

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERÍA DEL SOFTWARE

**APLICACIÓN HÍBRIDA PARA LA AYUDA A LA BÚSQUEDA DE
APARCAMIENTO**

A HYBRID APPLICATION FOR PARKING SEARCH SUPPORT

Realizado por
FRANCISCO JOSÉ REQUENA GARRIDO
Tutorizado por
EDUARDO GUZMÁN DE LOS RISCOS
Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JULIO 2017

Fecha defensa:
El Secretario del Tribunal

Resumen: Se ha desarrollado una aplicación móvil híbrida capaz de presentar datos en tiempo real sobre la ocupación de los aparcamientos de la ciudad de Málaga, de los que el Ayuntamiento ofrece información mediante consulta de una de sus APIs, incluyendo no sólo información sobre turismos, sino también sobre el nuevo sistema de transporte en bicicleta ofrecido por la EMT “málagabici”. Dicha aplicación también ofrece datos relativos a los precios de cada aparcamiento según el día actual, y la distancia y el tiempo de viaje según el tipo de transporte, exceptuando los precios para las bicicletas. También muestra en un mapa la localización actual y la del aparcamiento seleccionado, así como en un mapa en un apartado diferente, las localizaciones de los cortes de tráfico en la ciudad. Finalmente se ha desarrollado un módulo de aprendizaje automático con Azure para realizar predicciones sobre la ocupación de los aparcamientos de coches y bicicletas dado el día de la semana y la hora de interés.

Palabras clave: aplicación móvil híbrida, aprendizaje automático, MS Azure, datos abiertos.

Abstract: It has been developed a hybrid mobile application that is able to present real time information about parking occupations in Málaga, about which the Town hall offers information through the use of queries to one of its APIs, including not only information about cars, but also about the new offered EMT transport system called “málagabici”. It shows in a map the current location, the selected parking location, and it also shows the traffic cuts locations in another map in another section. This application also offers data relative to the price of each parking according to the current day, and the distance and travel time depending on the transport type, except for bicycle prices. Finally, it has developed a machine learning module with Azure to predict the bicycle and car occupation given the day of the week and the chosen hour.

Keywords: hybrid mobile application, machine learning, MS Azure, Open Data.

ÍNDICE

1 INTRODUCCIÓN	5
1.1 ANTECEDENTES	5
1.2 OBJETIVOS	6
2 TECNOLOGÍAS Y HERRAMIENTAS EMPLEADAS	7
2.1 HERRAMIENTAS UTILIZADAS	7
2.1.1 GITLAB	7
2.1.2 ORACLE DATAMODELER	7
2.1.3 VISUAL STUDIO CODE	7
2.1.3 NETBEANS	8
2.2 TECNOLOGÍAS EMPLEADAS	8
2.2.1 JAVA EE	8
2.2.2 IONIC 2	8
2.2.3 AZURE MACHINE LEARNING STUDIO	9
2.2.4 ANGULAR JS	10
3 ANÁLISIS DE REQUISITOS	11
3.1 ANÁLISIS	11
3.1.1 FICHERO DE APARCAMIENTOS DE SMASSA	11
3.1.2 FICHERO DE ESTACIONAMIENTOS DE BICICLETAS DEL SERVICIO MÁLAGABICI	12
3.2 CONSIDERACIONES PREVIAS	13
3.3 REQUISITOS FUNCIONALES	14
3.3.1 REQUISITOS DE LA APLICACIÓN WEB JAVA EE	14
3.3.2 REQUISITOS DE LA API REST	15

3.3.3 REQUISITOS DE LA APLICACIÓN MÓVIL.....	16
4 CASOS DE USO.....	19
5 METODOLOGÍA	25
6 DISEÑO	27
7 IMPLEMENTACIÓN	31
7.1 APLICACIÓN WEB JAVA EE.....	31
7.1.1 CONEXIÓN CON LA BASE DE DATOS	31
7.1.2 IMPLEMENTACIÓN DE LA APLICACIÓN	33
7.2 CREACIÓN DE LA API REST.....	34
7.2.1 CREACIÓN DE LOS SERVICIOS	34
7.2.2 CREACIÓN DE LOS MÉTODOS PARA EL SERVICIO DE APARCAMIENTO DE SMASA.....	35
7.2.3 CREACIÓN DE LOS MÉTODOS PARA EL SERVICIO DE ESTACIONAMIENTOS DE BICICLETAS	37
8 CREACIÓN DEL MODELO PREDICTIVO EN AZURE	39
8.1 CREACIÓN DEL EXPERIMENTO	39
8.2 DESPLIEGUE DEL SERVICIO	50
9 DESARROLLO DE LA APLICACIÓN MÓVIL	53
9.1 CREACIÓN DEL PROYECTO VACÍO.....	53
9.2 LISTADO DE LOS APARCAMIENTOS DE SMASA Y ESTACIONAMIENTOS DE BICICLETAS	53
9.3 ACTUALIZACIÓN DE LOS DATOS PERIÓDICAMENTE.....	55
9.4 CREACIÓN DE LA PÁGINA DE DETALLES	56
9.5 BÚSQUEDA POR RADIO.....	58

9.6 VISUALIZACIÓN DE LOS CORTES DE TRÁFICO	60
9.7 OBTENCIÓN DE PRECIOS PARA LOS APARCAMIENTOS DE SMASSA	61
9.7 OBTENCIÓN DE LAS DISTANCIAS Y TIEMPO DE VIAJE.....	62
9.8 REALIZACIÓN DE PREDICCIONES	63
9.9 AÑADIENDO IMÁGENES A LOS THUMBNAILS.....	64
10 CONCLUSIONES	67
10.1 CONCLUSIONES PERSONALES Y ACADÉMICAS	67
10.2 FUTURAS MEJORAS	68
BIBLIOGRAFÍA.....	69

1 INTRODUCCIÓN

En este apartado se realizará una exposición del contenido de esta memoria, así como sus antecedentes, motivación, objetivos y de las conclusiones obtenidas mediante el desarrollo de este trabajo.

1.1 ANTECEDENTES

En la actualidad, es muy común que cada unidad familiar se encuentre con más de un vehículo en su posesión. Este hecho demuestra una mejora en la economía y el fácil acceso que tienen las personas de diversos estatus sociales a la compra de un vehículo (quedando la gama del mismo en otro plano de estudio).

Aunque este hecho es, en general, beneficioso para la población ya que facilita el desplazamiento de los individuos, presenta un problema que se acentúa, sobre todo, en las grandes ciudades: el estacionamiento de los mismos. Este hecho que, a priori, podría traducirse únicamente en el tiempo que pierden las personas a la hora de buscar aparcamiento, si se analiza en profundidad, presenta otros graves inconvenientes: primero, que en las horas punta, prolonga las horas de tráfico y caravana, puesto que la ubicación de una plaza de aparcamiento no es inmediata y normalmente requiere de varias vueltas hasta su localización. Segundo, como consecuencia de lo anteriormente expuesto, se prolonga el tiempo de emisiones que contribuyen al efecto invernadero. Dicha problemática, en un siglo en el que la gente está tan concienciada con el cuidado del medio ambiente y en el que los gobiernos aprueban políticas para reducir la cantidad de emisiones, la informática juega un papel fundamental en colaboración con otros medios que buscan disminuir la cantidad de emisiones perjudiciales para la capa de ozono.

Es muy importante comentar también otro factor que influirá en el desarrollo de la aplicación que compete a dicho caso de estudio: la constante evolución de las llamadas “*smart cities*”. Estas ciudades inteligentes ofrecen una cantidad inmensa de datos en tiempo real del estado de la ciudad y que pueden ser aprovechados por diversos sistemas informáticos para múltiples usos.

En lo que concierne al desarrollo de la aplicación sobre la que versa dicho documento, se hará uso de los datos en abierto que ofrece el Ayuntamiento de Málaga.

1.2 OBJETIVOS

El objetivo principal de este trabajo es desarrollar una aplicación móvil híbrida, capaz de encontrar los parkings públicos más cercanos con plazas de aparcamiento libre, de los que se pueden obtener los datos mediante consultas a los servicios públicos ofrecidos por el Ayuntamiento de Málaga. El usuario podrá visualizar en tiempo real, información como la ocupación de cada uno de los parkings y la distancia hacia los mismos desde su posición actual, podrá seleccionar cualquiera de ellos y se establecerá la ruta a su destino. Además, la información de la ocupación del aparcamiento seleccionado se irá actualizando.

Finalmente, y puesto que recientemente se han construido múltiples puestos para el alquiler y estacionamiento de bicicletas del ayuntamiento, y fomentar así un medio de transporte más sano y respetuoso con el medio ambiente. Se intentará extender el ámbito de la aplicación para que sirva también para facilitar el estacionamiento de dichas bicicletas, con las mismas funcionalidades que la versión dedicada al estacionamiento de turismos.

El sistema contará, además, con un sistema de predicción de ocupación que estimará el número de plazas libres para un aparcamiento dado el día de la semana y la hora para las que se quiere dicha predicción.

2 TECNOLOGÍAS Y HERRAMIENTAS EMPLEADAS

En este apartado se justificará el uso de las tecnologías empleadas en este proyecto, así como los objetivos de cada una de ellas.

2.1 HERRAMIENTAS UTILIZADAS

2.1.1 GITLAB

Supone una alternativa a los repositorios de GitHub, que además de contar con sus mismas características y ser compatible con múltiples entornos de desarrollo que poseen herramientas específicas para su inclusión en los proyectos que contienen, cuenta con una característica esencial para muchos desarrolladores sin amplios recursos: proporciona repositorios privados gratuitos.

Este hecho motivó su uso frente a otros repositorios “gratuitos”, ya que los repositorios privados que ofrece GitLab no poseen ningún tipo de restricción frente a los públicos y sin perder ninguna funcionalidad.

2.1.2 ORACLE DATAMODELER

A pesar de que el sistema gestor de bases de datos es MySQL, se ha utilizado este editor para diseñar los modelos relacionales para un esquema de dicho tipo, por la facilidad de uso y la familiarización con el mismo. Ha sido escogido para realizar dicha tarea, aunque los scripts resultantes hayan tenido que ser modificados levemente a mano para que cumplan con la sintaxis permitida por MySQL.

En este documento se hará referencia al modelo de clases y objetos java en consonancia con el modelo de la base de datos, debido a la fuerte cohesión que existe entre los mismos, por lo que, para mostrar las relaciones entre los distintos objetos java, se mostrará el modelo lógico generado para la base de datos.

2.1.3 VISUAL STUDIO CODE

Por su ligereza y su interfaz simple y sencilla, este editor de código destaca frente a otros más pesados y rudimentarios, haciendo de la programación una experiencia más agradable ya que cuenta con innumerables extensiones que facilitan el desarrollo software para multitud de lenguajes de programación y frameworks diferentes. Este último hecho fue

determinante en la elección de dicho editor frente a otros potentes como Atom, WebStorm o Sublime Text, ya que era uno de los que ofrecían más ayudas y utilidades para la programación haciendo uso de Angular 2 y el framework Ionic.

2.1.3 NETBEANS

El uso de este entorno de programación a lo largo del grado, así como que éste constituya uno de los IDEs más potentes, sobre todo para la programación web, fue un hecho decisivo a la hora de elegir el entorno de trabajo más adecuado para la ejecución de esta tarea, ya que ofrece multitud de ayudas que reducen enormemente el tiempo de trabajo gracias a múltiples herramientas que facilita la generación de código automático.

2.2 TECNOLOGÍAS EMPLEADAS

2.2.1 JAVA EE

Plataforma de desarrollo de aplicaciones web basada en Java. Elegida por la amplia información distribuida en la web sobre la misma y la facilidad de uso frente a otros lenguajes tradicionales.

2.2.2 IONIC 2

Framework de desarrollo de aplicaciones móviles híbridas para Android y iOS. Por su facilidad de uso, los ejemplos disponibles en su web y su fácil integración, fue elegido como eje principal del trabajo del que se desarrolla dicha exposición. Basado en HTML5, CSS Y JS, está construido en Sass (Syntactically Awesome Style Sheets) y optimizado con Angular JS.

Para entender su funcionamiento, habrá que dar un repaso a conceptos básicos y muy importantes en la programación como es el uso del patrón MVC.

El MCV (Modelo-Vista-Controlador) es un patrón de diseño que se emplea para separar los datos de la lógica y de las interfaces de usuario. Como puede deducirse, está dividido en tres componentes: Modelo, Vista y Controlador.

- Modelo. Capa encargada de los datos. Realiza peticiones a bases de datos alojadas de forma local en nuestra aplicación o de forma remota en un servidor externo en la web.
- Vista. Encargada de mostrar la información al usuario.

- Controlador. Es la parte que se encarga de vincular la vista con el modelo a través de peticiones en los dos sentidos.

Una vez expuesto esto, se pasará a enumerar sus principales características y ventajas frente a otros frameworks de desarrollo de aplicaciones híbridas:

- Alto rendimiento, debido a la mínima manipulación del DOM, entre otros
- Uso de Angular JS. Debido a la robustez de este framework de desarrollo de aplicaciones web, su empleo en la implementación de aplicaciones híbridas hace que estas se vean bastante sencillas, abstrayendo al programador de ciertos elementos que dificultan algunos aspectos en la programación de manera transparente al mismo, dando como resultado aplicaciones con menor código que las nativas y, como característica adicional propia de Angular JS, robustas.
- Diseño Responsive. Ionic adapta el contenido según las características de la pantalla del dispositivo donde se esté visualizando.
- Ejecución nativa. Ionic emplea los SDK nativos de desarrollo móviles de las plataformas más importantes. De forma resumida, esto se traduce en un “único desarrollo y compilación para varios”.
- Diseño atractivo. La gran variedad de componentes altamente editados que posee Ionic hace posible que la codificación de la interfaz gráfica de usuario sea mínima, pues la mayoría de sus componentes son muy atractivos y lo mejor de todo, es que muchos de ellos se visualizan de manera diferente dependiendo de la plataforma en la que se esté ejecutando la aplicación, es decir, un mensaje de notificación, por ejemplo, se visualizará de manera diferente en un dispositivo Android que en otro dispositivo iOS.

2.2.3 AZURE MACHINE LEARNING STUDIO

Azure es una herramienta que ofrece Microsoft para el desarrollo de proyectos o experimentos de aprendizaje automático muy potente, tal como se mostrará a lo largo de este trabajo.

Una de sus principales características es su simplicidad a la hora de usarlo, ya que no hay que tener conocimientos informáticos y de programación, puesto que se basa en arrastrar diferentes módulos a una hoja de trabajo y conectarlos entre sí, como será explicado

posteriormente. Para usarlo sólo se necesitan ciertos conocimientos matemáticos sobre estadística y los datos que son objeto de estudio.

2.2.4 ANGULAR JS

AngularJS es un framework de javascript para el desarrollo de aplicaciones web que usa el patrón MVW (Model- View-Whatever). Contiene un conjunto de librerías de código abierto útiles para el desarrollo de aplicaciones web avanzadas en el lado del cliente, sobre todo para SPA (Single-Page Applications).

Está apoyado por Google, lo cual explica que cada vez más desarrolladores lo estén usando y que tenga un futuro prometedor.

3 ANÁLISIS DE REQUISITOS

3.1 ANÁLISIS

Previamente a la enumeración de los requisitos, es necesario realizar una introducción del entorno de trabajo en el que se va a desarrollar este proyecto.

Como se dijo anteriormente, los datos se van a obtener de una API que el Ayuntamiento de Málaga pone a disposición de quien la solicite de manera gratuita. Dichos datos pueden obtenerse en su mayoría, de dos formas: en formato CSV (Comma-Separated Values) o en formato JSON (JavaScript Object Notation), aunque también existen otros formatos disponibles, como pueden ser KML (Keyhole Markup Language) o GeoJSON, e incluso ofrece la posibilidad de incrustar cierto contenido web en las páginas de terceros mediante la incrustación de HTML.

En este caso concreto, los formatos interesados son el CSV y el JSON, si bien este último será de gran utilidad en la aplicación móvil debido a su facilidad de uso en el ecosistema de Ionic.

