Análisis de requerimientos y prototipado de una aplicación web accesible para personas ciegas basada en la API de Google Maps

Rubén Alcaraz Martínez

Tutora: Mireia Ribera Turró

10/06/2014

Curs 2013-2014

Màster en Gestió de Continguts Digitals

Facultat de Biblioteconomia i Documentació

Rubén Alcaraz Martínez

Junio 2014.

Los contenidos de este documento están sujetos a una licencia Creative Commons Reconocimiento - No Comercial - Compartir Igual 3.0.

Usted es libre de copiar, distribuir y comunicar públicamente la obra, así como de transformarla bajo las siguientes condiciones: reconozca los créditos de la obra: Rubén Alcaraz Martínez y el Máster en Gestión de Contenidos UB/UPF, no utilice la obra para fines comerciales y compártala bajo la misma licencia si altera o modifica su contenido. Puede consultar el texto completo de la licencia en:

http://creativecommons.org/licenses/by-nc-sa/3.0/es/

The power of the Web is in its universality.
Access by everyone regardless of disability is an essential aspect.
El poder de la Web está en su universalidad. El acceso por cualquier persona, independientemente de la discapacidad que presente es un aspecto esencial.

Tim Berners-Lee, *Director del W3C e inventor de la World Wide Web*.

Agradecimientos

Mireia Ribera, profesora de la Facultat de Biblioteconomia i Documentació de la Universitat de Barcelona y tutora de este trabajo.

Santi Moese, tiflotécnico de la ONCE.

Resumen

Objetivos: Explorar las posibilidades ofrecidas por las diferentes plataformas de cartografía digital, así como de otros lenguajes, librerías y estándares actuales del Web, para crear una aplicación accesible para personas ciegas que ofrezca servicios relacionados con información de carácter geográfico.
Metodología: Desarrollo informático de una prueba de concepto consistente en un mapa digital accesible para personas ciegas basado en la API de Google Maps, a partir de los requerimientos derivados del estudio de las características del colectivo objetivo, del análisis de las directrices para la accesibilidad del contenido web (WCAG 2.0) y de la especificación WAI-ARIA 1.0.
Resultados: Se ha obtenido un primer prototipo que permite realizar búsquedas sobre la base de datos de Google Places y obtener rutas entre dos puntos a partir del servicio de rutas de Google Maps, en el que destaca el acceso mayoritario a las funcionalidades de la aplicación mediante una interfaz de teclado, así como la aplicación de la ontología de roles, estados y propiedades de WAI-ARIA para anunciar los cambios en la interfaz provocados por la interacción del usuario con la aplicación.
Palabras clave: Google Maps API v3, Accesibilidad web, WCAG 2.0, WAI-ARIA, Mapas digitales, Servicios basados en la localización, Personas ciegas.

Resum

Objectius: Explorar les possibilitats oferides per les diferents plataformes de cartografia digital, així com per altres llenguatges, llibreries i estàndards del Web, per a crear una aplicació accessible per a persones cegues que ofereixi serveis relacionats amb informació de caràcter geogràfic.
Metodologia: Desenvolupament informàtic d'una prova de concepte consistent en un mapa digital accessible per a persones cegues basat en l'API de Google Maps, a partir dels requeriments derivats de l'estudi de les característiques del col·lectiu objectiu, de l'anàlisi de las directrius per a l'accessibilitat del contingut web (WCAG 2.0) i de l'especificació WAI-ARIA 1.0.
Resultats: S'ha obtingut un primer prototip que permet realitzar cerques a la base de dades de Google Places i obtenir rutes entre dos punts a partir del servei de rutes de Google Maps, en el que destaca l'accés a gairebé totes les funcionalitats del contingut mitjançant una interfície de teclat, així com l'aplicació de l'ontologia de rols, estats i propietats de WAI-ARIA per anunciar els canvis en la interfície provocats per la interacció de l'usuari amb l'aplicació.
Paraules clau: Google Maps API v3, Accessibilitat web, WCAG 2.0, WAI-ARIA, Mapes digitals, Serveis basats en la localització, Persones cegues.

