UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

# CityControlPanel: Visualización on-line de datos de ciudades

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor**: Marcos Giménez García

**Tutor**: Pietro Manzoni

**Curso académico:** 2014-2015

# Resumen

CityControlPanel tiene como objetivo diseñar una aplicación para navegadores web que presente, de forma gráfica, información disponible de una ciudad. Los datos se obtendrán desde bases de datos abiertas proporcionadas por el ayuntamiento y se ofrecerán a demanda del consumidor mediante un menú de opciones. Cada evento mostrado en el mapa tendrá su cajón de información descargada in situ desde el proveedor de datos.

**Palabras clave:** Ciudad Inteligente, Datos Abiertos, Web, OpenStreetMap, OpenLayers, Valencia

# Abstract

The aim of CityControlPanel is to design a web application that presents information about a city in a graphic way. Data will be obtained from open databases provided by the council. They will be offered to the user through a menu of options. Each event shown on the map will have its information, downloaded from the data provider, pop up.

**Keywords :** Smart City, Open Data, Web, OpenStreetMap, OpenLayers, Valencia

# Tabla de contenidos

# 1. Introducción

## 1.1 Motivación

El término Open Data ha ido instaurándose poco a poco desde el año 2010, numerosas organizaciones se han ido sumando a esta técnica (universidades, prensa, instituciones públicas) y ofrecen información, que antes estaba controlada mediante algún tipo de licencia, a disposición del público en general.

A través de estos datos CityControlPanel pretende ofrecer la información, que se pueda clasificar en un mapa, de una ciudad. Los grandes servidores de mapas existentes se limitan a mostrar la información del transporte u ocio de una ciudad mientras que los datos abiertos proporcionados por el ayuntamiento de una ciudad (o sus empresas públicas) albergan esa información además de muchos otros tipos, que pueden resultar muy interesantes para el ciudadano normal. Esta información es más detallada en muchas ocasiones por la cercanía de los servicios.

## 1.2 Objetivos del proyecto

Esencialmente, este proyecto pretende desarrollar una aplicación que recoja los datos de una ciudad, en este caso Valencia, y tratarlos con el fin de mostrar esta información al usuario de una forma agradable, sencilla y rápida usando como plataforma los navegadores web cotidianos. Para su éxito se deben cumplir los siguientes objetivos:

- Analizar sistemas que puedan ofrecer un servicio similar al propuesto en este proyecto y de esta forma trazar un camino a seguir basado en las virtudes y carencias existentes en el mercado.
- Listar las tecnologías necesarias para el desarrollo del proyecto.
- Características fundamentales que se deben satisfacer en la aplicación final.
- Crear un diseño conceptual de la aplicación.
- Desarrollar la aplicación en función de los resultados obtenidos en anteriores apartados y evaluar el resultado final.

## 1.3 Estructura de la memoria

El documento está estructurado en nueve capítulos, siguiendo las fases de desarrollo del proyecto paso por paso. En este capítulo se describe una idea general del proyecto así como objetivos a cumplir.

En el capítulo 2 **(Antecedentes)** se analizan los principales buscadores de mapas existentes y las características diferenciadoras de cada uno de ellos. Acto seguido se realiza un análisis de las aplicaciones más populares en el entorno de Valencia describiendo que ofrece cada una de ellas y con ello formalizar las características de la aplicación a desarrollar.

En el capítulo 3 **(Herramientas)** se describen las distintas tecnologías que se usan durante el desarrollo del proyecto. Concretamente los servicios de los que se nutre la aplicación, las librerías de las que hace uso para su implementación, los lenguajes en los que estará basada, los tipos de formato en los que recogerá los datos y el software del que se hará uso durante su desarrollo.

En el capítulo 4 **(Análisis)** se listan los requisitos necesarios de la aplicación basándose en la idea original de la aplicación y en el estudio del entorno realizado en el capítulo 2. Dichos requisitos están diferenciados según su desarrollo en el cliente o en el servidor.

En el capítulo 5 **(Diseño de la aplicación)** se especifica formalmente cómo será la aplicación en sus distintos niveles:

- Mediante los casos de uso se listan las distintas acciones que puede realizar el usuario.
- A través de bocetos se idea una base de cómo será la interfaz gráfica para posteriormente desarrollar entorno a ella.
- Los diagramas de sistema muestran gráficamente las acciones que realizará la aplicación en los distintos eventos posibles.

En el capítulo 6 **(Metodología de desarrollo)** se explica el desarrollo de la aplicación detallando las partes relevantes de este. La especificación del desarrollo se basa en tres apartados básicos: Nivel de presentación (interfaz gráfica final), nivel de persistencia (los ficheros de los que se obtiene información) y nivel de aplicación (detalle de las líneas de código relevantes para el desarrollo).

En el capítulo 7 **(Resultados)** se recuerdan los objetivos planteados en el capítulo 4 comprobando si se han ido cumpliendo cada uno de ellos.

En el capítulo 8 **(Conclusiones)** se detallan las conclusiones obtenidas una vez finalizado el proyecto, problemas que han existido a lo largo de su desarrollo y posibles mejoras para un futuro.

Para finalizar en el capítulo 9 **(Bibliografía)** se añaden referencias a documentos externos de los que se ha hecho uso para el desarrollo del proyecto.

# 2. Antecedentes

## 2.1 Introducción

En este apartado se realiza un análisis del entorno para obtener una descripción detallada de los sistemas existentes más conocidos en el área de la aplicación. Se consulta desde los grandes servicios de mapas, creados por las corporaciones más importantes en el sector de la informática, hasta aplicaciones en el ámbito local de la ciudad de Valencia.

## 2.2 Servicios de mapas globales

El entorno en el que se desarrolla este proyecto está vinculado directamente al servicio de mapas vía web. Desde que 'Google Inc.' comenzará su andadura con la aplicación web Google Maps, allá por el año 2005, han ido apareciendo distintos tipos de servicios, ofreciendo algo parecido a este o bien intentando cubrir una necesidad para diferenciarse. En las siguientes subsecciones se listan los detalles de los servicios más relevantes.

### 2.2.1 Google Maps

Google Maps es el servicio precursor de mapas en la web desarrollado por 'Google Inc.'. Este servicio tiene su variante en forma de aplicación de escritorio llamada 'Google Earth' y además ofrece su propia aplicación en dispositivos Android, iOS y Windows Phone. El logo actual de la aplicación se puede observar en la figura 2.1. Google Maps dispone de muchas características que se han ido implantando año tras año desde su puesta en marcha hasta convertirlo en el buscador de mapas más completo hasta la fecha. Incluye vista de satélite, mapa cartográfico, estado del tráfico en tiempo real, indicación de negocios, edificios públicos, monumentos históricos y servicios de transporte. También ofrece rutas para llegar a un punto mediante coche, bici, servicio de transporte o caminando. A continuación se listan sus principales características:


Figura 2.1: Google Maps

- **Google Street View:** Vistas de las calles de las principales ciudades y carreteras con imágenes a 360º. Esta herramienta ha sido cuestión de debate en muchos países porque podía violar la privacidad de las personas. De hecho, en países como Alemania acabo siendo retirado. Recientemente también se han incluido vistas en 360º de comercios que así lo soliciten o de vistas naturales, como la gran barrera de coral australiana en el proyecto Google Underwater Street View. También ofrece vistas de pájaro de las ciudades gracias a Google Aerial View.
- **Vista satélite:** Google dispone de imágenes de todo el planeta recogidas por satélite. En su mayoría son proporcionadas por la empresa DigitalGlobe y su satélite QuickBird. Aunque no todas las imágenes son recogidas vía satélite, en algunos casos están hechas desde aviones que vuelan a 10.000 metros de altura.

- **Variantes:** Además de Google Maps han surgido variantes a este centradas fuera de la Tierra:
    - **Google Moon:** En 2005, Google lanzó la aplicación web Google Moon, en honor al 36º aniversario del aterrizaje del hombre en la luna. El funcionamiento de la web era bastante similar al de Google Maps, y en el podíamos visualizar la superficie lunar con detalles sobre las zonas en las que ha habido un alunizaje.
    - **Google Mars:** Un caso similar al de Google Moon pero con el planeta rojo. Como añadido en este se pueden visualizar el mapa con imágenes infrarrojas y de relieve de Marte.
    - **Google Sky:** Una visualización del mapa estelar con las distintas constelaciones y galaxias descubiertas.
- **Google Local:** Sirve para buscar negocios alrededor de tu posición.
- **Google Traffic:** Visualización del estado del tráfico en tiempo real codificado en varios colores según la densidad de este.
- **Google Transit:** Este servicio de Google se encarga de calcular la ruta óptima para llegar a un lugar concreto, estima el tiempo que tardas en realizarla y te guía mientras la realizas. Esta ruta puede variar en función del tráfico o las obras en ese momento. En algunos lugares la capa que muestra las rutas está disponible, pero no así la guía.
- **Google My Maps:** Con esta función fue creada para que los negocios pudieran dibujar sobre el mapa sus propios marcadores o polígonos. Cada elemento en el mapa dispone de una etiqueta donde se puede insertar texto común, texto enriquecido, video embebido o cualquier contenido que pueda incluirse con etiquetas HTML.
- **Indoor Google Maps:** En 2011 Google incluyó mapas de espacios cerrados tales como aeropuertos, museos o centros comerciales. Está función fue incluida en dispositivos Android aunque más tarde fue lanzada también para iOS.

Aunque 'Google Maps' es de uso gratuito para el consumo, no ocurre lo mismo a la hora de desarrollar nuevas aplicaciones que hagan uso de esta. Debido a que la API de 'Google Maps' pasó a ser de pago en los últimos años, se produjo la migración de una parte de los desarrolladores a alternativas de libre uso.

### 2.2.2 Bing Maps

Bing Maps es el servicio de mapas creado por Microsoft para hacer frente a Google Maps. Pasando a lo largo de su vida por varios nombres (Live Search Maps, Windows Live Maps, Windows Live Local, MSN virtual Earth) es ahora cuando adquiere su actual nombre, a causa del navegador web Bing.

Figura 2.2: Bing Maps

En la figura 2.2 se observa el actual logo de la aplicación. La tecnología de Bing Maps está basada en otras ya existentes de Microsoft, como Microsoft MapPoint y TerraServer.

Bing Maps ofrece la información típica de callejero (paradas de metro, hospitales, universidades...) Así como puntos de interés público creados por los usuarios. En algunos países Bing Maps solo dispone de la información en grandes vías, dejando de lado las calles locales.

Además de la información de calles, Bing Maps también dispone de otros servicios:

- **Vista satélite:** Microsoft proporciona imágenes recogidas por satélite que varían en detalle dependiendo de la zona geográfica. En las zonas más detalladas dispone de una resolución de 4,5 pixeles por metro. Entre los países con mayor detalle se encuentran: Estados Unidos, Canadá, Reino Unido, Italia, Alemania, Australia, India, Nueva Zelanda y Japón.
- **Vista de pájaro:** En ciudades de Estados Unidos, Canadá, Japón, y Europa Bing Maps proporciona imágenes recogidas desde cuatro ángulos. Estás imágenes son más detalladas que las proporcionadas por la vista satélite. En ellas se pueden visualizar detalles como las señales o los peatones.
- **StreetSide:** Servicio similar a Google Street View. Las calles son recogidas con cámaras de 360º situadas en la parte superior de un coche. El servicio no está del todo extendido.
- **Mapas 3D:** Esta vista permite ver los edificios de las ciudades a 360º y con rotación angular. Es una vista realista gracias a que los edificios se muestran recogidos con imágenes recogidas con vista aérea.
- **ClearFlow:** Se trata de una tecnología realizada por Microsoft para mostrar la congestión de carreteras e incluso la previsión de ella, para así poder ofrecer la ruta más rápida al usuario.

### 2.2.3 Apple Maps

Apple Maps es un servicio de mapas proporcionado por la compañía Apple Inc. y distribuido en sus dos sistemas operativos iOS y OS X. Su logo actual es el mostrado en la figura 2.3. Fue lanzado en Septiembre de 2012 para reemplazar a Google Maps. Sus comienzos fueron complicados debido a problemas básicos en la aplicación, tales como situar ciudades en el punto geográfico erróneo y por carencias como un "Google Street View" propio.

Figura 2.3: Apple Maps

Su función más popular y por la que se diferenciaba de la competencia era "Flyover". Esta característica se trata de una visualización de los mapas de las principales ciudades en 3 dimensiones, dándoles una vista de pájaro realista. Además también cuenta con las siguientes características:

- Mapas recogidos por la propia compañía, sin fuentes de terceros.
- Vista satélite.
- Permite al usuario usar la aplicación "Yelp!". Una App que ofrece información sobre las empresas.
- Estado del tráfico, recogiendo datos anónimos desde todos los iPhone.

### 2.2.4 Yahoo! Maps

Yahoo! Maps [] fue un servicio de mapas gratuito y potenciado por Nokia's Here Maps. El servicio fue cancelado por Yahoo! en junio de 2015. En ese momento su imagen corporativa era la mostrada en la figura 2.4. Entre sus principales características

Figura 2.4: Yahoo Maps

figuraban:

- **Predicción del tiempo:** con la colaboración de "The weather channel".
- **Libro de direcciones:** Los usuarios de Yahoo! podían registrar las direcciones que más usaban.
- **Tráfico en vivo:** Marcadores con incidentes en el tráfico y la condición en la que se encontraba una autovía podían consultarse.
- **Buscador de puntos de interés:** Podía ser usado para buscar negocios o lugares turísticos cerca de tu punto geográfico.
- **Dirección de rutas:** Te indicaba los pasos a seguir para llegar a un punto mediante texto.

Los desarrolladores tenían a su disposición APIs para incluir los mapas de Yahoo! en sus páginas web. Existían APIs simples en XML, para flash y en AJAX.

### 2.2.5 Here

Here es un servicio de mapas creado por la compañía finlandesa Nokia. Es un servicio originalmente creado para los dispositivos móviles de dicha marca bajo el nombre de Nokia Maps y Ovi Maps. Actualmente se puede acceder y descargar los mapas de Here desde Windows (desktop y mobile) y desde Android e iOS. La aplicación se puede identificar con figura 2.5. Durante el desarrollo de este proyecto, concretamente el 3 de Agosto de 2015, Nokia ha anunciado un principio de acuerdo para vender este servicio al consorcio alemán de fabricantes de coches, encabezado por Mercedes, BMW y Audi.

Figura 2.5: Here

Las características principales de Here son las siguientes:

- Páginas detalladas de las ciudades más populares del mundo.
- Tiempo actual.
- Mapas en 3 dimensiones de 25 ciudades.
- Opciones del transporte público.
- Vista de satélite.

Solo para aplicación móvil

- Guía de voz a pie y en coche.
- Cálculo de ruta en vivo según el tráfico.
- Integración con redes sociales.
- Clima actual y predicción del tiempo de 7 días.
- Vista nocturna.
- Guardar lugares favoritos.
- Permite reportar errores en el mapa.

## 2.3 Servicios de mapas locales

El sector de información geográfica a nivel local está centrado casi en su totalidad a las APP móviles. Cada una de ellas con un propósito concreto (e.g. dar información sobre los horarios del metro o información turística). En los siguientes apartados se listan las aplicaciones más destacadas con datos geográficos de Valencia.

### 2.3.1 AppValencia

AppValencia es la aplicación oficial de Valencia proporcionada por el ayuntamiento. Esta APP sería la más relacionada con nuestro proyecto pues su fuente de datos es los datos abiertos del ayuntamiento de Valencia. Estos datos le permiten mostrar el tráfico de Valencia en tiempo real o informar de la parada de bus más cercana. Como añadido, permite gestionar trámites siempre y cuando dispongamos de un certificado digital, aunque este apartado queda fuera del ámbito del proyecto.

Esta aplicación está disponible en Android, a partir de la versión 2.3 Gingerbread y en iOS, a partir de la versión 6.0. En la imagen 2.6 se observa la interfaz gráfica de la aplicación.


Figura 2.6: AppValencia

### 2.3.2 EMT Valencia

La empresa municipal de transporte (EMT) de Valencia también dispone de su aplicación oficial diseñada por ellos mismo. En ella se facilita hasta tres opciones para viajar de un punto a otro de la ciudad, ya sea en autobús, metro, tranvía, valenbisi o a pie. También ofrece un listado de todos los horarios de sus servicios e incluso la opción de consultar y recargar los viajes de nuestro bono.

También dispone de una versión web de escritorio en la que se puede consultar los horarios de los servicios. La App está disponible en Android a partir de la versión 2.1 y en iOS a partir de la versión 6.0 (solo para iPhone). A la versión web se puede acceder desde http://www.emtvalencia.es/ciudadano/index.php. En las figura 2.7 se muestra la interfaz móvil y web.