Los datos objeto de estudio en esta primera versión de la aplicación serán los relativos a la ocupación de estacionamientos de bicicletas y aparcamientos de Málaga, de los que se pueden obtener datos mediante sensores que, de ahora en adelante, llamaremos aparcamientos de SMASSA. Ambos recursos son actualizados cada minuto. Una vez expuesto esto, se procederá al análisis de los datos contenidos en los ficheros y/o enlaces de interés para establecer unos requisitos principales.

3.1.1 FICHERO DE APARCAMIENTOS DE SMASSA

Sin más dilación, se procederá a enumerar los datos disponibles en el fichero nombrado en el título de este apartado y descritos en la página de datos abiertos del Ayuntamiento de Málaga (<http://datosabiertos.malaga.eu/dataset/ocupacion-aparcamientos-smassa/resource/0dcf7abd-26b4-42c8-af19-4992f1ee60c6>):

- **poiID** (Numérico): Identificador único del aparcamiento
- **nombre** (Cadena): Nombre del aparcamiento
- **dirección** (Cadena): Dirección del aparcamiento

- **telefono** (Numérico): Teléfono de contacto
- **correoelectronico** (Cadena): Correo electrónico de contacto
- **latitude** (Numérico): Coordenada geográfica de la ubicación del aparcamiento
- **longitude** (Numérico): Coordenada geográfica de la ubicación del aparcamiento
- **altitud** (Numérico): Coordenada geográfica de la ubicación del aparcamiento
- **capacidad** (Numérico): Número de plazas totales
- **capacidad_discapacitados** (Numérico): Número de plazas totales para discapacitados
- **fechahora_ultima_actualizacion** (Cadena): Representación textual de la fecha y la hora en la que se han tomado los datos del aparcamiento
- **libres** (Numérico): Número de plazas libres
- **libres_discapacitados** (Numérico): Número de plazas libres para discapacitados
- **nivelocupacion_naranja** (Numérico): Nivel de ocupación naranja
- **nivelocupacion_rojo** (Numérico): Nivel de ocupación rojo
- **smassa_sector_sare** (Numérico): Sector SARE

Los datos que se encuentran en cursiva son aquellos que se han utilizado, entre otros motivos, porque el resto suelen estar incompletos y no son de utilidad en nuestro uso, aunque la razón principal es la expuesta en primer lugar.

3.1.2 FICHERO DE ESTACIONAMIENTOS DE BICICLETAS DEL SERVICIO MÁLAGABICI

Como se ha hecho anteriormente, se procederá a una enumeración y descripción de los distintos datos ofrecidos en la web de recursos del Ayuntamiento.

- **ID** (Numérico): Identificador del estacionamiento
- **NOMBRE** (Cadena): Nombre del estacionamiento
- **NOMBRE_CIUDAD** (Cadena): Nombre de la ciudad (*MÁLAGA* en todos los casos)

- ***DIRECCION*** (Cadena): Dirección del estacionamiento
- **ID_ESTADO** (Numérico): Identificador del estado del aparcamiento
- **NOMBRE_ESTADO** (Cadena): Nombre del estado correspondiente al identificador del mismo
- **NUM_DERBIS** (Numérico): Número correspondiente al mecanismo de anclaje de la bicicleta con el soporte
- **NUM_LIBRES** (Numérico): Número de plazas libres en dicho estacionamiento
- **NUM_OCUPADOS** (Numérico): Número de plazas ocupadas
- **LAT** (Numérico): Coordenada geográfica de la ubicación del estacionamiento
- **LON** (Numérico): Coordenada geográfica de la ubicación del estacionamiento
- **ID_EXTERNO** (Numérico): No ha sido posible encontrar información alguna acerca de este dato, aunque se presupone, será un identificador que relacione dicho estacionamiento con algún elemento externo

Como en el caso anterior, los datos en cursiva son objeto de estudio, pues los otros no añaden valor adicional a la aplicación ni mayor exactitud al modelo predictivo, como se explicará posteriormente.

3.2 CONSIDERACIONES PREVIAS

En este apartado se va a plasmar algunas decisiones que se tomaron después de haber realizado el análisis de los datos extraídos de la página web del Ayuntamiento y que fueron tomadas en cuenta a la hora de realizar la captura de requisitos.

1. **Desarrollo de una aplicación web Java EE para la descarga y almacenamiento de los datos.** Se desarrollará esta aplicación para realizar la descarga de los ficheros CSV con los datos de los aparcamientos de SMASA y los estacionamientos del servicio de bicicletas, su procesamiento y la inclusión de los datos obtenidos en la base de datos que se va a implementar.

2. **Desarrollo de una API REST para dar servicio a la aplicación móvil.** En esta parte no se hará mucho hincapié, pues posteriormente será explicado con detalle la necesidad y el uso por el que fue creada.

3.3 REQUISITOS FUNCIONALES

Se va a proceder a listar los principales objetivos funcionales que han sido definidos para dicho proyecto teniendo en cuenta las consideraciones anteriormente expuestas. Dichos requisitos son los que determinan el funcionamiento de la aplicación y las posibles interacciones que puede tener el usuario con la misma.

Los requisitos serán divididos según los 3 ejecutables que serán generados como fruto de este trabajo.

3.3.1 REQUISITOS DE LA APLICACIÓN WEB JAVA EE

Como se expuso anteriormente, esta aplicación es la encargada de realizar la descarga periódica de los datos y de su inclusión en la base de datos. Su desarrollo es primordial, pues gracias a ella, podremos almacenar una cantidad ingente de datos con los que desarrollar el modelo predictivo en Azure para calcular el número de plazas libres en un aparcamiento de SMASSA y en los estacionamientos de bicicletas. Los requisitos de esta aplicación son los siguientes:

ID	Nombre	Descripción
AW1	Descarga de ficheros	La aplicación descargará y procesará los ficheros referentes a los aparcamientos de SMASSA y los estacionamientos del sistema de bicicletas.
AW2	Inclusión de datos en Base de Datos	La aplicación será capaz de guardar los datos extraídos de los ficheros en una base de datos relacional.
AW3	Tarea periódica	La aplicación realizará RF1 y RF2 periódicamente cada minuto.

3.3.2 REQUISITOS DE LA API REST

Este servicio web no contaba con una serie de requisitos fijos, ya que al haber seguido una metodología “ágil” como será descrita posteriormente, los requisitos de la misma han ido cambiando, traduciéndose, en cualquier caso, en la adición de algunos nuevos, por lo que se expondrán únicamente los contemplados inicialmente y a lo largo del desarrollo de este documento irán apareciendo el resto de los mismos.

ID	Nombre	Descripción
AR1	Obtención de los aparcamientos de SMASSA	El servicio web descargará y procesará los ficheros referentes a los aparcamientos de SMASSA y retornará los resultados obtenidos.
AR2	Obtención de los estacionamientos de bicicletas	El servicio web descargará y procesará los ficheros referentes a los estacionamientos de bicicletas y retornará los resultados obtenidos.
AR3	Búsqueda por radio de aparcamientos de SMASSA	El servicio web retornará los aparcamientos de SMASSA situados a una distancia por radio menor o igual al fijado.
AR4	Búsqueda por radio de estacionamientos de bicicletas	El servicio web retornará los estacionamientos de bicicleta situados a una distancia por radio menor o igual al fijado.
AR5	Obtención de los precios de estacionamiento	El servicio web devolverá el precio relativo a un aparcamiento en el día actual dado los minutos de estacionamiento.
AR6	Obtención de los cortes de tráfico	El servicio web devolverá una lista de las zonas con cortes de tráfico.

3.3.3 REQUISITOS DE LA APLICACIÓN MÓVIL

ID	Nombre	Descripción
AM1	Listado de aparcamientos de SMASSA	La aplicación obtendrá cada minuto el estado de los distintos aparcamientos de SMASSA de los que el Ayuntamiento ofrece información y los listará en la página principal de la aplicación.
AM2	Listado de estacionamientos de bicicletas	La aplicación obtendrá cada minuto el estado de los distintos estacionamientos de bicicletas de los que el Ayuntamiento ofrece información y los listará en una ventana secundaria cuando el usuario la habilite.
AM3	Visualización de los cortes de tráfico	Mediante la pulsación de un botón, la aplicación mostrará los cortes de tráfico de la ciudad en un mapa.
AM4	Visualización de los aparcamientos de SMASSA	El usuario podrá visualizar las localizaciones del aparcamiento seleccionado y de su posición en un mapa.
AM5	Predicción de la ocupación de los aparcamientos de SMASSA	El usuario podrá visualizar predicciones sobre la ocupación de los aparcamientos de SMASSA.
AM6	Predicción de la ocupación de los estacionamientos de bicicletas	El usuario podrá visualizar predicciones sobre la ocupación de los estacionamientos de bicicletas.
AM7	Visualización de los estacionamientos de bicicleta	El usuario podrá visualizar las localizaciones del estacionamiento seleccionado y de su posición en un mapa.

ID	Nombre	Descripción
AM8	Visualización de la distancia y tiempo de viaje	El usuario podrá visualizar la distancia desde su posición hasta el aparcamiento o el estacionamiento seleccionado y el tiempo de viaje según el medio de transporte (coche o bicicleta).
AM9	Cálculo de precios	El usuario podrá calcular el precio del aparcamiento seleccionado en el caso de los coches, según el día actual y los minutos introducidos.
AM10	Guía hacia el aparcamiento o estacionamiento seleccionado	El usuario podrá visualizar un icono en el mapa para ser redirigido a la aplicación de Google Maps y ser guiado al aparcamiento o estacionamiento seleccionado.
AM11	Búsqueda de aparcamientos de SMASSA por radio	El usuario podrá buscar los aparcamientos cercanos al mismo mediante la introducción de un radio de búsqueda.
AM12	Búsqueda de los estacionamientos de bicicletas por radio	El usuario podrá buscar los estacionamientos de bicicleta cercanos al mismo mediante la introducción de un radio de búsqueda.

4 CASOS DE USO

En esta sección se mostrarán una serie de tablas que recogen los principales casos de uso del usuario con la aplicación móvil, puesto que no hay interacción alguna entre el usuario y la aplicación y servicio web desarrollado para complementar a la aplicación móvil.

ID	CU1
Caso de uso	Cortes de tráfico
Descripción	Visualización de cortes de tráfico
Precondición	Estar en el apartado de navegación correspondiente a los coches
Escenario principal	<ol style="list-style-type: none"> 1. El usuario pulsa el botón correspondiente a los cortes de tráfico 2. El usuario es redirigido a otra ventana donde aparece un mapa con su posición en un marcador y la posición del resto de cortes de tráfico en otros marcadores
Escenario alternativo	-
ID requisitos asociados	AM1

ID	CU2
Caso de uso	Visualización de detalles aparcamientos
Descripción	Visualización de detalles de un aparcamiento de SMASSA
Precondición	Estar en el apartado de navegación correspondiente a los coches
Escenario principal	<ol style="list-style-type: none"> 1. El usuario busca el aparcamiento deseado y pulsa el botón correspondiente para visualizar los cortes 2. El usuario es redirigido a una página donde se muestra su posición y la del aparcamiento en un mapa y puede

	visualizar la distancia y el tiempo de viaje en coche
Escenario alternativo	-
ID requisitos asociados	AM4 y AM8

ID	CU3
Caso de uso	Visualización de detalles estacionamientos
Descripción	Visualización de detalles de un estacionamiento de bicicletas
Precondición	Estar en el apartado de navegación correspondiente a las bicicletas
Escenario principal	<ol style="list-style-type: none"> 1. El usuario busca el estacionamiento deseado y pulsa el botón de redireccionamiento 2. El usuario es redirigido a una página donde se muestra su posición y la del estacionamiento en un mapa y puede visualizar la distancia y el tiempo de viaje en bicicleta
Escenario alternativo	-
ID requisitos asociados	AM5 y AM8

ID	CU4
Caso de uso	Cálculo de precios
Descripción	Cálculo de precios para un aparcamiento
Precondición	Estar en el apartado de navegación correspondiente a los coches
Escenario principal	<ol style="list-style-type: none"> 1. El usuario busca el aparcamiento deseado y pulsa el botón de

	<p>redireccionamiento</p> <ol style="list-style-type: none"> 2. El usuario es redirigido a una página donde se muestra su posición y la del aparcamiento en un mapa y puede visualizar la distancia y el tiempo de viaje en coche 3. El usuario busca la entrada de precios, introduce los minutos para los que desea calcular el precio y pulsa el botón correspondiente 4. La aplicación muestra el precio del aparcamiento para los minutos seleccionados 4. La aplicación muestra un resultado de error
Escenario alternativo	
ID requisitos asociados	AM9

ID	CU5
Caso de uso	Pérdida tique
Descripción	Cálculo del precio por la pérdida del tique
Precondición	Estar en el apartado de navegación correspondiente a los coches
Escenario principal	<ol style="list-style-type: none"> 1. El usuario busca el aparcamiento deseado y pulsa el botón de redireccionamiento 2. El usuario es redirigido a una página donde se muestra su posición y la del aparcamiento en un mapa y puede visualizar la distancia y el tiempo de viaje en bicicleta 3. El usuario busca el botón correspondiente a la pérdida del ticket y lo selecciona 4. La aplicación muestra el precio por la pérdida del tique

	correspondiente al aparcamiento seleccionado.
Escenario alternativo	-
ID requisitos asociados	AM9

Para los casos de uso correspondientes a la búsqueda por radio y el direccionamiento hacia el aparcamiento de SMASSA y/o estacionamiento de bicicleta, por la semejanza que existe entre la versión para aparcamiento de coche y la de bicicleta, se van a juntar ambas versiones en un mismo caso de uso que explique de manera general, el escenario para cada requisito funcional mencionado.

ID	CU6 y CU7
Caso de uso	Búsquedas por radio
Descripción	Búsqueda de los aparcamientos cercanos
Precondición	Estar en el apartado correspondiente al tipo de transporte al que se desea realizar la consulta
Escenario principal	<ol style="list-style-type: none"> 1. El usuario avanza hacia el final del listado de aparcamientos donde se encuentra la entrada para la introducción del radio, lo introduce y realiza la búsqueda 2. El sistema muestra los aparcamientos de SMASSA o los estacionamientos de bicicletas que se encuentran en dicho radio
Escenario alternativo	-

ID requisitos asociados	AM11 y AM12
--------------------------------	-------------

ID	CU8 y CU9
Caso de uso	Direccionamiento
Descripción	Direccionamiento por GPS hacia el destino seleccionado
Precondición	Estar en el apartado correspondiente al tipo de transporte al que se desea realizar la consulta
Escenario principal	<ol style="list-style-type: none"> 1. El usuario busca el aparcamiento o estacionamiento deseado y pulsa el botón relativo a los detalles 2. El usuario es redirigido a una página donde se muestra su posición y la del aparcamiento en un mapa 3. El usuario selecciona el marcador y posteriormente, el primer icono inferior derecho 4. El usuario es redirigido a la aplicación de Google Maps 5. El usuario selecciona el tipo de transporte y selecciona la opción relativa a iniciar la ruta 6. El usuario es dirigido a su destino
Escenario alternativo	-
ID requisitos asociados	AM10

ID	CU10
Caso de uso	Predicciones
Descripción	Predicción de la ocupación del aparcamiento de SMASSA o estacionamiento de bicicleta seleccionad
Precondición	Estar en el apartado correspondiente al tipo de transporte al que se desea realizar la consulta
Escenario principal	<ol style="list-style-type: none"> 1. El usuario busca el aparcamiento o estacionamiento deseado y pulsa el botón relativo a los detalles 2. El usuario es redirigido a la página de detalles 3. El usuario selecciona el día de la semana y la hora para la que desea realizar la predicción 4. El sistema muestra la ocupación predicha
Escenario alternativo	-
ID requisitos asociados	AM5 y AM6

5 METODOLOGÍA

Durante la ejecución de este trabajo se ha seguido una metodología incremental iterativa, en la que únicamente participaba un desarrollador. A pesar de no ser un proyecto de gran envergadura, se definieron una serie de iteraciones (o *sprints*) y una planificación temporal.