Abstract

Objective: Explore the possibilities offered by the different web mapping platforms and other languages, libraries and current Web standards, to create an accessible application for blind people that offers services related to geographical information.
Methodology: Software development of a proof of concept consisting in an accessible digital map for blind people based on the Google Maps API and on the requirements derived from the study of the characteristics of the target group, the analysis of the Web content accessibility guidelines (WCAG 2.0) and the WAI-ARIA 1.0 specification.
Results: Creation of a first prototype that allows searching in the Google Places database and get directions between two points from the directions service of Google Maps. Most of the functionalities of the content are operable through a keyboard interface and also has been applied the ontology of roles, states and properties of WAI-ARIA for announcing interface changes caused by user interaction.
Keywords: Google Maps API v3, Web accessibility, WCAG 2.0, WAI-ARIA, Web mapping, Location based services, Blind people.

Sumario

1. Resumen ejecutivo

La información geográfica y sus diferentes sistemas de representación, con los mapas digitales a la cabeza, está ganando cada vez más importancia. La evolución de los teléfonos móviles hacia terminales inteligentes con potentes sistemas operativos y sistemas de posicionamiento GPS no ha hecho más que aumentar el interés por estas tecnologías. En este contexto, y conscientes de las posibilidades económicas existentes detrás del mundo de las búsquedas locales, las principales compañías de Internet como Google, Apple o Microsoft y un gran número de medianos y pequeños actores han desarrollado un sin fin de tecnologías, plataformas y sistemas relacionados con este tipo de información que se ha venido a sumar a la ofrecida desde hace años por las grandes compañías de la industria de los sistemas de información geográfica. Actualmente, son varias las aplicaciones que utilizan de alguna manera la posición del usuario para ofrecerle diferentes servicios relacionados con la planificación de rutas, navegación, localización de personas o información sobre puntos de interés cercanos. Paralelamente, estamos viviendo en los últimos años una constante integración de estas tecnologías en todo tipo de portales de Internet. Uno de los ejemplos más evidentes son los tradicionales callejeros de los webs de las administraciones locales. En pocos años, los ayuntamientos de nuestro territorio han pasado de ofrecer mapas prácticamente estáticos, a aplicaciones dinámicas que ofrecen tanto la tradicional información geográfica sobre el municipio, como nuevos servicios de valor añadido. Pero no son los únicos, buscadores de hoteles o los servicios de páginas amarillas, son sólo dos ejemplos más de servicios que han incorporado estas tecnologías en sus portales.

Todo este conjunto de tecnologías y aplicaciones hacen la vida más fácil al usuario. Sin embargo, el uso exclusivo de interfaces visuales para comunicar la información de carácter geográfico supone una importante barrera de acceso para los discapacitados visuales. La presentación y comunicación de la información que proporcionan estos sistemas debe adaptarse a las necesidades de este y de otros grupos de discapacitados. La supresión de las barreras tecnológicas constituye un tema de interés público además de un imperativo legal que permitirá alcanzar el derecho constitucional de igualdad de oportunidades. Asimismo, supone un importante factor de competitividad para aquellas empresas que incorporan la accesibilidad en el desarrollo de sus productos, permitiéndoles llegar a un mayor número de clientes, incluidas las administraciones públicas obligadas a contratar productos accesibles.

El usuario ciego presenta unas características particulares y sobretodo se caracteriza por una manera de acceder al contenido web completamente diferente a la del resto de usuarios. A diferencia de los usuarios sin problemas de visión, las personas ciegas no son capaces de tomar decisiones en base a la organización visual del contenido o a relaciones jerárquicas que no se puedan identificar por programación. En este sentido, una sintaxis correcta que aproveche el actual potencial semántico del lenguaje HTML permite a estos usuarios identificar la estructura de las páginas y localizar en ellas la información que necesitan. La imposibilidad de utilizar un periférico imprescindible para el resto de usuarios como el ratón, hace del teclado su principal aliado. De esta manera, la posibilidad de saltar bloques de información, el orden de tabulación y sobre todo, la capacidad de todos los elementos del contenido para tener el foco del teclado

y ser ejecutados mediante esta interfaz, resultan vitales para la correcta interacción del usuario con las funcionalidades de una página web o aplicación. Otro de los requisitos imprescindibles para poder acceder a la totalidad del contenido es la disponibilidad de alternativas de texto para todos los contenidos de carácter visual. Todos estos requerimientos quedan recogidos en forma de criterios de conformidad en las *Directrices para la accesibilidad del contenido web* (WCAG) y son analizados hasta el nivel de conformidad doble A en el apartado 6 de este trabajo con el objetivo de tenerlos en cuenta en el marco de la creación de un prototipo de mapa digital. En la misma línea, se aborda la especificación WAI-ARIA en el apartado 7, aplicable a ciertos elementos interactivos propios de las aplicaciones enriquecidas desarrolladas con AJAX. Tecnología presente en la actualidad en la mayoría de APIs o programas para la creación de mapas digitales y aplicaciones relacionadas. WAI-ARIA resulta de interés para mejorar la accesibilidad del prototipo en lo que respecta al anuncio de áreas del contenido que se actualizan como respuesta a determinadas acciones de los usuarios, para establecer roles que permitan a las ayudas técnicas entender la función de determinados elementos de la interfaz con independencia de su compatibilidad con el navegador y para mejorar la identificación y sugerencias ante errores, entre otras funciones.

