NumPlazas + "
bases").addTo(estacionesBarrio);
            numEstaciones = numEstaciones + 1;
            j = j + 1;
          }
      }

        if(popupMenu.isOpen()) {
          popupMenu.remove();
        }
      }

      //Funcion que define el borrado de la vista de las estaciones
      botonEliminarEstaciones.onclick = function () {
        var datosBarrio;
        for(var i = 0; i < marcadores.length; i++){
          //Comparamos el barrio de la estacion con el nombre del barrio en el
que se pulso el boton
          if
(marcadores[i].options.barrio.toLowerCase().localeCompare(this.value.toLowerCase(
)) == 0) {
           //Eliminamos los marcadores para las estaciones dentro del barrio
           marcadores[i].remove();
          }
        }

        if(popupMenu.isOpen()) {
          popupMenu.remove();
        }
      }


    //Funcion para crear los botones que se añaden al menu contextual de la capa
cuando hacemos click derecho
    function crearBoton(nombre, container, barrio) {
      var btn = L.DomUtil.create('button', 'btn btn-light btn-sm w-100 my-1',
container);
      btn.setAttribute('type', 'button');
      btn.innerHTML = nombre;
      btn.value = barrio;
      return btn;
    }
  });
});
```

### *5.2.1.2 Visualización de trayectos*

Para llevar a cabo una representación eficaz del volumen de trayectos en bicicleta en un rango temporal determinado, se ha optado por desarrollarla usando mapas de calor, en los que cada punto en el mapa representa un punto de la ruta de un usuario. A medida que más puntos se concentren en una zona, la intensidad de color irá aumentando para reflejar la diferencia de volumen de trayectos con el resto de las zonas de la ciudad. Uno de los objetivos del proyecto es desarrollar una interfaz dinámica e interactiva, por tanto, además de la visualización base de los trayectos, se han desarrollado opciones de filtrado de los datos que permitan estudiar el comportamiento por separado de un grupo de usuarios y, opciones de visualización dinámicas, que permitan observar la evolución de los trayectos a lo largo de las horas del día y entre varios días, representando dicha evolución con una sucesión de mapas de calor que pueda ser ajustada por el usuario utilizando un control de la animación similar al control de reproducción de un vídeo. De esta forma va a ser posible una representación de los trayectos interactiva y que se adecue a los resultados que se quieran extraer.

A continuación, se desarrollará el proceso de creación de la visualización, desde la selección y obtención de los datos hasta la creación, filtrado y animación de los mapas de calor.

#### 5.2.1.2.1 Selección y obtención de los datos

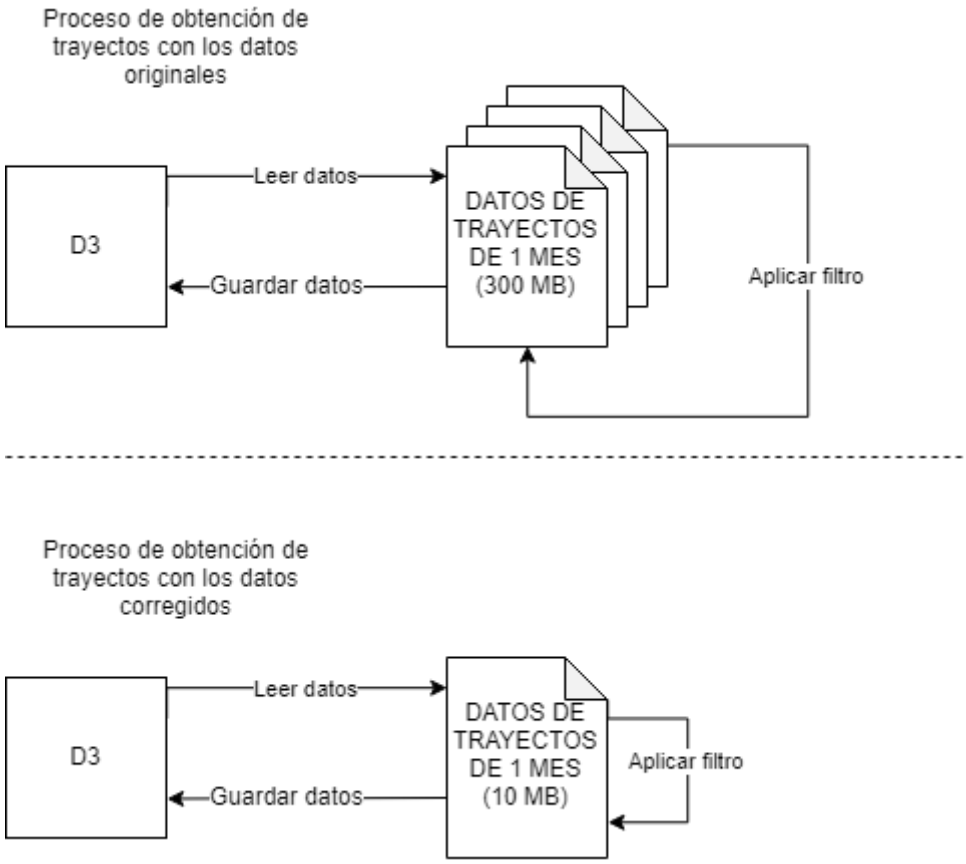
Los datos que se van a usar para esta visualización son los archivos JSON con la información de los trayectos (5.1.1). De entre los campos disponibles, los relevantes para el estudio serán:

- ageRange. El rango de edad permitirá conocer el comportamiento de los usuarios filtrando por edades.
- Track. Campo con la información de los trayectos.
- Unplug_hourTime. La hora de desenganche de la bicicleta se usará para clasificar los datos por rangos temporales para ofrecer una visualización clara de los trayectos.

- User_type. El tipo de usuario se usará también con fines de clasificación y filtrado de los datos para conocer el comportamiento de un grupo de usuarios determinado.

Como se detalló en el apartado 5.1.1, los datos se encuentran divididos en archivos por meses, sin embargo, el volumen de información que es necesario procesar para generar los mapas de calor si se mantiene la división mensual es muy grande, es decir, cada vez que se genere un mapa de calor de una hora de un día en un mes, se tendrá que ir recorriendo todo el mes, filtrando y guardando solo los que coincidan con la hora y el día seleccionados. Un procesamiento con esa cantidad de información a leer retrasará la animación de los mapas de calor hasta tal punto que pierda su objetivo de facilitar la visualización de la evolución de los trayectos. Por esta razón, los datos van a guardarse en archivos divididos de forma diaria. De esta manera, y replicando el ejemplo anterior, para generar un mapa de calor para los trayectos de una hora con la nueva estructura de ficheros sólo se tendrá que recorrer un día de trayectos filtrando por la hora que se quiere estudiar (Ilustración 11).

*Ilustración 11. Comparación entre procesos de obtención de datos de trayectos*



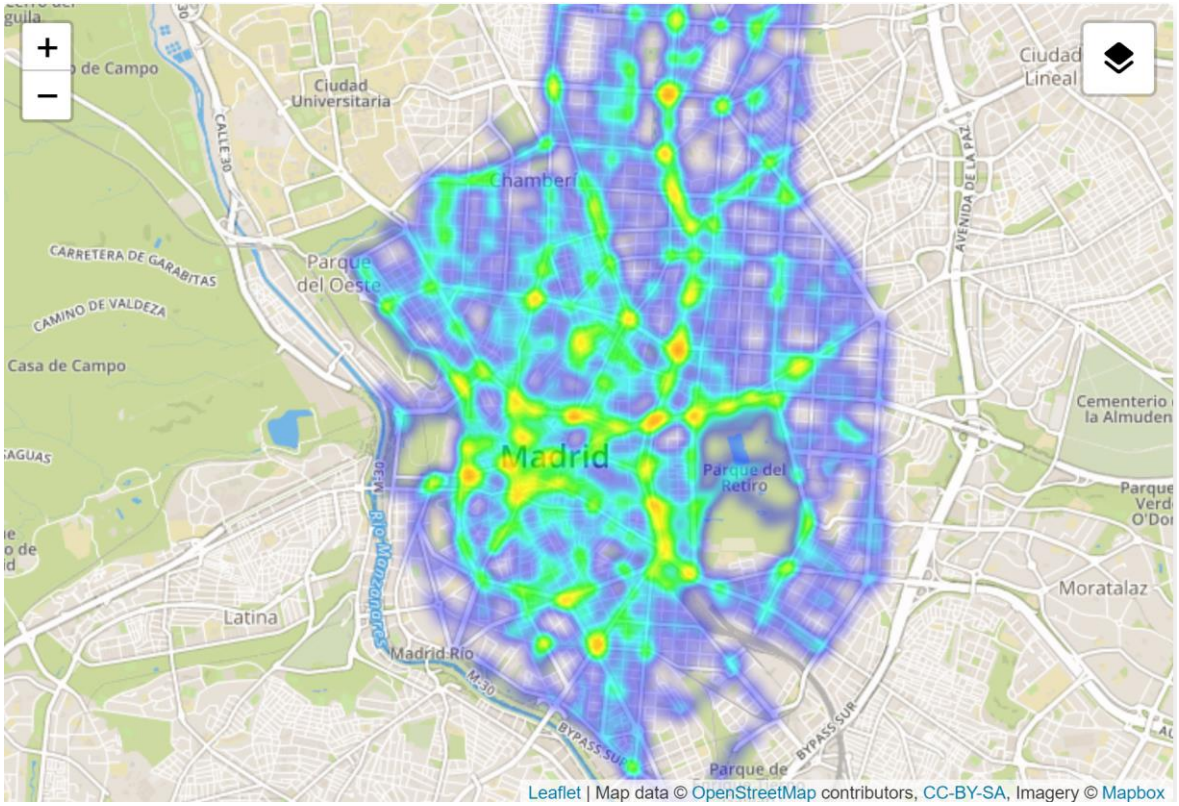

Una vez se ha corregido la estructura de los datos para una ejecución óptima, se puede comenzar con la creación de los mapas de calor.

#### 5.2.1.2.2 Creación de los mapas de calor

El primer paso para crear los mapas de calor es usar la función de lectura de archivos JSON de D3 para obtener los datos de los trayectos y guardarlos en una variable de tipo *array*. Una vez se han guardado los trayectos en la variable, el procedimiento para representar el mapa de calor es similar al seguido en el caso anterior para las estaciones, pero con diferencias en la forma de asignación de la intensidad a cada uno de los puntos. En el caso de las estaciones, se disponía del número de bases que tenía cada estación, pudiéndose usar como medida de la intensidad para ese punto, sin embargo, los puntos de los trayectos no cuentan con tal propiedad. El cambio de color en el mapa vendrá dado por la acumulación mayor o menor de puntos de trayectos. En intervalos horarios, se puede asignar un valor de

intensidad entre 0 y 1, ya que la acumulación de trayectos en el mapa no es suficiente como para saturarlo, sin embargo, no se puede asignar una intensidad arbitraria a los puntos para rangos diarios, ya que se distorsionarán los mapas de calor, ofreciendo siempre una escala de color oscura que hará pensar que el volumen de trayectos no cambia. Hay que normalizar la intensidad de los puntos ajustándola, dependiendo del número de trayectos que haya en el día, en relación con el mayor número de trayectos ocurridos durante un día. Realizando esta corrección, se mantiene la escala de color uniforme para todos los días, apreciándose con claridad cuando han ocurrido más o menos trayectos y donde han estado concentrados. En este punto se tiene todo definido para la visualización del mapa de calor, restando solamente la introducción de los puntos seleccionados en una variable que se pasará a la función para crear un mapa de calor del *plugin* de Leaflet y añadirlo al mapa de la ciudad. A continuación, se muestra el resultado obtenido para los trayectos realizados por los usuarios el día 1 de agosto de 2018 a las 15:00 (Ilustración 12).

*Ilustración 12. Trayectos del 1-08-2018 a las 15:00*



#### 5.2.1.2.3 Filtrado de los datos

Una vez se tiene definido la rutina de creación de los mapas de calor, se encapsula dentro de una función y se pasa a la creación de los controles de filtrado que permitan personalizar la vista de los mapas de calor. Como se ha descrito en la introducción del apartado 5.2.1.2, los controles de filtrado van a consistir en la selección de grupos de usuarios en función del:

- Tipo de usuario (ver tipos en 5.1.1)
- Rango de edad (ver rangos en 5.1.1)
- Rango temporal
    - Horario
    - Diario

Además de la selección para un grupo de usuarios o rango temporal particular, todas las combinaciones posibles de filtrado entre grupos se han tenido en cuenta en el desarrollo, para dotar de más libertad de elección al usuario para ver los datos que necesite en cada momento. En cada uno de los subapartados siguientes se detallará el proceso de creación de cada uno de los filtros.

*Desarrollo del filtro por tipo de usuario*

Para filtrar por tipo de usuario, se ha decidido implementar un control basado en un grupo de cuatro botones, en los que solo puede haber seleccionado uno a la vez y que definen cada uno el control de una variable que se activará o desactivará para marcar el tipo de usuario que se haya seleccionado en cada momento. El valor de la variable se usará como argumento en la función de creación del mapa de calor, filtrando los trayectos por el tipo de usuario marcado antes de añadir los puntos al mapa de calor.

*Desarrollo del filtro por rango de edad*

El control para filtrar por rango de edad se ha definido con una lista desplegable que contiene todas las opciones disponibles. Como en el caso anterior, el valor que se seleccione en la lista se usará como argumento en la función de creación del mapa de calor para filtrar los trayectos por el rango de edad seleccionado antes de añadir los puntos al mapa de calor.

*Desarrollo del filtro por rango temporal*

El control del rango temporal se basa en dos menús diferentes, el primero de ellos destinado al control de los rangos horarios dentro de un día, y el segundo destinado al control de los días completos.

Para la selección de la hora se ha usado un control de tipo *slider*, con 25 posiciones, cada una de las 24 primeras correspondientes a una hora del día (desde las 00:00 hasta las 23:00) y la última se ha dejado como acceso rápido a la visualización de los trayectos de todas las horas (seleccionando esta posición se representarán todos los trayectos del día). Acompañando al *slider* se proporciona una lista desplegable para seleccionar el día. En dos variables se guardan el valor seleccionado en el *slider* y la fecha introducida en la lista desplegable. Teniendo seleccionados ambos parámetros, se invoca a la función de creación del mapa de calor, pasando como argumentos la fecha, que corresponde al nombre del archivo que se leerá para obtener los datos de los trayectos y la hora, que servirá para filtrar los datos de trayectos que se pasen al mapa de calor para visualizarlos. También se tienen en cuenta como argumentos de la función el tipo de usuario seleccionado y el rango de edad, en caso de que los hubiera.

Para la selección del día, el menú consistirá en una lista despegable con la que se seleccionará el día que se desea visualizar. Cuando se marque un valor en la lista, se invocará a la función de creación del mapa de calor pasando como argumento la fecha para la selección del fichero de datos, y el tipo de usuario y rango de edad si los hubiera.

*Código de visualización de los trayectos*

A continuación, se incluye el código de la función de creación de la visualización para los mapas de calor de los trayectos, que incorpora todas las opciones descritas anteriormente.