Para no extender demasiado este apartado, se pasará a listar directamente a enumerar los sprints que han sido definidos y llevados a cabo durante el desarrollo del trabajo.

1. Modelado de la base de datos

- 1.1. Diseño del modelo lógico
- 1.2. Implementación del modelo
- 1.3. Inserción manual de los datos que así lo requieren

2. Desarrollo de la aplicación web para la extracción y almacenamiento de datos

- 2.1. Conexión con la base de datos
- 2.2. Descarga y procesamiento de los ficheros para guardar la información en la base de datos
- 2.3. Crear la tarea periódica asociada a la funcionalidad anterior

3. Desarrollo de la API REST

- 3.1. Búsqueda por radio en aparcamientos de SMASSA y estacionamientos de bicicletas.
- 3.2. Exportación de los datos almacenados en formato CSV para usar en Azure
- 3.3. Obtención de tarifas para los aparcamientos de SMASSA

4. Creación del módulo de aprendizaje automático en Azure

- 4.1. Creación de un experimento para los aparcamientos de coches, basado en regresión lineal, de prueba para buscar la mejor representación de los datos
- 4.2. Experimentación con otros tipos de algoritmos de aprendizaje y comparación de resultados

4.3. Creación del experimento asociado a los estacionamientos de bicicletas

4.4. Despliegue de los servicios webs asociados a dichos experimentos

5. Desarrollo de la aplicación móvil

5.1. Obtención de los listados de los aparcamientos de SMASSA y estacionamientos de bicicletas

5.2. Búsqueda por radio para la tarea anterior y actualización periódica de las listas

5.3. Creación de mapas para la visualización de resultados

5.4. Visualización de los cortes de tráfico

5.5. Inserción de la búsqueda de precios de los aparcamientos de coches

5.6. Obtención de distancias y tiempo de viaje según el tipo de transporte

6 DISEÑO

A continuación, se muestra el modelo lógico para la base de datos, diseñado para un óptimo almacenamiento de los datos extraídos.

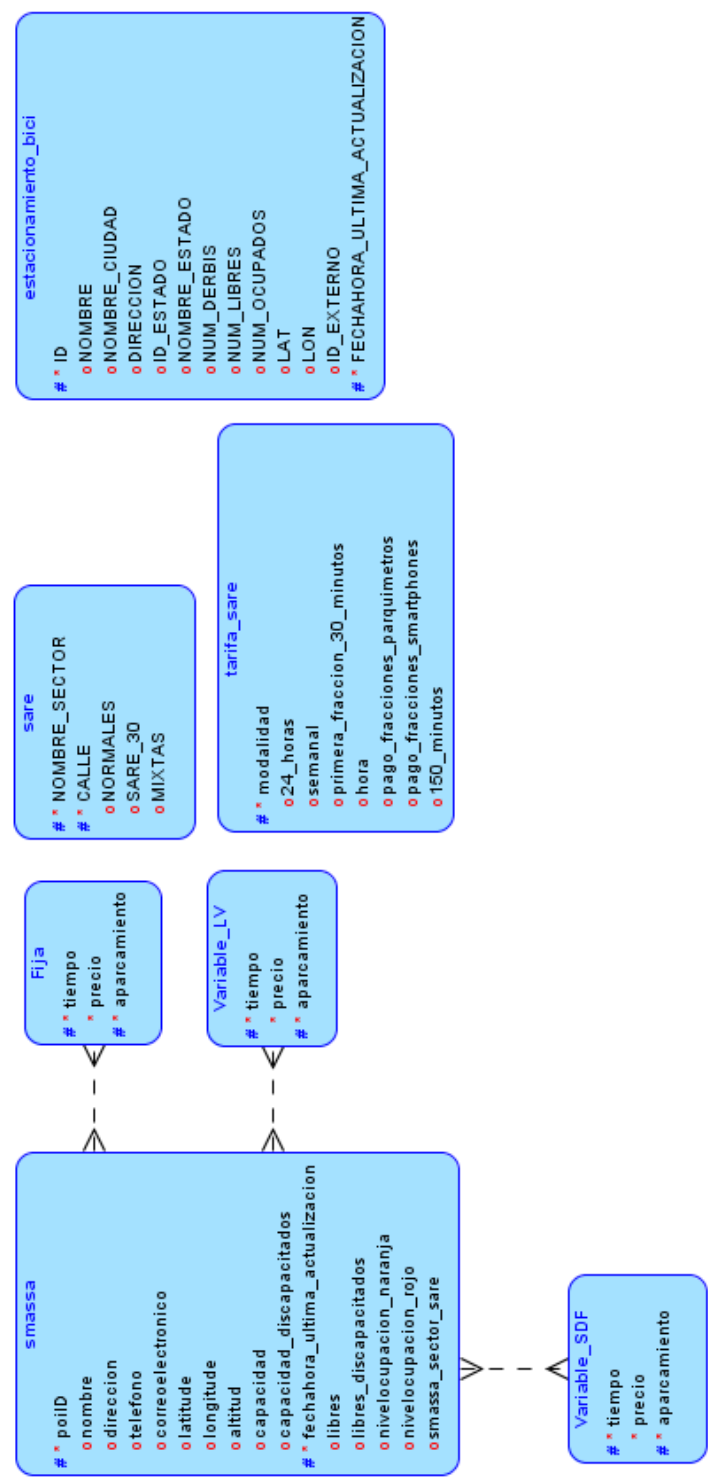


Ilustración 5.1 Diagrama lógico de la base de datos

Como puede observarse en el modelo, la estructura de las tablas para almacenar los precios de cada tarifa (tablas “*Fija*”, “*Variable_LV*” y “*Variable_SDF*”), están hechas de forma que sean independiente de los tiempos de las tarifas y únicas para cada aparcamiento mediante las relaciones, es decir, que en el caso de introducir un intervalo de tiempo nuevo para un aparcamiento determinado, esto se traducirá en la introducción de una nueva fila en la tarifa interesada y no supone cambio alguno en las funciones para buscar la tarifa en las funciones de las aplicaciones.

Sin embargo, hay que mencionar que la introducción de precios en la base de datos no se automatizó debido al formato de los ficheros que ofrecía el Ayuntamiento para las mismas, siendo un modelo de datos muy poco favorable para el desarrollo de este proyecto y que tuvo que ser personalizado. Para ser breves, se expondrá las columnas con las que cuenta el fichero: *aparcamiento*, *del minuto 0 al 30*, *del minuto 31 al 60*, *del minuto 61 al 720*, *del minuto 721 al 1440*, *a partir de la hora 24 a la hora 36*, *a partir de la hora 37 a la hora 48*, *pérdida de tique*.

Observando con cuidado estos atributos, se llega a la conclusión de que la introducción de una nueva tarifa en el modelo de datos construido de forma similar a los datos que ofrece el Ayuntamiento se traduce en la introducción de una nueva columna en la base de datos, que a su vez implicaría modificar las aplicaciones para buscar los precios de cada aparcamiento.

Finalizando este aspecto, este modelo de datos hace posible la introducción de nuevas tarifas sin tener que modificar en ningún aspecto la estructura de la base de datos y las aplicaciones, simplemente, añadiendo una fila a esta primera.

Por otro lado, puede observarse que se han incluido dos tablas relativas a los aparcamientos S.A.R.E., tanto para su localización como para sus precios. Inicialmente no se va a trabajar con estos datos, aunque se han introducido manualmente los datos relativos a cada tabla, importándolos desde un CSV, ya que ninguno de ellos va a ser actualizado periódicamente; las futuras modificaciones serán mínimas y no requerirán mucho trabajo para actualizar la base de datos. La introducción de los mismos se ha realizado con vistas a un futuro para extender la funcionalidad de la aplicación.

En este caso se ha optado a no relacionar las tablas de aparcamientos y tarifas inicialmente, si bien se puede ver que los datos en el caso anterior de tarifas para los aparcamientos SMASSA han sido optimizados, para este caso no y, por ende, no han sido relacionados a la espera de futuras mejoras.

Varios aspectos a tener en cuenta y que no aparecen en el modelo lógico de la base de datos son: primero, el borrado de las tablas de tarifa en cascada, para facilitar la eliminación de los datos, y el segundo, la creación de un evento periódico que se ejecuta cada minuto para borrar los datos relativos a los aparcamientos de SMASSA y los estacionamientos de bicicletas con una antigüedad superior a diez días. Se ha tomado esta última decisión debido a que, a lo largo del tiempo, la ocupación de los aparcamientos va variando, pues no es la misma en el mes de agosto, cuando normalmente la gente está de viaje, que en septiembre u octubre que empieza el curso académico y los adultos empiezan a trabajar.

La toma de esta decisión tiene consecuencias para el modelo predictivo, ya que tendrá que ser actualizado periódicamente y esta tarea ha de realizarse a mano, pues entre otros motivos y como se relatará posteriormente, Azure no puede descargar los datos de una API alojada en una máquina de usuario “común”, como ocurre en el caso del autor de dicha memoria, por lo que los datos han de ser exportados a un fichero CSV y dicho fichero tendrá que ser cargado en el conjunto de datos del entorno de trabajo.

Por último y para cerrar este capítulo, el lector podría pensar que no se le ha prestado importancia al modelo de clases que surgirá a raíz de este modelo de base de datos. Esto es así porque inicialmente, son cuatro tablas (las relacionadas con los aparcamientos de SMASSA y los estacionamientos de bicicletas), las que nos interesan y la aplicación móvil no guardará base de datos alguna para su ejecución, todo el proceso se realizará mediante consultas a la API del Ayuntamiento y a la API REST que se va a desarrollar, y es ésta, mediante consultas a la base de datos, la que obtendrá los datos de interés, por lo que realmente, las clases Java generadas para las dos aplicaciones web y las relaciones entre las mismas, es de mínima importancia.

7 IMPLEMENTACIÓN

7.1 APLICACIÓN WEB JAVA EE

7.1.1 CONEXIÓN CON LA BASE DE DATOS

Para esta parte del proyecto se hizo uso de las tecnologías EJB (Enterprise JavaBeans) y JPA (Java Persistence API), para facilitar la conexión con la base de datos y obtener de forma automática las clases y entidades necesarias para conectar con la base de datos de manera transparente al programador e independiente del tipo de base de datos al que se conectase. Esto fue posible gracias al uso del JDBC (Java Database Connectivity), utilidad que permite al programador realizar operaciones sobre la base de datos en lenguaje java independientemente del tipo de base de datos al que esté conectado.

Desde NetBeans, todo esto se traduce en que, una vez hemos creado la base de datos MySQL, antes o después de haber creado las tablas e insertado los datos que requerían su importación manual desde el CSV, se crearía una nueva conexión asociada a dicha base de datos.

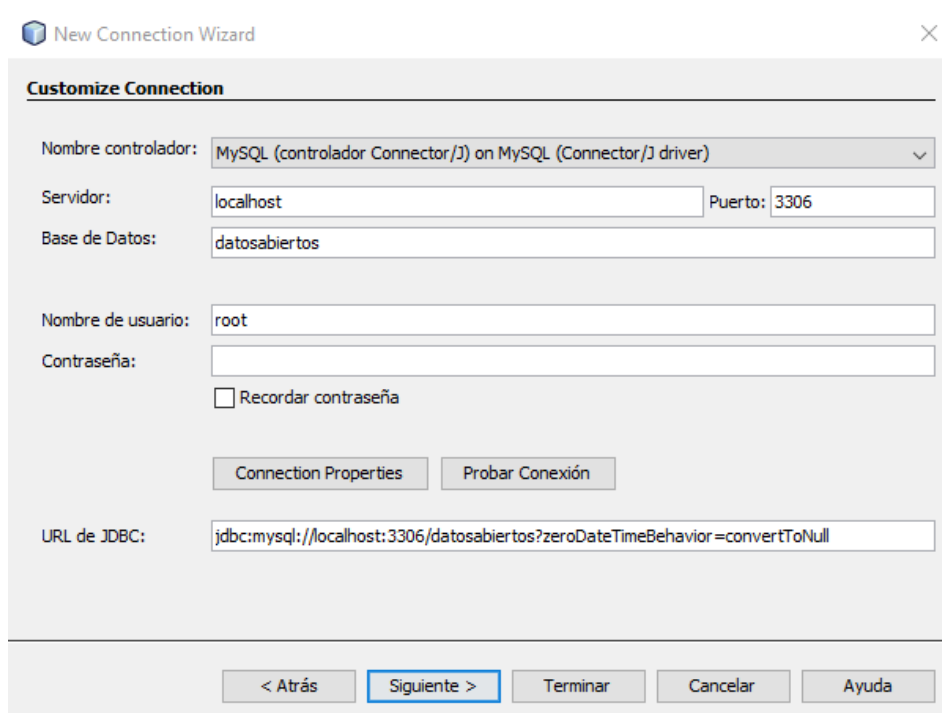


Ilustración 7.1 Parámetros de la conexión con la base de datos

Como resultado, en el apartado de “*Prestaciones*”, en el desplegable de “*Bases de Datos*”, se tendría que tener algo similar a esto:

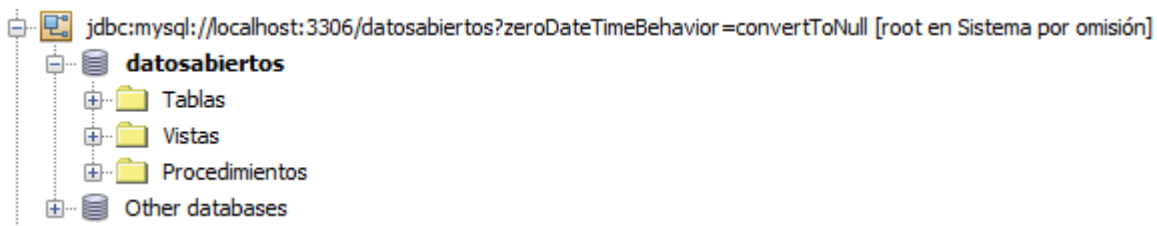


Ilustración 7.2 Resultado de la conexión con la base de datos

Esta conexión junto con el “*pool de conexión*” y el recurso JDBC que se va a crear, servirá tanto a esta aplicación web como al servicio web que será creado posteriormente, para conectar con la base de datos.

Como se ha mencionado, esto solo no es suficiente para establecer la conexión con la base de datos y empezar a programar, primero se ha de crear el *pool de conexión* con la base de datos de la siguiente manera:

1. En el paquete EJB del proyecto empresarial se abrirá el menú desplegable pulsando el botón derecho del ratón, se pinchará en *Nuevo y Otro...*
2. Dentro de las categorías, se buscará la carpeta *GlassFish* y en el tipo de archivo, se seleccionará *JDBC Connection Pool*
3. Se asignará el nombre deseado, se extraerá de la conexión creada previamente y se aceptarán los pasos siguientes

Finalmente, para terminar de establecer la conexión con la base de datos, se creará el recurso asociado al *pool de conexión*, siguiendo estos pasos:

1. En el paquete EJB del proyecto empresarial se abrirá el menú desplegable pulsando el botón derecho del ratón, se pinchará en *Nuevo y Otro...*
2. Dentro de las categorías, se buscará la carpeta *GlassFish* y en el tipo de archivo, se seleccionará *JDBC RESOURCE*
3. En la siguiente ventana, se seleccionará la opción de usar un *pool de conexión* existente y se seleccionará el creado anteriormente, como no se desean parámetros adicionales, se seleccionará *Terminar* y se tendrá el recurso JDBC asociado.

Si se han seguido satisfactoriamente todos los pasos, en la pestaña de *Prestaciones*, dentro del desplegable *Servidores/ GlassFish Server/ Resources/ JDBC/ Connection Pools*, debería estar el *pool de conexión* creado y en *Servidores/ GlassFish Server/ Resources/ JDBC/ JDBC Resources*, el recurso asociado al *pool de conexión*

7.1.2 IMPLEMENTACIÓN DE LA APLICACIÓN

Llegados a este punto, se generan las entidades EJB y JPA a partir de la base de datos y se tiene el proyecto preparado para el primer sprint de programación. Para este sprint se ha creado una clase Java normal, que ha sido anotada con las anotaciones *@Singleton* y *@Startup* para que se cree una sola instancia y se ejecute al iniciar la aplicación.