A los requerimientos de las directrices de accesibilidad se le suma el *feedback* obtenido en una entrevista/test de usuario en el que una persona ciega, experta en el uso de ayudas técnicas y usuario habitual de la tecnología que nos ocupa, probó diferentes prototipos, así como algunas aplicaciones comerciales disponibles en línea.

Actualmente, disponemos de un amplio mercado de APIs y librerías para desarrollar mapas digitales y servicios basados en la localización, si bien es cierto que las más populares son las de Google Maps, Bing Maps y OpenStreetMaps. En este trabajo se ha seleccionado la API de Google Maps para la creación de una prueba de concepto con el objetivo de poner en práctica las técnicas necesarias para desarrollar un mapa digital accesible. Los motivos de la elección tienen que ver con la mayor documentación disponible, el hecho de que la API cuente con un servicio de rutas completamente desarrollado y con el mayor nivel de adopción de esta API entre los desarrolladores con respecto a sus competidoras. Esta prueba de concepto consiste en la implementación de una aplicación que incorpora como principales funcionalidades la capacidad de realizar búsquedas de lugares sobre la base de datos de Google Places, devolviendo al usuario aquellos resultados situados a una determinada distancia de su posición, así como un servicio de rutas entre dos puntos, el primero de los cuales puede ser, si así se desea, la ubicación actual del usuario. El objetivo, no es tanto obtener una aplicación totalmente operativa, como sí lo es, una vez identificados los problemas de accesibilidad inherentes al resultado, identificar y aplicar las alternativas necesarias en vistas a corregirlos.

Finalmente, se planifica la ejecución del proyecto y se estima un presupuesto en el marco ficticio de un ayuntamiento que desea implementar una aplicación de estas características.

2. Alcance y objetivos del proyecto

2.1. Definición

El proyecto consiste en la exploración de las posibilidades ofrecidas por las diferentes plataformas de cartografía digital, así como de otras herramientas y estándares actuales del Web, para crear una aplicación accesible para personas ciegas que ofrezca servicios relacionados con información de carácter geográfico.

2.2. Alcance

Se trata de un trabajo esencialmente exploratorio. No obstante, una parte de las averiguaciones se han puesto a prueba en un prototipo en fase beta.

2.3. Objetivos

Los objetivos del proyecto son:

- Explicar las principales características del colectivo de usuarios ciegos, concretamente aquellas que determinan la manera en cómo éstos acceden a la información de carácter geográfico.
- Analizar el mercado de APIs, librerías y estándares más populares para la creación de mapas digitales y servicios basados en la localización.
- Determinar una serie de requerimientos teóricos y técnicos a partir del análisis de las directrices internacionales de accesibilidad web y del estudio de otros proyectos similares.
- Describir el proceso de creación de una aplicación accesible que permita geolocalizar al usuario, ofrecerle puntos de interés cercanos y le proporcione un servicio de rutas entre dos puntos.
- Desarrollar una prueba de concepto (prototipo) de la aplicación descrita.

3. Justificación

El campo de la información y la comunicación constituye un elemento esencial en la vida social, política, cultural y económica de las personas. Los avances producidos dentro del ámbito de las tecnologías de la información y la comunicación han abierto nuevas posibilidades de acceso a estos elementos, pero al mismo tiempo también han propiciado, de manera indirecta, la creación de nuevas barreras para las personas con discapacidad. Por lo que respecta al disfrute y uso de todo tipo de bienes, servicios y tecnologías, el marco regulatorio debería abarcar todas las situaciones en las cuales se ponen a disposición del público en general. Según lo dicho, resulta poco relevante quién proporcione el bien o preste el servicio, ya sea el propio Estado o una empresa privada. Si el bien o servicio se pone a disposición del público en general, resulta lógico que deba adaptarse a unos requisitos de accesibilidad determinados, como recoge la disposición final sexta de la *Ley 51/2003, de 2 de diciembre, de igualdad de oportunidades, no discriminación y accesibilidad universal de las personas con discapacidad*. Los incumplimientos del principio de accesibilidad universal son considerados por esta ley como supuestos de discriminación y de violación del derecho a la igualdad de oportunidades. La accesibilidad universal es definida en la ley como "la condición que deben cumplir los entornos, procesos, bienes, productos y servicios, así como los objetos o instrumentos, herramientas y dispositivos, para ser comprensibles, utilizables y practicables por todas las personas en condiciones de seguridad y comodidad y de la forma más autónoma y natural posible. Presupone la estrategia de 'diseño para todos' y se entiende sin perjuicio de los ajustes razonables que deban adoptarse".