```
function representarMapaCalor(datosMapa, hora, tipoUsuario, rangoEdad) {
  var horaCorregida;

  //Corregimos los digitos de la hora para comparar
  if(hora.length == 1) {
    horaCorregida = "0" + hora;
```

```
      } else {
        horaCorregida = hora;
      }

      //Si se pulsa un boton sin haber seleccionado una fecha no se lee el
archivo
      if(seleccionDia.value.localeCompare("Seleccione una fecha") == 0) {

      } else {
        d3.json(datosMapa+'?' + Math.floor(Math.random() * 1000), function(error,
json) {
          var datos;
          var fecha = 0;
          var puntosTotales = [];

          if(error){
            actualizarTexto("infoTrayectos1", "Seleccione un grupo");
            actualizarTexto("infoTrayectos2", "Seleccione un grupo");
            actualizarTexto("diaHora", "Fecha y Hora");
          }  else {
            datos = json.features;
            var numViajes = 0;
            var puntos = [];
            var k = 0;
            var cuentaHora = 0;

            //Si no se ha introducido un tipo de usuario / rango de edad o se
seleccionan todos se representa el mapa original
            if (tipoUsuario == null || tipoUsuario == 0) {
              if (rangoEdad == null || rangoEdad == 0){
                for (var i = 0; i < datos.length; i++) {

if(datos[i].unplug_hourTime.$date.substring(11,13).localeCompare(horaCorregida)
== 0){
                    //Cambiamos las coordenadas (Latitud por longitud y longitud
por latitud)
                    for (var j = 0; j < datos[i].coordenadas_ruta.length; j++) {
                      puntos[j] = [datos[i].coordenadas_ruta[j][1],
datos[i].coordenadas_ruta[j][0]];
                      puntosTotales[k] = [puntos[j][0],puntos[j][1], 0.8];
                      k = k + 1;
                    }
                  fecha = datos[i].unplug_hourTime.$date.split("T");
                  numViajes = numViajes + 1;
                }
              }

              //Si no ha habido trayectos con las caracteristicas seleccionadas
borramos el mapa de calor
              if(k == 0) {
                var fechaCorreccion = datos[k].unplug_hourTime.$date.split("T");
                var diaCorreccion = fechaCorreccion[0].split("-");
```

```
                diaCorreccion = diaCorreccion[2] + "-" + diaCorreccion[1] + "-" +
diaCorreccion[0];
                var horaCorreccion = sliderHorario.value;
                if(horaCorreccion.length == 1) {
                  horaCorreccion = "0" + horaCorreccion + ":00:00";
                } else {
                  horaCorreccion = horaCorreccion + ":00:00";
                }
                actualizarTexto("infoTrayectos1", "No hay trayectos");
                actualizarTexto("infoTrayectos2", "No hay trayectos");
                actualizarTexto("diaHora", "Dia: " + diaCorreccion + " Hora: " +
horaCorreccion);
                mostrarAlerta();
                puntosTotales = [[0,0,0]];
                heatMap.setLatLngs(puntosTotales).addTo(calorTrayectos);
              //Si hay trayectos se representan con una escala ligeramente
diferente
            } else {
              dia = fecha[0].split("-");
              dia = dia[2] + "-" + dia[1] + "-" + dia[0];

              hora = fecha[1].substring(0, 8);

              actualizarTexto("infoTrayectos1", "Trayectos: " + numViajes);
              actualizarTexto("infoTrayectos2", "Trayectos: " + numViajes);
              actualizarTexto("diaHora", "Dia: " + dia + " Hora: " + hora);
                //Añadimos los puntos al mapa de calor
                heatMap.setLatLngs(puntosTotales);
                heatMap.setOptions({
                  gradient: {
                    .4: "blue",
                    .6: "cyan",
                    .7: "lime",
                    .8: "yellow",
                    1: "red"
                  },
                  radius: 4.5,
                  minOpacity: 0.50,
                  max: 0.4,
                  blur: 8
                }).addTo(calorTrayectos);
              }

            } else {
              for (var i = 0; i < datos.length; i++) {

if(datos[i].unplug_hourTime.$date.substring(11,13).localeCompare(horaCorregida)
== 0 && datos[i].ageRange == rangoEdad){
                //Cambiamos las coordenadas (Latitud por longitud y longitud
por latitud)
                for (var j = 0; j < datos[i].coordenadas_ruta.length; j++) {
                  puntos[j] = [datos[i].coordenadas_ruta[j][1],
datos[i].coordenadas_ruta[j][0]];
```

```
                    puntosTotales[k] = [puntos[j][0],puntos[j][1], 0.8];
                    k = k + 1;
                  }
                  fecha = datos[i].unplug_hourTime.$date.split("T");
                  numViajes = numViajes + 1;
                }
              }

              //Si no ha habido trayectos con las caracteristicas seleccionadas
borramos el mapa de calor
              if(k == 0) {
                var fechaCorreccion = datos[k].unplug_hourTime.$date.split("T");
                var diaCorreccion = fechaCorreccion[0].split("-");
                diaCorreccion = diaCorreccion[2] + "-" + diaCorreccion[1] + "-" +
diaCorreccion[0];
                var horaCorreccion = sliderHorario.value;
                if(horaCorreccion.length == 1) {
                  horaCorreccion = "0" + horaCorreccion + ":00:00";
                } else {
                  horaCorreccion = horaCorreccion + ":00:00";
                }
                actualizarTexto("infoTrayectos1", "No hay trayectos");
                actualizarTexto("infoTrayectos2", "No hay trayectos");
                actualizarTexto("diaHora", "Dia: " + diaCorreccion + " Hora: " +
horaCorreccion);
                mostrarAlerta();
                puntosTotales = [[0,0,0]];
                heatMap.setLatLngs(puntosTotales).addTo(calorTrayectos);
              //Si hay trayectos se representan con una escala ligeramente
diferente
            } else {
              dia = fecha[0].split("-");
              dia = dia[2] + "-" + dia[1] + "-" + dia[0];

              hora = fecha[1].substring(0, 8);

              actualizarTexto("infoTrayectos1", "Trayectos: " + numViajes);
              actualizarTexto("infoTrayectos2", "Trayectos: " + numViajes);
              actualizarTexto("diaHora", "Dia: " + dia + " Hora: " + hora);
                //Añadimos los puntos al mapa de calor
                heatMap.setLatLngs(puntosTotales);
                heatMap.setOptions({
                  gradient: {
                    .4: "blue",
                    .6: "cyan",
                    .7: "lime",
                    .8: "yellow",
                    1: "red"
                  },
                  radius: 4.5,
                  minOpacity: 0.50,
                  max: 0.4,
                  blur: 8
```

```
                }).addTo(calorTrayectos);
              }
            }
          } else {
            if (rangoEdad == null || rangoEdad == 0){
              for (var i = 0; i < datos.length; i++) {

if(datos[i].unplug_hourTime.$date.substring(11,13).localeCompare(horaCorregida)
== 0 && datos[i].user_type == tipoUsuario){
                //Cambiamos las coordenadas (Latitud por longitud y longitud
por latitud)
                for (var j = 0; j < datos[i].coordenadas_ruta.length; j++) {
                  puntos[j] = [datos[i].coordenadas_ruta[j][1],
datos[i].coordenadas_ruta[j][0]];
                  puntosTotales[k] = [puntos[j][0],puntos[j][1], 0.8];
                  k = k + 1;
                }
                fecha = datos[i].unplug_hourTime.$date.split("T");
                numViajes = numViajes + 1;
              }
            }
            //Si no ha habido trayectos con las caracteristicas seleccionadas
borramos el mapa de calor
            if(k == 0) {
              var fechaCorreccion = datos[k].unplug_hourTime.$date.split("T");
              var diaCorreccion = fechaCorreccion[0].split("-");
              diaCorreccion = diaCorreccion[2] + "-" + diaCorreccion[1] + "-" +
diaCorreccion[0];
              var horaCorreccion = sliderHorario.value;
              if(horaCorreccion.length == 1) {
                horaCorreccion = "0" + horaCorreccion + ":00:00";
              } else {
                horaCorreccion = horaCorreccion + ":00:00";
              }
              actualizarTexto("infoTrayectos1", "No hay trayectos");
              actualizarTexto("infoTrayectos2", "No hay trayectos");
              actualizarTexto("diaHora", "Dia: " + diaCorreccion + " Hora: " +
horaCorreccion);
              mostrarAlerta();
              puntosTotales = [[0,0,0]];
              heatMap.setLatLngs(puntosTotales).addTo(calorTrayectos);
            //Si hay trayectos se representan con una escala ligeramente
diferente
            } else {
              dia = fecha[0].split("-");
              dia = dia[2] + "-" + dia[1] + "-" + dia[0];

              hora = fecha[1].substring(0, 8);

              actualizarTexto("infoTrayectos1", "Trayectos: " + numViajes);
              actualizarTexto("infoTrayectos2", "Trayectos: " + numViajes);
              actualizarTexto("diaHora", "Dia: " + dia + " Hora: " + hora);
                //Añadimos los puntos al mapa de calor
```

```
                heatMap.setLatLngs(puntosTotales);
                heatMap.setOptions({
                  gradient: {
                    .4: "blue",
                    .6: "cyan",
                    .7: "lime",
                    .8: "yellow",
                    1: "red"
                  },
                  radius: 4.5,
                  minOpacity: 0.50,
                  max: 0.4,
                  blur: 8
                }).addTo(calorTrayectos);
              }
            } else {
              for (var i = 0; i < datos.length; i++) {

if(datos[i].unplug_hourTime.$date.substring(11,13).localeCompare(horaCorregida)
== 0 && datos[i].user_type == tipoUsuario && datos[i].ageRange == rangoEdad){
                //Cambiamos las coordenadas (Latitud por longitud y longitud
por latitud)
                for (var j = 0; j < datos[i].coordenadas_ruta.length; j++) {
                  puntos[j] = [datos[i].coordenadas_ruta[j][1],
datos[i].coordenadas_ruta[j][0]];
                  puntosTotales[k] = [puntos[j][0],puntos[j][1], 0.8];
                  k = k + 1;
                }
                fecha = datos[i].unplug_hourTime.$date.split("T");
                numViajes = numViajes + 1;
              }
            }
            //Si no ha habido trayectos con las caracteristicas seleccionadas
borramos el mapa de calor
            if(k == 0) {
              var fechaCorreccion = datos[k].unplug_hourTime.$date.split("T");
              var diaCorreccion = fechaCorreccion[0].split("-");
              diaCorreccion = diaCorreccion[2] + "-" + diaCorreccion[1] + "-" +
diaCorreccion[0];
              var horaCorreccion = sliderHorario.value;
              if(horaCorreccion.length == 1) {
                horaCorreccion = "0" + horaCorreccion + ":00:00";
              } else {
                horaCorreccion = horaCorreccion + ":00:00";
              }
              actualizarTexto("infoTrayectos1", "No hay trayectos");
              actualizarTexto("infoTrayectos2", "No hay trayectos");
              actualizarTexto("diaHora", "Dia: " + diaCorreccion + " Hora: " +
horaCorreccion);
              mostrarAlerta();
              puntosTotales = [[0,0,0]];
              heatMap.setLatLngs(puntosTotales).addTo(calorTrayectos);
```