Inicialmente no se encontró ningún enlace en la API del Ayuntamiento para la descarga de los datos en formato JSON, por lo que, al principio, esta aplicación buscaría descargar un fichero CSV, lo guardaría temporalmente hasta que fuera procesado y luego sería borrado. Esto presentaba un problema: al descargar los ficheros, el formato en el que están guardados los datos en el mismo, al no ser UTF-8 y abrirlos, ocurre que no reconocen los caracteres especiales como las tildes e introducen en su lugar, otros que los hacen inservibles para nuestro uso, dando lugar a una serie de líneas de código necesarias para sustituir estos caracteres.

En la **Ilustración 7.3** se muestran los enlaces de acceso al servicio donde poder descargar los ficheros CSV con los datos necesarios para el desarrollo de este proyecto

```
urlSmassa = "http://datosabiertos.malaga.eu/recursos/aparcamientos/smassa/smassa_ocupacion.csv";  
urlBicis = "http://datosabiertos.malaga.eu/recursos/transporte/EMI/estacionamientos/Estacionamientos.csv"
```

Ilustración 7.3 Enlaces a los servicios de datos de aparcamientos de SMASSA y estacionamiento de bicicletas

Durante el proceso de descarga no hubo ningún problema; sin embargo, durante el procesamiento del fichero, tuvieron lugar algunos problemas con el uso de librerías específicas para procesar un fichero CSV, entre otros motivos, debido a la estructura de las entidades creadas a partir de la base de datos, pues dichas entidades contenían algunos atributos estructurados que no pueden ser generados por las librerías, que sólo generan objetos Java con atributos simples a partir de los ficheros CSV. Por ende, se optó por el

procesamiento tradicional del fichero de texto para generar las clases con mayor facilidad, las relaciones entre las mismas y la inclusión de los objetos en la base de datos mediante los EJB.

El procesamiento tradicional del fichero ocasionó otro problema: el reconocimiento de cada elemento dentro de una línea. En este caso, no concurren ninguno de los siguientes hechos que facilitan el procesado de un archivo CSV: los elementos están separados por comas y dentro de cada uno de ellos, no hay comas, o los elementos se encuentran separados por comas y aunque haya comas dentro de los elementos, cada uno va entre comillas dobles o simples.

La solución encontrada a este problema fue la creación de un patrón especial para la división de una línea en varios elementos de forma correcta como se muestra a continuación.

```
Pattern pattern = Pattern.compile("(?=[^\\\"]*\"[^\"]*\"|\"[^\"]*\"|(?![^\\\"]*\"))")
```

Ilustración 7.4 Patrón para procesado de líneas de texto

Una vez hecho esto y habiendo comprobado que los resultados son los esperados mediante la visualización de los resultados por consola, en el caso de los aparcamientos de SMASSA se realizaron tres métodos en la clase Java de su EJB para buscar la tarifa correspondiente a dicho aparcamiento según su nombre, establecer la relación y guardarlo en la base de datos.

Para comprobar finalmente, que dicha aplicación realiza su fin correctamente, se desplegó en el servidor de aplicaciones y se puso en funcionamiento varios minutos con un resultado satisfactorio.

7.2 CREACIÓN DE LA API REST

En este apartado se usará la conexión a la base de datos creada para el apartado anterior.

7.2.1 CREACIÓN DE LOS SERVICIOS

Para comenzar con esta tarea, se empezó por generar los servicios a partir de la base de datos de forma autogenerada haciendo uso de las facilidades que ofrece NetBeans, como se dijo anteriormente, dando como resultado a un paquete de servicios con los EJB correspondientes para conectar a la base de datos y hacer las peticiones, y otro paquete con las

entidades Java correspondientes a la tablas guardadas en la base de datos, siendo realmente, iguales que las entidades generadas en el proyecto anterior.

Estos EJBs de los que se han hablado, realmente son también iguales que los del proyecto anterior, solo que en sus métodos se encuentran anotaciones específicas para devolver el resultado, en nuestro caso, en JSON. Es por este motivo, que crear otro proyecto para generar la API no tiene ninguna ventaja y solo sirve para duplicar código que hace lo mismo, pero como se dijo previamente, por problemas internos y para evitar perder más tiempo, ya que no es un proyecto muy grande, se tomó la decisión de crear otro proyecto aparte.

7.2.2 CREACIÓN DE LOS MÉTODOS PARA EL SERVICIO DE APARCAMIENTO DE SMASSA

Como se comentó anteriormente, al principio no se encontró ningún enlace a un servicio que devolviera los datos de los aparcamientos de SMASSA y estacionamientos de bicicleta en formato JSON, aunque se adelanta que posteriormente sí se encontró por casualidad, porque estaba muy mal indicado dentro de la página. Debido a esto, se tuvieron que copiar los métodos del proyecto anterior para descargar el fichero y procesarlo, solo que, a diferencia del caso anterior, no se guardan los objetos en la base de datos ni se realizan las relaciones de los aparcamientos con sus precios, sino que se devuelven las entidades de los aparcamientos y los estacionamientos en formato JSON, evidentemente, cada uno por separado en su servicio correspondiente.

En particular, para este servicio tuvo que crearse otro método adicional con el que no cuenta el servicio correspondiente a los estacionamientos de bicicletas: la búsqueda del precio correspondiente a un aparcamiento en un tiempo determinado.

La idea de este servicio se basa en que el usuario, una vez haya seleccionado el aparcamiento que le interesa, buscará el precio que le costará estacionar su vehículo durante los minutos que estime oportunos. Se hace especial hincapié en el hecho de introducir los datos en minutos, pues es como se encuentran divididos los datos en la base de datos y como se indicará en la aplicación móvil.

Este servicio además servirá para buscar el precio de la pérdida del tique, aunque no se menciona anteriormente. Para ello, se mostrará primero una entrada de la base de datos sobre las tarifas para un aparcamiento aleatorio.

aparcamiento	tiempo	precio
C. HAYA	30	0.052
C. HAYA	60	0.0062
C. HAYA	720	0.00307
C. HAYA	1440	0.001
C. HAYA	2160	0.0307
C. HAYA	2880	0.0001
C. HAYA	999999	21.5468

Ilustración 7.5 Tarifa fija para el aparcamiento C. Haya

Si se observa con detenimiento y se vuelve al capítulo 6 de este documento, cuando se enumeran las columnas que posee el fichero CSV de las tarifas, hay una llamada *perdida_ticket* que no se encuentra en este modelo, sin embargo, ha sido sustituida por la última franja de tiempo que se encuentra en la **Ilustración 7.5**. Se cambió por dicho número para dejar espacio por si se añadieran nuevos intervalos de tiempo superiores y para tener una representación numérica aproximada de dicha columna. Dicho esto, se diluye que cuando se busque el costo que supone la pérdida de tique, internamente se realizará una petición con un valor numérico grande, para superar las franjas existentes, pero sin superar el valor correspondiente a la pérdida del tique.

Finalmente, aunque como se verá, se añadirán nuevos métodos a este servicio, para exportar los datos al Azure se creó un método que copiara el comportamiento del servicio del Ayuntamiento desde donde las aplicaciones y Azure pueden descargar los datos: un enlace que, al ser llamado, descarga automáticamente un fichero.

Para ello, se creó un método que devolviese un *StreamingOutput*, donde escribiera en formato de CSV el fichero con todos los datos almacenados en la base de datos. Esto se consiguió con un buffer que escribiera línea a línea los datos en el stream de datos: primero escribía la cabecera con el nombre de las columnas y luego, tras sacar todos los datos de la

base de datos, generaba una línea de texto con todos los datos separados por comas de manera similar a como vienen en el fichero que se puede descargar desde la web del Ayuntamiento.

7.2.3 CREACIÓN DE LOS MÉTODOS PARA EL SERVICIO DE ESTACIONAMIENTOS DE BICICLETAS

La creación de los métodos necesarios para las consultas relacionadas con los estacionamientos de bicicletas es relativamente sencilla una vez han sido creados los métodos del servicio anterior. Ya se explicó el motivo por el cual se creaba un método para obtener los datos de los aparcamientos o estacionamientos actuales, por lo que en este apartado sólo se mencionará que se realizó el método equivalente al de obtener los resultados en formato CSV para el Azure, siguiendo el mismo procedimiento que para los aparcamientos de SMASSA, pero adaptándolo al caso de los estacionamientos de bicicletas.

8 CREACIÓN DEL MODELO PREDICTIVO EN AZURE

8.1 CREACIÓN DEL EXPERIMENTO

Como iniciación en Azure, se empezó por crear un experimento simple tal como se muestra en la **Ilustración 8.1**, con el fichero CSV de los aparcamientos de SMASSA que se descarga al seleccionar el enlace correspondiente y, tras la imagen, se procederá a la explicación del experimento.

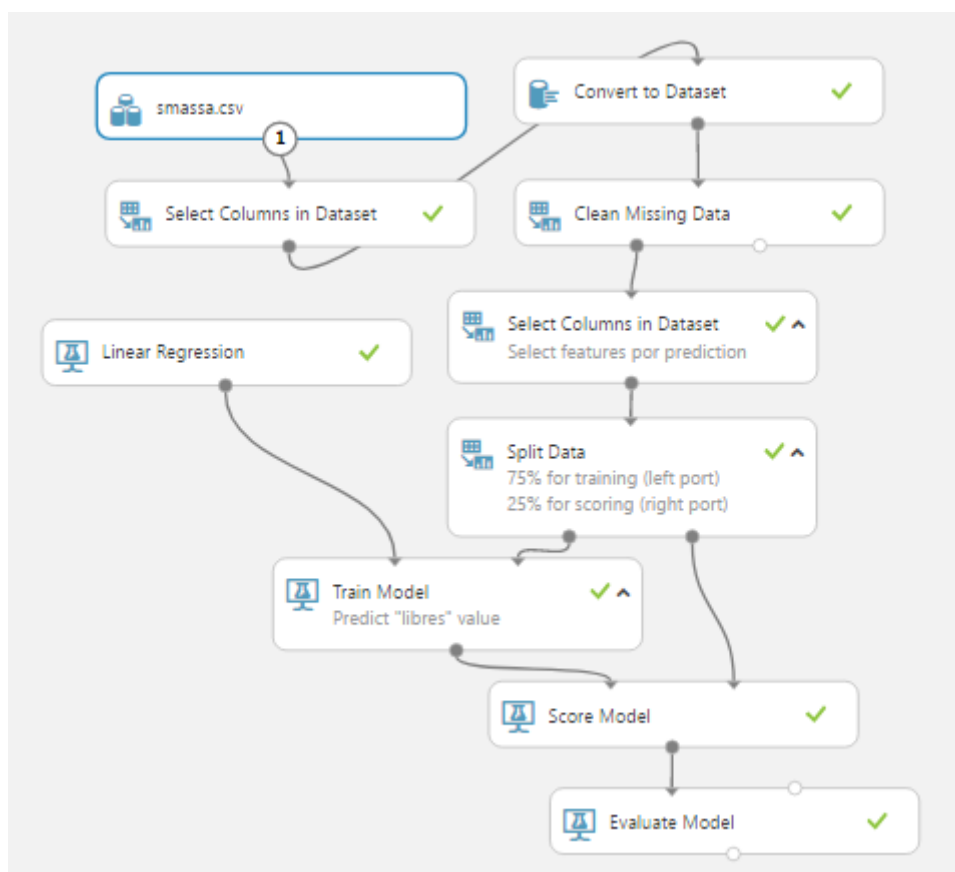


Ilustración 8.1 Primer experimento de Azure

Como se decía anteriormente, crear un experimento en Azure es una tarea muy sencilla por las facilidades que tiene y ofrece resultados muy buenos, posee una amplia gama de funcionalidades para hacer todo tipo de experimentos y la **Ilustración 8.1** es ejemplo de ello.

De la figura se observa que el funcionamiento básico de *Azure Machine Learning Studio* se basa en arrastrar módulos de datos y módulos de lógica a una paleta de trabajo y comunicándolos entre ellos mismos. En este caso, el módulo llamado “smassa.csv”, se

corresponde con el conjunto de datos resultante de importar el fichero CSV a nuestro conjunto de datos en Azure. El siguiente módulo con el que está enlazado, servirá para seleccionar ciertas columnas del conjunto de datos e incluirlas en el experimento o eliminarlas. Para este experimento, se eliminarán las siguientes columnas: *capacidad_discapacitados*, *libres_discapacitados*, *teléfono*, *correoelectronico*, *smassa_sector_sare*, *direccion*, *latitude*, *longitude* y *altitud*.

El siguiente módulo con el que está enlazado, sirve para limpiar el conjunto de datos de valores incorrectos. Para el caso que concierne a este documento, se procederá a cambiar aquellos con valor ‘-1’ a valor ‘0’, por considerar inicialmente que el ‘-1’ se corresponde con un aparcamiento que no se puede usar y que el valor ‘0’ puede realizar la misma función sin introducir anomalías inicialmente en el modelo. Posteriormente, se eliminarán aquellas filas en cuyas columnas falte valores. El siguiente módulo, como se indica en su descripción, sirve para seleccionar las columnas de interés para el experimento, en este caso: *capacidad*, *fechahora_ultima_actualizacion*, *nombre* y *libres*. Los datos para el experimento, llegados a este punto, ya están preparados. En el siguiente paso, el módulo de *Split Data* los usaremos para elegir el porcentaje de datos que queremos usar para entrenar al modelo y el porcentaje de datos que servirán de prueba para evaluarlo. Esta selección es aleatoria. Finalmente, se elegirá el tipo de algoritmo de aprendizaje que se quiere usar y dentro de los de tipo regresión, se seleccionará la regresión lineal. En el siguiente paso, se conectan los dos últimos módulos especificados al módulo de *Train Model* que se encargará de entrenar al modelo para predecir la columna que queremos predecir, en el caso de este proyecto: *libres*. Finalmente, para obtener los resultados, se conecta el módulo de entrenamiento y el de *Split Data* al módulo de *Score Model*, de forma que, al tener el modelo entrenado y el porcentaje de datos especificados para probarlos, realice la estimación.

Finalmente, si se desean obtener valores estadísticos sobre la exactitud del modelo, se conectará el módulo de puntuación con el módulo *Evaluate Model*.

Para visualizar mejor las diferencias entre estos dos módulos, se mostrará en la **Ilustraciones 8.2, 8.3 y 8.4** el resultado obtenido por los módulos *Score Model* y el módulo *Evaluate Model* para otro de los experimentos realizados.






nombre	capacidad	fecha_hora_ultima_actualizacion	libres	Scored Labels
				
Cervantes	414	2017-08-05T01:05:00	352	306.562904
El Carmen	186	2017-06-05T13:45:00	0	-0.256323
Cruz De Humilladero	104	2017-09-05T21:24:00	83	96.132488
El Carmen	186	2017-09-05T21:28:00	0	0.212577
Cruz De Humilladero	104	2017-06-05T19:14:00	99	95.664763

Ilustración 8.2 Resultados del módulo ‘Score Model’

Metrics

Mean Absolute Error	39.927933
Root Mean Squared Error	65.089746
Relative Absolute Error	0.292796
Relative Squared Error	0.187595
Coefficient of Determination	0.812405

Ilustración 8.3 Métrica del módulo ‘Evaluate Model’

Error Histogram

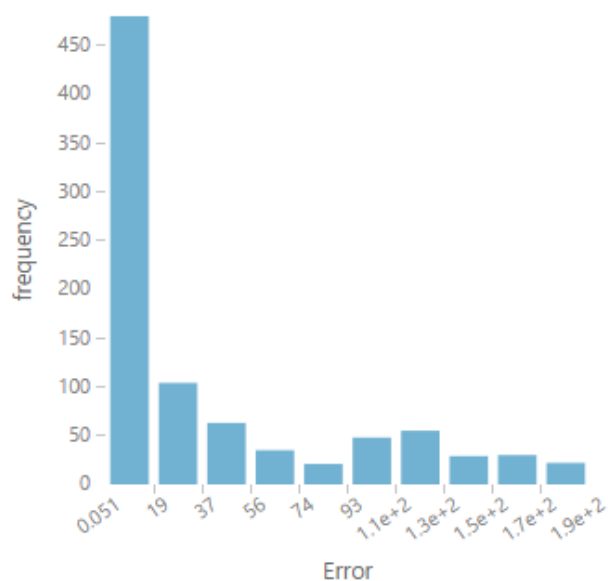


Ilustración 8.4 Histograma de errores del módulo ‘Evaluate Model’

Estos resultados se obtuvieron para un experimento como el descrito, pero con un conjunto de datos tomado durante unas 3 horas.