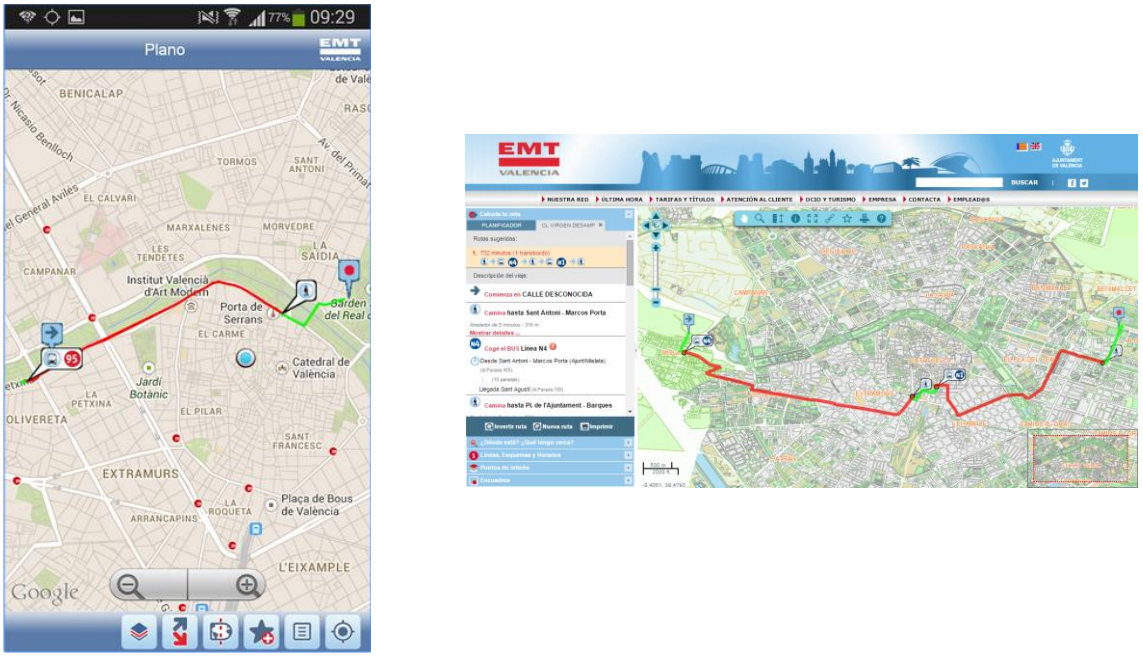


Figura 2.7: EMT Valencia

## 2.3.3 App Guía Valencia

APP Guía Valencia es una aplicación orientada al sector del turismo. Esta App fue desarrollada por Clever Work System SL. En ella se encuentra la oferta de restaurantes y hoteles de Valencia, así como el ocio nocturno o las fallas. Para mostrar la información geográfica usa los mapas nativos de cada sistema operativo. En el caso de Android Google Maps y en el caso de iOS, Apple Maps.

La aplicación se puede descargar a partir de la versión 2.3.3 de Android y en iOS a partir de la 7.0. En la figura 2.8 se observa la interfaz de la aplicación en Android.


Figura 2.8: APP Guía Valencia

### 2.3.4 Aplicaciones metro

En el sector del servicio de metroValencia existen varias aplicaciones disponibles, todas ellas ofrecen en esencia lo mismo, en la siguiente lista se enumera cada una de ellas y su factor diferencial:

- **MetroValencia (SeleRED):** Esta aplicación cuanto con el apoyo de MetroValencia y de Ferrocarriles de la Generalitat Valenciana (FGV). Te informa de los horarios para tu viaje solicitado y los transbordos que debes realizar. Además añade la posibilidad de indicarte donde está la parada de metro más cercana. Disponible en Android a partir de la versión 2.2 y en iOS 6.0 o posterior. También se puede visualizar las paradas de metro sobre mapa en su web oficial. En la figura 2.9 se observa la interfaz de la aplicación.
- **Metro Valencia (Jomofer):** Con esta aplicación puedes consultar los horarios, ver transbordos, mapa general del metro, incidencias y consultar el saldo de viajes. En Android 3.0 o superior. En la figura 2.10 se observa la interfaz con las paradas de metro señaladas.
- **Metro Valencia offline:** Aplicación diseñada por GreeGrowApps. Permite consultar los horarios del metro sin necesidad de estar conectado a internet. Si los horarios cambiaran, la aplicación se los descargaría en el momento de tener conexión. Una opción interesante para consultar algún dato en el caso de hallarse dentro del metro y no disponer de conexión. Disponible en Android 2.2 o versiones superiores e iOS 8.1. En la figura 2.11 se indica la interfaz de la aplicación.

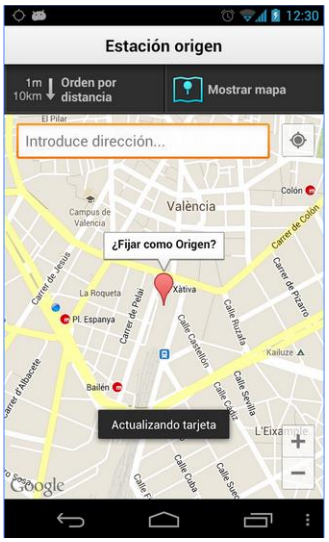


Figura 2.9: MetroValencia (SeleRed)

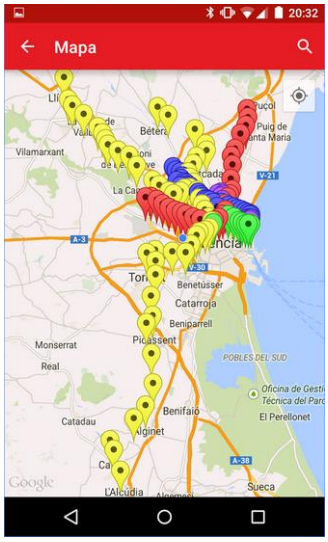


Figura 2.10: Metro Valencia (Jomofer)

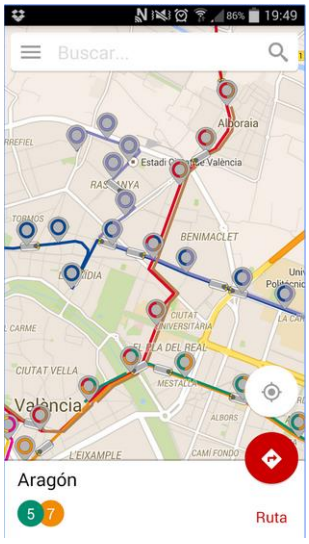


Figura 2.11: Metro Valencia offline

### 2.3.5 VLC Valencia

VLC Valencia es una aplicación cuyo fin es informar de las opciones turísticas de Valencia en general. Se lista distintas opciones de hoteles, restaurantes, museos, monumentos y el ocio. Esta App, desarrollada por la Fundación de turismo de Valencia, muestra cada una de esas opciones en una capa basada en Google Maps para orientar la localización.

Esta App está disponible para versiones Android 2.2 o superior y a partir de la versión 6.0 de iOS.

### 2.3.6 Valencia Maps and Walks

Valencia Maps and Walks es una opción de guía turística desarrollada por GPSmyCity.com, Inc. orientada mayoritariamente al usuario extranjero por encontrarse en inglés, aunque bien es cierto que incluye la opción de traducir con Google Translate. Una característica destacada de esta aplicación es que al contrario que la mayoría no usa la API de Google Maps sino la opción libre de OpenStreetMap.

Existe disponible una versión Lite de la aplicación de forma gratuita. La versión completa se puede obtener por 4,99 €.

Se puede encontrar en Android a partir de la versión 4.0 IceCream y en iOS 7.0 en adelante. En la figura 2.12 se muestra su interfaz en un iPad.

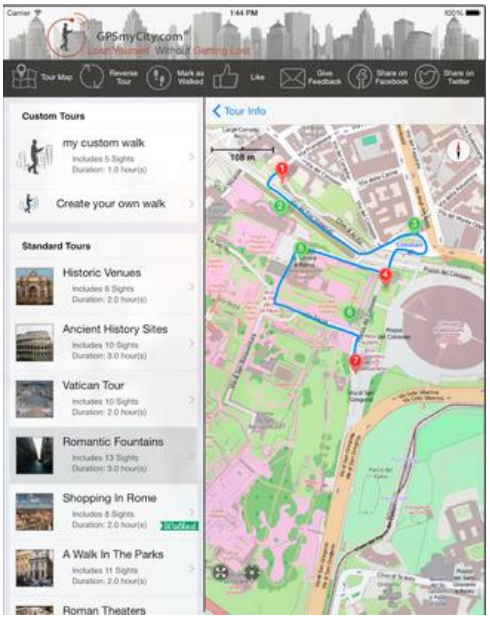


Figura 2.12: Valencia Maps and Walks

### 2.3.7 Now Valencia

Now Valencia es una opción para descubrir la oferta de ocio de Valencia, restaurantes, discotecas y pubs, diseñada por Imagina Labs. Todas sus opciones se pueden visualizar en un mapa cartográfico de la ciudad. Como añadido, se pueden guardar los lugares favoritos para poder encontrarlos en el mapa siempre que se desee y compartirlos en redes sociales.

Esta App está disponible en dispositivos Android 2.3 en adelante y en iPhone e iPad con iOS 6.0 y siguientes versiones. La interfaz de la aplicación se indica en la figura 2.13.

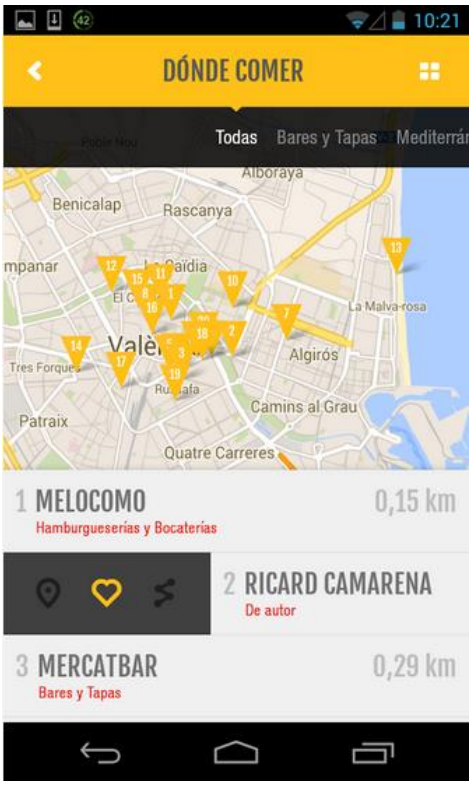


Figura 2.13: Now Valencia

### 2.3.8 Guía de Valencia – mi nube

Guía de Valencia es una opción disponible solo en la App Store y Android. Desarrollada por 'minube' en ella se visualiza en el mapa lugares turísticos a visitar, restaurantes, hoteles y guardar las opciones favoritas.

Requiere de S.O Android 2.2 mínimo y de iOS 7.0 o posterior. La interfaz en iPad se muestra en la figura 2.14.

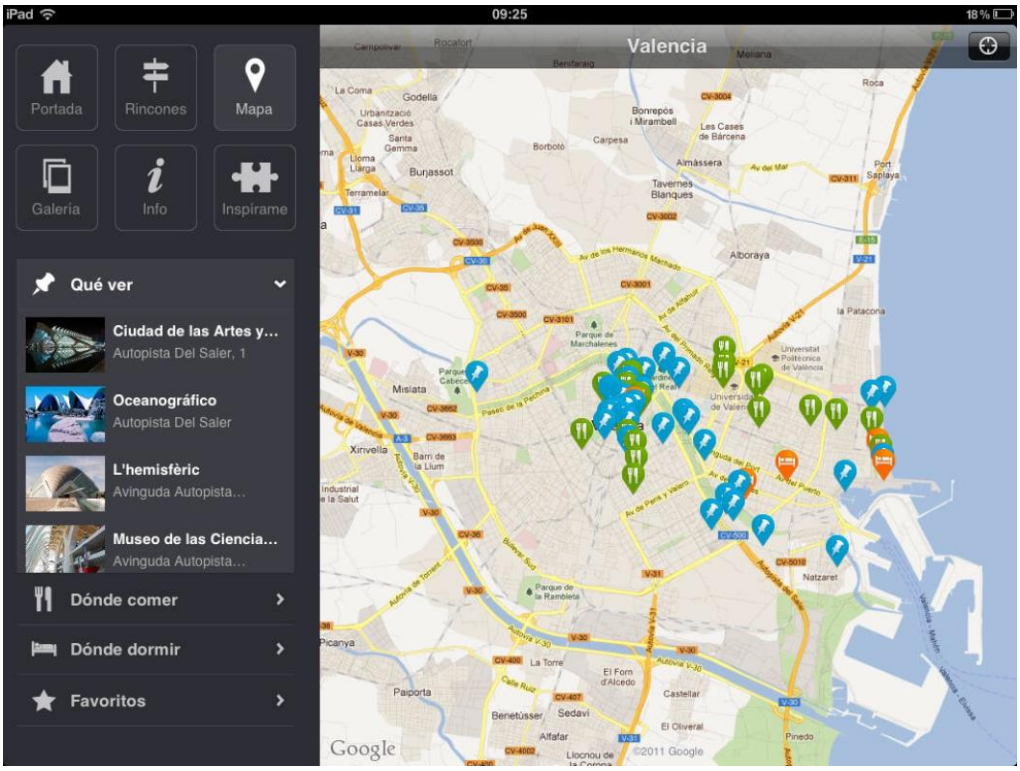


Figura 2.14: Guía de Valencia – Mi nube

### 2.3.9 Bici Valencia (Valenbisi)

La aplicación Bici Valencia, desarrollada por GreenGrowApps, permite ver sobre el mapa todas las estaciones de Valenbisi. Dentro de cada estación se puede visualizar las bicis disponibles y los bornes libres, además de consultar los carriles bicis de toda la ciudad.

La aplicación está disponible en Android 1.6 y versiones posteriores e iOS 5.0 en adelante. La interfaz en Android se puede observar en la figura 2.15. Se puede visitar la página oficial de Valenbisi, donde se facilitan los datos de cada estación sobre un mapa de la ciudad.

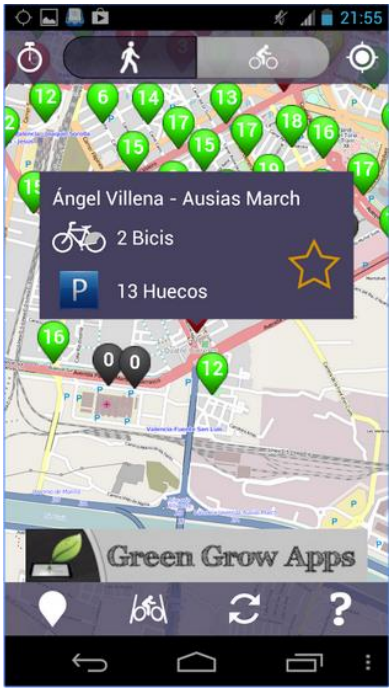


Figura 2.15: Valenbisi

## 2.4 Comparativa

Existen más servicios de información disponibles que ofrecen datos de la ciudad de Valencia. En el apartado anterior se han seleccionado los de mayor popularidad y diversidad. A continuación se expone una comparativa de las aplicaciones locales analizadas:

### 2.4.1 Servicios de mapas

En la gráfica 2.16 se expone el servicio de mapeado que usan las aplicaciones analizadas.

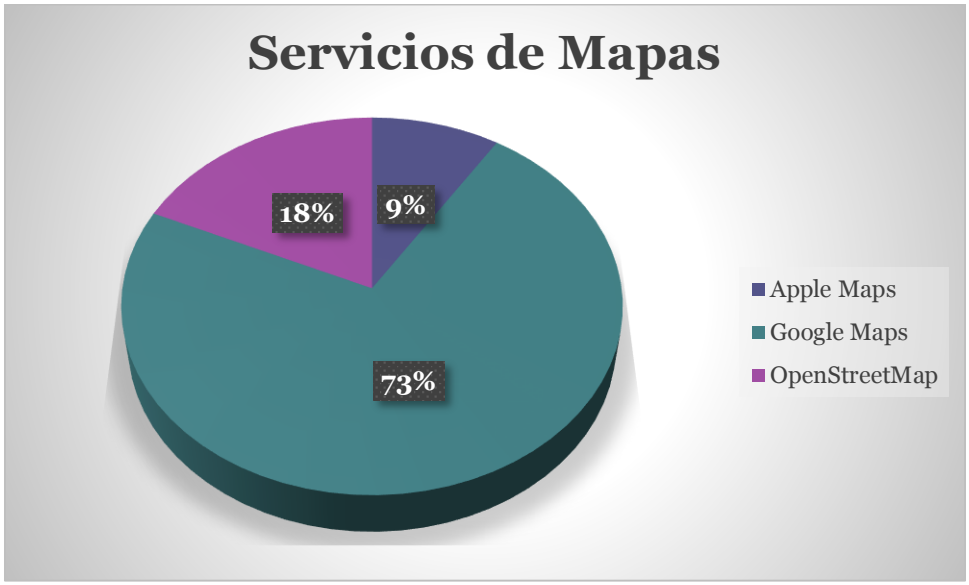


Figura 2.16: Servicios de Mapas

Con los datos de la gráfica demuestran que el 73% de los desarrolladores han usado la API de Google Maps para su aplicación, aunque pueda conllevar un gasto económico en el caso de superar el número mínimo de visualizaciones.

En segundo lugar se posiciona la opción de OpenStreetMap con un 18% de uso y finalmente Apple Maps obtiene un 9% por aplicaciones que lo usan en sus versiones de iOS.