```
              //Si hay trayectos se representan con una escala ligeramente
diferente
            } else {
              dia = fecha[0].split("-");
              dia = dia[2] + "-" + dia[1] + "-" + dia[0];

              hora = fecha[1].substring(0, 8);

              actualizarTexto("infoTrayectos1", "Trayectos: " + numViajes);
              actualizarTexto("infoTrayectos2", "Trayectos: " + numViajes);
              actualizarTexto("diaHora", "Dia: " + dia + " Hora: " + hora);
                //Añadimos los puntos al mapa de calor
                heatMap.setLatLngs(puntosTotales);
                heatMap.setOptions({
                  gradient: {
                    .4: "blue",
                    .6: "cyan",
                    .7: "lime",
                    .8: "yellow",
                    1: "red"
                  },
                  radius: 4.5,
                  minOpacity: 0.50,
                  max: 1,
                  blur: 8
                }).addTo(calorTrayectos);
              }
            }
          }
        }
      });
    }
  }
```

#### 5.2.1.2.4 Animación de los mapas de calor

Como se ha descrito en el apartado 5.2.1.2, uno de los objetivos es introducir una visualización dinámica de los trayectos. Esta visualización consiste en la sucesión de varios mapas de calor en un intervalo de tiempo, de forma que se cree una animación que permita visualizar la evolución de los trayectos en el rango temporal seleccionado. Para crear la animación ha sido necesario introducir nuevas funcionalidades, tanto en la interfaz de control, como en la lógica de funcionamiento interna.

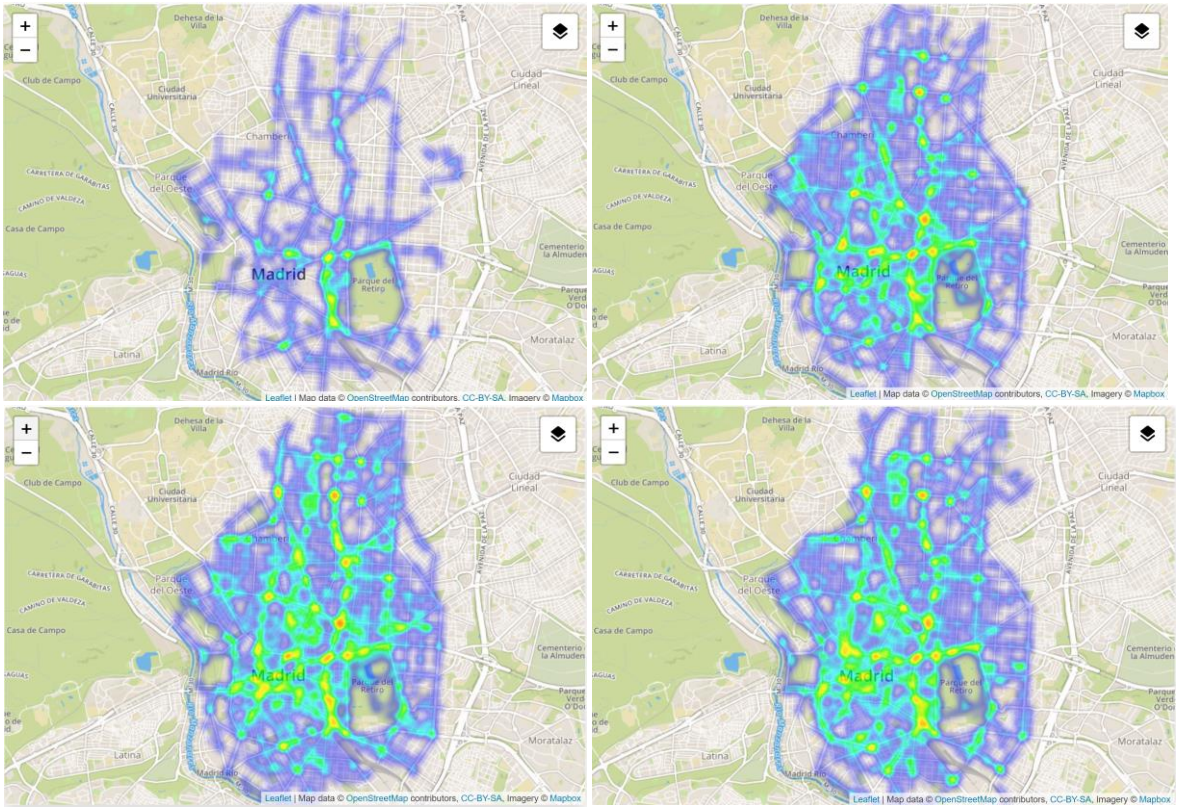
En cuanto al menú de selección de horas, ha sido necesaria la introducción de dos botones en la interfaz para iniciar y parar la animación. Con el *slider* introducido en el apartado

anterior, se selecciona la hora de comienzo de la animación y pulsando en el botón de reproducir se activa. La lógica interna ha consistido en la creación de una función que se activa cuando se pulse el botón de reproducir y que se encarga de, a través de la función *setTimeout()* de JavaScript, invocar cada segundo a la función de creación de un mapa de calor, a la que se pasa como argumento la hora de comienzo de la animación seleccionada en el *slider*. En cada ejecución de la función, el valor de la hora del día de la que se extraen los trayectos se va actualizando para generar un nuevo mapa de calor y representarlo, eliminando el anterior. A continuación, se muestra el código para el control de la animación:

```
//Definimos el comportamiento del boton de reproducción
botonPlayHorario.onclick = function () {
  //Guardamos el día introducido en la lista
  diaReproduccion = seleccionDia.value;
  //Guardamos la hora de comienzo introducida en el slider
  horaInicialReproduccion = sliderHorario.value;

  if (diaReproduccion.localeCompare("Seleccione una fecha") == 0) {
    mostrarAlerta("Seleccione un dia para reproducir los trayectos");
  } else {
    if(horaInicialReproduccion == null) {
      horaInicialReproduccion = 0;
    }
    //Arrancamos la animacion
    idTimerHorario = setInterval(reproducirHoras, 1000);
  }

}

//Definimos el comportamiento del boton de pausa
botonPauseHorario.onclick = function() {
  //Cuando se pulse el boton, se parara la animacion
  clearInterval(idTimerHorario);
}

//Funcion que controla la animación, comprobado la hora de reproduccion e
invocando a la funcion para representar el mapa de calor correspondiente
function reproducirHoras() {
  if(parseInt(horaInicialReproduccion) <= 23) {
    representarMapaCalor("static/data/dias/historicos_" + diaReproduccion +
".json", horaInicialReproduccion, tipoUsuario, rangoEdad);
    horaInicialReproduccion = (parseInt(horaInicialReproduccion) + 1).toString();
  } else {
    //Volvemos al principio del dia y detenemos la animacion
    horaInicialReproduccion = 0;
    clearInterval(idTimerHorario);
  }
}
```

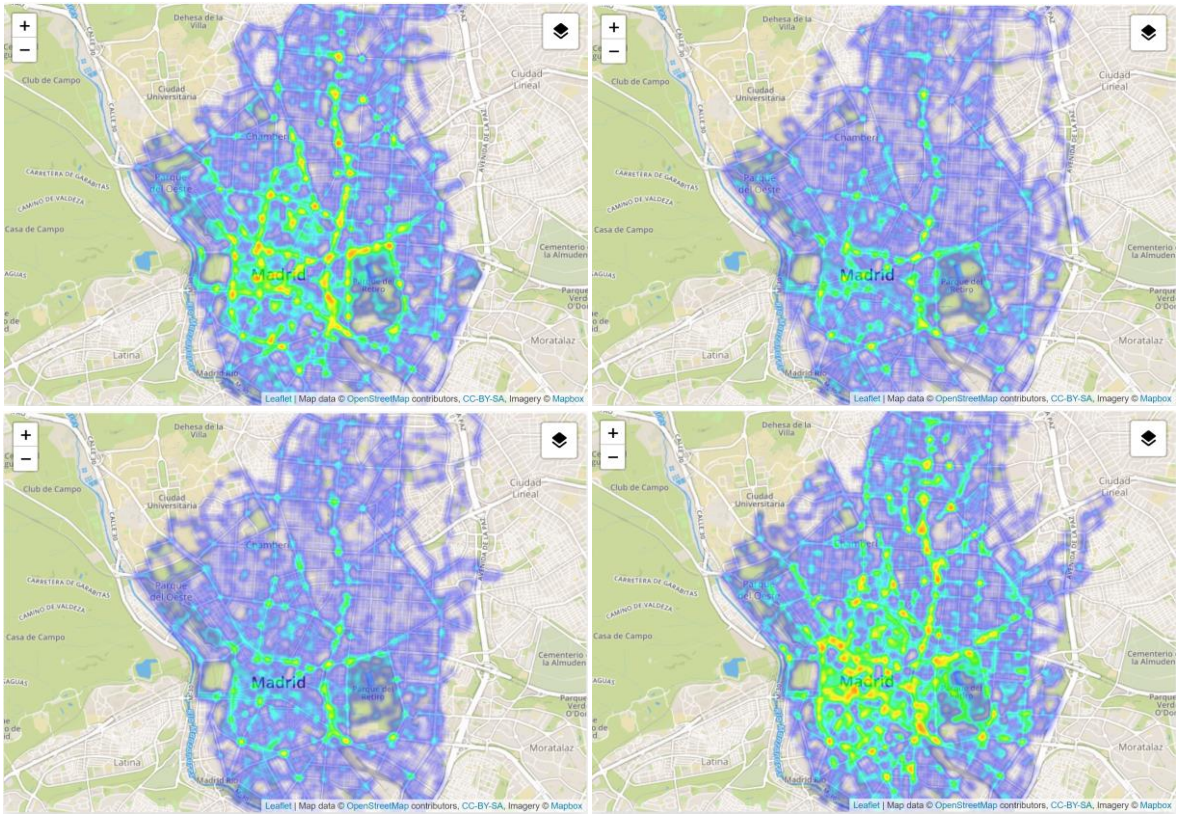
En la Ilustración 13, se muestra un ejemplo de evolución de los trayectos por horas desde las 6:00 hasta las 9:00 del 1 de agosto de 2018, de izquierda a derecha.

*Ilustración 13. Evolución de los trayectos desde las 6:00 a las 9:00. 1-08-2018*




Con respecto a la reproducción de días completos, se añadió a la interfaz los respectivos botones de reproducción y pausa y una nueva lista desplegable, para permitir la selección de un rango de días. Cuando se introduce una fecha en cada una de las listas y se pulse el botón de reproducir, se invoca a la función asociada, que comprueba si el rango de fechas introducido es correcto y como en el caso anterior, haciendo uso de la función *setTimeout(),* se invoca cada 2 segundos a la función para crear un mapa de calor con los trayectos de todo el día, aumentando en cada iteración el día hasta que se llegue al indicado en la nueva lista introducida. En la siguiente ilustración se muestra un ejemplo de la evolución de los trayectos desde el viernes 3 de agosto hasta el lunes 6 de agosto para cada uno de los días, de izquierda a derecha (Ilustración 14).

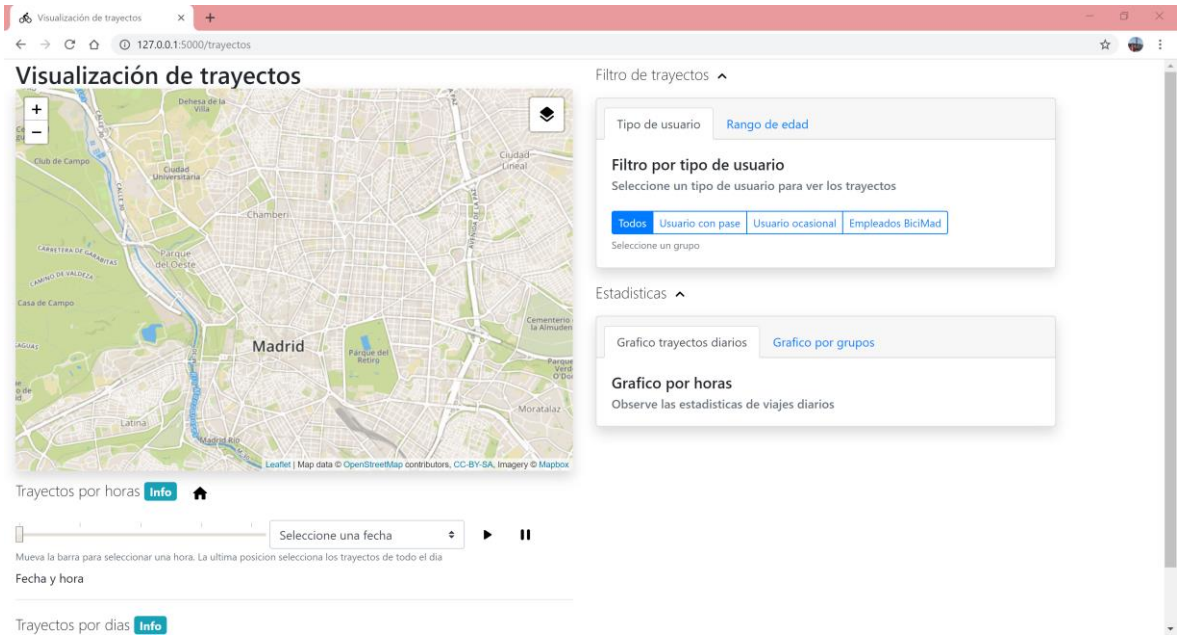
*Ilustración 14. Evolución de los trayectos. Viernes 3 - Lunes 6. Agosto 2018*




## 5.2.2 DESARROLLO DE LA INTERFAZ DE LA APLICACIÓN

El siguiente paso en el desarrollo de la aplicación es dar forma a la interfaz e introducir los nuevos componentes de la librería Bootstrap para organizar el contenido (Ilustración 15). Así, siendo el mapa el principal componente de la aplicación, se ha decidido darle un protagonismo mayor en la parte superior izquierda. El resto de la aplicación está compuesta por las interfaces para cada una de las opciones introducidas en el apartado anterior. Además, se va a introducir en esta fase una nueva pestaña de estadísticas para incorporar a la visualización información útil sobre los trayectos. La creación de este componente se detallará a continuación, junto con el desarrollo de la interfaz del resto de opciones.

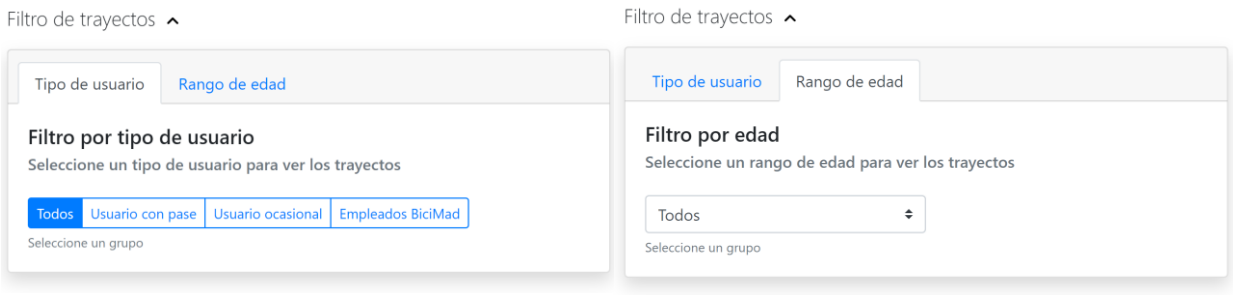
*Ilustración 15. Interfaz de la aplicación*




### *5.2.2.1 Interfaz para las opciones de filtrado*

La interfaz para filtrar los datos consistirá en una tarjeta flotante, situada a la derecha del mapa, que podrá desplegarse y en la que se incluyen todas las opciones de filtrado organizadas por tipo. Esta tarjeta desplegable se ha creado usando el componente base *Card* de Bootstrap. A continuación, se muestra el componente (Ilustración 16) y el código HTML.

*Ilustración 16. Tarjeta con opciones de filtrado*