Una vez se comprobó que el experimento funcionaba, se usó el conjunto de datos mencionado en el párrafo anterior y se realizaron 6 experimentos nuevos para jugar con el valor de la fecha de actualización y, en vez de establecer las casillas con valor ‘-1’ a ‘0’, borrar la fila correspondiente. Cuando se dice que se “jugó” con el valor de la fecha de actualización, se quiere decir que se jugó con su formato. Para que se vea claramente, el formato en la base de datos es de tipo “datetime” y la representación del mismo es la siguiente: YYYY-MM-DD HH:MM:SS. Realizar una predicción con una fecha así, a este nivel puede dar lugar a modelos muy inexactos, por lo que se optó por probar diferentes formatos para buscar el más exacto. Estos fueron: el formato anterior, el formato EEE HH, donde ‘EEE’ hace referencia al día de la semana (lunes, martes, ...) y, finalmente, separar este último formato en dos columnas, una para el día de la semana y otro para la hora.

Una vez elegido el formato, se realizaron distintos experimentos para comparar resultados con el mismo conjunto de datos con la variación de la eliminación de las filas con valores ‘-1’ en sus casillas o la sustitución de dichos valores por ‘0’ y sin eliminarlos.

Estos han sido los resultados estadísticos obtenidos:

Metrics

Mean Absolute Error	39.927933
Root Mean Squared Error	65.089746
Relative Absolute Error	0.292796
Relative Squared Error	0.187595
Coefficient of Determination	0.812405

Ilustración 8.5 Fecha en formato ‘YYYY-MM-DD HH:MM:SS’ y sustituyendo valores ‘-1’ por ‘0’

Metrics

Mean Absolute Error	51.613567
Root Mean Squared Error	74.914249
Relative Absolute Error	0.407469
Relative Squared Error	0.273866
Coefficient of Determination	0.726134

Ilustración 8.6 Fecha en formato 'YYYY-MM-DD HH:MM:SS' y eliminando filas con valores '-1'

Metrics

Mean Absolute Error	34.497035
Root Mean Squared Error	45.805348
Relative Absolute Error	0.255722
Relative Squared Error	0.0933
Coefficient of Determination	0.9067

Ilustración 8.7 Fecha e formato 'EEE HH' y sustituyendo valores '-1' por '0'

Metrics

Mean Absolute Error	34.589532
Root Mean Squared Error	44.788829
Relative Absolute Error	0.286039
Relative Squared Error	0.105754
Coefficient of Determination	0.894246

Ilustración 8.8 Fecha en formato 'EEE HH' y eliminando filas con valores '-1'

▲ Metrics

Mean Absolute Error	34.85526
Root Mean Squared Error	45.13318
Relative Absolute Error	0.256131
Relative Squared Error	0.088776
Coefficient of Determination	0.911224

Ilustración 8.9 Fecha separada en dos columnas 'EEE' y 'HH' y sustituyendo valores '-1' por '0'

▲ Metrics

Mean Absolute Error	35.104767
Root Mean Squared Error	45.752573
Relative Absolute Error	0.2903
Relative Squared Error	0.110354
Coefficient of Determination	0.889646

Ilustración 8.10 Fecha separada en dos columnas 'EEE' y 'HH' y eliminando filas con valores '-1'

Por medio de la experimentación se puede concluir en que la mejor distribución para los datos es la que se usa en la **Ilustración 8.9**, ya que ofrece los mejores resultados.

Una vez llegados a esta conclusión, Azure ofrece muchos más módulos de aprendizaje. Para este caso interesan los de regresión, ya que el resto son de clasificación o predicciones de otro tipo de datos. Para localizar el mejor módulo, se empleará la experimentación usando los modelos de regresión disponibles en la paleta de trabajo y que se puedan usar en el experimento anterior, por lo que, para ello, se sustituirá simplemente el módulo correspondiente a la regresión lineal por el módulo a probar y se guardarán los resultados de la prueba.

Antes de proceder con esta tarea, se realizará una modificación en el experimento para ofrecer mayor exactitud a los resultados y que tengan mayor validez estadísticamente hablando. Para ello se sustituirá el módulo de *Score Model* por el de *Cross Validate Model*.

De manera resumida, lo que hace este módulo es evaluar los resultados durante varias iteraciones para así garantizar que los resultados son independientes de la partición de los datos. Para ello, simplemente realiza una media aritmética de las evaluaciones obtenidas en cada iteración.

Los distintos experimentos han sido llevados a cabo con este nuevo módulo sustituyendo. A continuación, se muestran los distintos módulos de aprendizaje empleados y los resultados obtenidos en las pruebas:

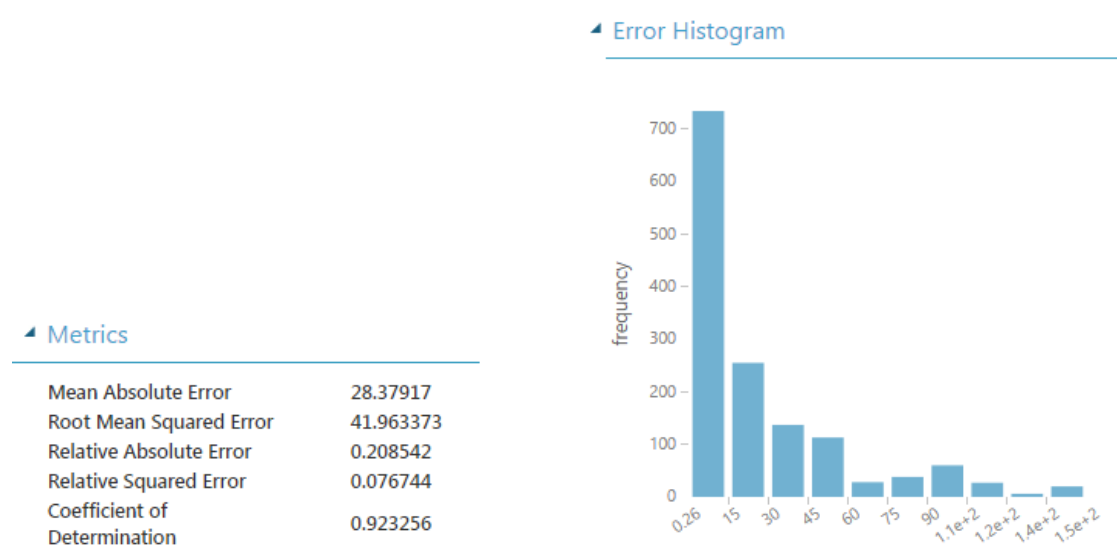


Ilustración 8.11 Métrica Regresión de Poisson

Ilustración 8.12 Histograma de errores de Regresión de Poisson

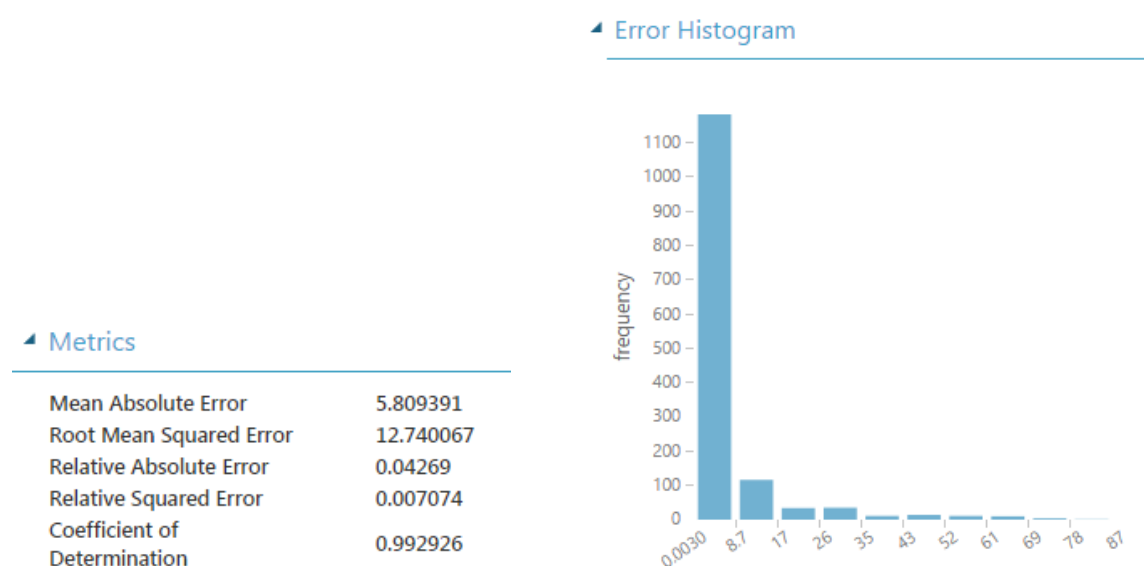


Ilustración 8.13 Métrica Regresión de red neuronal

Ilustración 8.14 Histograma de errores de Regresión de red neuronal

Negative Log Likelihood	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
71058110525.48943	4.474033	12.027327	0.032877	0.006304	0.993696

Ilustración 8.15 Regresión de Bosques de Decisión

Negative Log Likelihood	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
7430.669041	34.653717	45.576089	0.25465	0.090527	0.909473

Ilustración 8.16 Regresión Bayesiana

Metrics

Mean Absolute Error	3.761585
Root Mean Squared Error	10.559783
Relative Absolute Error	0.027642
Relative Squared Error	0.00486
Coefficient of Determination	0.99514

Ilustración 8.17 Métrica Regresión de árbol de decisión mejorado

Error Histogram

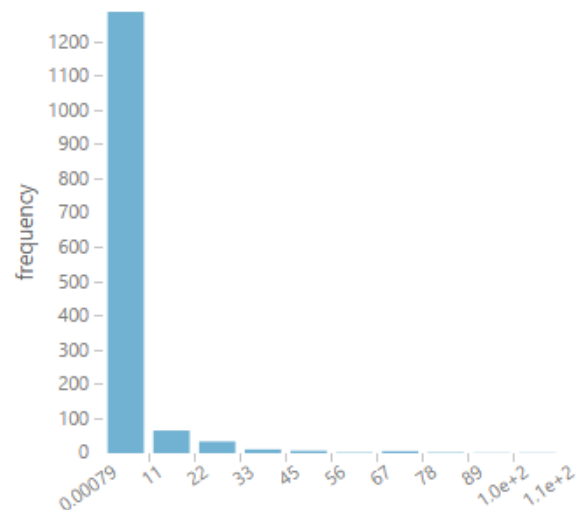


Ilustración 8.18 Histograma de errores de Regresión de árbol de decisión mejorado

Observando cuidadosamente los resultados de los experimentos realizados, se ve que hay tres que destacan por encima del resto por su exactitud la minimización de los errores absolutos y relativos. La importancia en el análisis de estos dos errores reside en que, aunque a priori un modelo parezca bastante preciso, como puede ser la regresión lineal, al trasladarlo al caso del modelo de predicción de los aparcamientos de bicicletas para un conjunto de datos tomados en el mismo tiempo que para los casos anteriores, los resultados que arroja son muy pobres.

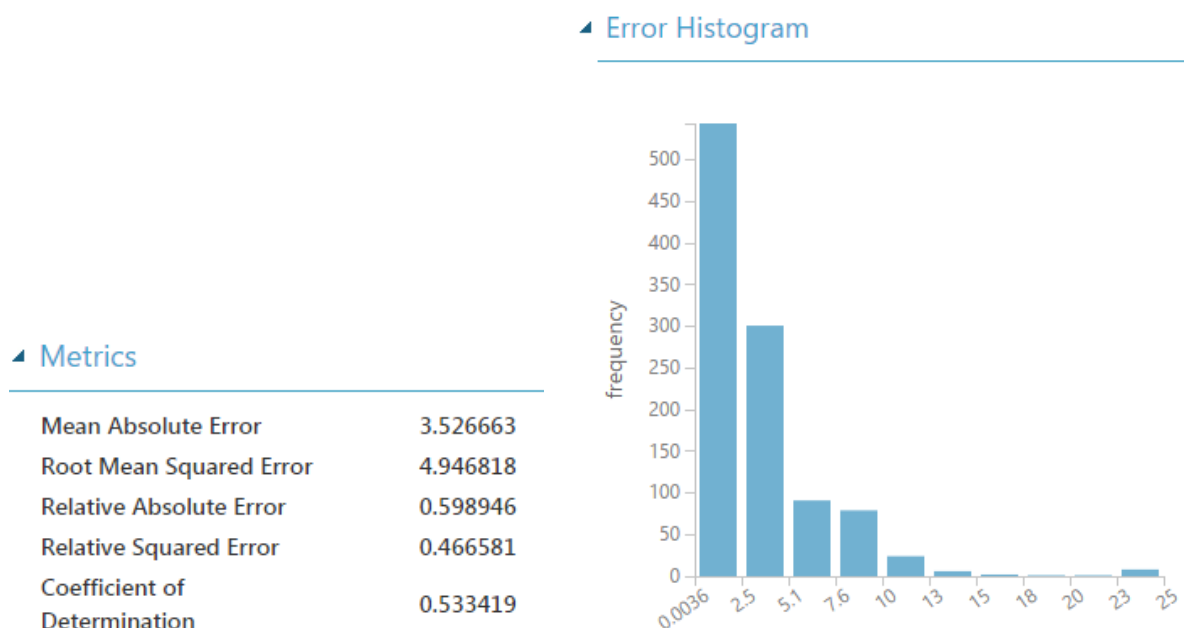


Ilustración 8.19 Métrica Regresión lineal para la predicción de los estacionamientos de bicicletas

Ilustración 8.20 Histograma de errores de Regresión lineal para los estacionamientos de bicicletas

Se puede ver a simple vista que los resultados de la regresión lineal en este caso dejan bastante que desear a pesar de que en el caso anterior eran bastante buenos. Esto es así porque en el caso de los aparcamientos de SMASSA, la capacidad de los mismos y el número de plazas libres oscila entre valores numéricos muy grandes, por lo que un error de ± 35 , por ejemplo, se disimula bastante bien, mientras que, en el caso de los estacionamientos de bicicletas, en los que todos rondan la capacidad de 20 – 25, estos errores se ven acentuados y producen un modelo de predicción bastante malo.

En el caso de los aparcamientos de SMASSA, los módulos de aprendizaje que daban mejores resultados son: la regresión de red neuronal, regresión de bosques (o árbol) de decisión y la regresión de árbol de decisión mejorada. De estos tres, el último mencionado es

el que posee la menor media de errores y un coeficiente de correlación más cercano a uno, por lo que será usado para definir el modelo de predicción tanto en el caso de los aparcamientos de SMASSA como en el caso de los estacionamientos de bicicleta.

A continuación, se muestran los resultados obtenidos tras el cambio de módulo en el experimento de aprendizaje para la predicción de la ocupación en los estacionamientos de bicicletas y estableciendo el porcentaje de datos para entrenar al modelo al 90%.