### 2.4.2 Sector orientado

Otro aspecto fundamental a considerar es el ámbito en el que se orientan las aplicaciones. En la gráfica 2.17 se visualizan los resultados obtenidos:


Figura 2.17: Ámbito aplicación

El resultado revela que la mayor parte de las aplicaciones están orientadas al transporte (uniendo servicios de autobuses, metro y alquiler de bicicletas). Concretamente un 46,15% están orientada a ellas.

En segundo y tercer lugar la opción preferida es el turismo y el ocio, respectivamente. Destaca, por su ausencia, las aplicaciones de gestión e información de la ciudad. Tan solo la aplicación oficial del ayuntamiento tiene opciones de este tipo, lo que supone el 7,69% del total.

### 2.4.3 Plataforma

En la gráfica 2.18 se observa el entorno al que están orientadas las distintas opciones.

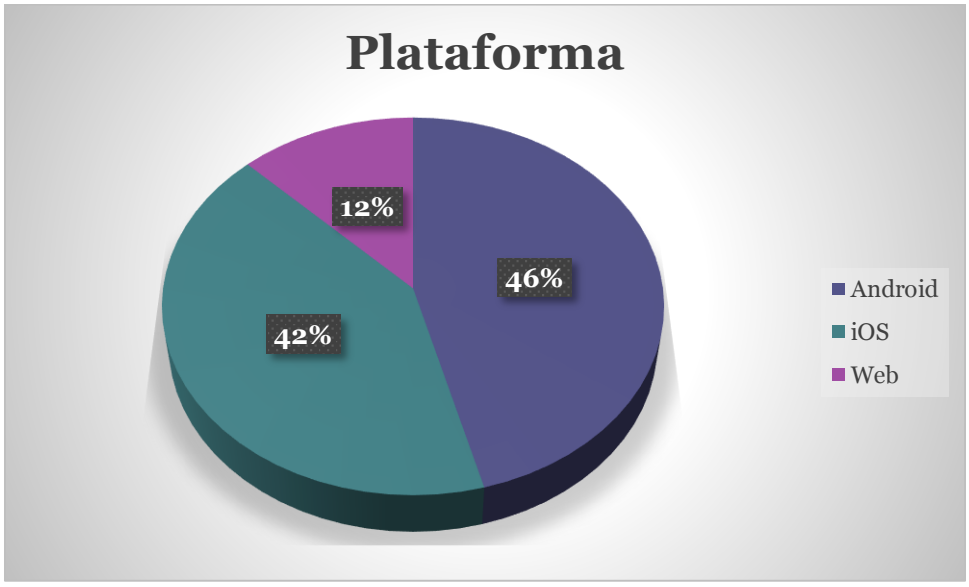


Figura 2.18: Plataforma

La gran mayoría de opciones de información cartográfica de la ciudad se hayan para los dispositivos móviles (Android 45,8% e iOS 41,6%). En detrimento de las opciones vía web, que

solo consiguen un 12,5% del total, gracias a las webs oficiales de los medios de transporte de Valencia.

### 2.4.4 Coste

Por último, se obtiene la gráfica 2.19, sobre el coste de obtención de cada aplicación:

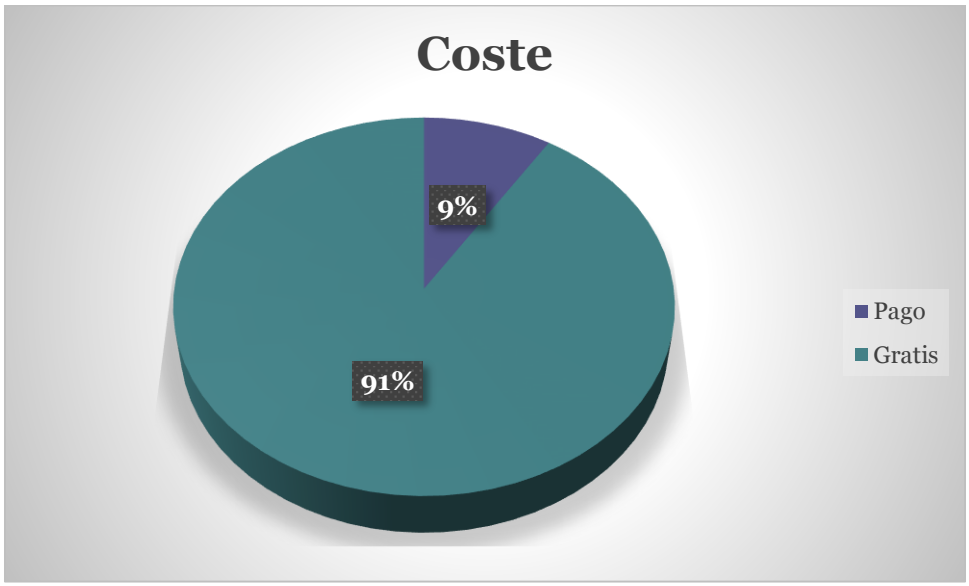


Figura 2.19: Coste

Queda claramente demostrado que la inmensa mayoría de aplicaciones están disponibles de forma gratuita para su uso (más del 90%). Esto es debido a que, generalmente, los datos de la ciudad están disponibles gratuitamente para terceros y que el público general probablemente no estaría dispuesto a pagar por un servicio de este ámbito.

## 2.5 Conclusión

Basándome en las distintas comparativas realizadas con los servicios analizados se obtienen las siguientes conclusiones:

- En el mercado la práctica habitual es recurrir al servicio de mapas de Google debido a la familiaridad que le transmite al usuario y que su API dispone de opciones muy variadas. No obstante, la naturaleza de este proyecto es crear el servicio en base a herramientas de uso completamente libre, lo que hace el uso de OpenStreetMap un reto a la hora de transmitir facilidad de uso de la aplicación al usuario.
- Las aplicaciones de información de la ciudad basan su oferta en transportes y turismo casi en su totalidad. Deja en evidencia que el usuario reclama este tipo de información, pero no por ello es incompatible con información de otro tipo como pueden ser datos demográficos de la ciudad o información de otro tipo más útil para el ciudadano de Valencia y no para el turista.
- Las plataformas por excelencia hoy en día para estos servicios son los sistemas operativos móviles de Google y Apple, dejando de lado la consulta vía web o de escritorio. Usar la

web no supone dejar de lado estos dispositivos, pues las webs actuales pueden ser perfectamente adaptativas.

- La aplicación a desarrollar debe ser de uso totalmente gratuito, porque el mercado así lo requiere y porque el proyecto está cimentado en software de uso gratuito.

En resumen la aplicación a desarrollar estará orientada al turismo, pero también a toda la información que pueda ser útil para el ciudadano de Valencia. Está aplicación se basará en la plataforma web, pero deberá adaptarse a cualquier dispositivo. Y por último será de uso libre y gratuito.

# 3. Herramientas

## 3.1 Servicios

A la hora de desarrollar una aplicación que haga uso de mapas una de las primeras cosas que hay que analizar es el proveedor de mapas a utilizar. Para ello hay dos factores esenciales a tener en cuenta, por un lado la calidad y detalle de los mapas y por otro el coste para poder trabajar sobre ellos.

En cuestión de calidad seguramente la API más completa hoy en día sea la de 'Google Maps', el problema está en su política. Cierto es que la API es gratuita siempre que tus desarrollos con dicha API no superen las 25.000 cargas de mapa por día y durante 90 días consecutivos. Si se cumplen estos requisitos sería cuando hay que realizar un pago para el uso de la API. Lógicamente esto no supondría un problema para el desarrollo del proyecto, pero la idea inicial es desarrollar todo el trabajo con herramientas 'free', en el sentido de libre uso y sin ningún tipo de restricción.

Ante esto han surgido alternativas de uso gratuito para desarrolladores, entre las que destaca OpenStreetMap.

### 3.1.1 OpenStreetMap

OpenStreetMap (OSM) [1] se trata de un proyecto para crear mapas libres y editables por la comunidad de usuarios. Este fue fundado el 1 de Julio de 2004 por el británico Steve Coast. La idea surgió debido a los altos precios que cobraba la agencia cartográfica Británica por aquel entonces. En el año 2006 pasó a convertirse en una fundación con registro en Inglaterra y Gales. El icono actual del servicio es el mostrado en la figura 3.1.

Figura 3.1: OpenStreetMap

Los mapas son creados utilizando la información geográfica proporcionada por dispositivos GPS móviles, fotografías y diversas fuentes de acceso libre. Toda esta información es distribuida bajo la "Licencia Abierta de Bases de Datos". OSM hace uso de una estructura de datos topológica. Los datos son almacenados bajo el "European Petroleum Survey Group" (EPSG:4326). Los elementos más básicos de OSM son los siguientes:

- **Nodos:** Posición geográfica de dos coordenadas.
- **Vías:** Conjunción de nodos que formas un polígono.
- **Relaciones:** Grupos de nodos o vías que tienen algo en común.
- **Etiquetas:** Se pueden asignar a los tres anteriores y constan de un par clave.

En base a los mapas proporcionados por OSM se han creado aplicaciones de todo tipo, así como librerías en varios lenguajes de programación para poder hacer uso de estos mapas.

### 3.1.2 Portal de datos abiertos de Valencia

Otro servicio del que se debe nutrir la aplicación es el proveedor de información. Durante estos últimos años muchas ciudades se han sumado a la filosofía del 'Open Data'.

Figura 3.2: Portal datos abiertos Valencia

#### 3.1.2a ¿Qué es Open Data?

'Open Data' es una filosofía que busca la distribución de datos de forma libre a cualquier persona que los necesite. Los datos distribuidos se publican sin Copyright ni patentes u otro tipo de restricción. Estos datos se pueden obtener sin procesar, estructurados y en formatos conocidos que faciliten la reutilización.

**Open Data en las ciudades**

El sector público de una ciudad controla gran cantidad de datos que son útiles para ciudadanos y empresas. La apertura de estos datos permite que se puedan crear nuevas ideas a partir de ellos y así, la propia ciudad, disponer de un mejor acceso a sus servicios e incluso poder crear nuevos. Con la distribución de los datos se consiguen cuatro objetivos:

- **Transparencia:** La fiabilidad del origen de los datos hacen de estos un excelente medio para comunicar la gestión pública realizada, la rendición de cuentas y control externo a la gestión todo ello destinado a ser transparente en la gestión del servicio público y generar confianza en la población.
- **Reutilización de la información pública:** Una organización pública genera grandes cantidades de información útil para muchos ciudadanos. El libre acceso de esta busca su reutilización por parte de la ciudadanía.
- **Generación e incentivación de economías:** La generación de economías a partir de los datos abiertos tiene múltiples facetas como son la eficiencia en la gestión ahorros debidos a la reutilización de la información eliminando gastos duplicados, eficiencias asociadas a una mayor visibilidad de los procesos y posibilidad de establecer acciones de mejora sobre ellos, creación de servicios nuevos, permitir establecer mecanismos de colaboración internos o externos, permitir mecanismos de generación de servicios a bajo coste, establecer modelos de comercialización de los datos, …
- **Fuente de innovación:** La distribución da pie a crear nuevas ideas, permitiendo reforzar líneas de negocios, crear nuevos servicios…

La población que accede a los datos abiertos se puede diferenciar en tres roles:

- **Desarrolladores:** Se les facilita la creación de nuevos servicios, favorece la innovación y les ayuda a disminuir la inversión necesaria para el desarrollo de una aplicación.
- **Ciudadanía:** El libre acceso a los datos genera un uso de la gestión pública transparente, consistente y fiable.
- **Administración:** La propia administración gana haciendo uso de los datos. De esta forma elimina problemas como la redundancia de datos y mejora la eficiencia de la gestión.

**Open Data del ayuntamiento de Valencia**

Dentro de la página del ayuntamiento de la ciudad de Valencia [2] existe una sección en la que se pone a disposición de forma gratuita ciertos datos de la ciudad. La idea es que cualquier persona u organización pueda construir sobre los datos una nueva idea para crear nuevos servicios. La web distribuye los datos en 6 secciones:

- Medio ambiente
- Sociedad y bienestar
- Transporte
- Urbanismo e infraestructuras
- Salud
- Turismo

Cada sección dispone de muchos servicios, por ejemplo, dentro de transporte están las paradas del bus o las estaciones de Valenbisi con datos asociados a ellos. Cada uno dispone de distintos formatos con los que recoger los datos, en total el portal contiene los siguientes: SHP, GML, WFS, WMS, KML, CSV, JSON, JSON-LD, XML, TURTLE, N3. Aunque no todas las opciones disponen de todos los formatos.

Además, el portal ofrece una API para poder mostrar datos según tu posición geográfica dentro de la ciudad aunque requiere registro previo gratuito. La API dispone de dos opciones:

- **Cerca de Mi:** Obtiene los tres elementos más cercanos a la posición del usuario entre una serie de categorías.
- **Puntos de interés:** Obtiene los distintos puntos de interés en base a una posición y con un radio de acción determinado y agrupado por varias temáticas.

En el lado negativo los datos sufren problemas de actualización. Es posible ver tipos de datos que varían cada minuto en los que su última actualización fue hace tres días.

### 3.1.3 MapIcons

MapIcons [3] es un proyecto destinado a proveer más de 1000 iconos para un uso libre. Está optimizado para Google Maps, pero se pueden descargar los iconos desde su portal web para cualquier uso. El proyecto nació en el año 2009 de la mano de Nicolas Mollet. Inicialmente el proyecto se hospedaba en 'Google Code' bajo el nombre de "Google Map Icons". Actualmente el servicio se identifica con el icono de la figura 3.3.

Figura 3.3: MapIcons

Los iconos están divididos en 13 categorías: Comercios, Naturaleza, Turismo, Gente, Deportes, Salud y educación, Eventos, Cultura y entretenimiento, Restaurantes y hoteles, Negocios, Industria, Transportes, Media.

Cada categoría dispone de multitud de iconos y dentro de cada icono se puede personalizar el color y seleccionar entre uno de los siete diseños disponibles.

## 3.2 Librerías

Una vez se ha escogido la plataforma a usar para visualizar los mapas y se dispone de los datos el siguiente paso es decidir cómo se van a superponer unos sobre otros. Dentro de las librerías JavaScript que trabajan sobre OSM destacan 'Leaflet' y 'OpenLayers'. La primera se trata de una librería JavaScript de código libre con un diseño atractivo, uso sencillo, muy ligero y con un gran rendimiento. En el lado negativo requiere de plugins adicionales para muchas funciones, no es tan maduro como el proyecto OpenLayers (y por lo tanto menos testado) y está orientado a dispositivos móviles (este proyecto también está orientado a entornos de escritorio). Con todo ello se ha seleccionado la segunda opción, descrita a continuación.

### 3.2.1 OpenLayers

OpenLayers [4] es un proyecto del Open Source Geospatial Foundation (OSGeo). Se trata de una biblioteca realizada en JavaScript de código abierto para mostrar mapas interactivos vía web. OpenLayers ofrece una API con la cual acceder a distintos proveedores de datos cartográficos entre los que se encuentran Google Maps, Bing, Yahoo u OpenStreetMap. Actualmente se encuentra en la versión 3.7.0. Desde la versión 3 de la librería la imagen de esta es la de la figura 3.4.


Figura 3.4: OpenLayers

OpenLayers basa su estructura en dos objetos: mapas y capas. El mapa es el lienzo sobre el que se aplican los distintos tipos de datos (las capas). Existen varios tipos de capas: las que muestran la información cartográfica, capas vectoriales que nos ayudan a aplicar formas geométricas al mapa, capas para añadir marcadores...

Al declarar la capa del mapa cartográfico se deben establecer ciertos parámetros como la fuente de la que se recogen los datos (en el caso que nos ocupa será OpenStreetMap), las coordenadas del punto central del mapa, el zoom aplicado, controles gráficos en pantalla... Y sobre esta capa se añaden el resto de capas con su información correspondiente.

### 3.2.2 Slide & Push Menus

Es un componente para crear menús fijos que se deslizan para estar visibles u ocultarse. Las opciones permiten que se puedan deslizar desde cualquier ángulo de la página y en el caso de los ángulos izquierdo y derecho permite mover el cuerpo de la web. Está realizado con HTML, CSS y JavaScript.

## 3.3 Lenguajes

Dado que este proyecto se basa en ofrecer un servicio vía web existen lenguajes del lado del cliente que son estrictamente necesarios para el desarrollo (HTML, CSS y JavaScript).

En el lado del servidor se trabajará con PHP por ser uno de los lenguajes del lado del servidor más potentes, flexibles y con un alto rendimiento. Además de ser un lenguaje en el que me desenvuelvo con soltura. PHP será complementado con AJAX para cumplir los objetivos de la página web.

### 3.3.1 HTML

HTML (HyperText Markup Language o lenguaje de marcas de hipertexto) es un lenguaje de marcas estandarizado para la elaboración de páginas web. Define una estructura básica y un código para la definición de contenido de una página web (texto, imágenes, video…) Es un estándar a cargo de la W3C (World Wide Web Consortium), organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web.

Los elementos están escritos en forma de etiquetas, rodeadas por corchetes (<, >). Los elementos tienen dos propiedades básicas: atributos y contenido. Cada atributo y contenido tiene ciertas restricciones para que se considere válido el documento HTML. Un elemento tiene una etiqueta de inicio (<ejemplo>) y una etiqueta de cierre (</ejemplo>), aunque existen algunas excepciones.