```
<div class="shadow card text-left collapse show" id="cartaFiltro">
          <div class="card-header">
            <ul class="card-header-tabs nav nav-tabs" id="myTab" role="tablist">
              <li class="nav-item">
                <a class="nav-link active" id="user-tab" data-toggle="tab"
href="#user" role="tab" aria-controls="user" aria-selected="true">Tipo de usuario
</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" id="age-tab" data-toggle="tab" href="#age"
role="tab" aria-controls="age" aria-selected="false">Rango de edad </a>
              </li>
            </ul>
          </div>
          <div class="tab-content card-body" id="myTabContent">
            <div class="tab-pane fade show active" id="user" role="tabpanel"
aria-labelledby="user-tab">
              <h5 class="card-title">Filtro por tipo de usuario</h5>
              <h6 class="card-subtitle text-muted">Seleccione un tipo de usuario
para ver los trayectos</h6>
              <br>
              <div class="btn-group btn-group-toggle btn-group-sm" data-
toggle="buttons">
                <label class="btn btn-outline-primary active">
                  <input type="radio" name="options" id="option1"
autocomplete="off" checked> Todos
                </label>
                <label class="btn btn-outline-primary">
                 <input type="radio" name="options" id="option2"
autocomplete="off"> Usuario con pase
               </label>
               <label class="btn btn-outline-primary">
                 <input type="radio" name="options" id="option3"
autocomplete="off"> Usuario ocasional
               </label>
               <label class="btn btn-outline-primary">
                <input type="radio" name="options" id="option4"
autocomplete="off"> Empleados BiciMad
              </label>
            </div>
            <small class="form-text text-muted" id="infoTrayectos1">Seleccione un
grupo</small>
          </div>
          <div class="tab-pane fade" id="age" role="tabpanel" aria-
labelledby="age-tab">
            <h5 class="card-title">Filtro por edad</h5>
            <h6 class="card-subtitle text-muted">Seleccione un rango de edad para
ver los trayectos</h6>
            <select class="custom-select align-bottom mt-4 w-50"
id="seleccionEdad">
              <option value="0">Todos</option>
              <option value="1">Rango 1 (0-16 años)</option>
              <option value="2">Rango 2 (17-18 años)</option>
```

```
              <option value="3">Rango 3 (19-26 años)</option>
              <option value="4">Rango 4 (27-40 años)</option>
              <option value="5">Rango 5 (41-65 años)</option>
              <option value="6">Rango 6 (66- años)</option>
            </select>
            <small class="form-text text-muted" id="infoTrayectos2">Seleccione un
grupo</small>
          </div>
        </div>
      </div>
```

### *5.2.2.2 Interfaz para el control de la reproducción de la animación y selección de días*

Los controles de reproducción de la animación y la selección de días se han situado bajo el mapa. La distribución de los controles es la misma que la descrita en el apartado 5.2.1.2.4, es decir, un menú para selección de horas con un *slider*, una lista desplegable y los botones de reproducción y un menú para la selección de los días con dos listas desplegables y los botones de reproducción. El código HTML para la creación del componente se muestra a continuación y el resultado se muestra en la Ilustración 17.

```
<div class="mt-2 align-middle">
  <p class="lead" id="tituloSeccionHoras">Trayectos por horas <button class="btn
btn-info badge badge-info" data-toggle="popover" data-trigger="focus"
title="Controles" data-html="true" data-content="…">Info</button> <button
class="btn d-inline" onclick="restaurarVista()"> <img src="…"> </button></p>
  <input type="range" id="rangoHorario" min="0" max="24" value="0" class="slider
d-inline align-middle ml-0" style="width: 45%" list="ticks" data-toggle="tooltip"
data-html="true" data-placement="top" title="">
      <datalist id="ticks">
        <option value="0">
        <option value="6">
        <option value="12">
        <option value="18">
        <option value="23">
      </datalist>
      <select class="custom-select h-100 ml-0 d-inline align-middle"
id="seleccionDia" style="width: 35%">
        <option>Seleccione una fecha</option>
        {% for dia in archivos%}
          <option value={{ dia }}>{{ dia }}</option>
        {% endfor %}
      </select>
      <div class="d-inline">
        <button class="btn btn-sm ml-1 align-middle" id="botonPlayHorario" data-
toggle="tooltip" data-placement="bottom" title="Pulse para reproducir"><img
src="…"></button>
```

```
      </div>
      <div class="d-inline">
        <button class="btn btn-sm ml-1 align-middle" id="botonPauseHorario" data-
toggle="tooltip" data-placement="bottom" title="Pulse para parar"><img
src="…"></button>
      </div>
      <small class="form-text text-muted mt-0 pt-0">Mueva la barra para
seleccionar una hora. La ultima posicion selecciona los trayectos de todo el
dia</small>
</div>
<div class="mt-1 w-50">
  <span class="align-middle" id="diaHora">Fecha y hora</span>
</div>
<div class="mt-1 w-25 d-inline">
  <div id="alerta"></div>
</div>
<hr>
<p class="lead" id="tituloSeccionHoras">Trayectos por dias <button class="btn
btn-info badge badge-info" data-toggle="popover" data-trigger="focus"
title="Controles" data-html="true" data-content="…">Info</button></p>
<div>
  <small class="form-text text-muted mt-0 pt-0 d-inline">Desde</small>
  <select class="custom-select h-100 ml-0 align-middle" id="seleccionDiaInicial"
style="width: 35%">
    <option>Seleccione una fecha</option>
    {% for dia in archivos%}
    <option value={{ dia }}>{{ dia }}</option>
    {% endfor %}
  </select>
    <small class="form-text text-muted mt-0 pt-0 d-inline">Hasta</small>
    <select class="custom-select h-100 ml-0 align-middle" id="seleccionDiaFinal"
style="width: 35%">
      <option>Seleccione una fecha</option>
      {% for dia in archivos%}
      <option value={{ dia }}>{{ dia }}</option>
      {% endfor %}
    </select>
  <div class="d-inline">
    <button class="btn btn-sm ml-1 align-middle" id="botonPlayDiario" data-
toggle="tooltip" data-placement="bottom" title="Pulse para reproducir"><img
src="…"></button>
  </div>
  <div class="d-inline">
    <button class="btn btn-sm ml-1 align-middle" id="botonPauseDiario" data-
toggle="tooltip" data-placement="bottom" title="Pulse para parar"><img
src="{{url_for('static', filename='…')}}"></button>
  </div>
  <div class="mt-1 w-50">
    <span class="align-middle" id="dia">Fecha y hora</span>
  </div>
</div>
```

*Ilustración 17. Interfaz para el control de la visualización de trayectos*


### *5.2.2.3 Interfaz para estadísticas adicionales*

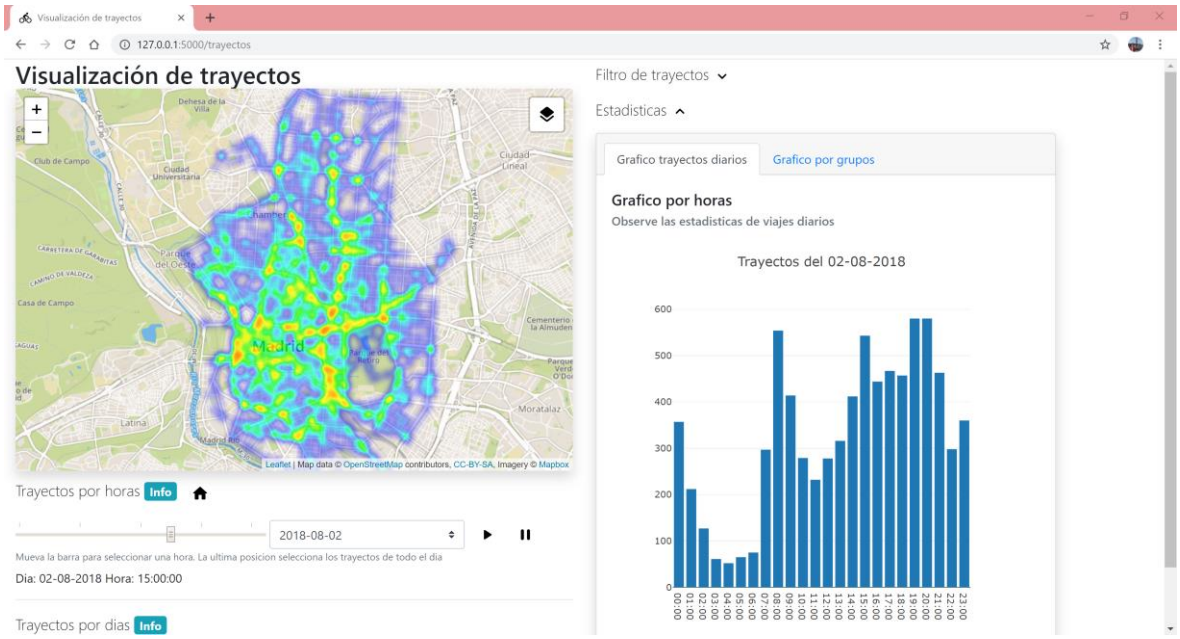
Bajo la tarjeta flotante con las opciones de filtrado se ha incluido el nuevo componente, que consiste en la representación a través de un diagrama de barras de los trayectos por horas ocurridos en el día seleccionado. Además, se incluye un segundo diagrama de barras agrupado para representar la cantidad de trayectos realizados por cada uno de los grupos en cada una de las horas. Ambos diagramas se agrupan en su propia tarjeta flotante.

Para el cálculo de las estadísticas ha sido necesaria la programación de una nueva función, mediante la cual se leen los datos del archivo de trayectos que se está analizando con D3, para guardarlos en una variable e ir recorriéndola con un bucle, agrupando los trayectos por horas para poder representar el diagrama de barras. Para el cálculo de las estadísticas de grupos, se sigue el mismo procedimiento, pero en vez de agrupar los trayectos únicamente por horas, se incluyen en su propio grupo dependiendo del tipo de usuario que lo realizó. A continuación, se muestra el código HTML del componente.