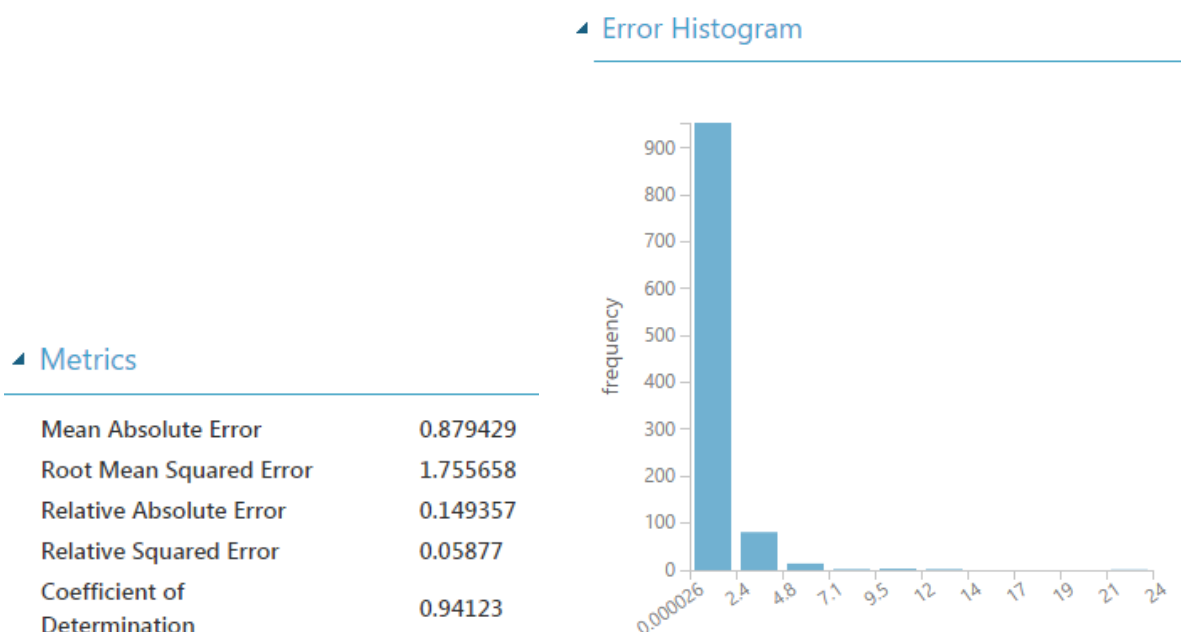


Ilustración 8.21 Métrica final para la predicción de los estacionamientos de bicicletas

Ilustración 8.22 Histograma de errores final para la predicción de los estacionamientos de bicicletas

En el experimento final para la predicción de los aparcamientos de SMASSA también se establecerá un porcentaje de datos para entrenar al modelo del 90%, quedando el resto para evaluarlo. A continuación, se muestran los experimentos finales.

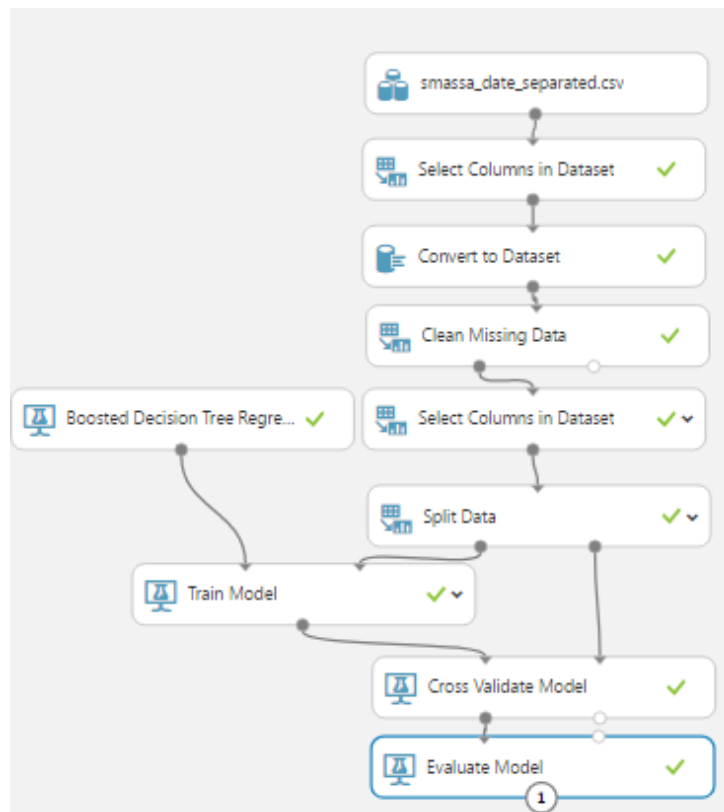


Ilustración 8.23 Experimento final para la predicción de aparcamientos de SMASSA

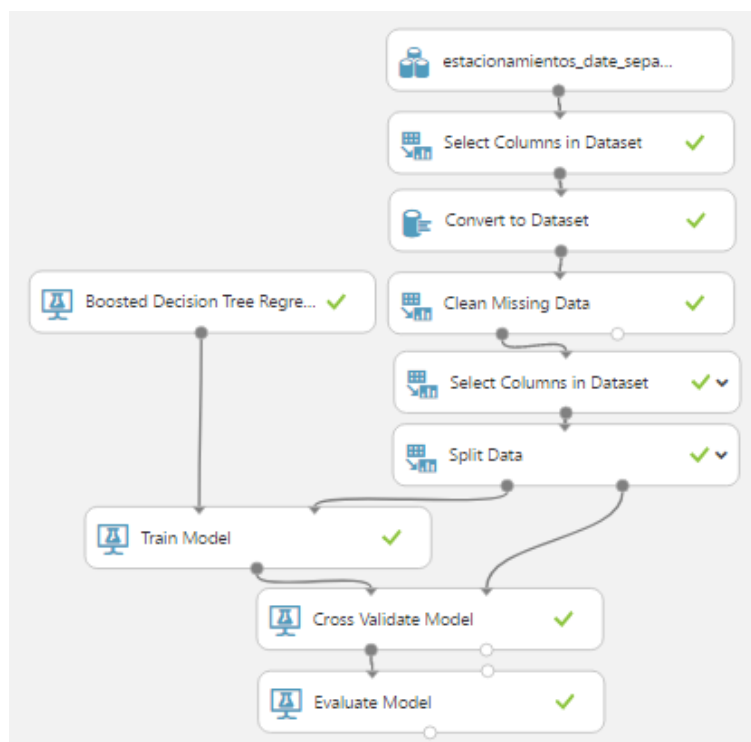


Ilustración 8.24 Experimento final para la predicción de la ocupación en los estacionamientos de bicicletas

8.2 DESPLIEGUE DEL SERVICIO

Una vez finalizados con éxito los pasos anteriores, se desplegará el servicio web asociado a los dos experimentos predictivos simplemente.

En primer lugar, se preparará el servicio web pulsando el botón que se muestra a continuación en la **Ilustración 8.25**.



Ilustración 8.25 Preparación del servicio web

En segundo lugar, el módulo correspondiente a la entrada de datos se moverá de sitio como se mostrará posteriormente, en el módulo de selección de columnas conectado a la entrada de los datos almacenados, se excluirán las columnas que no quieran añadirse como parámetro para la predicción y, finalmente, en el módulo de selección de columnas conectado con la salida de la predicción, se incluirá la columna correspondiente al valor predicho y se excluirán el resto.

Una vez hecho esto, se pulsará sobre el botón de ejecución del experimento y, posteriormente, tras ser el experimento verificado y ejecutado, se pulsará sobre el botón de despliegue.



Ilustración 8.26 Despliegue

Si todo el proceso ha ido bien, se abrirá una ventana con los datos relativos a nuestro servicio y con ejemplos para poder usarlo en diversos lenguajes de programación (C#, Python, Python 3+ y R), y el servicio web estará listo para ser usado.

Experimentos listos para el despliegue de los servicios:

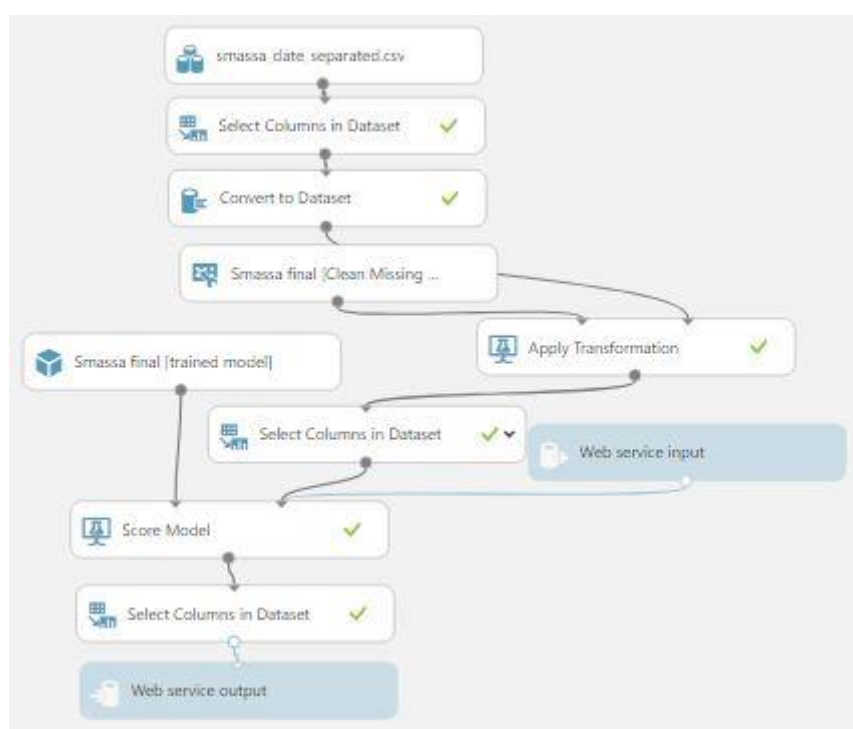


Ilustración 8.27 Servicio de aparcamiento de SMASSA

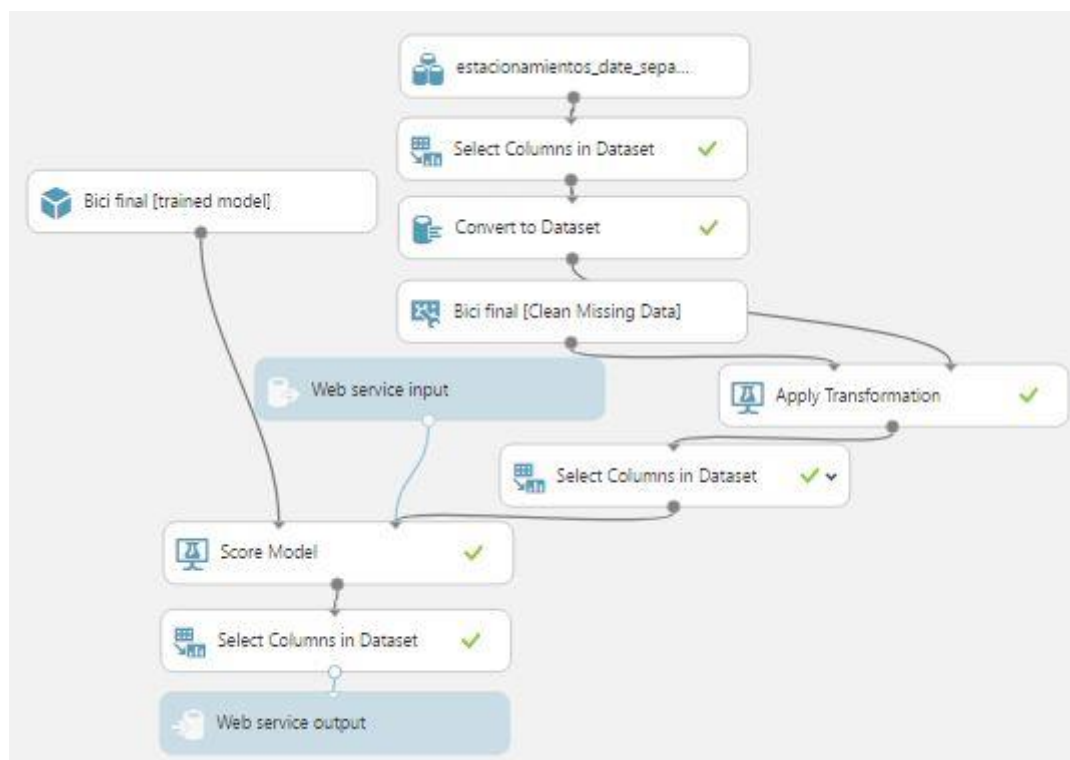


Ilustración 8.28 Servicio de estacionamiento de bicicletas

Finalmente, se procedió a cambiar el módulo de datos por uno que fuera capaz de llamar al método del servicio web que se desarrolló para exportar los datos de la base de datos a un fichero CSV. Sin embargo, aunque el método era correcto, pues llamando a la URL se descargaba un fichero como sucede cuando se llama a la URL del servicio del Ayuntamiento, se lanzaba un error al llamar a la URL del servicio creado.

Se probó a desarrollar dos métodos más, uno que retornará la información de la base de datos en texto plano, pero formateada como un CSV, y otro que abriera un *Stream* de datos que se irían devolviendo en correspondencia con el formato CSV. Aun así, ninguno funcionó, ya que no deja hacer llamadas a direcciones *localhost* y porque, aunque se cambiara dicha dirección por la dirección de la red WiFi, no se podía autenticar el destino, por lo que el procedimiento adoptado para exportar los datos a Azure fue el comentado en primer lugar: llamar a la URL del servicio que descargara el fichero con todos los datos almacenados y, manualmente, exportarlos al conjunto de datos de Azure.

9 DESARROLLO DE LA APLICACIÓN MÓVIL

9.1 CREACIÓN DEL PROYECTO VACÍO

En esta sección se narrará el proceso de creación de la aplicación móvil y los problemas que fueron surgiendo durante su desarrollo. Por la similitud en la implementación de la lógica relacionada con los aparcamientos de SMASSA y los estacionamientos de bicicletas, se explicará el proceso de creación de las funcionalidades para el primero, realizándose para los estacionamientos de bicicletas de forma análoga.

Realizada la instalación del software necesario para el uso de Ionic, se crea un nuevo proyecto con la plantilla “*tabs*”, que crea un proyecto nuevo con 3 pestañas en blanco listas para usarse, a partir del comando: `ionic start [nombre] tabs`, resultando en algo similar a lo que se muestra en la siguiente imagen.

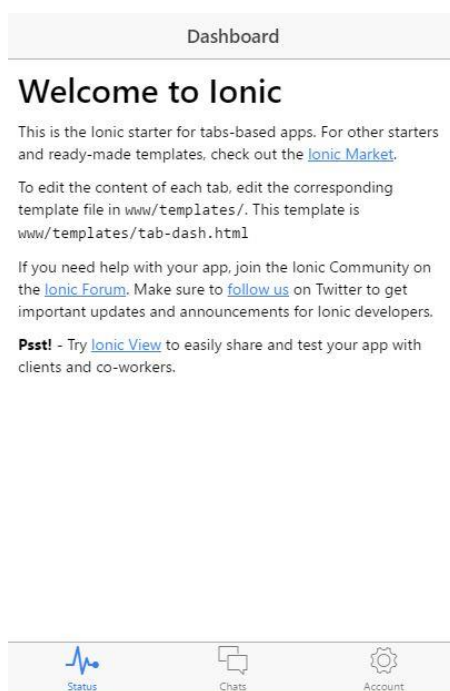


Ilustración 9.1 Proyecto nuevo Tabs

9.2 LISTADO DE LOS APARCAMIENTOS DE SMASSA Y ESTACIONAMIENTOS DE BICICLETAS

Como primer paso, se decidió que la página principal mostrase la lista de aparcamientos de SMASSA con datos como la dirección del aparcamiento, la capacidad, las

plazas libres y un botón para redirigir a la página de detalles, pero para eso, primero había que obtener la lista de aparcamientos que, como se dijo, se obtendría a partir del CVS que se puede descargar mediante peticiones a la web del Ayuntamiento, por lo que se plantearon dos alternativas: la primera, descargarlo desde la aplicación móvil y procesar el fichero generando una lista de objetos, y la segunda, aprovechar la implementación de la aplicación web para incorporar a nuestro servicio un método que descargase el fichero, lo procesara y devolviera en la respuesta un JSON con los datos. Se escogió ese formato por la facilidad de uso en la aplicación.