### 3.3.2 CSS

CSS (Cascading Style Sheets u hoja de estilo en cascada) es un lenguaje usado para definir el estilo de documentos escritos en HTML o XML. El W3C es el encargado de establecer la especificación de las hojas de estilo que servirán de estándar.

El objetivo que se persigue con el desarrollo del documento CSS es el de separar la implementación de un documento de su diseño. Con esta idea se obtienen ventajas como poder cambiar el diseño de una web en cualquier momento simplemente cambiando la ruta del diseño CSS en la cabecera del documento HTML.

La información de los estilos puede estar definida dentro del mismo fichero que el HTML o en un fichero aparte para obtener una mejor estructura del código. Para llamar a un documento CSS dentro del HTML bastará con añadir la etiqueta <style> en la cabecera con unos cuantos parámetros y la ruta al fichero.

### 3.3.3 JavaScript

JavaScript [5] es un lenguaje de programación interpretado. Se utiliza principalmente en el lado del cliente (el código se ejecuta en el navegador del cliente). El lenguaje permite mejoras en la interfaz del usuario y lo más importante, convertir a una página web en dinámica, esto es que dependiendo de las acciones del usuario, la página web mostrará una información u otra. Aunque también hay que destacar que JavaScript se usa en aplicaciones externas a la web (documentos PDF o aplicaciones de escritorio).

JavaScript está diseñado con una sintaxis similar a C y adopta nombre del lenguaje Java. Aunque dista de la semántica y propósitos de estos.

Todos los navegadores actuales saben interpretar JavaScript. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

### 3.3.4 PHP

PHP [6] es un acrónimo recursivo que significa PHP HyperText Pre-processor. Es un lenguaje de programación del lado del servidor diseñado para el desarrollo web de contenido dinámico. Puede ser usado en la mayoría de servidores web y en cualquier plataforma.

PHP se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos. No en vano, aplicaciones como Facebook, con un tráfico enorme, hacen uso de esta tecnología.

El lenguaje fue creado en 1995 por Rasmus Lerdorf y hoy en día sigue en desarrollo. Forma parte del software libre bajo la licencia PHP.

Al ser un lenguaje con similitudes a los lenguajes típicos de programación estructurada (C, Perl) permite que los programadores se desenvuelvan con fluidez en un corto espacio de tiempo.

Su funcionamiento en una página web es el siguiente:

- El cliente realiza una petición al servidor.
- El servidor ejecuta el intérprete de PHP.
- El servidor procesa el Script solicitado que procesará el contenido dinámicamente.
- El resultado se envía por el intérprete al servidor.
- Y el servidor lo envía al cliente.

PHP permite la conexión a distintos tipos de bases de datos, tales como: MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, FireBird, SQLite o MongoDB.

Para diferenciar en una página con extensión PHP cuando comienza el código con este lenguaje se usa la sintaxis “<?php” y se finaliza con “?>”.

### 3.3.5 AJAX

AJAX (Asynchronous JavaScript And XML o JavaScript asíncrono y XML) [7] es una técnica de desarrollo para crear aplicaciones interactivas en la web. AJAX funciona en múltiples plataformas y es utilizable en la mayoría de sistemas operativos y navegadores dado que está basado en estándares abiertos como JavaScript o DOM.

Cuando se lanza el código AJAX se realiza una comunicación asíncrona con el servidor en segundo plano. Con ello se logra eliminar las recargas de páginas para mostrar contenido nuevo y esto repercute en una mayor interactividad, velocidad y usabilidad de las aplicaciones.

## 3.4 Formatos de datos / Arquitecturas

Como se ha descrito en el apartado 3.1.2 el portal de datos abiertos de la ciudad de Valencia pone a disposición muchos formatos de datos. En el proyecto se va a tratar las extensiones CSV y GeoJSON. Que son y los motivos de su elección vienen descritos a continuación:

### 3.4.1 CSV

CSV (Comma Separated Values o valores separados por coma) es un tipo de documento en formato abierto destinado a representar los datos en forma de tabla donde la coma indica la separación entre una columna y otra y la distinción de filas se hace mediante el salto de línea. Cuando se requiera que una coma la recoja como parte del texto esta deberá ir contenida dentro de comillas dobles.

Este formato ha sido uno de los escogidos por las siguientes razones:

- Es muy sencillo.
- No usa un juego de caracteres concreto.
- Es un formato usado durante la carrera.
- Fácil de diseccionar mediante código.

### 3.4.2 GeoJSON

GeoJSON es una derivación de JSON (JavaScript Object Notation). JSON es un formato ligero para el intercambio de datos. Su simplicidad ha dado lugar a que sea una alternativa a XML. GeoJSON usa el formato JSON para representar colecciones de elementos (“Features”) geográficos simples. Los tres elementos básicos que guarda GeoJSON son los puntos, las líneas y los polígonos. Al representar un “Feature” se tiene que indicar su tipo, el total de coordenadas que lo representan en el orden adecuado y sus propiedades si las tuviera. La figura 3.5 muestra un ejemplo de la representación de un “Feature” en concreto.


Figura 3.5: GeoJSON

Las razones por las que se ha escogido GeoJSON como formato para el proyecto son las siguientes:

- Formato muy ligero. De esta forma se evitan largos tiempos de carga cuando el usuario solicite una opción que trabaje con este formato.
- Simple de tratar con múltiples lenguajes. Para leer un fichero JSON basta con usar la función eval() en JavaScript o JSONdecode() en PHP.

- Más intuitivo que otros formatos a la hora de representar figuras lineales o poligonales en un mapa.

### 3.4.3 REST

REST [8] (REpresentational State Transfer o Transferencia de estado representacional) es una arquitectura basada en el intercambio de mensajes que basa su funcionamiento en el protocolo HTTP (HyperText Transfer Protocol). Fue creado por Roy Fielding en el año 2000.

REST permite desarrollar servicios y aplicaciones que pueden ser usadas en cualquier dispositivo compatible con el protocolo HTTP. Esto lo convierte en una opción muy simple y accesible en detrimento a otras opciones de protocolos de intercambio de mensajes como SOAP que destacan por tener una mayor complejidad. Los sistemas basados en REST suelen adquirir la denominación RESTful.

Otra característica fundamental de REST es que no se requiere que ni cliente ni servidor guarden el estado de las comunicaciones entre mensajes. Con ello el desarrollador puede escalar sin preocuparse del almacenamiento de variables de sesión.

El conjunto de operaciones permitidas por REST están bien definidas debido a que se tratas de las operaciones conocidas por el protocolo HTTP. Las más usadas son GET, POST, PUT y DELETE.

La sintaxis universal para identificar los recursos aporta una gran flexibilidad. Cada recurso es direccionable únicamente a través de su URI.

## 3.5 Software

### 3.5.1 Sublime Text

Para la realización del proyecto sirve cualquier editor de texto que sea capaz de reconocer todos los lenguajes y formatos de datos descritos en los apartados anteriores. Ya que a la hora de elegir los servicios a usar se ha escogido una opción de libre uso, para la elección del software no va a ser menos. Por ello el editor de texto escogido es Sublime Text.


Figura 3.6: Sublime Text

Sublime Text es un editor de textos escrito en C++ y que usa Phyton para los plugins. Este editor destaca por tener características interesantes como la capacidad de leer 43 lenguajes de programación, sistema de pestañas o visualización de la estructura del código en un minimapa. El programa se puede identificar con la imagen de la figura 3.6. Los principales motivos por la elección de este editor son los siguientes:

- **Sencillez:** El programa en si es un editor de textos simple, dispone de las opciones esenciales que un editor debe tener acompañado de un diseño agradable a la vista. Distingue la sintaxis y la colorea para facilitar la lectura de ella y autocompleta palabras que ya existan en el código.
- **Formatos compatibles:** Lee gran parte de los formatos de programación, entre los que se incluyen todos los citados anteriormente.

- **Uso gratuito:** La herramienta no es software libre, pero aun así su uso es gratuito. Solo en el caso de que lo desees puedes pagar una licencia que no sirve más que para ayudar a los desarrolladores a continuar con el proyecto.

### 3.5.2 WAMP Server

WAMP es un acrónimo que define las tecnologías con las que está hecho:

Figura 3.7: WAMP Server

- **Windows:** Sistema operativo en el que se basa.
- **Apache:** Servidor web.
- **MySQL:** El gestor de bases de datos.
- **PHP:** Lenguaje de programación.

WAMP está basando en Windows, pero también tiene sus sistemas análogos LAMP (basado en Linux) y MAMP (basado en Macintosh). WAMP es un servidor web, un gestor de bases de datos y un intérprete del lenguaje de programación PHP. Todo esto lo convierten en un entorno de desarrollo web ideal para este proyecto. WAMP server está disponible gratuitamente bajo la licencia GPML. Su actual imagen se muestra en la figura 3.7.

### 3.5.3 Google Chrome / Firefox / Internet Explorer / Safari / Opera

Para el trabajo diario se ha hecho uso de Google Chrome, pero dentro de cada nuevo módulo instalado se testea con los cinco navegadores más usados a día de hoy, mostrados en la figura 3.8. Concretamente en sus siguientes versiones:

Figura 3.8: Navegadores

- **Google Chrome:**
  - Versión Windows: 44.0.2403.125 m
  - Versión iOS: 44.2403.67
  - Versión Android: 44.0.2403.133
- **Mozilla Firefox:**
  - Versión Windows: 39.0.3
  - Versión Android: 40.0.3
- **Internet Explorer:** 11.0.9600.17914
- **Safari:**
  - Versión Windows: 5.34.57.2
  - Versión iOS: 8.4.1
- **Opera:**
  - Versión Windows: 31.0.1889.174
  - Versión iOS: 10.2.0
  - Versión Android: 30.0.1856.93524

# 4. Requisitos

En esta sección se realiza un análisis exhaustivo de los elementos del problema y las pautas que se van a seguir para alcanzar la solución sin entrar en los detalles de la implementación, que se desarrollarán en el siguiente apartado. Se plantea que requisitos debe cumplir la aplicación (en el lado del cliente y en el del servidor).

## 4.1 Requisitos en el cliente

El lado del cliente (o la capa de presentación) es la encargada de mostrar visualmente todas las opciones de las que dispone la aplicación, el diseño de esta y su funcionalidad. Debe contener unos requisitos para su óptimo funcionamiento.

Dentro del desarrollo en el lado del servidor se podrían diferenciar dos partes, el desarrollo de la interfaz gráfica y la programación en el lado del cliente, que será interpretada por el navegador en cuestión. A continuación se detallan los requisitos para las dos secciones:

### 4.1.1 Interfaz gráfica

- **Sencillez:** La interfaz gráfica debe caracterizarse por su facilidad de uso. Dentro de un mismo menú se listarán todas las opciones disponibles, estando cada una de ellas como máximo a dos clicks de distancia.
- **Robustez:** Evitar inconsistencias en el diseño de la web, véase 'divs' mal encuadrados o links sin el diseño oportuno.
- **Adaptabilidad:** El diseño de la web debe ser responsivo, es decir, se adaptará al tamaño de pantalla. Con este hecho se logra que la aplicación sea utilizable en dispositivos como tabletas y sobretodo smartphones donde la pantalla puede llegar a ser un hándicap.
- **Carisma:** Colores cálidos y amigables. Estos variarán cuando una opción ha sido seleccionada o con un evento como onMouseOver.
- **Visualización:** Los objetos con una posición de dos coordenadas se mostrarán mediante marcadores (e.g. Una parada de bus o una cámara de tráfico) y los elementos con múltiples posiciones geográficas se indicarán a través de formas geométricas (división de barrios o indicación de carriles bici).
-

### 4.1.2 Desarrollo

- La programación en el cliente debe encargarse de dar distintas funcionalidades a un mismo objeto según su estado.
- Mediante el uso de una librería externa se mostrarán los datos cartográficos en un marco que ocupará la totalidad del espacio disponible.
- Los objetos que se plasmen sobre el mapa ocuparán un espacio que se redimensionará según el zoom aplicado para no dificultar la visualización del callejero.

## 4.2 Requisitos en el servidor

En el apartado anterior se han visto las posibilidades de las que dispone el usuario para interactuar con la aplicación, ahora bien, estas acciones requieren de un trabajo por parte del servidor que provee la información.

En esta lista se detallan los requisitos que deberá tener el desarrollo centrado en el lado del servidor:

- **Actualización:** Mediante cada solicitud de datos el servidor deberá proveer los más recientes, es decir, en cada solicitud realizada por el usuario el servidor recogerá los datos dispuestos de su lugar de origen y los tratará para acto seguido ser mostrados en pantalla.
- **Optimización:** Las llamadas al servidor por parte del usuario no deberán suponer una dificultad de uso de la aplicación por su larga espera.
- **Variedad:** El servidor deberá ser capaz de tratar dos tipos de formatos distintos de datos, mediante una función que disponga el lenguaje de programación usado o a través de código paso a paso.
- **Adaptabilidad:** Para cada tipo de solicitud, se manejarán los datos adecuándose a las necesidades de dicho tipo, ya sea por la forma de plasmarlo en el mapa o por la información que disponga.
- **Comunicación:** La transmisión de datos entre cliente y servidor será asíncrona, para así facilitar la fluidez de la aplicación.
- **Seguridad:** Los datos enviados de cliente a servidor se transmitirán mediante método POST para así mantener la comunicación por un canal más seguro, evitando la manipulación de datos por parte del cliente que pueda dar lugar a errores inesperados.
- **Robustez:** El servidor debe encargarse de cubrir cualquier incidencia durante el desarrollo de las solicitudes evitando que un tipo de error se muestre en el terminal del usuario. También debe ser el encargado de solucionar posibles incompatibilidades entre dos tipos de datos si se solicita mostrarlos simultáneamente.
- **Escalabilidad:** El lenguaje de servidor debe ser completamente compatible con la gran mayoría de sistemas. Con ello se evitan dificultades a la hora de migrar.

# 5. Diseño de la aplicación

Llegado a este punto hay que especificar como va a ser la aplicación a desarrollar. Se detallará las acciones de las que dispondrá el usuario, los procesos que encadenan así como la interfaz de presentación.

## 5.1 Diagrama de casos de uso

Un diagrama de caso de uso muestra gráficamente los pasos o actividades que se requieren para iniciar un proceso. Dentro del diagrama existen tres elementos fundamentales:

- **Entidad o actor:** Es el elemento externo que interaccionará con la aplicación.
- **Caso de uso:** El proceso en sí que puede llegar a producirse.
- **Conector:** Indicador que relaciona un elemento con cualquier otro.

Dentro del ámbito de los diagramas se ha ido utilizando en los últimos años UML (Unified Modeling Language o Lenguaje unificado de modelado).

### 5.1.1 UML

Unified Modeling Language (UML) [9] es un "lenguaje de modelado" para especificar métodos o procesos que está estandarizado por la ISO/IEC 19501:2005. Es utilizado para describir un sistema, para detallar los artefactos en el sistema y para documentar y construir. UML ofrece un estándar para describir en un plano aspectos como los procesos de negocio, las funciones del sistema o esquemas de bases de datos, entre otros.

### 5.1.2 Casos de uso CityControlPanel

Utilizando el estándar UML se ha especificado la totalidad de acciones que puede realizar el usuario dentro de la aplicación ha desarrollar. El resultado se observa en la figura 5.1.

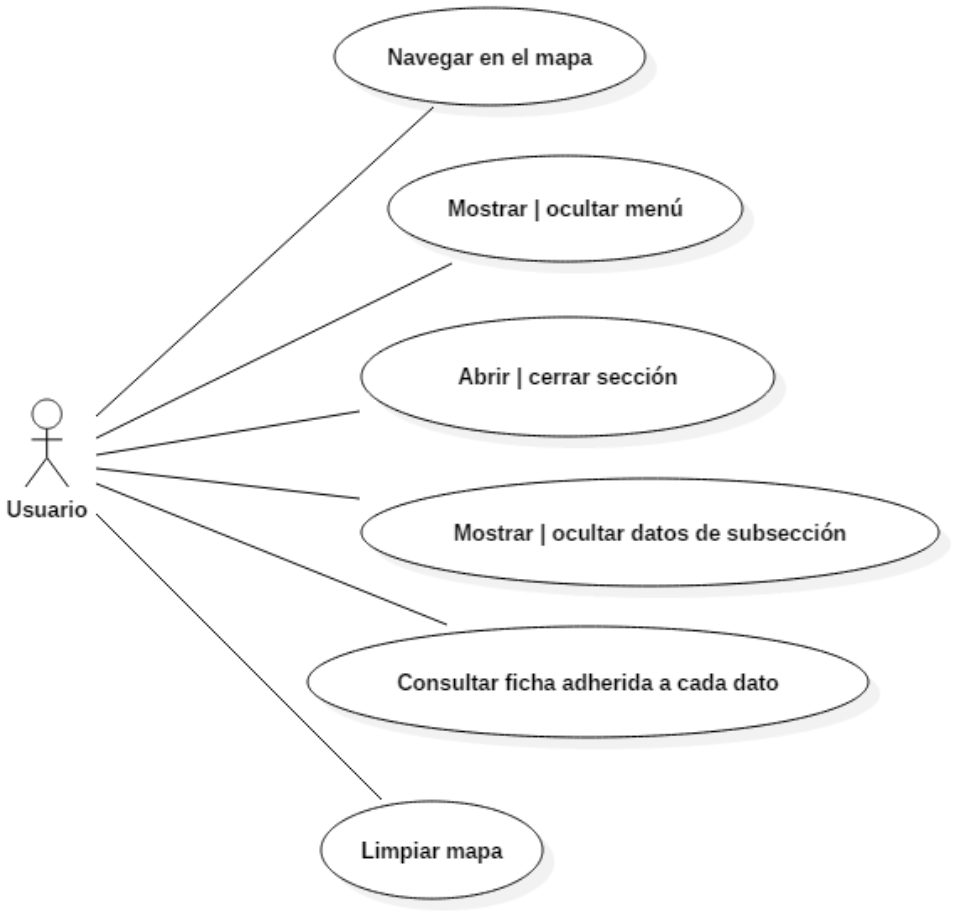


Figura 5.1: Casos de uso

## 5.2 Interfaz gráfica

Una de las ideas básicas sobre el diseño antes de comenzar el desarrollo del proyecto es la sencillez y la austeridad sin dejar de lado el atractivo. Se pretende conseguir una aplicación web en la que cualquier opción disponible este a dos clicks de distancia, sin largos tiempos de espera y que pueda ser usada por cualquier usuario independientemente de su edad o conocimientos informáticos.