```
<div class="shadow card text-left collapse show" id="cartaEstadisticas">
  <div class="card-header">
    <ul class="card-header-tabs nav nav-tabs" id="myTab2" role="tablist">
      <li class="nav-item">
        <a class="nav-link active" id="graphic-tab" data-toggle="tab"
href="#graphic" role="tab" aria-controls="graphic" aria-selected="true">Grafico
trayectos diarios </a>
      </li>
      <li class="nav-item">
        <a class="nav-link" id="other-tab" data-toggle="tab" href="#other"
role="tab" aria-controls="other" aria-selected="false">Grafico por grupos </a>
      </li>
    </ul>
  </div>
  <div class="tab-content card-body" id="myTabContent2">
    <div class="tab-pane fade show active" id="graphic" role="tabpanel" aria-
labelledby="graphic-tab">
      <h5 class="card-title">Grafico por horas</h5>
      <h6 class="card-subtitle text-muted">Observe las estadisticas de viajes
diarios</h6>
      <div class="w-100" id="graficoEstadisticas"></div>
      </div>

    <div class="tab-pane fade" id="other" role="tabpanel" aria-labelledby="other-
tab">
      <h5 class="card-title">Grafico por grupos</h5>
      <h6 class="card-subtitle text-muted">Observe las estadisticas de viajes por
grupos</h6>
      <div class="w-100" id="graficoEstadisticasGrupos"></div>
      </div>
    </div>
  </div>
</div>
```

El resultado final de la aplicación, incluyendo todas las opciones se muestra en la Ilustración 18.

*Ilustración 18. Vista en uso de la herramienta de visualización*




### 5.2.3 IMPLEMENTACIÓN DEL SERVIDOR

El último paso para finalizar el desarrollo de la herramienta es la implementación de un servidor Python haciendo uso del *framework* Flask. Puesto que en un futuro se generarán nuevos ficheros de datos de trayectos, es necesario automatizar el proceso de búsqueda de los archivos, ya que hay componentes en la aplicación, como las listas desplegables, que dependen directamente de esta información. El servidor se encarga de la tarea de buscar en el directorio correspondiente todos los archivos con datos de trayectos, guardando los nombres en una variable y pasándola a la aplicación cuando esta se carga.

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/trayectos')
def trayectos():
       import os
       path = 'C:\\...\\datos'
       # variable para guardar los nombres de los ficheros
       files = []
```

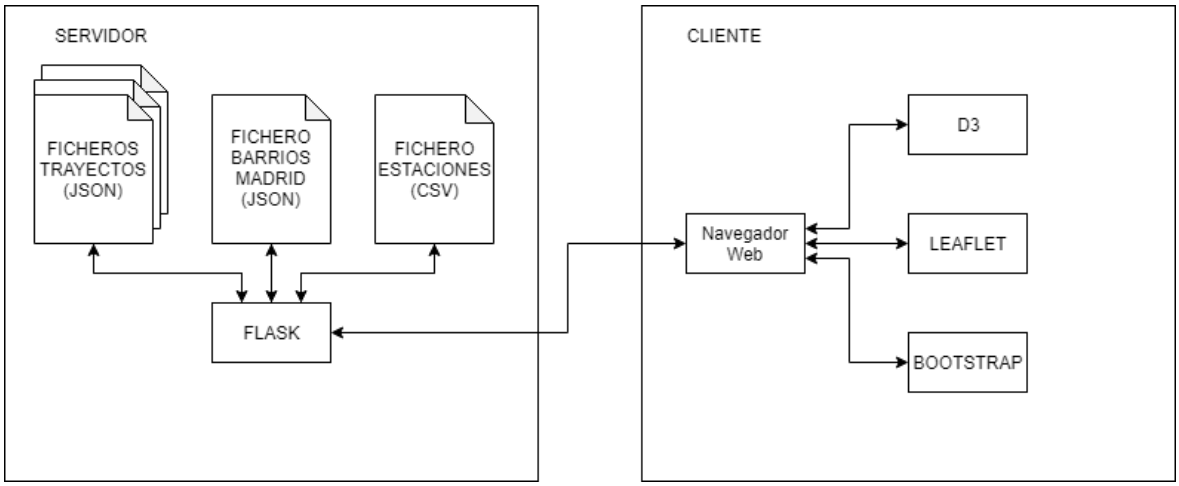
```
        # r=raiz, d=directorios, f = ficheros
        for r, d, f in os.walk(path):
                for file in f:
                        # se comprueba el tipo de fichero y se añade el nombre a la
                        variable
                        if '.json' in file:
                                files.append(file[11:-5])
        return render_template("trayectos.html", archivos=files)

if __name__ == "__main__":
    app.run()
```

Para acceder a la información sobre los ficheros dentro de la aplicación, Flask proporciona unas plantillas que permite insertar código Python en la aplicación web que será procesado por el servidor. Para la creación de listas desplegables que dependen de los ficheros guardados, se accede al contenido de la variable *archivos*, usando la siguiente plantilla en la aplicación:

```
          <select class="…" id="…" style="…">
            <option>Seleccione una fecha</option>
            {% for dia in archivos%}
              <option value={{ dia }}>{{ dia }}</option>
            {% endfor %}
          </select>
```

Una vez se ha implementado el servidor, todos los componentes de la aplicación quedan integrados siguiendo el esquema mostrado en la en la Ilustración 19.

*Ilustración 19. Diagrama de la aplicación*



# Capítulo 6. ANÁLISIS DE RESULTADOS

En este apartado se tratarán los resultados más relevantes de cada una de las fases del proyecto, acompañados de las gráficas, visualizaciones y esquemas correspondientes. Cada uno de los apartados siguientes desarrollará los resultados obtenidos en las fases del proyecto.

## *6.1 RESULTADOS DEL ANÁLISIS INICIAL DE DATOS*

Como se ha descrito en el apartado 5.1, en esta fase inicial del proyecto se realizó un estudio general de los conjuntos de datos con dos finalidades. La primera de ellas, obtener conclusiones relevantes de los datos para comprobar si satisfacen las necesidades del proyecto y la segunda, conocer la estructura y el contenido de los datos, con el fin de enfocar el trabajo sobre los conjuntos de datos seleccionados para las fases posteriores. Tal y como se indicó en la introducción del Capítulo 5. los resultados obtenidos y el trabajo realizado en esta fase han servido como punto de partida para el desarrollo de cada una de las fases del proyecto Big Data conjunto indicado en la Ilustración 6. A continuación, se muestran los resultados más relevantes obtenidos para los tres conjuntos de datos principales.

### 6.1.1 RESULTADOS DEL ANÁLISIS DE TRAYECTOS

Los conjuntos de datos de trayectos representan una de las bases fundamentales del proceso de visualización llevado a cabo. El estudio de los trayectos ha permitido conocer la estructura de los datos, como están divididos y su contenido. El conocimiento de la estructura y la división de los datos ha sido fundamental para llevar a cabo la visualización, ya que ha determinado el trabajo con las diferentes variables y el proceso de adaptación de los datos para ajustarlo a una visualización correcta (5.2.1.2.1)

*Ilustración 20. Estructura de los datos de trayectos*

```
root
 |-- _id: struct (nullable = true)
 |    |-- $oid: string (nullable = true)
 |-- ageRange: long (nullable = true)
 |-- idplug_base: long (nullable = true)
 |-- idplug_station: long (nullable = true)
 |-- idunplug_base: long (nullable = true)
 |-- idunplug_station: long (nullable = true)
 |-- track: struct (nullable = true)
 |    |-- features: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- geometry: struct (nullable = true)
 |    |    |    |    |-- coordinates: array (nullable = true)
 |    |    |    |    |    |-- element: double (containsNull = true)
 |    |    |    |    |-- type: string (nullable = true)
 |    |    |    |-- properties: struct (nullable = true)
 |    |    |    |    |-- secondsfromstart: long (nullable = true)
 |    |    |    |    |-- speed: double (nullable = true)
 |    |    |    |    |-- var: string (nullable = true)
 |    |    |    |-- type: string (nullable = true)
 |    |-- type: string (nullable = true)
 |-- travel_time: long (nullable = true)
 |-- unplug_hourTime: struct (nullable = true)
 |    |-- $date: string (nullable = true)
 |-- user_day_code: string (nullable = true)
 |-- user_type: long (nullable = true)
 |-- zip_code: string (nullable = true
)
```

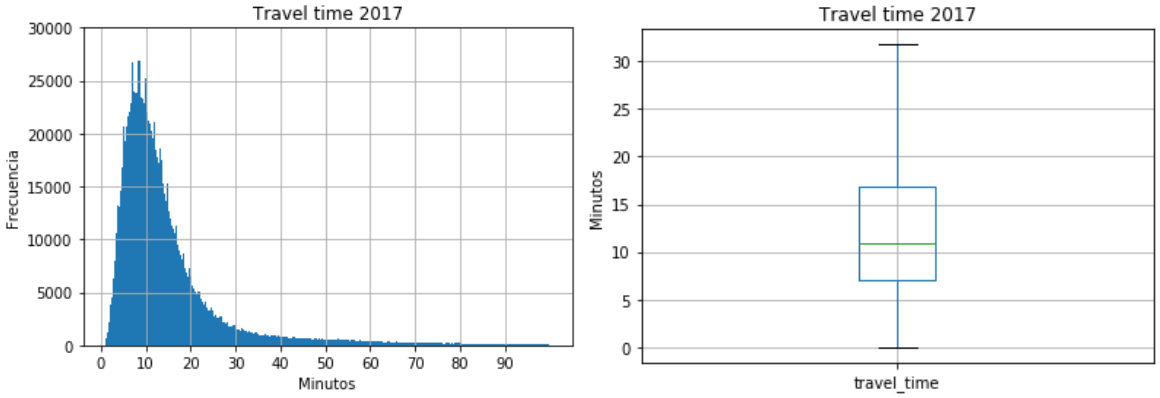
Como se puede observar en la Ilustración 20, el conjunto de datos de trayectos cuenta con gran diversidad de tipos de variables (*strings*, *longs* y *structures* entre otros), siendo cada uno de ellos único en su método de estudio y de trabajo. El estudio de la estructura ha servido para familiarizarse con todos los tipos de datos que se van a usar, de cara al análisis general y a la visualización.

La división de los conjuntos de trayectos en meses también ha influido en el desarrollo posterior, tal como se explica en el apartado 5.2.1.2.1, ya que ha sido necesario un proceso de adaptación de los datos para cumplir con el objetivo del proyecto de ofrecer una visualización dinámica.

Una vez se ha completado la aproximación a los datos, se realizó el estudio de algunas de las variables más importantes, que arrojan conclusiones sobre el uso que se hace del servicio. Así, se ha podido ver el comportamiento general de los usuarios en cuanto al tiempo de viaje (Ilustración 21) y al número de veces que un mismo usuario utiliza el servicio en un día (Ilustración 22).

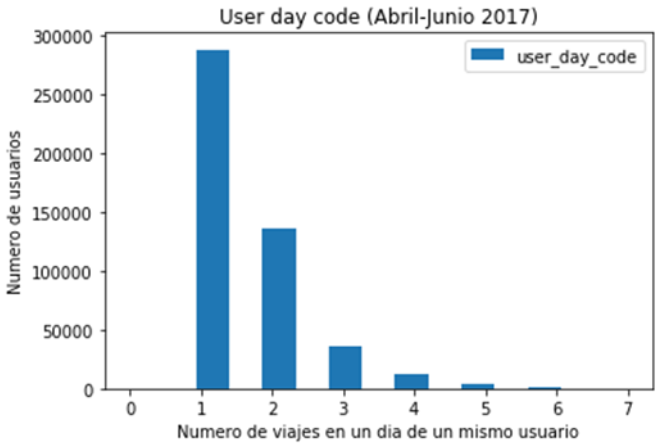
En cuanto al tiempo de viaje, como se puede observar en la Ilustración 21, la mayoría de los trayectos realizados son viajes cortos (el 75 % de los trayectos está concentrado en tiempos entre 0 y 17 minutos). En el año 2017 entre abril y diciembre hubo 2 689 803 trayectos.

*Ilustración 21. Estudio del tiempo de viaje*




Gracias a la variable *user_day_code*, que asigna a un usuario un id único para el día, se han podido conocer los hábitos de uso en cuanto al número de veces que una persona utiliza el servicio en el día. Los resultados para abril, mayo y junio de 2017 se presentan en la Ilustración 22, donde se puede ver como la mayor parte de los usuarios sólo utilizan el servicio una vez al día, alrededor de 280 000 trayectos, seguidos por 150 000 usuarios que lo utilizaron dos veces al día y un número de usuarios prácticamente residual a partir de 4 trayectos diarios.

*Ilustración 22. Estudio del código de usuario diario*




### 6.1.2 RESULTADOS DEL ANÁLISIS DE ACCIDENTES

El estudio de los accidentes proporcionará conclusiones fundamentalmente al trabajo de análisis del proyecto Big Data conjunto (segunda fase del proyecto, Ilustración 6). El objetivo con este conjunto de datos ha sido comprobar si ofrecen la profundidad suficiente como para realizar un trabajo de análisis posterior basado en estos datos. Se han obtenido resultados sobre la localización de los accidentes (Ilustración 23) y la influencia de diferentes parámetros adicionales incluidos en los datos, como el distrito (Ilustración 24), la hora del día (Ilustración 25) o el día de la semana (Ilustración 26).