Por su facilidad, se optó por la segunda, puesto que, de esa manera, se reutilizaba código funcional y no se perdía excesivo tiempo en descubrir cómo se hacía con el lenguaje de typescript.

Para llevarla a cabo, se creó un servicio inyectable que realizase las peticiones a las URLs necesarias cuando lo necesitásemos y que se importaría en todos los componentes que necesitasen realizar llamadas a una URL.

Como resultado, se obtuvo lo siguiente:



Ilustración 9.2 Listado de aparcamientos

En la ilustración anterior se puede observar cómo además se cambió el título de la ventana y los iconos y nombres de cada *tab*. Para el listado de los estacionamientos de bicicletas se realizó un proceso similar, por lo que no se mostrará una imagen del resultado obtenido.

Durante la ejecución de esta idea, se descubrió que la página de datos abiertos del Ayuntamiento contaba con una dirección a una API a la que poder realizar consultas y obtener el fichero en formato JSON, por lo que, después de haber realizado los listados anteriores con el método del servicio web del proyecto, se cambió la llamada de la aplicación móvil a la API de datos.

9.3 ACTUALIZACIÓN DE LOS DATOS PERIÓDICAMENTE

Una vez han sido realizados los dos listados, hay que tener en cuenta que la aplicación debe actualizar los datos en tiempo real, por lo que se hará uso de dos elementos para realizar este comportamiento. El primero, es el uso de los llamados *data binding* de angular. Esto no es más que un enlace entre la vista y las variables y métodos definidos en el ámbito del componente. El segundo, será la creación de una tarea periódica que se ejecute cada minuto (tiempo en el que se actualizan los datos objeto de estudio), que obtenga los datos cada periodo. Como resultado, se obtendrá una función que llame a al método del servicio inyectable que se encarga de obtener los datos y, por medio de los *data binding*, se actualizará sola la vista sin necesidad de incluir código adicional.

```
getPosts() {  
  let loading = this.loadingCtrl.create({  
    content: 'Please wait...'  
  });  
  loading.present();  
  
  this.datosAbiertosService.getAllParking().subscribe(response => {  
    this.items = response.result.records;  
  });  
  loading.dismiss();  
}
```

Ilustración 9.3 Llamada al método que obtendrá los datos

El código adicional representa el típico elemento de carga que aparece en pantalla mientras se obtienen los datos, que aparecerá en más secciones de la aplicación.

```
this.timer = Observable.interval(1000 * 60); //tiempo en ms
this.subscription = this.timer.subscribe(x => {
  this.getPosts();
});
```

Ilustración 9.4 Estableciendo los métodos periódicos

Tal como está la aplicación actualmente, cuando se abre, no muestra los datos al iniciarse hasta pasado el primer minuto, en el que se obtienen los primeros datos. Este hecho supone un comportamiento no deseable, pues lo ideal sería que, al abrir la aplicación, se cargaran los datos actuales y se fueran actualizando, posteriormente, cada minuto.

Afortunadamente, esto es fácil de solucionar gracias a una serie de eventos asociados al ciclo de vida de la aplicación, por lo que a través del método **ngOnInit**, que es llamado al crearse el componente y ejecuta el método que obtiene los datos del servicio del Ayuntamiento.

```
ngOnInit() {
  this.getPosts();
}
```

Ilustración 9.5 Método asociado a la creación del componente

9.4 CREACIÓN DE LA PÁGINA DE DETALLES

El siguiente paso a seguir, sería la creación de una versión inicial de la página de detalles de los aparcamientos de SMASSA y de los estacionamientos de bicicletas. Se ha realizado una única página para visualizar los datos en cada caso, pues los requerimientos de cada uno eran similares y los casos en los que difieren pueden solucionarse fácilmente si a la página de detalles se le pasan parámetros como el tipo de aparcamiento que tiene que visualizar.

Inicialmente se buscará obtener en un mapa la posición del usuario y la del aparcamiento o estacionamiento deseado. Debido a la versión de AngularJS e Ionic usada en el proyecto, no se pudo realizar en javascript, ya que no se permite para dichas versiones, sino que fue realizado en typescript con los componentes de GoogleMaps que se ofrecen para

Ionic tras la inclusión en el proyecto del *plugin* necesario (**cordova-plugin-googlemaps**). Aunque dichos *plugins* facilitan la creación de los mapas respecto al método tradicional de javascript, no cuentan con algunos métodos, o al menos, no han sido encontrados, para realizar tareas como el ajuste automático del zoom, por lo que tuvo que realizarse a mano.

A continuación, se muestran los parámetros necesarios que son pasados a este componente desde los componentes relativos al listado de los aparcamientos de coches y estacionamiento de las bicicletas, para la creación del mapa, y el código usado para la creación de los mapas a través de los componentes que ofrece el *plugin* instalado.

```
this.navCtrl.push(DetailsPage, {  
  lat: item.latitude,  
  lon: item.longitude,  
  name: item.nombre,  
  type: 'car',  
});
```

Ilustración 9.6 Parámetros enviados al componente de detalles

```
loadMap(postion: Geoposition) {  
  let element: HTMLElement = document.getElementById('map');  
  this.map = this.googleMaps.create(element);  
  this.currentLocation = new LatLng(postion.coords.latitude,  
  postion.coords.longitude);  
  let bounds: LatLngBounds = new LatLngBounds(this.currentLocation);  
  this.map.one(GoogleMapsEvent.MAP_READY).then(  
    () => {  
      let target: LatLng = new LatLng(this.lat, this.lon);  
      bounds.extend(target);  
      let targetMarker: MarkerOptions = {  
        position: target,  
        flat: true,  
        title: this.name  
      };  
      let currentLocationMarker: MarkerOptions = {  
        position: this.currentLocation,  
        title: 'Me'  
      };  
      this.map.addMarker(targetMarker)  
        .then((marker: Marker) => {  
          marker.showInfoWindow();  
        });  
      this.map.addMarker(currentLocationMarker)  
        .then((marker: Marker) => {  
          marker.showInfoWindow();  
        });  
      this.map.setCenter(bounds.getCenter());  
      this.map.setZoom(10);  
    })  
  );  
}
```

Ilustración 9.7 Mapas en Ionic

Para realizar los mapas, se hizo uso de otro plugin para la obtención de la posición actual haciendo uso de la geolocalización (**cordova-plugin-geolocation**).

9.5 BÚSQUEDA POR RADIO

Para realizar dicha tarea, previamente se creó en el servicio web que forma parte de este proyecto, dos métodos (uno para los aparcamientos de SMASSA y otro para los estacionamientos de bicicleta), que se encargase de realizar dicha búsqueda y que devolviese la lista de resultados.

```
@GET
@Path("/findNearSmassa")
@Produces(MediaType.APPLICATION_JSON)
public List<Smassa> findCerca(@QueryParam("latitude") double latitud, @QueryParam("longitude") double longitud,
    @QueryParam("radius") double radio) {
    List<Smassa> list = getCurrentSmassa();
    List<Smassa> response = new ArrayList<Smassa>();
    double distance;
    for (Smassa s : list) {
        distance = 6371 * Math.acos(Math.cos(Math.toRadians(latitud)) * Math.cos(Math.toRadians(s.getLatitude()))
            * Math.cos(Math.toRadians(s.getLongitude()) - Math.toRadians(longitud)) + Math.sin(Math.toRadians(latitud))
            * Math.sin(Math.toRadians(s.getLatitude())));
        if (distance <= radio) {
            response.add(s);
        }
    }
    return response;
}
```

Ilustración 9.8 Búsqueda por radio

En la vista, se añadirá una entrada para la introducción del radio y un botón de aceptación, para que el controlador del componente llame al servicio inyectado, que a su vez llamará a un nuevo método que se habrá de crear para llamar a la URL del nuevo método del servicio que se ha creado y le pase el radio como parámetro.

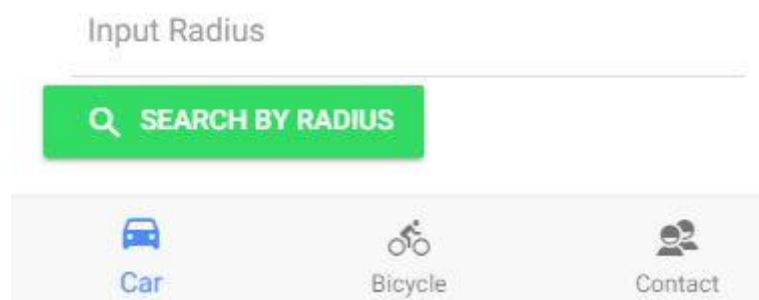


Ilustración 9.9 Búsqueda por radio en la vista

A continuación, se muestra el método del controlador que llama al método del servicio inyectable que realiza la petición al servicio web.

```
searchRadio() {  
  let loading = this.loadingCtrl.create({  
    content: 'Please wait...'  
  });  
  loading.present();  
  
  this.hidden = false;  
  
  this.geolocation.getCurrentPosition().then(response => {  
    this.latitude = response.coords.latitude;  
    this.longitude = response.coords.longitude;  
  
    this.datosAbiertosService.getNearSmassa(this.latitude,  
    this.longitude, this.radius).subscribe(response => {  
      this.items = response;  
    });  
  })  
  .catch(error => {  
    console.log(error);  
  })  
  loading.dismiss();  
  this.subscription.unsubscribe();  
  this.subscription = this.timer.subscribe(x => {  
    this.getNearSmassa();  
  })  
}
```

Ilustración 9.10 Búsqueda por radio en el controlador

Si se prueba esta nueva funcionalidad, si no se realiza ninguna acción en la vista, al cabo de un minuto se vuelven a actualizar los datos y desaparece la búsqueda realizada, por lo que, para mantener dicha búsqueda, se tuvo que eliminar el método de actualización de los datos de la tarea periódica en el momento en el que se realizaba la búsqueda.

```
this.subscription.unsubscribe();  
this.subscription = this.timer.subscribe(x => {  
  this.getNearSmassa();  
})
```

Ilustración 9.11 Eliminación del método suscrito a la tarea periódica e inclusión de un nuevo método

Como se puede observar, al evento periódico se le ha añadido un nuevo método después de eliminar el método relativo a la obtención de los datos de todos los aparcamientos. Este nuevo método lo que hace es cargar cada minuto los datos de la búsqueda, con la

posición y el radio que tiene guardados el controlador, para evitar hacer un uso intensivo de los datos al actualizar la posición constantemente. Adicionalmente, para recalcular la posición manualmente para este nuevo método, se añadió un botón a la vista para refrescar la posición.

Posteriormente, para restablecer el comportamiento habitual, se añadió un botón para limpiar la búsqueda y añadir el método de búsqueda de los datos a la tarea periódica.

```
clear() {  
  this.hidden = true;  
  this.radius = null;  
  this.getPosts();  
  this.subscription.unsubscribe();  
  this.subscription = this.timer.subscribe(x => {  
    this.getPosts();  
  });  
}
```

Ilustración 9.12 Método clear

9.6 VISUALIZACIÓN DE LOS CORTES DE TRÁFICO

En la página inicial del listado de aparcamientos de coches, se contempló la posibilidad de añadir un botón que redirigiera a un mapa para mostrar los cortes de tráfico que hay por toda la ciudad de Málaga.

Al principio dicha tarea se realizó incrustando cierto código HTML que se ofrece en la página del servicio web del Ayuntamiento de datos abiertos. Sin embargo, el mapa mostraba mucha información irrelevante y enlaces a sitios externos que no eran de interés para la tarea que concierne este trabajo. Por ende, se creó un nuevo método en el servicio inyectable, que obtuviese los cortes de tráfico, para que se mostraran en un mapa simple y mejor, ya que, entre otros motivos, el HTML incrustado hace uso de unas imágenes personalizadas para los marcadores y al incrustarlo en la aplicación, dichos marcadores no se pueden visualizar.

Sin embargo, al realizar la llamada, surgió un problema en la petición llamado CORS (Cross-Origin Resource Sharing), protocolo usado para realizar llamadas entre dominios diferentes. Dicho problema suele solucionarse en el lado del servidor, habitualmente, dando acceso a determinados dominios para realizar llamadas a sus direcciones, pero en el lado del cliente, es más difícil solucionarlo, ya que deben coincidir varias cabeceras, protocolos y, en muchos casos, el dominio debe poder autenticarse.

Se probó a añadir las cabeceras necesarias, pero no funcionó, sin embargo, antes de desistir, se encontró que NetBeans, en muchos casos, podía evitar este problema creando una clase específica para manejar esta situación.

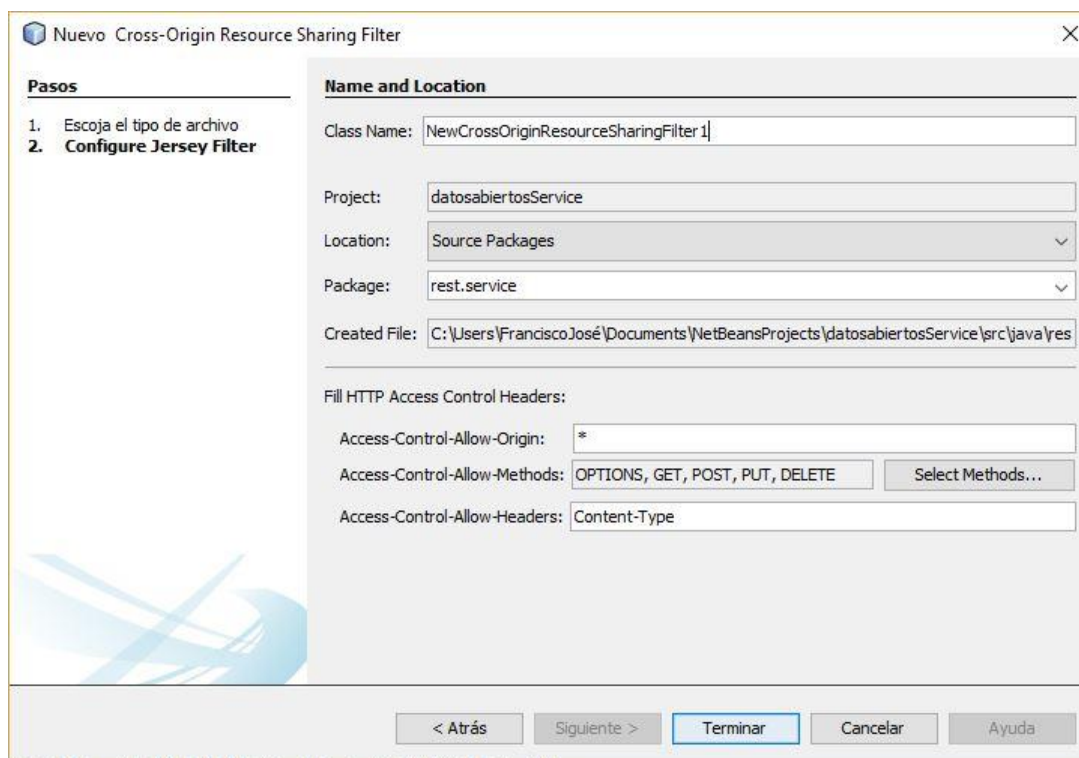


Ilustración 9.13 Creación del recurso

En consecuencia, se creó un método en el servicio web de dicho proyecto para obtener la información desde la web de datos abiertos del Ayuntamiento y que dicho método del servicio fuera llamado desde la aplicación móvil, solventando así, dicho problema y realizando la búsqueda de los cortes de tráfico.

9.7 OBTENCIÓN DE PRECIOS PARA LOS APARCAMIENTOS DE SMASSA

Como se mencionó en capítulos anteriores, la forma de obtención de precios y almacenamiento de los mismos según el modelo ofrecido por la web de datos abiertos del Ayuntamiento es muy poco eficiente y muy dependiente a los cambios, por lo que directamente se creará un método dentro del servicio web de dicho proyecto para obtener los precios para cada aparcamiento.

Una vez creado y probado el método, se realizó un nuevo método en el servicio inyectable para llamar a la URL de este método y obtener el precio de cada aparcamiento para unos minutos determinados.