### 5.2.1 Situación inicial

En este apartado se detalla la situación de la interfaz una vez se carga la aplicación. Cabe destacar que toda la interactividad de la aplicación está pensada para desarrollarse sobre la página principal (‘index’). La idea es que al cargar la página se muestre una vista general del callejero de Valencia que ocupe todo el espacio disponible de la web. El zoom por defecto será en el que se encuadre toda la ciudad (obviando pedanías). Por supuesto el usuario puede acercar

o alejar el mapa a su antojo (cuanto más cercano mayor detalle). El único añadido será un botón que al pulsarlo mostrará el menú de opciones. Algo similar al boceto de la figura 5.2.

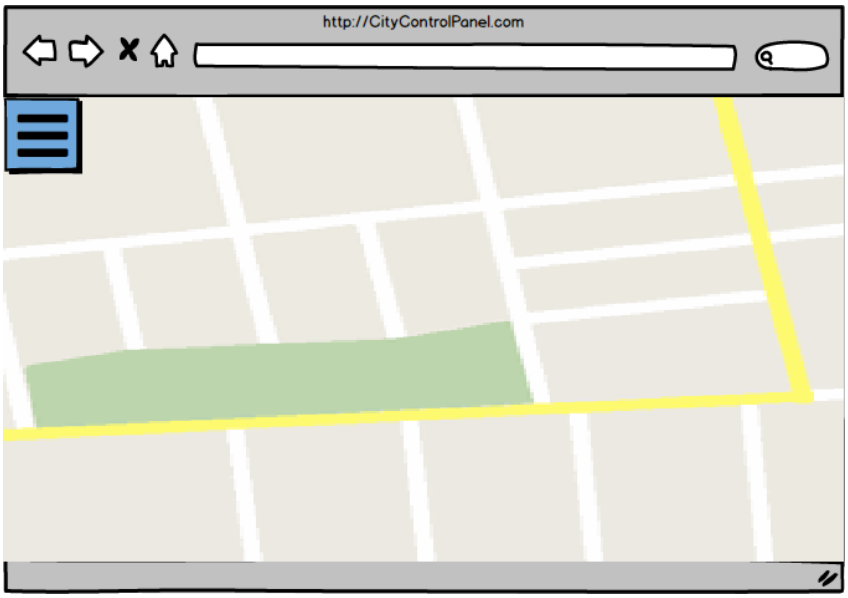


Figura 5.2: Boceto 1

Evidentemente este boceto y los siguientes mostrados son una idea del proyecto, la localización de cada objeto puede variar según las necesidades o restricciones que se vayan ocasionando.

### 5.2.2 Menú de opciones (secciones)

Al pulsar el botón incluido en la interfaz principal (situado en la sección superior izquierda) se desplegará un menú de opciones en formato vertical. Este menú estará estructurado en secciones y subsecciones, siendo las primeras un campo en común de las segundas (Turismo, transporte, urbanismo…). Inicialmente solo se mostrarán las secciones. También se añadirá una sección 'especial' basada en función de la localización del usuario. Finalmente se añadirá en algún lugar del menú el nombre de la aplicación 'CityControlPanel'. La idea se puede observar en la figura 5.3.

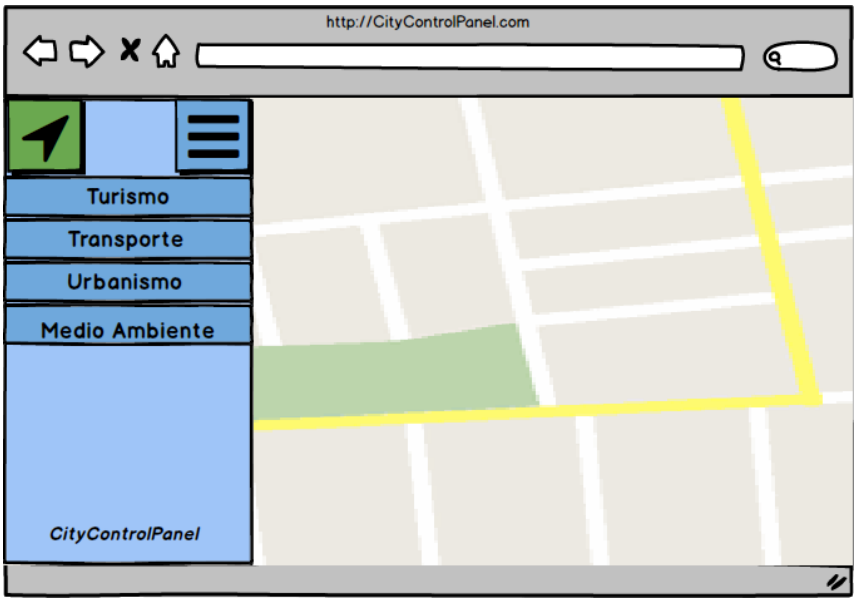


Figura 5.3: Boceto 2

### 5.2.3 Subsecciones

Al interactuar con una sección del menú se desplegará debajo de esta un submenú con cada una de las opciones correspondientes haciendo descender a las secciones situadas por debajo de esta. Por ejemplo, escogiendo la sección 'Transporte' se despliegan una serie de opciones de este ámbito como pudieran ser el tráfico actual, la estructura de carril bici de la ciudad, estado del servicio de alquiler de bicicletas públicas o las paradas de bus. Algo similar al boceto de la figura 5.4.

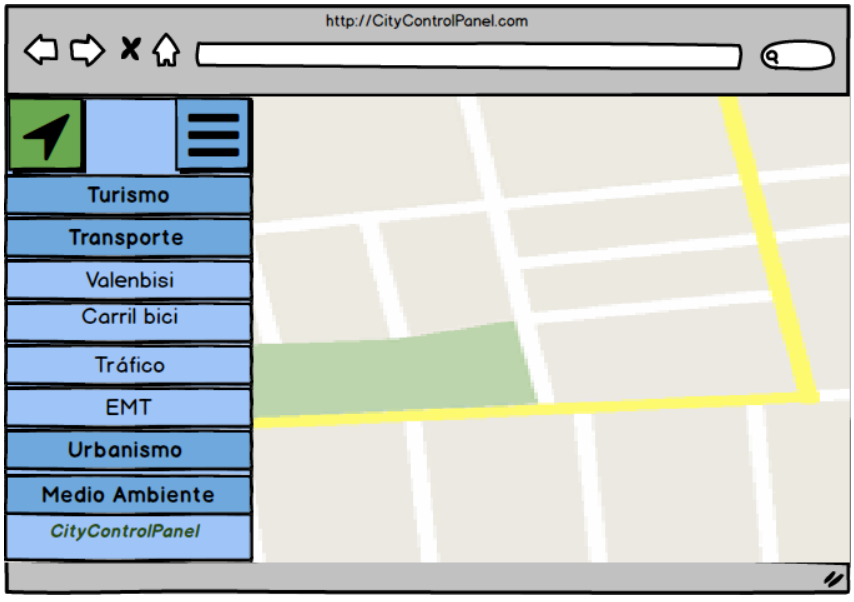


Figura 5.4: Boceto 3

Al interactuar con una de las opciones, por ejemplo 'Valenbisi', la aplicación pasará a mostrar en el mapa la ubicación de las estaciones de 'Valenbisi' en el mapa de Valencia. Esta acción requiere un proceso que puede alargarse en el tiempo y al no cargar una página nueva en el

navegador no se informa de ello. Debido a esto se debe informar al usuario de que el proceso se está llevando a cabo mediante alguna forma, como el círculo de carga mostrado de la figura 5.5.


Figura 5.5: Boceto 4

Una vez el proceso de carga haya finalizado se mostrarán los datos en el mapa en forma de marcadores o de cualquier otro tipo si así lo requiere, situados en sus correspondientes coordenadas. Al interactuar con un marcador se mostrarán datos de este en una pequeña ventana, algo similar a lo mostrado en la imagen 5.6.


Figura 5.6: Boceto 5

### 5.2.4 Geo posición

Por último, la sección especial destacada en color verde en el menú mostrará datos según la localización del usuario. Podrá ser información de cualquiera de las otras secciones disponibles. En el mapa además de los marcadores con las opciones encontradas, se hallará un icono en el lugar de la posición geográfica, para así indicar la distancia de esta al marcador dado. En la figura 5.7 se observa la localización del usuario y los tres elementos más próximos a su posición.

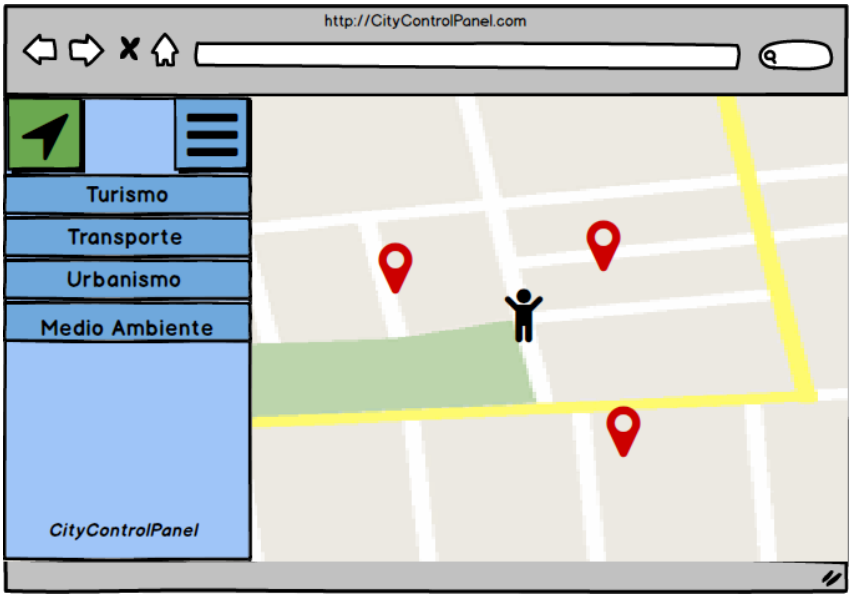


Figura 5.7: Boceto 6

### 5.2.5 Interfaz móvil

Aunque la aplicación web se diseñe originalmente sobre un entorno de escritorio el objetivo es que la interfaz se adapte a las características de la pantalla, técnica conocida como diseño responsivo. Con ello se consigue que la web se visualice correctamente también en tabletas y teléfonos inteligentes.

## 5.3 Diagramas de sistema

En esta sección se va a representar mediante diagramas como va a ser el sistema a desarrollar y su funcionamiento interno a grandes rasgos con los distintos eventos que puedan llegar a producirse.

### 5.3.1 Inicio de la aplicación

El diagrama de la figura 5.8 representa el conjunto de acciones que realiza la aplicación al cargarse en el cliente. Para comenzar, comprueba si tiene una conexión a internet establecida (en el caso de acceder en local podría abrir la web pero no acceder a funciones que requieren solicitudes externas). Si no está establecida bloquea la página y notifica del suceso. En caso afirmativo, averigua si el terminal cliente tiene la tecnología capaz de proporcionar la posición geográfica. En caso afirmativo el navegador preguntará al usuario si quiere facilitarla. En caso de no obtener la posición, se mostrará un mensaje indicando el problema, ya sea porque el usuario ha rechazado la solicitud, la tecnología no dispone esa opción, se ha intentado obtener pero el tiempo de espera ha superado el máximo o por cualquier otro tipo de error desconocido.

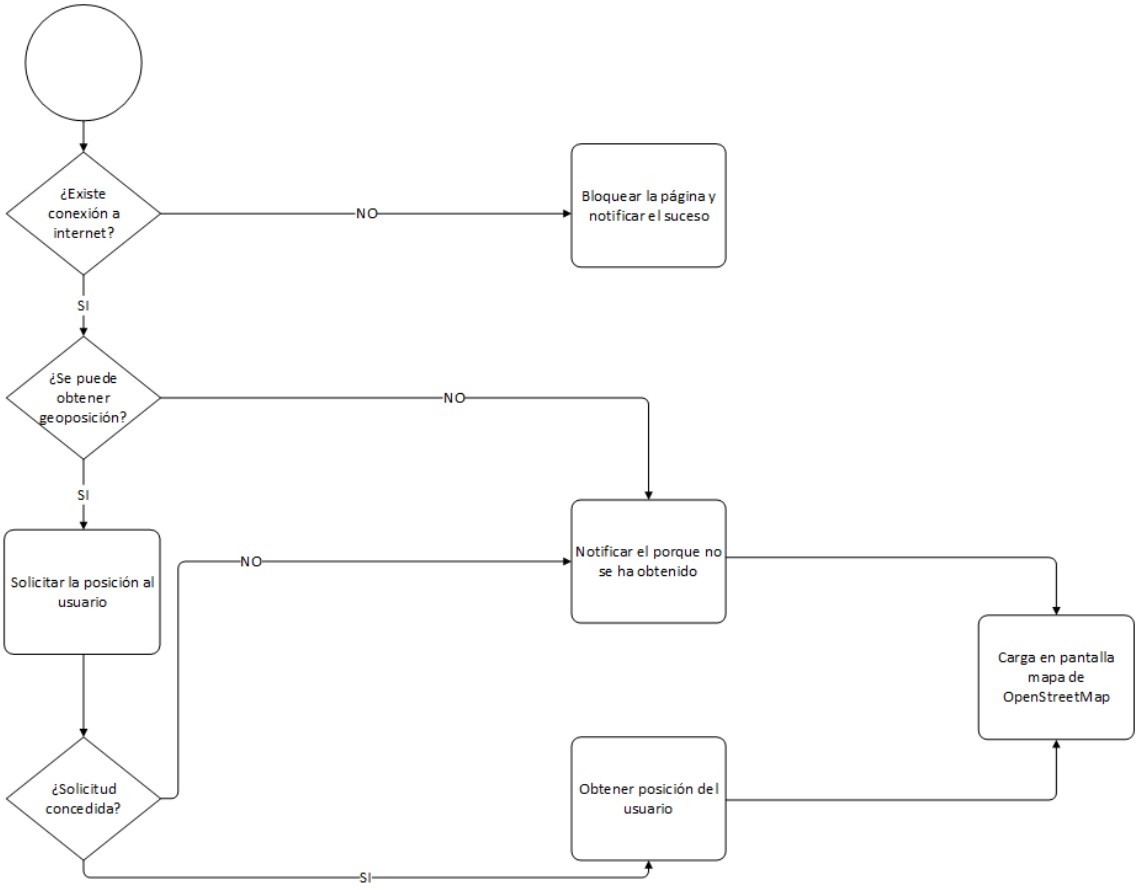


Figura 5.8: Diagrama de sistema 1

### 5.3.2 Solicitar un tipo de dato

Cada vez que el usuario solicite a la aplicación la muestra de un tipo de dato, esta a su vez, lo descarga del proveedor (el servicio de datos abiertos de Valencia). Este hecho no puede tener lugar si el cliente ha perdido la conexión a internet, por ello, antes de iniciar el proceso de descarga, se hace una comprobación de conexión para evitar que el usuario encuentre errores inesperados en pantalla.

Es posible que el enlace con el tiempo haya cambiado o que en ese instante no se encuentre disponible, por ello, también hay que contemplar ese error. Como ya se ha indicado el programa trabaja con distintos tipos de formato, según cual sea se ejecutará un parte del código u otra.

En el caso de GeoJSON existen dos variantes, representadas por líneas (e.g. carril bici) o por polígonos (e.g. barrios). Y en el caso de JSON obtenido por posición puede darse la posibilidad de que no haya ningún lugar cercano a la posición del usuario, hecho que también debe ser notificado. Todas estas decisiones se observan gráficamente en la figura 5.9.