*Ilustración 23. Mapa de calor con la localización de los accidentes*


*Ilustración 24. Número de accidentes por distritos en 2018*

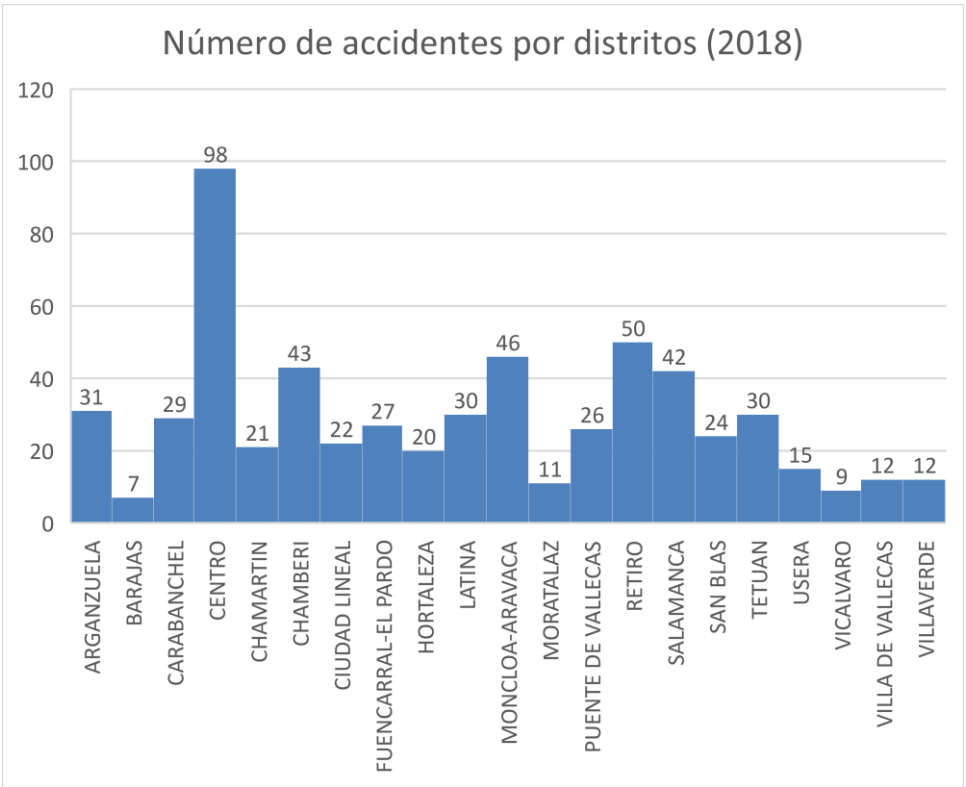


*Ilustración 25. Número de accidentes por horas en 2018*

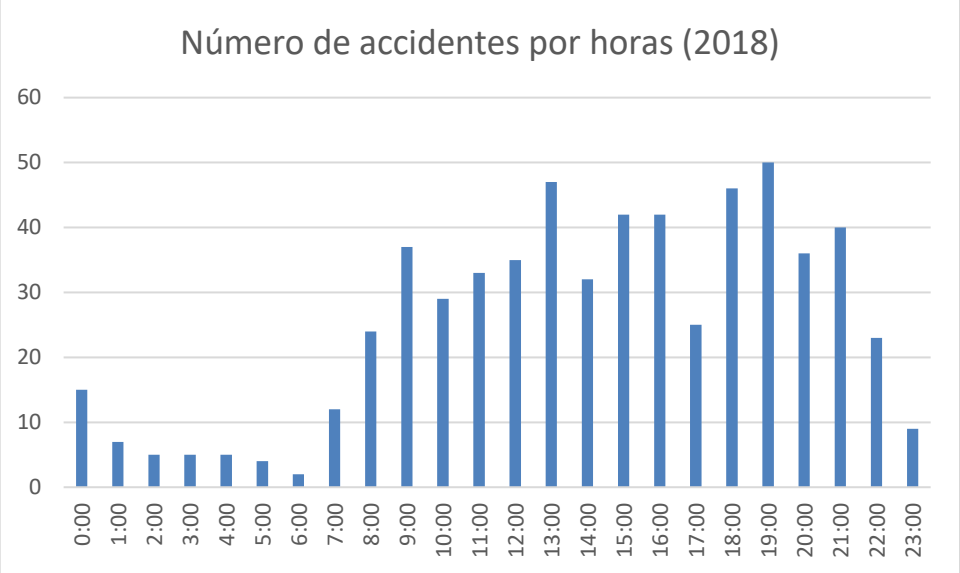
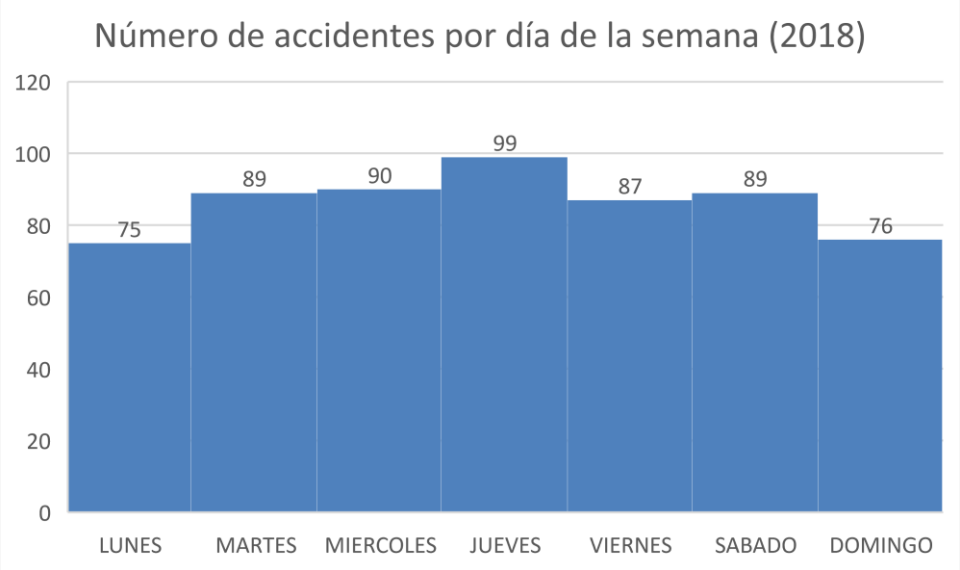



*Ilustración 26. Número de accidentes por día de la semana en 2018*




Presentados los gráficos se pueden obtener conclusiones sobre su utilidad, habiendo parámetros que introducen significado relevante al análisis, como las localizaciones de los accidentes, que como se puede observar en la Ilustración 23, se concentran de forma desigual en la ciudad, o el distrito de la ciudad en el que se produjo el accidente y la hora del día (Ilustración 24 e Ilustración 25). Sin embargo, como se puede ver en la Ilustración 26, el día

de la semana no es un parámetro relevante para el análisis de los accidentes, ya que el número de accidentes prácticamente no varía a lo largo de los días, salvo en lunes y domingos, donde disminuye.

### 6.1.3 RESULTADOS DEL ESTUDIO DE LA INFORMACIÓN DE ESTACIONES

Una parte de la visualización de la herramienta se encarga de mostrar la cobertura del servicio en la ciudad con marcadores, mapas de calor y mapas superpuestos. Este estudio ha servido para familiarizarse con el formato de los datos y las distintas variables que se incluyen para cada estación. El conocimiento de estos datos ha sido la base del desarrollo posterior de la visualización, ya que ha determinado la forma de acceder a los datos y su utilización (5.2.1.1).

## *6.2 RESULTADOS DEL DESARROLLO DE LA HERRAMIENTA*

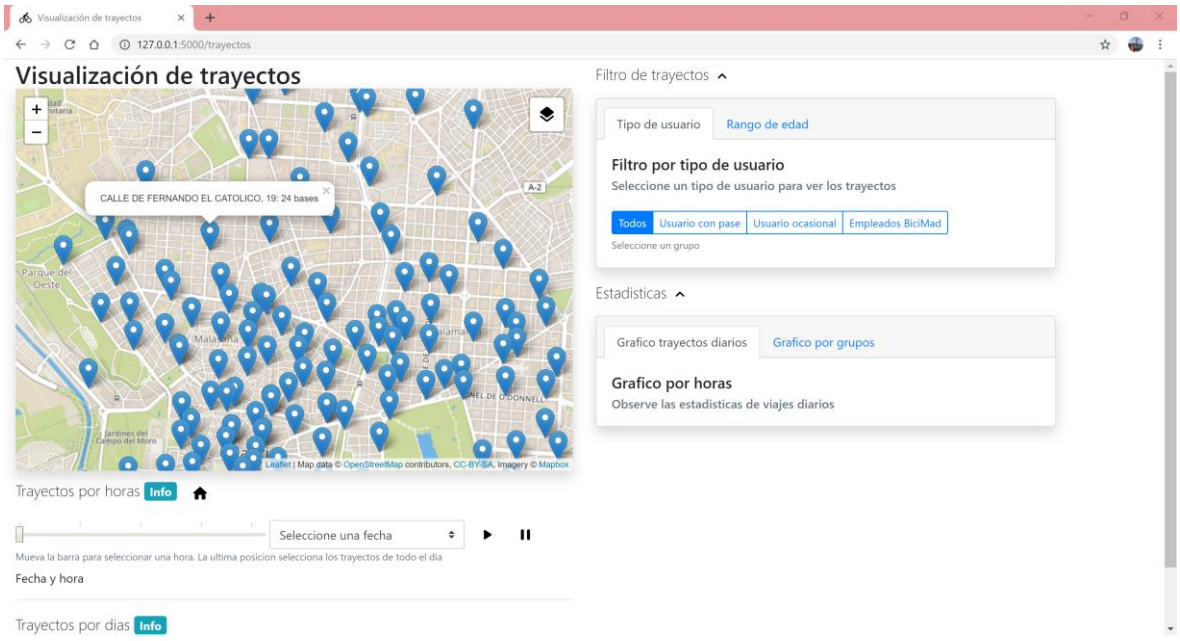
En este apartado se mostrarán los resultados de las diferentes visualizaciones de la herramienta, que se dividirán en los correspondientes a la visualización de los datos de cobertura del servicio y los datos de trayectos.

### 6.2.1 RESULTADOS DE LA VISUALIZACIÓN DE LA COBERTURA DEL SERVICIO

En la herramienta se incluyen tres visualizaciones para la cobertura del servicio que ofrecen distintas perspectivas. Todos los datos de las visualizaciones se cargan cuando se accede a la aplicación y se organizan en sus respectivas capas, para permitir un acceso rápido a la información desde el mapa.

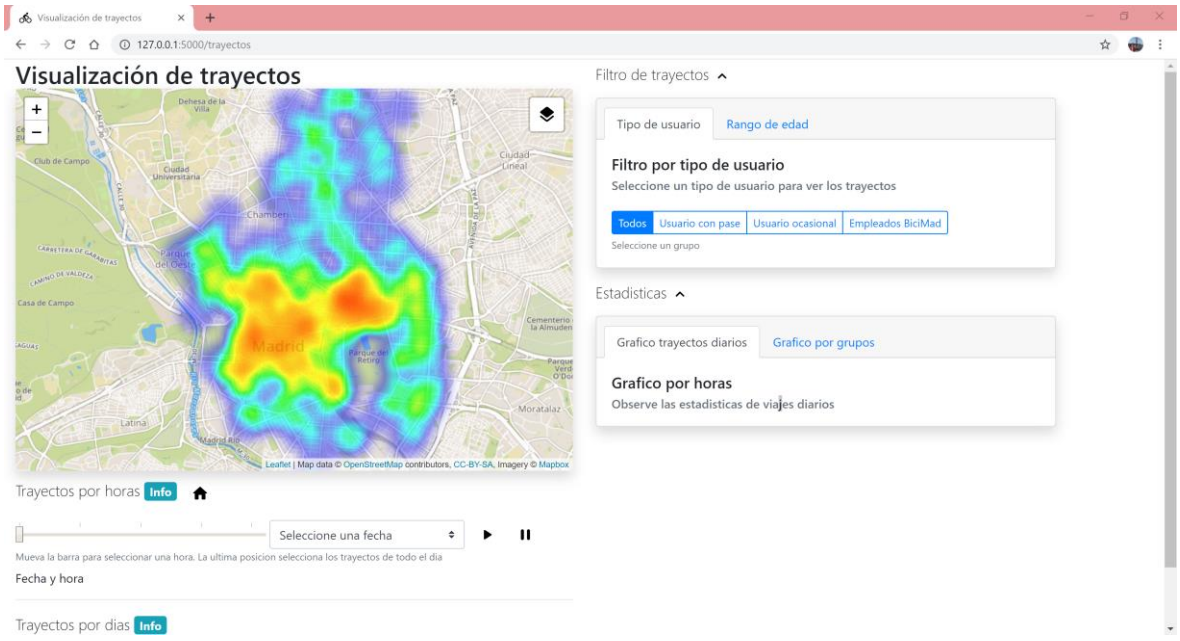
En la visualización de las estaciones con marcadores (Ilustración 27), se ofrece una visión de la cobertura del servicio en el que cada estación se ha representado con un marcador en el que se puede pulsar para acceder a la información de la estación, como la dirección o el número de bases de carga.