Posteriormente, en la página de detalles de un aparcamiento, se añadió la entrada correspondiente para la introducción de los minutos deseados, un botón de aceptación y un campo para mostrar los datos, que quedará oculto en el caso de que los detalles que se estén mostrando, sean los correspondientes a un estacionamiento de bicicletas.

Price:

Minutes aprox

Q SEARCH PRICE

TICKET LOST

Ilustración9.14 Obtención de precios en la vista de detalles

Finalmente, en la página de detalles, se introdujo también un botón para el cálculo del costo de la pérdida del tique del aparcamiento, que igualmente llamará al método del cálculo de precios, pero con el máximo de minutos, que se corresponde con la pérdida del tique. Todos estos elementos relativos a las tarifas quedarán ocultos para el caso de los estacionamientos de bicicletas.

9.7 OBTENCIÓN DE LAS DISTANCIAS Y TIEMPO DE VIAJE

Para la realización de esta funcionalidad, se hizo uso de una de las APIs de Google específica para esta tarea (Distance Matrix API).

Su uso consiste en la creación de una URL a la que se le pasan parámetros como las coordenadas de origen, destino y modo de transporte, y devuelve un JSON con la respuesta, en la que incluye la distancia y el tiempo de viaje para el transporte seleccionado.

Se ha de comentar que los resultados no se muestran con el formato usado en España para los decimales y los miles, es decir, el punto representa los decimales y la coma los miles.

Como con el resto de llamadas a las APIs, se creó un nuevo método en el servicio inyectable de la aplicación móvil que es llamado desde el controlador de la página de detalles y que muestra los resultados en dos campos de la vista previstos para ello. Sin embargo, como ocurrió para la obtención de los cortes de tráfico, surgió un problema de CORS y el método de llamada a la API de Google se tuvo que crear en el servicio web que se creó para dar soporte a la aplicación móvil, y en el método inyectable, se llamó a la URL de este nuevo servicio creado.

Distance: 4.5 km Time: 12 mins

Ilustración 9.15 Resultado para el aparcamiento de SMASSA de la calle Salitre

Para finalizar, comentar que en un principio se intentó que esta funcionalidad mostrara los datos en la página donde se realiza el listado de los aparcamientos de SMASSA y los estacionamientos de bicicletas, pero puesto que se estaba haciendo uso de *data binding*, se realizaban multitud de consultas cada segundo y, aparte de que Google pueda cobrar por ello, la aplicación se quedaba bloqueada, por lo que para no aumentar la dificultad innecesariamente, se decidió añadir dichos datos a la página de detalles.

9.8 REALIZACIÓN DE PREDICCIONES

Un elemento extra que va añadir mucho valor a la aplicación va a ser la posibilidad de predecir las plazas libres de los aparcamientos de coches y los estacionamientos, para un día y una hora determinada.

En la realización de esta tarea se hará uso del servicio de Azure que se desplegó relativo a los experimentos predictivos llevados a cabo para los aparcamientos de SMASSA y los estacionamientos de bicicletas. Se emplearán los ejemplos de uso de estos servicios que ofrece la plataforma para un servicio determinado cuando es desplegado, para realizar los métodos que realicen las peticiones al servicio.

Se empezó por la creación en la página de detalles, de un apartado dedicado a esta funcionalidad. Fue necesario pasarle otro atributo a este componente: la capacidad del aparcamiento de SMASSA o del estacionamiento de bicicleta, según el caso. Para evitar problemas en la introducción de los datos por parte del usuario, se introdujeron dos menús de

selección desplegables, para seleccionar el día y la hora para la que se quiere realizar la predicción, un botón de aceptación y el campo correspondiente para mostrar los resultados.

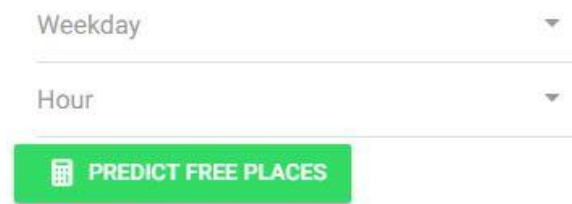
The image shows a user interface for making a prediction. It features two dropdown menus: the first is labeled 'Weekday' and the second is labeled 'Hour'. Below these menus is a prominent green button with a white calendar icon and the text 'PREDICT FREE PLACES' in white capital letters.

Ilustración 9.16 Menú de predicción

Posteriormente, se creó un método en el controlador de la página de detalles, que distinguiera, según si se trataba de un aparcamiento de SMASSA o un estacionamiento de bicicleta, a qué método del servicio inyectable tenía que llamar para que realizara la petición a los servicios de predicción creados en Azure. Esto es sencillo ya que, como se comentó, uno de los parámetros que se pasan a esta vista es el tipo del elemento que realiza la petición a la misma. Sin embargo, como en el caso anterior, surgió el problema del CORS, a pesar de que, en este caso, pasamos un *token* de autenticación a la URL cuando realizamos la petición al servicio.

Para solventarlo, se procedió como en casos anteriores, creando un nuevo método en el servicio web de soporte para la aplicación móvil y realizando la aplicación móvil la petición a este nuevo método.

9.9 AÑADIENDO IMÁGENES A LOS THUMBNAILS

Finalizando con la programación de la aplicación móvil, se procedió a agregar imágenes para los aparcamientos y estacionamientos disponibles. Para ello y para evitar realizar multitud de peticiones a servicios web para obtener imágenes de los sitios, se descargaron manualmente y se nombraron según el “id” del estacionamiento o aparcamiento, para posteriormente, en la vista, dentro de la etiqueta para cargar la imagen, en la fuente, llamar a un método que se encuentra dentro del componente que la procesa y que se encarga de devolver la ruta de dicho fichero.

```
<img src={{getImageUrl(item.poiID)}}  
onerror="this.src='assets/images/smassa/Vehicles_and_cars_12-128.png'"/>
```

Ilustración 9.17 Visualización de la imagen

```
getImageUrl(id){
  return 'assets/images/smassa/'+id+'.jpg';
}
```

Ilustración 9.18 Obtención de la ruta de la imagen

Se añadió una imagen por defecto para el aparcamiento o estacionamiento en el caso de que el fichero no existiera o no fuera encontrado, resultando así, el listado de aparcamientos de la siguiente manera.

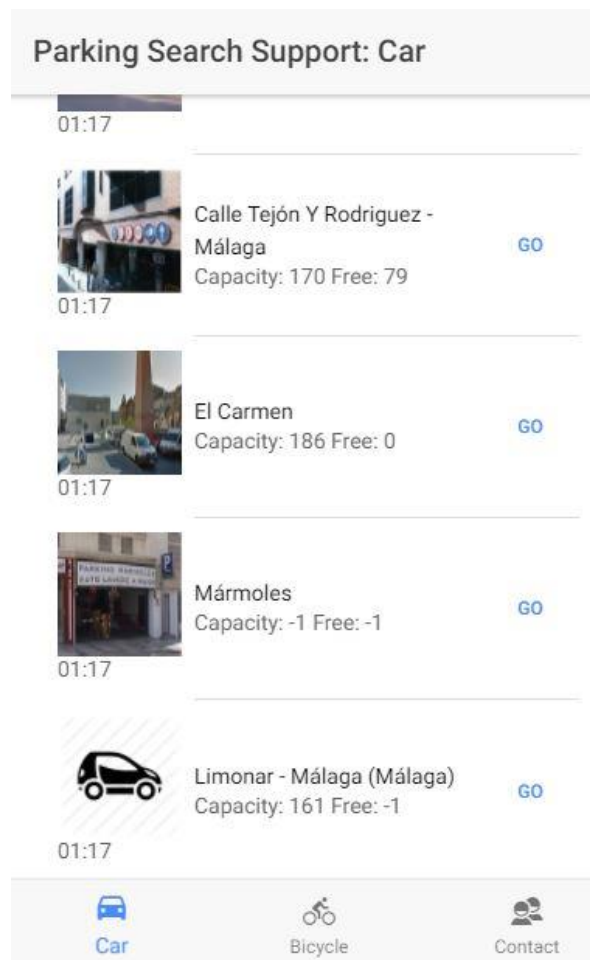


Ilustración 9.19 Listado de aparcamientos final

La página correspondiente con el listado de estacionamientos de bicicletas se visualiza de forma casi idéntica a esta, por lo que no se mostrará.

10 CONCLUSIONES

En este último apartado, se hablará de las conclusiones a nivel académico y personal, alcanzadas tras el desarrollo de dicho proyecto, así como futuros planes para el mismo.

10.1 CONCLUSIONES PERSONALES Y ACADÉMICAS

En lo académico, este proyecto me ha servido para conocer y aprender nuevas tecnologías, como son Ionic, AngularJS y Azure.

Si tuviera que destacar algo de cada uno, de Ionic destaca la facilidad con la que se pueden crear interfaces visualmente muy atractivas, pero por contrapartida, el proyecto es excesivamente pesado, pues importa una cantidad enorme de librerías que luego no son usadas. En concreto, este proyecto pesa **284 MB**.

A AngularJS no puedo sacarle ningún defecto. Aparte de ser ligero, con una cantidad mínima de código, es capaz de realizar tareas que en otros lenguajes requerían amplios conocimientos del mismo y muchas más líneas. Además, su uso es muy sencillo incluso para realizar funcionalidades bastante complejas, y aunque nos encontremos con algo más difícil de la cuenta, es una tecnología que está en expansión y usándose cada vez más y cuenta con multitud de foros donde hay soluciones para prácticamente cada problema que nos podamos encontrar.

Respecto a Azure, y en concreto, al servicio de **Azure Machine Learning Studio**, incluso para haber sido desarrollado por un gigante como es Microsoft, me ha sorprendido por lo tremendamente potente que es y la facilidad de uso. No requiere ningún concepto de programación para realizar experimentos ni desplegar servicios webs, los únicos conocimientos que se requieren, son los relativos al ámbito del objeto de estudio, expandiendo de manera significativa, el número de usuarios a los que puede ser accesible.

Sin duda, este proyecto me ha servido para emplear todas las buenas prácticas que he ido aprendiendo a lo largo de toda esta etapa que llega a su fin. Durante la realización del mismo, he podido averiguar que, a partir de los errores que han ido surgiendo y que he ido solventando, soy capaz de gestionar y llevar adelante un proyecto, solucionar los problemas que aparezcan y dar soluciones eficaces. Me ha servido para echar la vista atrás y ver cómo he

evolucionado desde mi primer año en la carrera, los conocimientos que he adquirido y cómo he madurado a lo largo de todo el recorrido.

En lo personal, la finalización de este TFG supone el cierre de una intensa etapa, en la que no ha sido todo de color de rosas, pero que ha servido para forjarme como programador, como ingeniero y como persona. Con esto, se pone fin al último capítulo de esta etapa, pero se inicia otra mucho más intensa, mucho más dura, pero que se llevará exitosamente gracias a las experiencias adquiridas y a todo lo que nos han enseñado los profesores durante estos años.

10.2 FUTURAS MEJORAS

En la aplicación móvil se ha dejado una pestaña de navegación vacía con vistas de futuro, para añadir nuevas funcionalidades. El trabajo, globalmente, ha sido abordado con vistas al futuro. Esta afirmación no pretende ser meramente una frase elegante, en esta profesión, el aprendizaje constante forma parte del día a día y qué mejor forma de actualizarse, que mejorar un proyecto que se inició en una etapa temprana e ir contemplando la evolución de uno mismo a través de su trabajo.

Entre las mejoras planteadas, destacan:

- **Adición de los servicios de aparcamientos S.A.RE.** La información sobre los mismos se encuentra actualmente almacenada en la base de datos, solamente queda añadirlo a la aplicación móvil en su apartado correspondiente, de manera similar a los aparcamientos y estacionamientos.
- **Adición de los servicios de aparcamientos de motos.** Aunque el servicio de datos abiertos del Ayuntamiento no ofrece información relacionada sobre la ocupación en tiempo real sobre los aparcamientos de motos, sí ofrece la localización y las tarifas de los mismos, por lo que supone una buena idea para extender el uso de la aplicación
- **Búsqueda de los aparcamientos de movilidad reducida**

BIBLIOGRAFÍA

<http://datosabiertos.malaga.eu/dataset/ocupacion-aparcamientos-smassa> - Ocupación aparcamientos SMASSA – Datos Abiertos Ayto. Málaga - [Accedido el 4 de marzo de 2017]

<http://datosabiertos.malaga.eu/dataset/malaga-bici> - Málaga bici – Datos Abiertos Ayto. Málaga - [Accedido el 4 de marzo de 2017]

<http://datosabiertos.malaga.eu/dataset/tarifas-aparcamientos-l-d> - Tarifas coches aparcamientos L-D – Datos Abiertos Ayto. Málaga - [Accedido el 4 de marzo de 2017]

<http://datosabiertos.malaga.eu/dataset/tarifas-coches-aparcamientos-l-v> - Tarifas coches aparcamientos L-V – Datos Abiertos Ayto. Málaga - [Accedido el 4 de marzo de 2017]

<http://datosabiertos.malaga.eu/dataset/tarifas-coches-aparcamientos-s-d-f> - Tarifas coches aparcamientos S-D-F – Datos Abiertos Ayto. Málaga - [Accedido el 4 de marzo de 2017]

<http://datosabiertos.malaga.eu/dataset/plazas-aparcamientos-s-a-re> - Plazas de aparcamientos S.A.RE.– Datos Abiertos Ayto. Málaga - [Accedido el 4 de marzo de 2017]

<http://datosabiertos.malaga.eu/dataset/tarifas-s-a-re> - Tarifas S.A.RE. – Datos Abiertos Ayto. Málaga - [Accedido el 4 de marzo de 2017]

<https://ionicframework.com/docs/intro/installation/> - Installing Ionic – [Accedido el 16 de marzo de 2017]

<https://stackoverflow.com/questions/14096429/how-to-delete-a-mysql-record-after-a-certain-time> - How to delete a MySQL record after a certain time – Stackoverflow – [Accedido el 19 de marzo de 2017]

<https://www.mkyong.com/java/how-to-read-and-parse-csv-file-in-java/> - How to read and parse CSV file in Jav – Mkyong – [Accedido el 22 de marzo de 2017]

<https://vichargrave.github.io/articles/2011-07/java-csv-parser-using-regular-expressions> - Java CSV Parser Using Regular Expressions – [Accedido el 22 de marzo de 2017]

<https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-create-experiment> - Machine learning tutorial: Create your first data science experiment in Azure Machine Learning Studio – Microsoft Docs – [Accedido el 27 de marzo de 2017]

<https://stackoverflow.com/questions/10100936/file-downloading-in-restful-web-services> - File downloading in restful web services – Stackoverflow – [Accedido el 17 de abril de 2017]

<http://datosabiertos.malaga.eu/dataset/cortes-de-trafico> - Cortes de tráfico - Datos Abiertos Ayto. Málaga - [Accedido el 19 de abril de 2017]

<https://www.joshmorony.com/ionic-2-how-to-use-google-maps-geolocation-video-tutorial> - Ionic 2 & 3: How to Use Google Maps & Geolocation – Joshmorony – [Accedido el 21 de abril de 2017]

<https://ionicframework.com/docs/components/#overview> – Components – Ionic Component Documentation [Accedido el 22 de abril de 2017]

<https://github.com/tomchentw/react-google-maps/issues/260> - Auto center based on marker's location – GitHub – [Accedido el 2 de mayo de 2017]

<https://stackoverflow.com/questions/10343892/glassfish-how-to-set-access-control-allow-origin-header> - How to set Access-Control-Allow-Origin header – Stackoverflow – [Accedido el 9 de mayo de 2017]

<http://blog.enriqueoriol.com/2016/11/ionic2-lifecycle-events.html> - Navegación en Ionic 2: lifecycle events – [Accedido el 13 de mayo de 2017]

<https://stackoverflow.com/questions/36600291/angular2-stop-cancel-clear-observable-in-routerondeactivate> - Angula2 – stop / cancel /clear observable in routerOnDeactivate() – Stackoverflow – [Accedido el 20 de mayo de 2017]