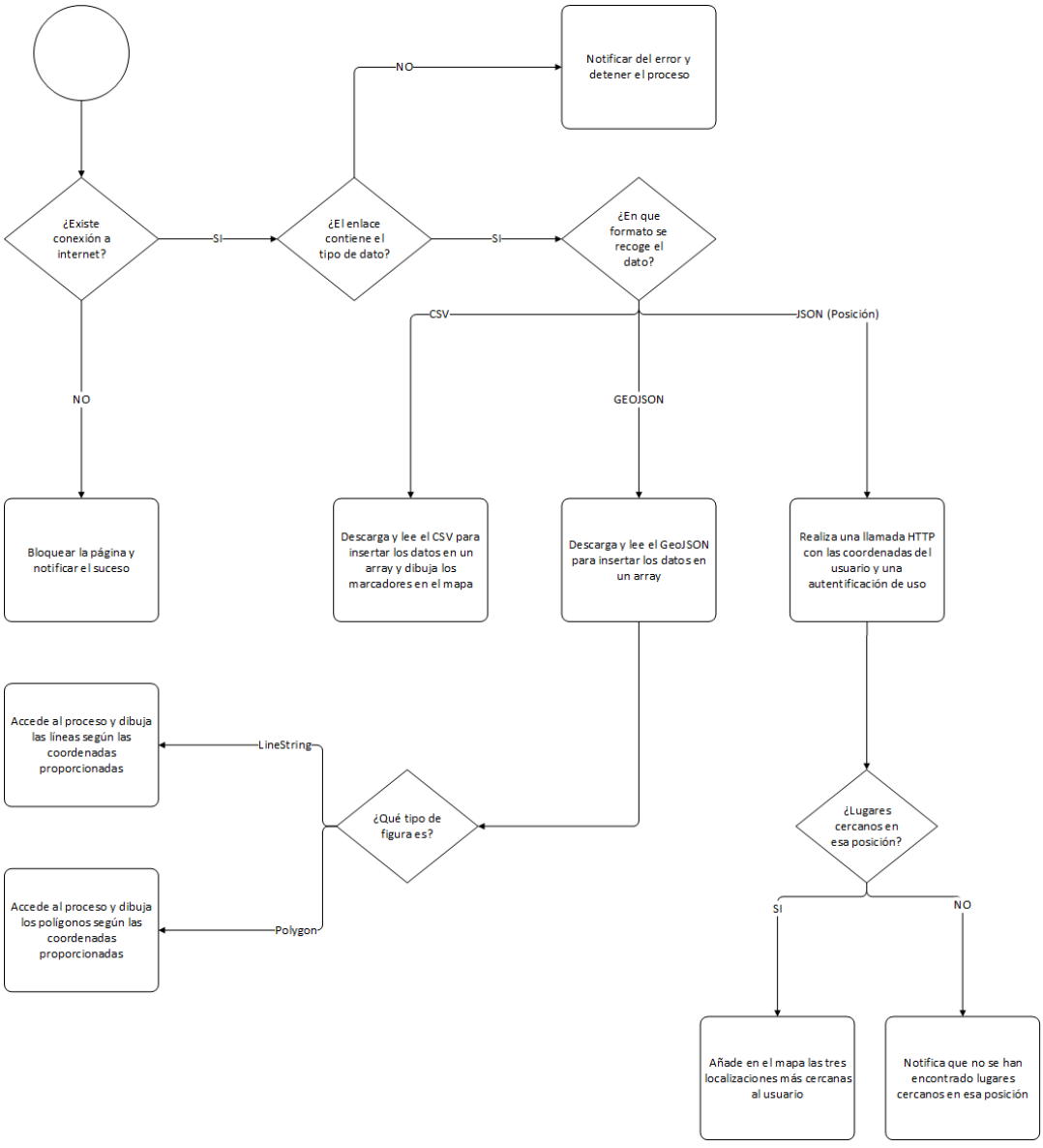


Figura 5.9: Diagrama de sistema 2

### 5.3.3 Menú y suprimir figuras

Por último, en la figura 5.10 se especifica dos tareas menores que puede realizar el usuario. Una es las posibilidades que ofrece el menú de opciones para maximizar o minimizar y la otra las distintas opciones para suprimir datos del mapa.

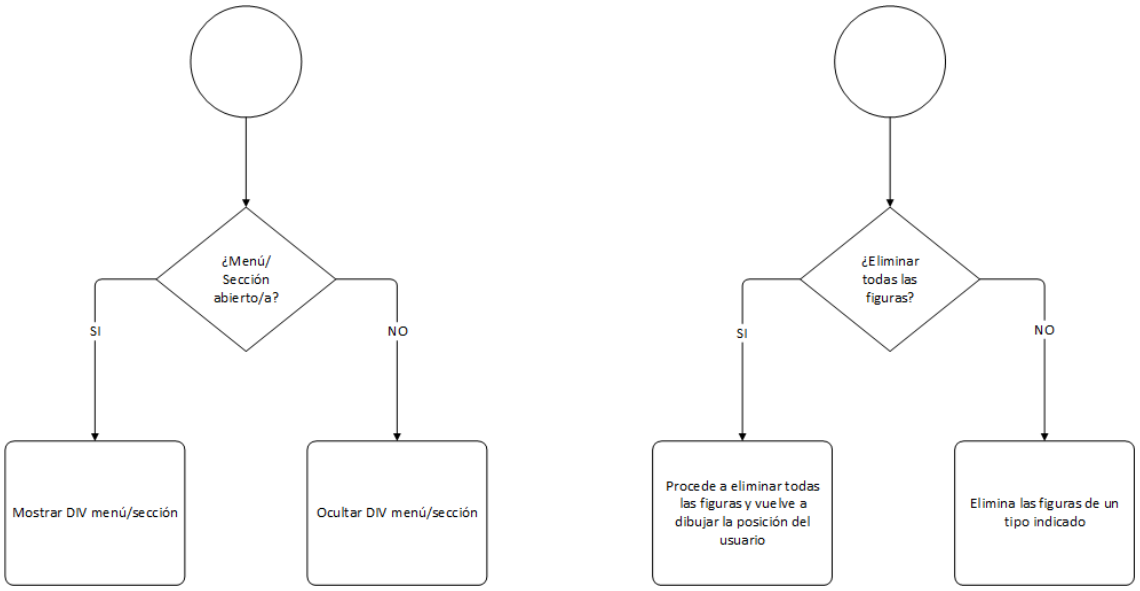


Figura 5.10: Diagrama de sistema 3 y 4

# 6. Metodología de desarrollo

## 6.1 Introducción

En el apartado anterior se ha podido ver qué acciones y comprobaciones va a realizar la aplicación, así como la interactividad del usuario con ella y su interfaz gráfica. A continuación se detalla como todas estas ideas de diseño se van a implementar con las tecnologías necesarias para su fin y en todos y cada uno de los puntos.

## 6.2 Estructura de los ficheros

En la ilustración 6.1 se muestra la estructura que tienen los distintos ficheros del proyecto. Conviene detallar que función ocupa cada uno porque serán mencionados a lo largo de los detalles de la implementación.

- **index.html:** En esta página se desarrolla toda la interactividad del usuario con la aplicación. El resto de ficheros orientados al cliente (JavaScript y CSS) están incluidos.
- **funciones.js:** Se encarga de realizar la interoperabilidad entre usuario y programa y de comunicarse con el servidor.
- **ol.js y ol.css:** Los archivos correspondientes a la librería y diseño de OpenLayers.
- **CSV y JSON:** Los archivos ubicados en estas carpetas son los encargados de recoger los datos del proveedor de servicios, tratarlos y enviarlos al cliente (funciones.js).

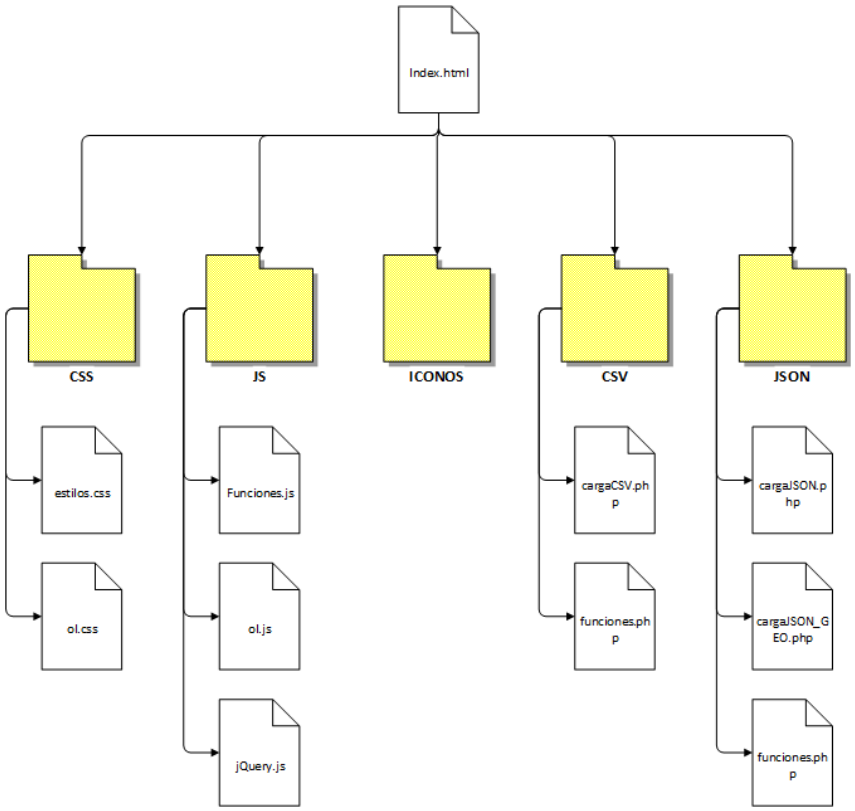


Figura 6.1: Estructura de los ficheros

## 6.3 Nivel de presentación

En este apartado se muestra el diseño definitivo de la aplicación en base a los bocetos que se realizaron anteriormente en función de las necesidades requeridas.

### 6.3.1 Situación inicial

El resultado final de la aplicación es bastante similar a lo ideado en general. Una vez la aplicación se carga, el div general que ocupa todo el ancho y alto contendrá la capa del mapa proporcionada por OSM, añadiendo a la esquina superior izquierda el botón de menú con el icono que se suele usar en la actualidad para este evento. En la figura 6.2 se observa una imagen del resultado.

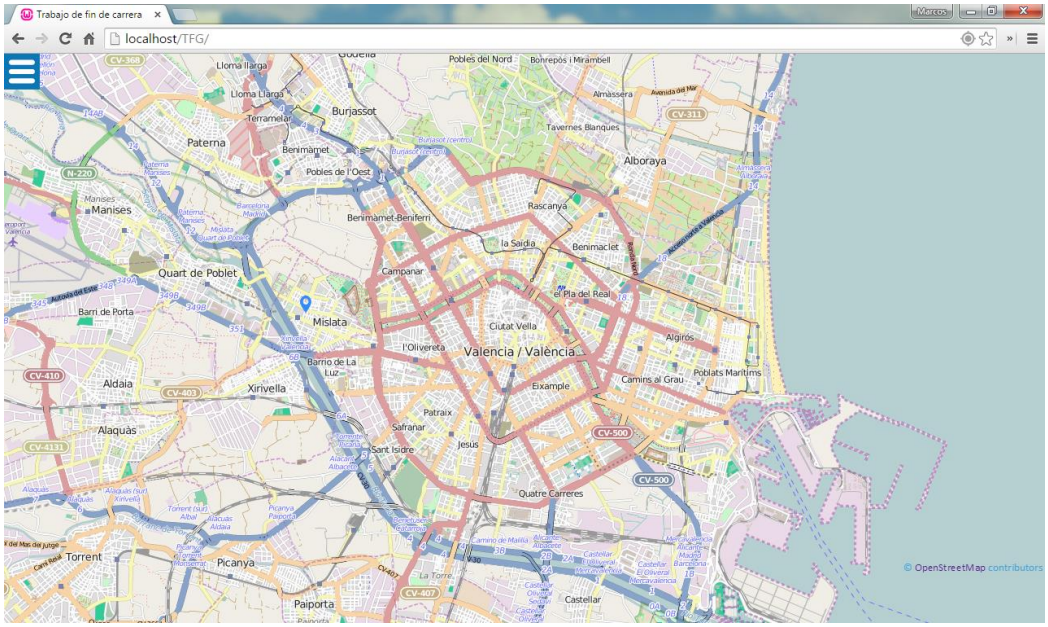


Figura 6.2: Presentación 1

Como detalle se observa un icono azul en la localidad de Mislata. Este icono nos indica nuestra ubicación actual en el caso de que haya podido obtenerla. En caso contrario, avisa de ello y no se podrá usar las funciones que se basan en la posición del usuario. Por ello, en la primera ocasión que se acceda a la aplicación, esta mostrará una solicitud de ubicación como la de la imagen 6.3 (varia la forma según el navegador).

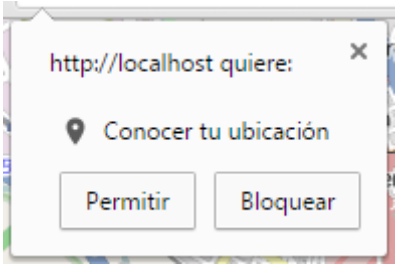


Figura 6.3: Presentación 2

### 6.3.2 Menú de opciones

Finalmente las fuentes de datos recogidas se dividen en: Turismo, Transporte, Urbanismo y Cerca de mí, que muestra información de todas las secciones en función a la posición del usuario. También se dispone, remarcado con color rojo de una opción para limpiar el mapa. En la figura 6.4 se observa el resultado.

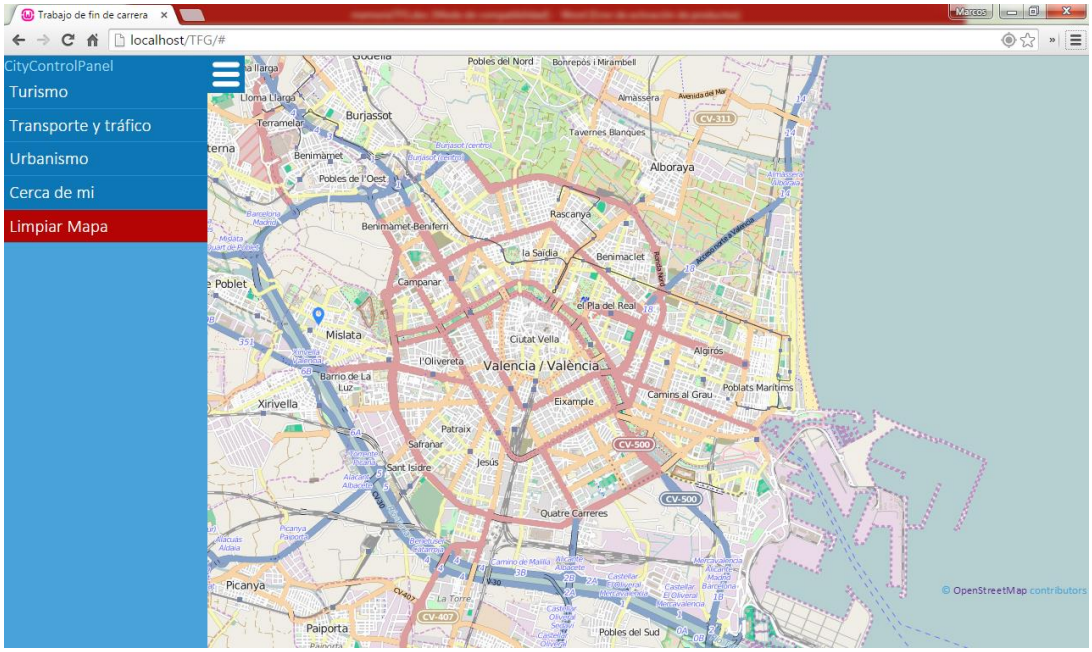


Figura 6.4: Presentación 3

### 6.3.3 Subsecciones

A continuación se lista el total de subsecciones o fuentes de datos que se pueden visualizar en la aplicación:

- **Turismo**
  - Monumentos turísticos
  - Monumentos falleros
  - Monumentos falleros infantiles
- **Transporte y tráfico**
  - Valenbisi
  - Carril bici
  - Tráfico
  - EMT
  - Cámaras de tráfico
  - Cortes tráfico fallas
- **Urbanismo**
  - Barrios municipales
  - Distritos municipales
  - Manzanas Catastrales
- **Cerca de mi**
  - Aparcamientos

- Valenbisi disponible
- Parada taxi
- Puntos WiFi
- Contenedores

Si se selecciona una opción, la aplicación informa mediante un 'boot loader' como el de la figura 6.5, mientras realiza la petición al servidor, en ese momento se bloquea la interfaz para evitar generar posibles errores.