*Ilustración 27. Visualización de la localización de las estaciones*



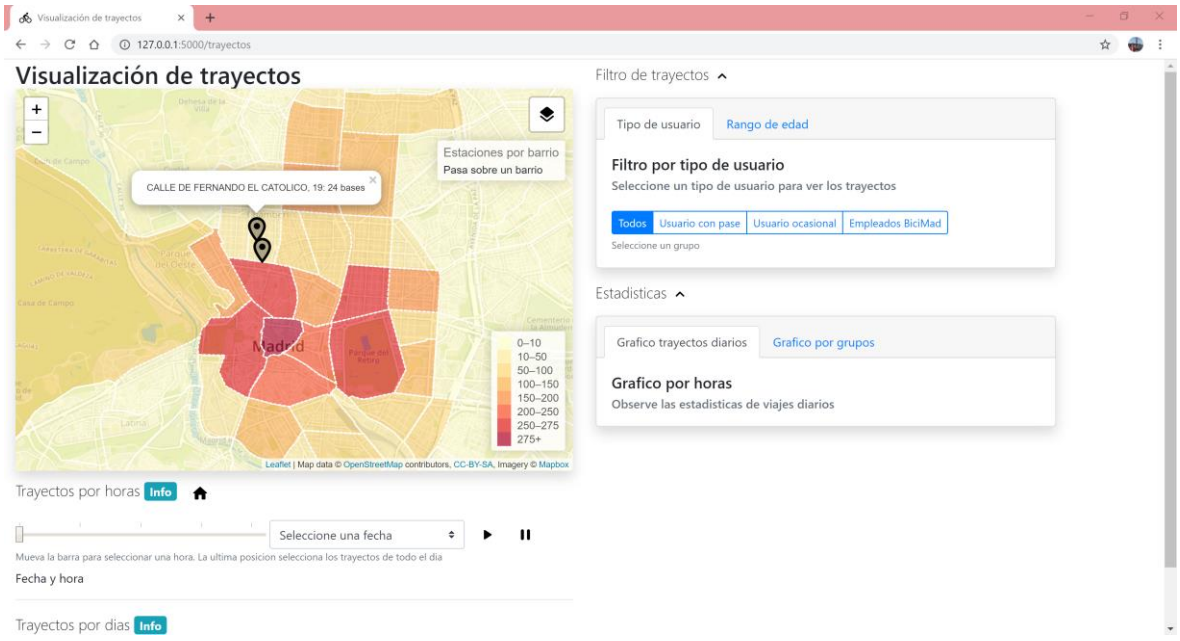

En la Ilustración 28 se incluye el resultado de la visualización de las estaciones en un mapa de calor. Según el número de bases de la estación se le ha asignado mayor intensidad de color al punto. Así, se permite ver la concentración general de las estaciones, basándose en el número de bases de las mismas y su cercanía a otras.

*Ilustración 28. Visualización de la distribución de las estaciones*




Para ofrecer una visión de la cobertura de estaciones específica en la ciudad, se desarrolló la tercera visualización, que se presenta en la Ilustración 29, que permite estudiar que barrios cuentan con un mayor número de bases de carga. Además, se ha incluido la posibilidad de visualizar la localización de las estaciones dentro de un barrio haciendo uso del menú desplegable.

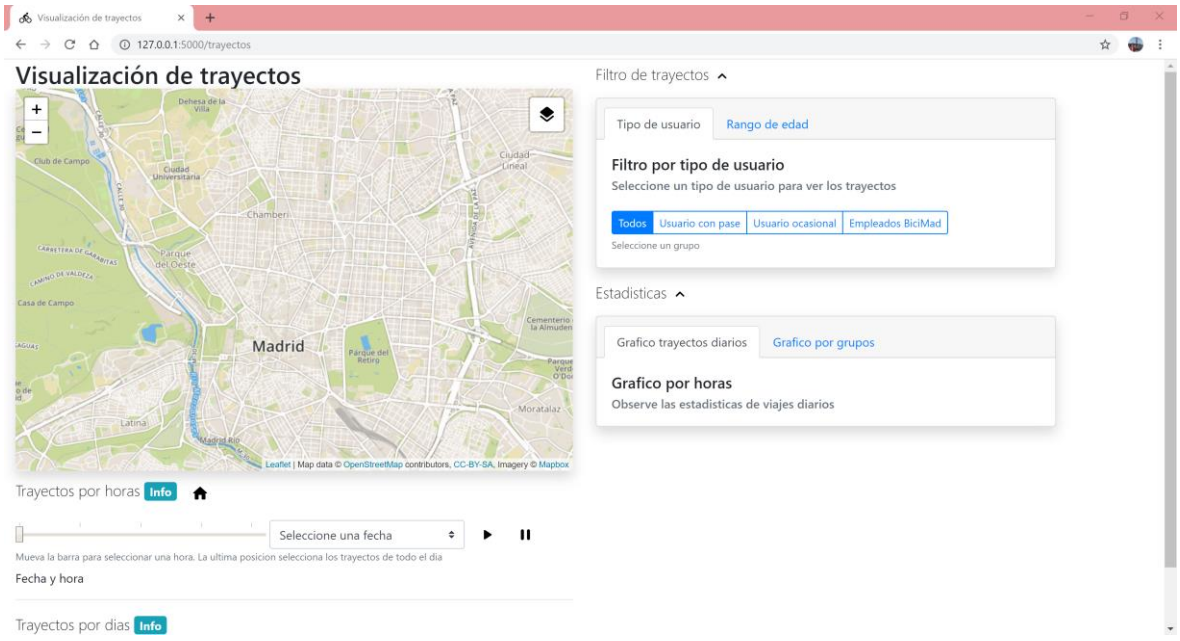
*Ilustración 29. Visualización de la cobertura del servicio por barrios*




### 6.2.2 RESULTADOS DE LA HERRAMIENTA Y DE LA VISUALIZACIÓN DE LOS TRAYECTOS

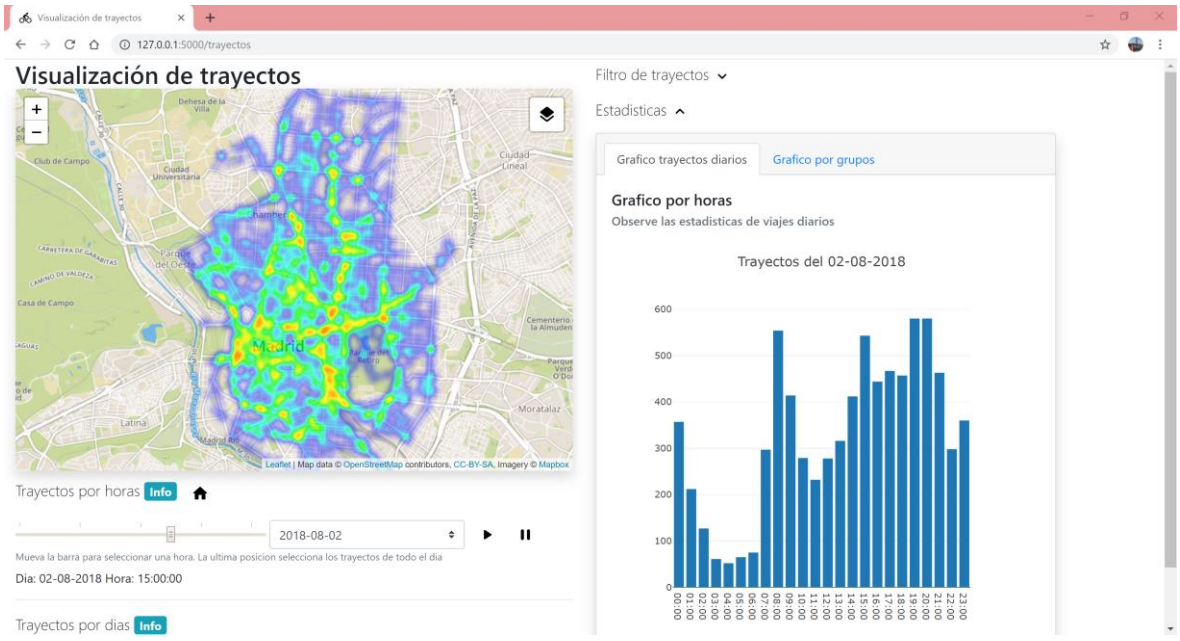
Al ser la visualización de datos de trayectos más compleja, ha sido necesario incluir opciones específicas en una interfaz fuera del mapa, para permitir el control del filtrado de los datos y el control de la reproducción de la animación de los trayectos. La interfaz de la aplicación se presenta en la Ilustración 30, donde se puede observar el resultado obtenido. Los elementos de la interfaz se pueden minimizar para controlar el contenido que se quiera ver en cada momento y están divididos por funcionalidades, por un lado, las opciones de filtrado, por otro las opciones de reproducción de la animación y por último la tarjeta con información estadística de los trayectos.

*Ilustración 30. Vista de la aplicación*



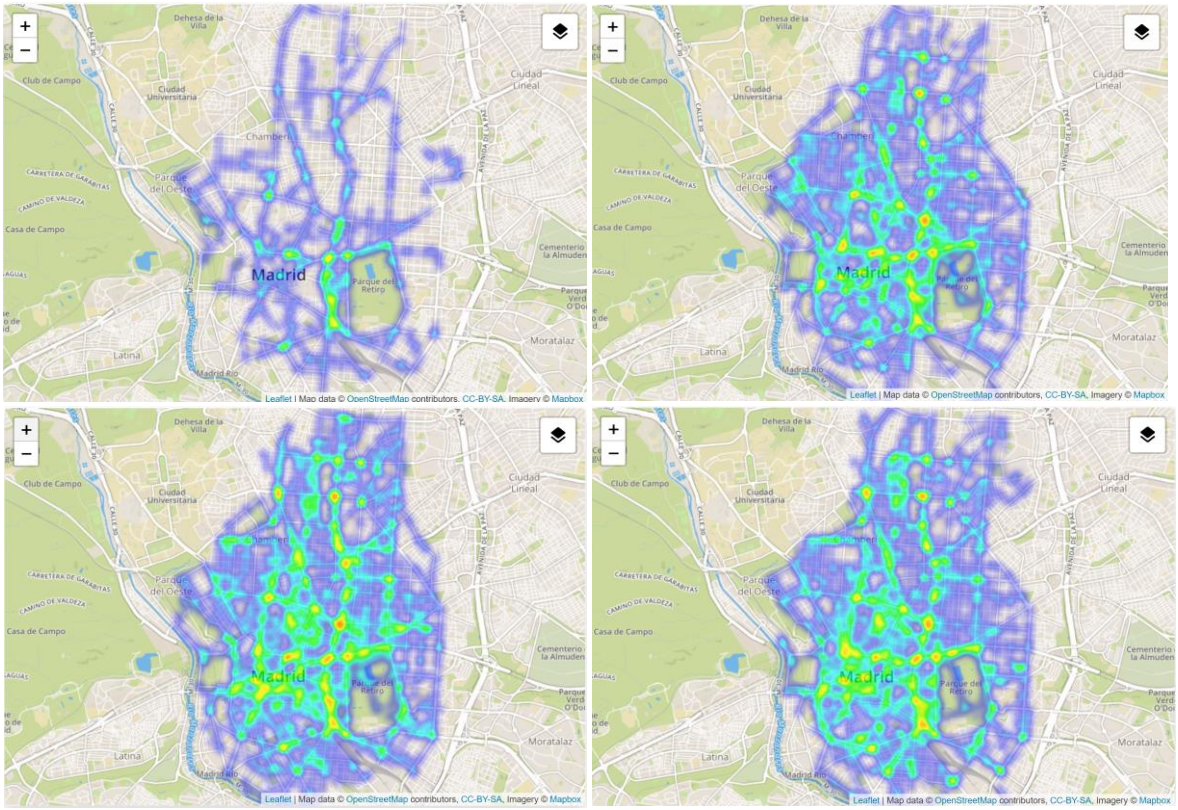

Cuando la aplicación esté en uso y se seleccione un conjunto de trayectos para visualizar, se mostrará como en la Ilustración 31, permitiendo ver la concentración de los trayectos en el mapa y un resumen del día en forma de gráfico de barras.

*Ilustración 31. Vista en uso de la aplicación*



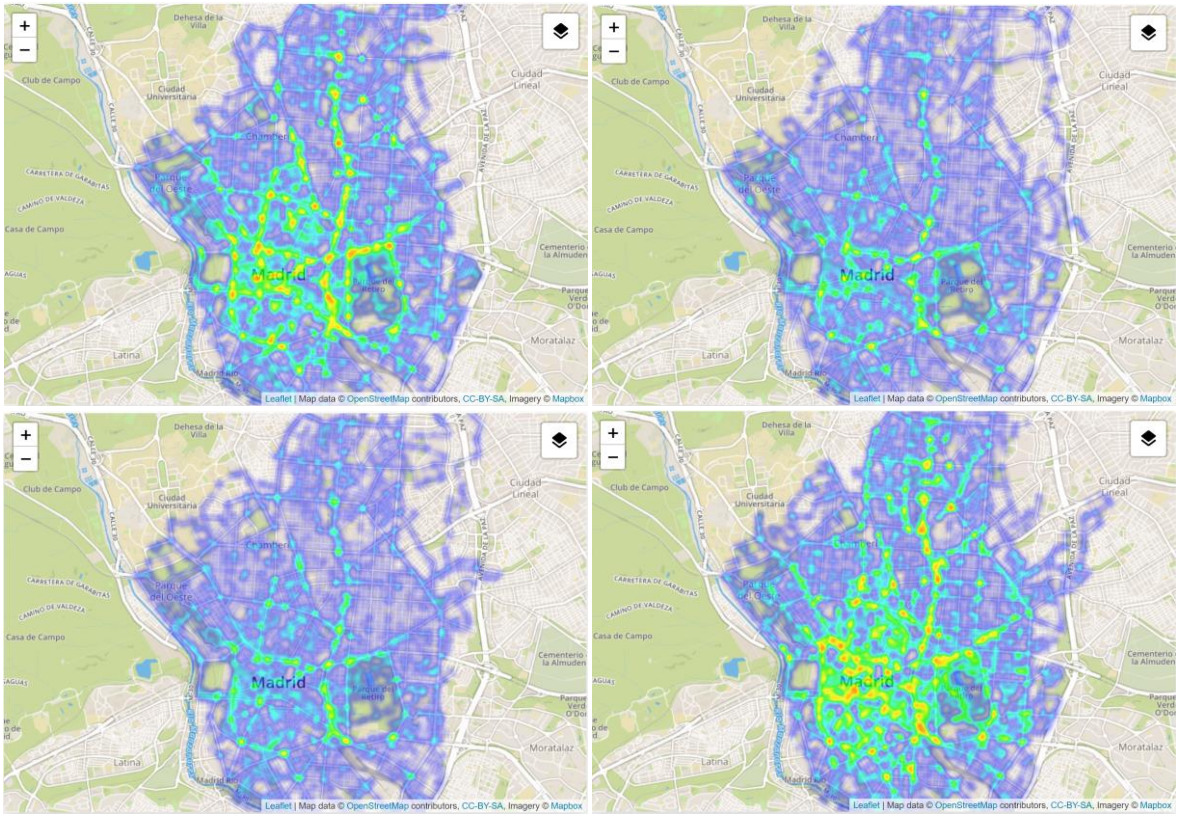
Con la introducción de la reproducción interactiva de los trayectos, se amplían las posibilidades de visualización, permitiendo observar la evolución temporal de los trayectos a lo largo de las horas del día y entre días. En la Ilustración 32 se puede observar un ejemplo comparativo entre los trayectos a diferentes horas del día, desde las 6:00 hasta las 9:00, de izquierda a derecha, pudiéndose observar cómo cambia el flujo de trayectos, especialmente a partir de las 7:00 en la imagen superior derecha.