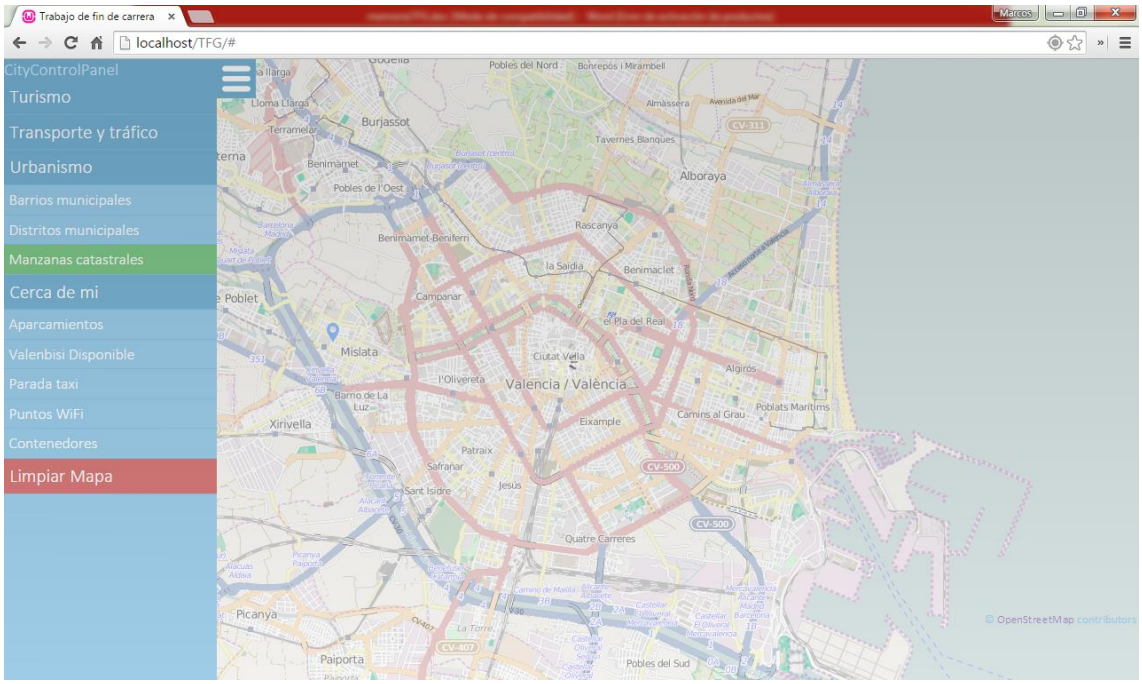


Figura 6.5: Presentación 4

Una vez descargado e impreso los datos en pantalla quedan mostrados como en las figuras 6.6, 6.7 y 6,8 , cada uno con un popup de los datos que acompañan a la posición:

<u>Marcadores</u>


Figura 6.6: Presentación 5

Lineas


Figura 6.7: Presentación 6

Polígonos

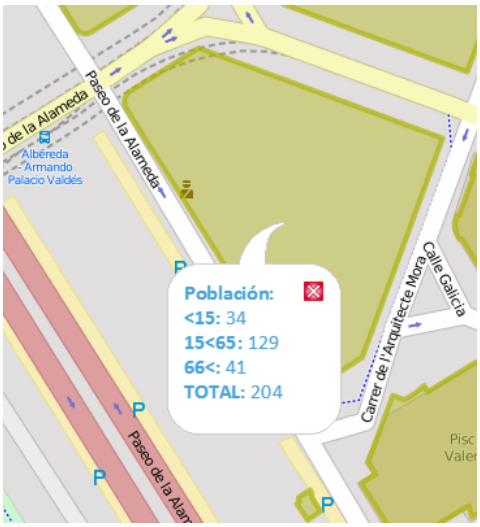


Figura 6.8: Presentación 7

### 6.3.4 Geo posición

La última sección correspondiente a datos en función de la posición geográfica se muestra de forma similar a los marcadores, con el añadido de que se obtienen los tres más cercanos y dentro de la información del popup se pueden ver los metros de distancia. Se observa de forma gráfica en la figura 6.9.


Figura 6.9: Presentación 8

### 6.3.5 Interfaz móvil

Por último, para mostrar como la aplicación se adapta al tamaño de pantalla, se muestran las imágenes 6.10, 6.11 y 6.12, realizadas en una tableta y un Smartphone.

Captura en iPad (Pantalla de 9,7 pulgadas y navegador Safari)

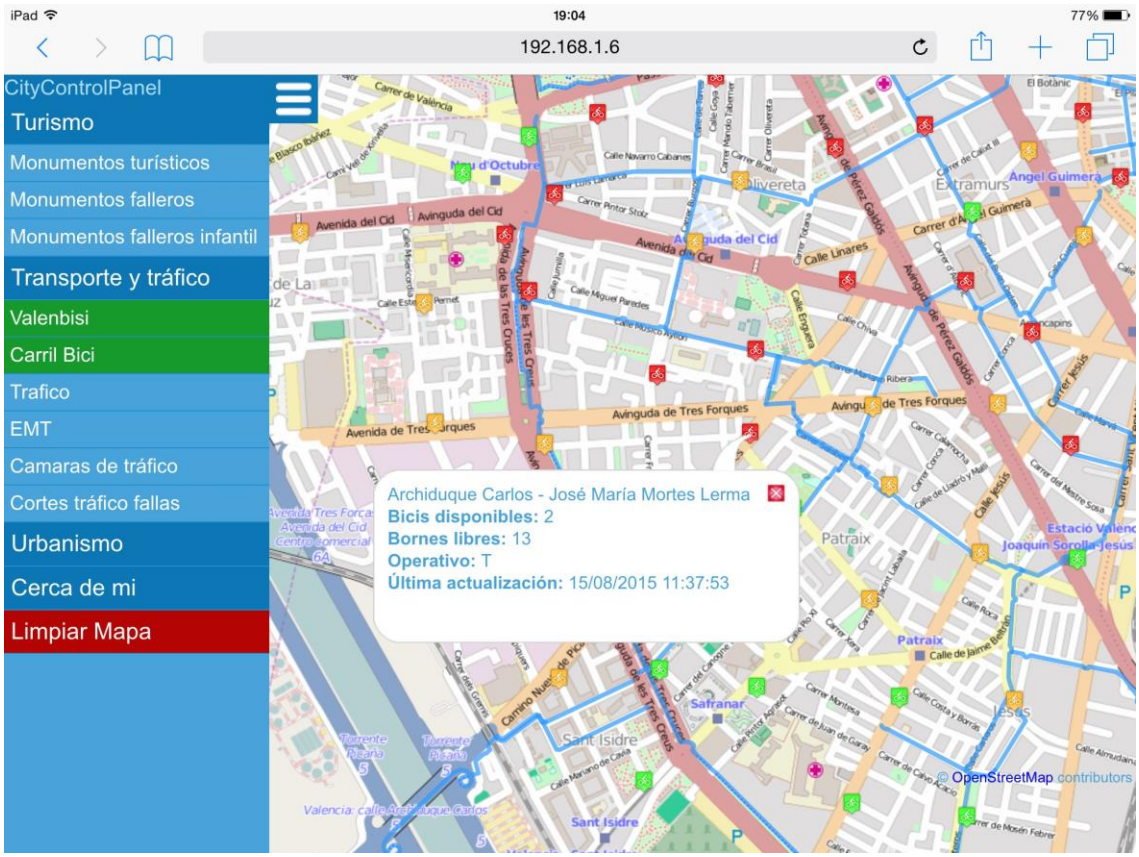


Figura 6.10: Presentación 9

Capturas en Motorola Moto G 2013 (pantalla de 4,5 pulgadas y navegador Chrome)


Figura 6.11: Presentación 10

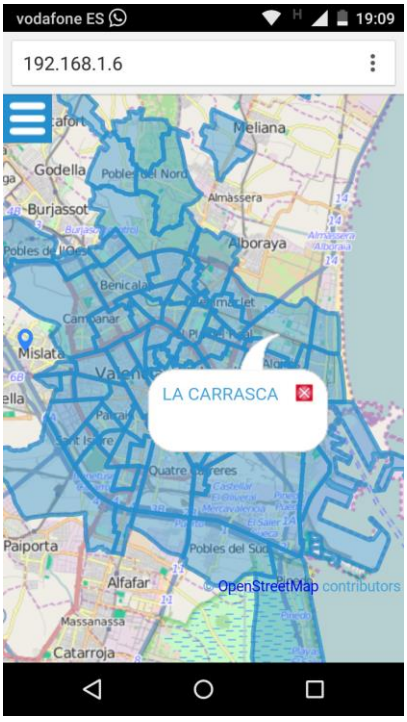


Figura 6.12: Presentación 11

## 6.4 Nivel de persistencia

Todos los datos recogidas provienen del portal web datos abiertos del ayuntamiento de Valencia. En los siguientes apartados se muestra la estructura de cada formato en el momento que lo se descarga del servidor para comenzar a ser tratado.

### 6.4.1 CSV

Con este formato se tratan los siguientes ámbitos de datos: Monumentos turísticos, monumentos falleros, monumentos falleros infantiles, valenbisi, EMT y cámaras de tráfico. La figura 6.13 corresponde al CSV de valenbisi.

```
1  X;Y;name;number;address;open;available;free;total;ticket;updated_at
```

Figura 6.13: CSV 1

En la anterior imagen se observa la cabecera del fichero. Esta no será tratada como el resto de datos, sino para poder llamar al resto. Cada parámetro indica lo siguiente:

- **X e Y:** Referencia de un punto geográfico.

- **name:** Nombre la estación de Valenbisi.

- **number:** Número de la estación.

- **Adress:** Dirección de la estación.

- **Open:** Indica si la estación se encuentra operativa.

- **Available:** Número de bicicletas disponibles para su recogida.

- **Free:** Bornes libres para depositar la bicicleta.

- **Total:** Número de bornes totales.

- **Ticket:** Indica la posibilidad del pago mediante tarjeta.

- **Updated_at:** última actualización del estado.

Un ejemplo de registro es el mostrado en la imagen 6.14:

```
3   -0.323401216004035;39.464437274407885;163_PASEO NEPTUNO;163;Paseo Neptuno 32-34;T;2;17;20;T;19/06/2015 23:54:55
```

Figura 6.14: CSV 2

Como se puede observar el comienzo de un registro nuevo se indica a través de los saltos de línea y la separación de un parámetro a otro mediante el punto y coma. Más adelante se explica cómo se ha separado y tratado cada registro y parámetro en el código.

### 6.4.2 GeoJSON

En este caso se hace uso del formato GeoJSON para representar figuras, concretamente en los siguientes casos: Carril bici, tráfico, cortes de tráfico en fallas, barrios municipales, distritos municipales y manzanas catastrales. Cada fichero JSON contiene una cabecera al comienzo como la mostrada en la figura 6.15:

```
{
"type": "FeatureCollection",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },

"features": [
```

Figura 6.15: GeoJSON 1

En ella se informa de que tipo de JSON se trata, en este caso “Feature Collection” y la estructura que van a tener los registros (nombre y propiedades del registro). Por último con el símbolo de corchete abierto se indica el comienzo de los ‘features’. En la figura 6.16 se detalla un registro del fichero de barrios municipales.

```
{ "type": "Feature", "properties":
    {
        "codbarrio": "5",
        "nombre": "RAFALELL-VISTABELLA",
        "coddistbar": "175",
        "coddistrit": "17"
    },
    "geometry":
    { "type": "Polygon",
      "coordinates": [ [ [ -0.317646067169912, 39.564894807846137 ],
                         [ -0.317309280940658, 39.564630283519506 ],
                         [ -0.317012010626102, 39.564718006521424 ],
                         [ -0.316639082260081, 39.565353514135161 ],
                         [ -0.316107600562493, 39.56523312946161 ],
                         [ -0.315888864995411, 39.566547842529083 ],
                         [ -0.31491907193738, 39.566381286710609 ],
                         [ -0.31439016610305, 39.566175363035171 ],
                         [ -0.313723974940282, 39.565907714663794 ],
                         [ -0.312978610534738, 39.565656247859884 ],
                         [ -0.311585950113824, 39.565232134293126 ],
                         [ -0.310974219987385, 39.565024286855881 ],
                         [ -0.309827338016867, 39.56477250190521 ],
                         [ -0.309179052253804, 39.564644875580811 ],
                         [ -0.308913521378219, 39.564908980557256 ],
                         [ -0.308196071702572, 39.564820284674056 ],
                         [ -0.307578633361471, 39.564778945983903 ],
                         [ -0.307129949931522, 39.56471899570635 ]
                     ] ]
    }
}
```


Figura 6.16: GeoJSON 2

Cada registro está dividido en dos secciones diferenciadas, por un lado “properties” y por el otro “geometry”. En el primero se obtienen los datos característicos del registro, como puede ser el nombre del barrio o el estado del tráfico. Y en el segundo se informa del tipo de figura que es (LineString o Polygon) y del total de las coordenadas que la forman.

### 6.4.3 JSON Geo posición

El último formato a recoger corresponde a la API cerca de mí, que devuelve los tres lugares más cercanos de un tipo en base a la posición del usuario, enviada dentro de la llamada a la API. Esta ofrece un fichero en formato JSON con la estructura mostrada en la figura 6.17:

```
[
    {
        "latDestino":39464486,
        "lonDestino":-368766,
        "distancia":1541,
        "titulo":"APARCAMIENTOS",
        "mensaje":"REGNE\nPlazas libres: 93"
    },
    {
        "latDestino":39468347,
        "lonDestino":-379767,
        "distancia":2529,
        "titulo":"APARCAMIENTOS",
        "mensaje":"BARON DE CARCER - S. AGUSTIN - S. VICENTE\nPlazas libres: 453"
    },
    {
        "latDestino":39474518,
        "lonDestino":-375397,
        "distancia":2769,
        "titulo":"APARCAMIENTOS","mensaje":"PLAÇA DE LA REINA\nPlazas libres: 177"
    }
]
```


Figura 6.17: JSON

Como se observa, la API envía tres registros en orden de cercanía al usuario. Sus parámetros contienen las coordenadas del lugar, la distancia a este en metros, a qué sección pertenece y el mensaje del lugar que se añadirán al popup de la figura.

## 6.5 Nivel de aplicación

En este apartado se muestran las secciones fundamentales del código con una breve descripción en cada una de ellas.

### 6.5.1 Instancia del mapa

La única función añadida al archivo principal del proyecto está diseñada para crear la capa del mapa. El div en el que se implanta ocupa el 100% del tamaño de pantalla. Para poder crear el mapa deseado se incluye dentro del index el archivo ‘ol.js’ (librería de OpenLayers). El código resultante se observa en la figura 6.18.

```
map = new OpenLayers.Map("main");

map.removeControl(map.getControlsByClass('OpenLayers.Control.Zoom')[0]);
map.addLayer(new OpenLayers.Layer.OSM());

var coordenadas = format_coor( -0.3749, 39.4721 );
var zoom=13;
map.setCenter (coordenadas, zoom);

var MarcadoresLayer = new OpenLayers.Layer.Vector("MarcadoresLayer");

map.addLayer(MarcadoresLayer);

var controls = {
  selector: new OpenLayers.Control.SelectFeature(MarcadoresLayer, { onSelect: createPopup, onUnselect: destroyPopup })
};

map.addControl(controls['selector']);
controls['selector'].activate();


geoposicion_ini();
```

Figura 6.18: Instancia del mapa

Para comenzar se crea una nueva instancia de la clase 'OpenLayers.Map' llamada 'map'. Sobre esta instancia se modificarán propiedades del mapa y se añadirán capas.

En la primera capa a añadir se indicará el proveedor de mapas, en este caso OpenStreetMap. La forma de declaralo es con 'new OpenLayers.Layer.OSM()'.

Las propiedades se editan en las siguientes tres líneas. Se indican las coordenadas y el zoom por defecto al inicio de la aplicación mediante el método 'setCenter()'.

A continuación, se añade la capa de vectores sobre la que se van a cargar todas las figuras que seleccione el usuario. Se instancia con 'OpenLayers.Layer.Vector()' y la se añade al mapa mediante 'map.addLayer()'.

Para finalizar se incluye en la capa de vectores la posibilidad de añadir 'Popups', declarada en la variable 'controls'.

### 6.5.2 Obtener posición

Al arranque de la aplicación se llama al proceso 'geoposicion_ini'. Este comprueba si el navegador usado soporta la geo localización. En la actualidad, casi la totalidad de navegadores pueden obtener la posición del usuario, pero las versiones antiguas pueden no tener soporte.

Si es posible, el método 'busca_posicion' (visible en la figura 6.19) la obtendrá a través de 'getCurrentPosition'. Los dos primeros parámetros de la función son los nombres de los métodos a llamar en caso de obtener la posición y el caso contrario. Por último, se establece un tiempo máximo en que el navegador buscará la posición. Transcurrido dicho tiempo 'timeout' se interpretará que la posición no puede ser obtenida.

```
function busca_posicion(){
  navigator.geolocation.getCurrentPosition(pos_correcto, pos_error,{
        enableHighAccuracy: true,
        maximumAge:         5000,
        timeout:            20000
      });
}
```

Figura 6.19: Obtener posición

En el método ‘pos_correcto’ se guarda la posición del usuario en dos variables globales y en caso negativo se ejecuta el método ‘pos_error’.

En ‘pos_error’ se contemplan tres tipos de errores:

- **Permission_Denied:** El usuario ha rechazado la solicitud de acceso a la localización.
- **Position_Unavailable:** Durante la obtención de la posición se produjo un error desconocido.
- **Timeout:** Se ha superado el tiempo máximo para la obtención de la posición.

En caso de que el motivo por el que no se haya podido obtener la posición no corresponda a ninguno de los tres anteriores se mostrará un error por defecto.