*Ilustración 32. Visualización de la evolución de los trayectos desde las 6:00 hasta las 9:00. 1-08-2018*




La misma reproducción puede ser usada para estudiar diferencias entre días, como se puede ver en la Ilustración 33, en la que se representan los trayectos diarios desde el viernes 3 de agosto al lunes 6 de agosto de izquierda a derecha. Como se puede ver, hay diferencias notables entre los días de diario y el fin de semana, habiendo en este caso particular la mitad de viajes un día del fin de semana que en un día de diario.

*Ilustración 33. Visualización de la evolución de los trayectos. Viernes 3 - Lunes 6 de agosto*




La aplicación también permite el estudio de los viajes hechos por diferentes grupos de usuarios. En la Ilustración 34 se presentan las visualizaciones de los trayectos realizadas por los usuarios con pase anual (401 trayectos, imagen superior izquierda), por los usuarios ocasionales (12 trayectos, imagen superior derecha) y los empleados del servicio de alquiler de bicicletas (58 trayectos, imagen inferior) a las 16:00 el 1 de agosto de 2018. Como se puede observar, hay una disminución sustancial del número de trayectos para los usuarios ocasionales y los empleados con respecto a los usuarios con pase anual. Esta situación se mantiene durante todo el día, circunstancia que puede estudiarse más fácilmente observando el diagrama de barras agrupado del día. En la Ilustración 35 se muestra a modo de ejemplo el diagrama de barras agrupado de los días 1 y 3 agosto de 2017, donde se observa esta situación.

*Ilustración 34. Comparación de trayectos por tipos de usuario. 16:00 del 1-08-2018*

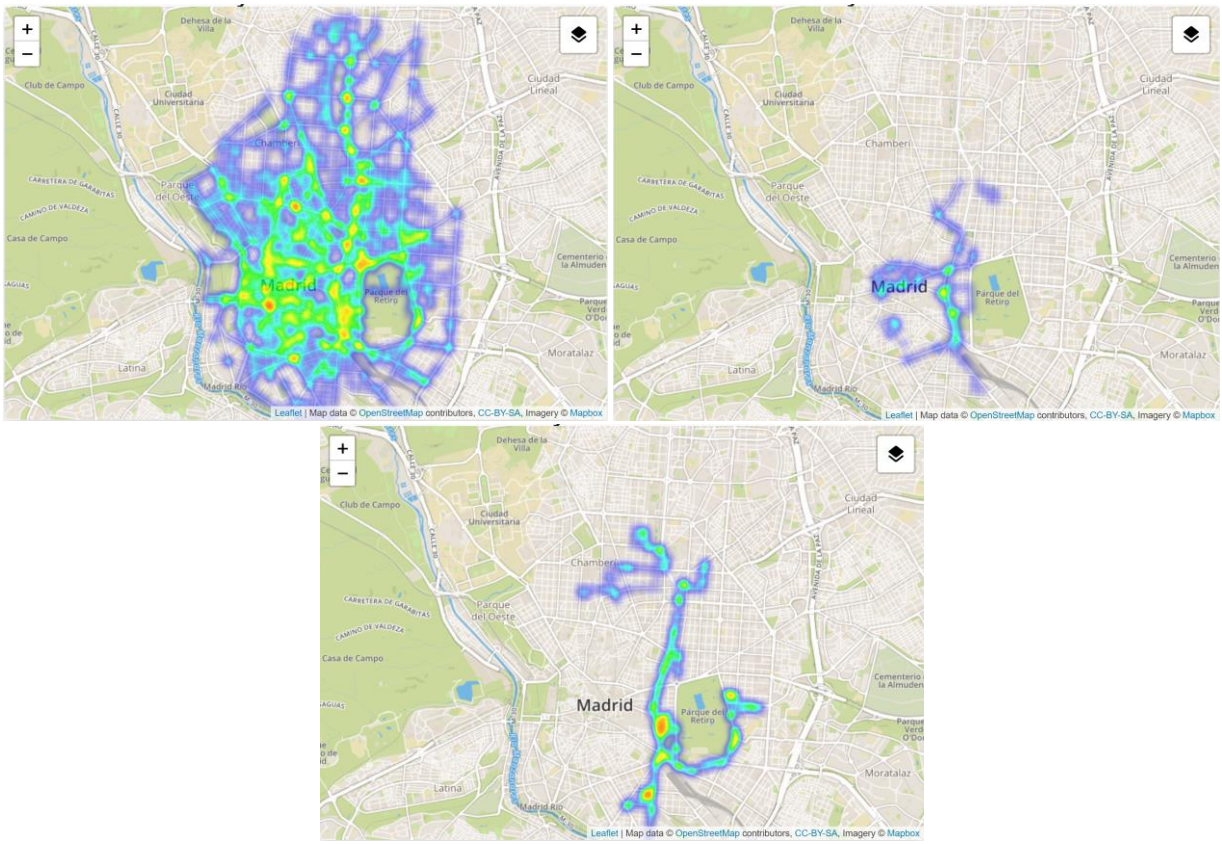
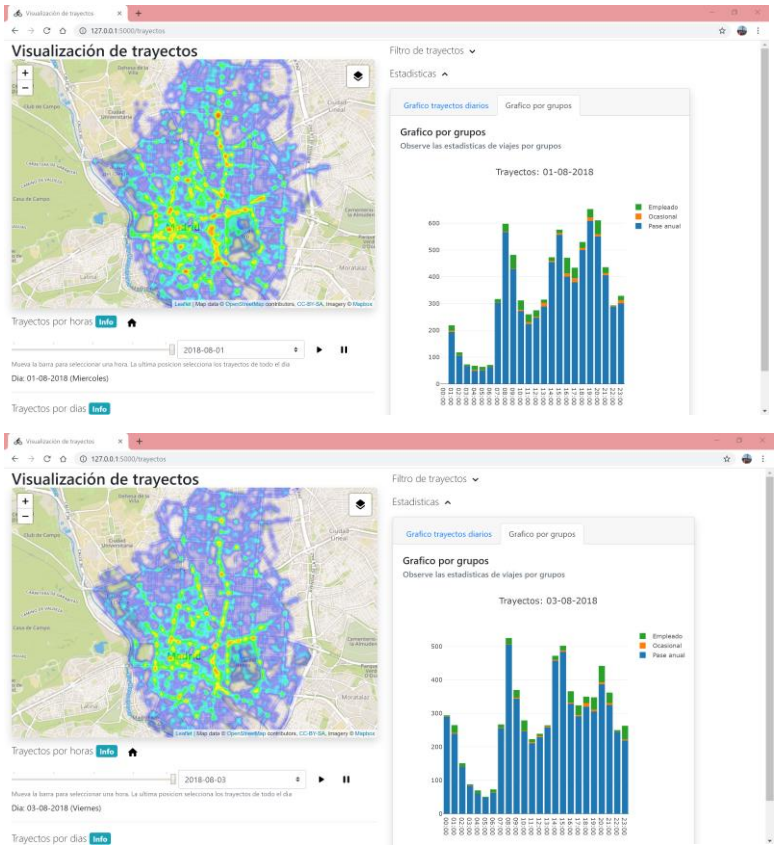
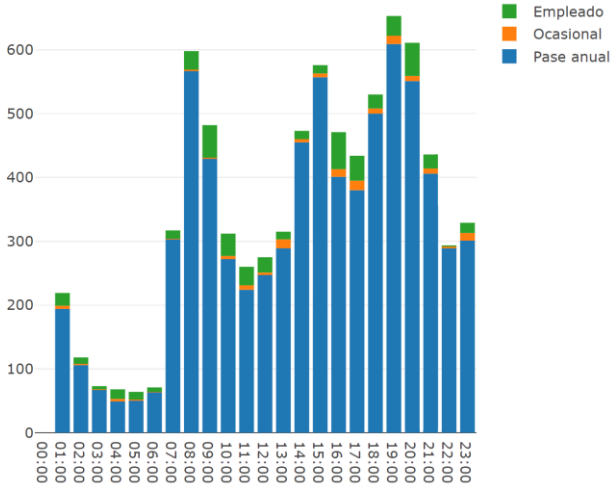


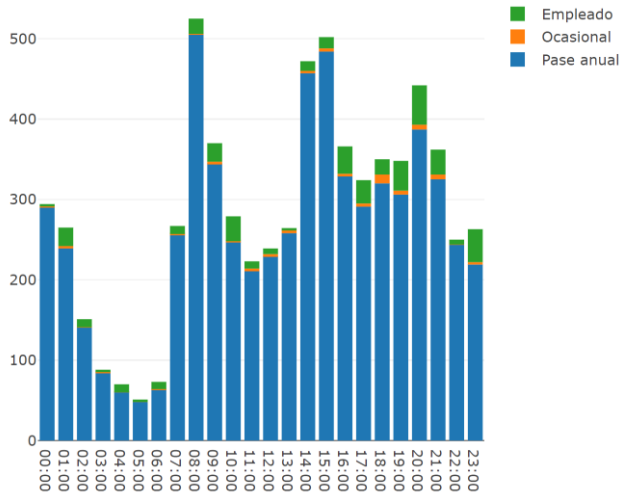
*Ilustración 35. Trayectos por horas agrupados por tipo de usuario. 1-08-2018, 3-08-2018*




Trayectos: 01-08-2018




Trayectos: 03-08-2018



# Capítulo 7. CONCLUSIONES

Como se describió en el apartado 4.3, se ha seguido una metodología de desarrollo iterativa, centrada en el desarrollo cíclico de nuevas funciones y su incorporación a la aplicación principal. Dado el carácter del desarrollo, que ha consistido en trabajar sobre varios conjuntos de datos, ampliando cada vez más la funcionalidad ofrecida hasta llegar a la meta propuesta, esta forma de trabajo basada en ciclos cortos de desarrollo ha sido fundamental para lograr la consecución de los objetivos planteados.

Todos los objetivos planteados inicialmente para el proyecto se han cubierto en el desarrollo: se ha realizado un análisis inicial de los conjuntos de datos que se iban a usar, que ha permitido conocer su estructura y planificar las tareas posteriores de adaptación y visualización, se han estudiado diferentes herramientas de visualización en función de los resultados que se querían conseguir y se ha desarrollado la aplicación, que cuenta con diferentes módulos que satisfacen todas las características previstas, como la visualización de la cobertura del servicio de alquiler de bicicletas y la visualización de los trayectos. Por último se ha desarrollado la interfaz, dando forma a la herramienta incorporando todas las opciones de control dinámico de la representación de los datos y el filtrado.

# Capítulo 8. REFERENCIAS

[1] A. Imawan, F. Putri y J. Kwon, «TiQ: A Timeline Query Processing System over Road Traffic Data,» de *IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, Chengdu, 2015.

[2] J. Huete, «La EMT mejorará su movilidad urbana gracias al Big Data de Moovit,» 30 Octubre 2018. [En línea]. Available: https://www.innovaspain.com/emt-movilidad-urbana-big-data-moovit/.

[3] M. Bostock, «D3 Documentation,» 22 Marzo 2018. [En línea]. Available: https://github.com/d3/d3/wiki.

[4] V. Agafonkin, «Leaflet,» [En línea]. Available: https://leafletjs.com/.

[5] Bootstrap Team, «Bootstrap,» [En línea]. Available: https://getbootstrap.com/.

[6] A. Ronacher, «Flask,» [En línea]. Available: http://flask.pocoo.org/.

[7] G. Romanillos, B. Moya-Gómez, M. Zaltz-Austwick y P. J. Lamíquiz-Daudén, «The pulse of the cycling city: visualising Madrid bike share system GPS routes and cycling flow,» *Journal of Maps*, vol. 14:1, pp. 34-43, 2018.

[8] H. Zhiyuan, Z. Liang, X. Ruihua y Z. Feng, «Application of big data visualization in passenger flow analysis of Shanghai Metro network,» de *2nd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, Singapur, 2017.