### 6.5.3 Función enlaces

El evento ‘onClick()’, situado en los elementos del menú, llama a la función enlaces como se muestra en la figura 6.20:

```
<a id="fallasInfantil" href="#" onClick="enlaces('fallasInfantil', 15, './csv/cargaCSV.php');"   >Monumentos falleros infantil</a>
```

Figura 6.20: Función enlaces 1

La función enlaces recibe como parámetros el tipo de dato, un código de este y la ruta del archivo que contendrá como destino la llamada AJAX. El código se observa en la figura 6.21:

```
function enlaces(tipo, numInicio, formato){
  if (navigator.onLine == false){
    document.getElementById("log").style.display="block";
    document.getElementById("carga").style.display="block";
  }
  else{
    if (this.inicios[numInicio])
    {
        realizaProceso(
          direcciones(numInicio),
          tipo,
          formato
        );
        BotonPulsado(tipo, 1);
    }
    else
    {

        elimina_marcadores(tipo);
        BotonPulsado(tipo, 0);

    }
    this.inicios[numInicio]=!this.inicios[numInicio];
    map.removePopup(map.popups[0]);
  }

}
```

Figura 6.21: Función enlaces 2

Al comienzo comprueba si la conexión a internet del usuario sigue activa. En caso afirmativo llamará a la función AJAX ‘realizaProceso’ o eliminará los marcadores del mapa si ya se había llamado a este tipo de dato.

### 6.5.4 realizaProceso

El cliente se comunica con el servidor a través de esta función, visible en la figura 6.22. En la transmisión de datos se envían dos parámetros:

- **Valor:** La ruta URL donde se encuentra el archivo con los datos a descargar.
- **Tipo:** Nombre del tipo de dato.

```
function realizaProceso(valor, tipo, destino){
    var parametros = {
            "valor" : valor,
            "tipo"  : tipo
    };
    $.ajax({
            data:  parametros,
            url:   destino,
            type:  'post',
            beforeSend: function () {
                    $('#carga').css({display:'block'});
            },
            success:  function (response) {
                    $("#resultado").html(response);
                    $('#carga').css({display:'none'});
            }
    });
}
```

Figura 6.22: Realiza Proceso

Con 'beforeSend' se muestra el div de carga durante los instantes que tarda en recibir los datos del servidor y 'succes' recibe la respuesta del servidor y oculta el div de carga.

### 6.5.5 cargaCSV.php

Siguiendo el camino que haría el programa en una petición del cliente, ahora se detallan las acciones del servidor. En primer lugar, se comentará el caso de las opciones que hacen uso de CSV. Lo primero que hay que realizar es la recogida, mediante POST, de los datos enviados por la función AJAX.

En segundo lugar, se accede al fichero en la ruta especificada si está disponible para su lectura. Esta comprobación se observa en la figura 6.24.

```
if (($fichero = fopen($resultado, "r")) !== FALSE) {
```

Figura 6.23: Carga CSV 1

En la figura 6.24 se observa como la obtención de los datos se realiza gracias al método fgetcsv(), introduciendo como parámetro el limitador de cada campo. En base a los datos obtenidos se crea un array asociativo con el nombre de la columna y el dato que contiene. Finalmente se cierra la conexión con el fichero.

```php
// Lee los nombres de los campos
$nombres_campos = fgetcsv($fichero, 0, ";", "\"", "\"");
$num_campos = count($nombres_campos);
// Lee los registros
while (($datos = fgetcsv($fichero, 0, ";", "\"", "\"")) !== FALSE) {
    // Crea un array asociativo con los nombres y valores de los campos
    for ($icampo = 0; $icampo < $num_campos; $icampo++) {
        $registro[$nombres_campos[$icampo]] = $datos[$icampo];
    }
    // Añade el registro leido al array de registros
    $registros[] = $registro;
}
fclose($fichero);
```

Figura 6.24: Carga CSV 2

Posteriormente se accede al método único de cada tipo de dato. Recorriendo la variable $registros se puede indicar las coordenadas y el texto que se añadirá al popup. El ejemplo de la figura 6.25 corresponde a las cámaras de tráfico.

```php
function camaras($registros, $tipo){
  $icono = "./iconos/camaras.png";
  echo "<script>";
  for ($i = 0; $i < count($registros); $i++) {
      $popup = "<b>URL:</b> <a href=".$registros[$i]["url_trafico"]." target='_blank'>".$registros[$i]["descrip"]."</a>";
      $popup = addslashes($popup);
      $popup = mb_convert_encoding($popup, "UTF-8", "Windows-1252");
      echo 'set_marcadores( ' . $registros[$i]["X"] . ', ' . $registros[$i]["Y"] .
                          ', "' . $popup . '", "' . $icono .'", "' . $tipo . '");';
   }
  echo "</script>";
}
```

Figura 6.25: Carga CSV 3

Por cada elemento del array '$registros' se realiza una llamada al método del cliente 'set_marcadores'. Como parámetros se envían las coordenadas, el popup al que se le ha dado forma dentro del mismo bucle, la ruta al icono del marcador, y el tipo de dato que se trata.

### 6.5.6 Set_marcadores

De vuelta en el cliente, el método 'set_marcadores' crea un elemento o 'feature' por cada llamada realizada. Los marcadores son una figura de tipo 'Geometry.Point'. Se añaden las coordenadas y dos atributos opcionales: Description para el texto popup y externalGraphic para la imagen del ícono. La creación de un 'feature' se observa en la figura 6.26.

```javascript
var feature = new OpenLayers.Feature.Vector(
    new OpenLayers.Geometry.Point( longitud, latitud ).transform(epsg4326, projectTo),
    {description: textoPopUp} ,
    {externalGraphic: rutaIcono, graphicHeight: 20.76, graphicWidth: 18}
);
```

Figura 6.26: Set Marcadores

Cada figura se añade finalmente al mapa con el método 'addFeatures' asociado a la capa de marcadores creada en el index.

### 6.5.7 cargaJSON.php

En este apartado se ha vuelto al lado del servidor para explicar cómo se tratan los ficheros en el caso de que sean JSON. Al igual que en 'cargaCSV.php' se recogen los datos enviados por AJAX mediante POST. Se vuelca el fichero JSON con 'file_get_contents', variable tratada de dos formas, ambas mostradas en las figuras 6.27 y 6.28, respectivamente:

```
$str_datos = file_get_contents($resultado);
```

Figura 6.27: Carga JSON 1

Esta forma, basada en los ficheros GeoJSON habituales. Como parámetro se añade la URL donde se ubica el fichero únicamente.

```
$str_datos = file_get_contents('http://user:password@'.$resultado);
```

Figura 6.28: Carga JSON 2

O bien, añadiendo en la llamada HTTP usuario y contraseña. Este caso se utiliza para las llamadas que usan como base la posición del usuario. En ellas se requiere de autentificación para el uso de la API. Dentro de $resultado se ha añadido previamente las coordenadas del usuario.

La decodificación de los datos JSON para su posterior lectura se realiza con el método PHP 'json_decode' (figura 6.29). Como parámetros se añade la variable en la que se ha recogido el fichero y la opción booleana 'true' para indicar que se desea decodificar en un array asociativo.

```
$geojson = json_decode($str_datos,true);
```

Figura 6.29: Carga JSON 3

A partir de este momento, y al igual que ocurriera en el caso de CSV se tratan los datos de distinta forma dependiendo del tipo. Como ejemplo descriptivo se usa el caso del carril bici.

Se realiza una iteración de bucle por cada elemento que haya en el JSON. En cada iteración se realiza otro bucle para recoger el total de coordenadas que forman en este caso una línea o 'lineString' que se irán añadiendo al array coordenadas. Se incluye el texto conveniente al popup y se llama al método 'set_lineas' para que el cliente muestre en pantalla la figura geométrica.

```
echo "<script>";
for ($i = 0; $i < count($geojsonObjeto); $i++){

  $coordenadas = [];
  $popup = "<b>Estado:</b> ". $geojsonObjeto[$i]["properties"]["estado"];

  for ($j = 0; $j < count($geojsonObjeto[$i]["geometry"]["coordinates"]); $j++){
      $coordenadas[$j][0] = $geojsonObjeto[$i]["geometry"]["coordinates"][$j][0];
      $coordenadas[$j][1] = $geojsonObjeto[$i]["geometry"]["coordinates"][$j][1];
  }

  echo 'set_lineas('.json_encode($coordenadas).', '.json_encode("#2E9AFE").
              ', '.json_encode($tipo).', '.json_encode($popup).');';
}

echo "</script>";
```

Figura 6.30: Carga JSON 4

'Set_lineas' recibe como parámetros las coordenadas, el color con el que dibujar la línea, el tipo de dato y el texto del popup. Todo el código se observa en la figura 6.30.

### 6.5.8 Set_lineas

OpenLayers dibuja las geometrías en base a una serie de puntos. Por ello se recorre el array de coordenadas, como el de la figura 6.31, creando una serie de puntos.

```
for (var i = 0; i<coordenadas.length; i++){
  points[i] = new OpenLayers.Geometry.Point( coordenadas[i][0], coordenadas[i][1]).transform(epsg4326, projectTo);
}
```

Figura 6.31: Set líneas 1

En base a esos puntos se crea la figura, que puede ser o bien un polígono o una línea (figuras 6.32 y 6.33.

Polígono

```
var ring = new OpenLayers.Geometry.LinearRing(points);
var polygon = new OpenLayers.Geometry.Polygon([ring]);

var feature = new OpenLayers.Feature.Vector(
  polygon,
  {description: textoPopUp},
  estilo
);
```

Figura 6.32: Set líneas 2

Línea

```
var feature = new OpenLayers.Feature.Vector(
  new OpenLayers.Geometry.LineString(points),
  {description: textoPopUp},
  estilo
);
```

Figura 6.33: Set líneas 3

# 7. Resultados

Con cada iteración durante el desarrollo del proyecto se comprobaba que requisitos de los listados en el capítulo 4 se había cumplido. En este instante, la aplicación ya ha sido desarrollada y es momento de examinarla para dar el proyecto por finalizado. Para ello se van a comprobar por última vez todos los requisitos descritos anteriormente.

## 7.1 Requisitos en el cliente

### 7.1.1 Interfaz gráfica

- **Sencillez:** Se ha simplificado al máximo las tareas a realizar por parte del usuario. Cada una de ellas se accionan con un click y se retiran del mismo modo. Cuentan con un nombre descriptivo para no dar pie a confusiones sobre su utilidad.
- **Robustez:** Todos los elementos visuales se comportan correctamente en los distintos navegadores bajo distintos dispositivos.
- **Adaptabilidad:** Todos los elementos visuales han sido testeados en distintos dispositivos con diferentes tamaños de pantalla y su adaptación ha sido satisfactoria.
- **Carisma:** Predomina el color azul en las opciones del menú. Mientras que cada tipo de dato en pantalla se muestra con un color único para asociarlo a él. Además, en casos como Valenbisi o el tráfico varia el color en función del estado en el que se encuentre el elemento.
- **Visualización:** La idea inicial de como mostrar cada dato en pantalla se ha cubierto completamente y su funcionamiento en ella es el esperado.

### 7.1.2 Desarrollo

Para el desarrollo en el cliente finalmente se usó la librería OpenLayers que ha resultado ser una gran herramienta para trabajar en aplicaciones de mapeado vía web. Tanto los marcadores, como los objetos poligonales, se adaptan al zoom aplicado para así no cubrir elementos del mapa que pudieran ser útiles. Por último, los elementos del menú varían su función según su estado.

## 7.2 Requisitos en el servidor

En cuanto a los requisitos en el servidor las comprobaciones realizadas proporcionan los siguientes resultados:

- **Actualización:** Cada evento lleva asociada la dirección por donde obtener los datos. Cuando el evento se activa, descarga los datos correspondientes in situ, de esta forma siempre se obtienen los datos más recientes.
- **Optimización:** La dificultad que entraña los tiempos de espera reside en el tamaño del fichero de datos a obtener, generalmente esto no supone un problema. La excepción se encuentra en la llamada a las manzanas catastrales, donde el tamaño del fichero da lugar a unos segundos de espera hasta la visualización de ellos.
- **Variedad:** Se cumple el objetivo de tratar varios formatos. Se disponen de ficheros en CSV y GeoJSON, a lo que se debe sumar los datos basados en la posición que están formados en JSON.
- **Adaptabilidad:** Los datos con una posición única se visualizan con marcadores y los que disponen de varias a través de líneas o polígonos. Además los objetos que tengan un estado variable varían de color en función de este.
- **Comunicación:** Se ha implantado AJAX para que todo el desarrollo de la aplicación resida en la misma web.
- **Seguridad:** Toda la transmisión interna de datos se realiza con el método POST.
- **Robustez:** Se han cubierto los posibles errores por error en la conexión a internet o por fallo en la obtención de los datos.
- **Escalabilidad:** La gran mayoría de servidores web soportan el lenguaje PHP por su gran uso. No debe suponer un problema la migración de un servidor a otro.

# 8. Conclusiones

## 8.1 Problemas encontrados

A lo largo del proyecto existen problemas de cualquier índole, muchos de ellos evitables o simplemente por error del desarrollador en algunos de los pasos. En esta sección se listan los problemas destacados:

- **Documentación:** La librería de OpenLayers dispone de una wiki extensa en la que se puede obtener las propiedades de cualquier método de ella. No obstante, los ejemplos sobre su uso son escasos y nada esclarecedores en muchos casos, obligando al usuario a tener que buscar la información necesaria en foros externos de programación, con la dedicación de tiempo que ello conlleva.

- **Vectores:** Esta dificultad está enlazada con la anterior. En el momento de hacer zoom en el mapa con varios elementos de tipo vectorial (líneas y polígonos) existía un problema con el redimensionamiento de estos. Los objetos se deben redimensionar para no albergar un tamaño del mapa que impida visualizar otros elementos, sin embargo durante el proceso los objetos perdían sus atributos, aplicándose a todos el del objeto más reciente. Como resultado todos los elementos se mostraban con el mismo color al aplicar zoom in o zoom out.

- **Adaptación:** Durante el desarrollo de la aplicación se ha usado Google Chrome como navegador, así que se modificaban errores en función de cómo se visualizara en este navegador. Para adaptar la aplicación a distintos navegadores y en distintas plataformas había que testear en todos ellos cuando se finalizaba una fase del proyecto.

- **Datos abiertos:** En ciertos casos, la información que proporciona el portal de datos abiertos de Valencia no está bien detallada o se encuentra desactualizada. Esto repercute directamente en la calidad de la aplicación.

## 8.2 Conclusiones del trabajo

Durante el desarrollo del proyecto se han analizado las ingentes cantidades de información de la que puede disponer una ciudad. Estos datos pueden tener un gran valor para la población. Dando un acceso libre a los datos de la ciudad se aporta una gran herramienta, con la que desarrolladores de la comunidad pueden crear aplicaciones y de esta forma apoyar al sector tecnológico de la región y a su vez ofrecer un mejor servicio del lugar a los ciudadanos.

Pero el Open Data también requiere de un buen mantenimiento. Si el ayuntamiento de la localidad no se compromete a tener unos datos consistentes y actualizados, los desarrolladores a la larga no harán uso de ellos.

## 8.3 Trabajos futuros

El proyecto CityControlPanel se ha desarrollado únicamente en la ciudad de Valencia, por ser el lugar de residencia del autor y localidad de la UPV. Pero la idea es que la aplicación pueda recoger datos de cualquier ciudad que los facilite en España y Europa.

Una ampliación interesante es la conversión de la aplicación en un portal de consulta ciudadana. Mediante la colaboración con el ayuntamiento se mostrarían consultas de documentación que demande la población. Así de esta forma evitar el tener que ir a la administración correspondiente para solicitar un documento que puede ser obtenido perfectamente por internet mediante el DNI electrónico.

Aunque la web se adapta a cualquier tipo de dispositivo una mejora sería el desarrollo de una App para Android e iOS. Con ello se generaría un mayor tráfico del proyecto debido a que se atraería a un público más general.

# 9. Bibliografía

[1] **"OpenStreetMap: be your own cartographer" Autor:** Jonathan Bennet. **Editorial:** Birmingham. **Año:** 2010. **ISBN:** 9781847197504.

[2] **"Portal transparencia y datos abiertos Valencia" URL:** "http://gobiernoabierto.valencia.es/". **Palabra clave:** Open Data. **Nombre empresa:** Ajuntament de València. **Fecha visualización:** 15/03/2015.

[3] **"MapIcons" URL:** "https://mapicons.mapsmarker.com/". **Palabra clave:** MapIcons. **Nombre empresa:** Nicolas Mollet. **Fecha visualización:** 23/03/2015.

[4] **"Manual OpenLayers" URL:** "http://openlayers.bicimap.es/manualOpenLayers.html". **Palabra clave:** OpenLayers. **Nombre empresa:** Ingemoral S.L. **Fecha visualización:** 20/02/2015.

[5] **"JavaScript" Autor:** Astor de Caso Parra. **Editorial:** Anaya Multimedia. **Año:** 2011. **ISBN:** 9788441530485.

[6] **"Manual de PHP" URL:** "https://secure.php.net/manual/es/index.php". **Palabra clave:** PHP. **Nombre empresa:** Ingemoral S.L. **Fecha visualización:** 20/02/2015.

[7] **"AJAX for web application developers" Autor:** Kris Hadlock. **Editorial:** Indianapolis, Ind. **Año:** 2006. **ISBN:** 0672329123.

[8] **"Representational State Transfer" URL:** "https://es.wikipedia.org/wiki/Representational_State_Transfer". **Palabra clave:** REST. **Nombre empresa:** Wikipedia. **Fecha visualización:** 09/05/2015.

[9] **"UML" Autor:** Howard Podeswa. **Editorial:** Anaya Multimedia. **Año:** 2010. **ISBN:** 9788441527195.

# Tabla de figuras