Diversos autores (Nielsen, 2000; Romero, 2001; Arch y Letourneau, 2002) han destacado la importancia de la accesibilidad no sólo como cuestión ética, sino también como factor de competitividad en el entorno empresarial. Teniendo en cuenta tanto el importante porcentaje que representan las personas con algún tipo de discapacidad entre los usuarios o clientes potenciales de cualquier web o aplicación, como el elevado grado de exigencia existente en el entorno web fruto de una competitividad cada vez mayor, la integración de los estándares de accesibilidad en el desarrollo de estos productos puede resultar un rasgo diferencial frente a la competencia. En este sentido, las grandes empresas desarrolladoras de software ya hace años que disponen de departamentos de I+D encargados exclusivamente de las normativas internas de usabilidad y accesibilidad. Los casos de Microsoft[1], Apple[2], IBM[3] u Oracle[4], son sólo algunos ejemplos. Además, la accesibilidad también resulta de utilidad en otros ámbitos, como en el

[1] *Microsoft accessibility*. <http://www.microsoft.com/enable/>. [Consulta: 31/05/2014].

[2] *Apple accesibilidad*. >http://www.apple.com/es/accessibility/>. [Consulta: 31/05/2014].

[3] *IBM accessibility: Human Ability and Accessibility Center*. <http://www-03.ibm.com/able/>. [Consulta: 31/05/2014].

[4] *Oracle's accessibility program*. <http://www.oracle.com/us/corporate/accessibility/index.html>. [Consulta: 31/05/2014].

posicionamiento en motores de búsqueda (Dolson, 2012; Nielsen, 2012; Carreras, 2013)[5] o en la mejora de las versiones móviles de los portales web, entre otros.

El proyecto que nos ocupa beneficia principalmente a las personas ciegas o con el órgano de la vista afectado severamente por cualquier circunstancia, pero también es aplicable o útil para usuarios con otros tipos de discapacidades. La incorporación de las pautas o requerimientos que se enumeran en este proyecto son de utilidad para cualquier persona o empresa interesada en desarrollar aplicaciones accesibles. Se trata, además, de mejorar un tipo de contenido que prácticamente todas las administraciones públicas ya han incorporado en sus respectivos portales en Internet, generalmente de una manera no accesible. Algunos ejemplos sobre los que se podría aplicar son los planos digitales de las ciudades, los servicios de rutas en transporte público o incluso mapas que se han creado en otros ámbitos públicos o privados, como los directorios de empresas o los portales de búsqueda de hoteles, entre otros.

Fig. 1. En los últimos años los mapas digitales se han convertido en un tipo de contenido presente en la mayoría de portales de la administración. También en todo tipo de servicios privados. En la figura, el buscador de hoteles de *Trivago* y el *Plànol de BCN*, dos ejemplos no accesibles.

En el campo que nos ocupa en este proyecto cabe destacar que las características inherentes al tipo de información con el que vamos a trabajar y especialmente el medio o producto sobre el cual se muestra (el mapa), presenta como principal característica un marcado carácter visual. La experiencia de ver un mapa es prácticamente imposible de sustituir a través de

[5] Nielsen utiliza la frase "*the world's richest blind user*" para referirse a Googlebot, el robot de Google responsable del rastreo de webs, al tratarse de un visitante similar a los discapacitados visuales en tanto que no es capaz de "ver" las imágenes, vídeos, animaciones, de seguir los enlaces que dependen de Flash o Javascript, etc.

cualquier alternativa tecnológica, con la excepción de las impresiones en relieve o 3D (García; Ruíz, 2010). No obstante, cualquier información que se transmita a través de él (puntos de interés, rutas, leyendas, etc.), o que se desprenda de su mera visualización (como por ejemplo, la distancia entre dos puntos), puede ser transmitida de manera efectiva a un usuario ciego o con cualquier otro tipo de discapacidad.

Además de las consideraciones de tipo moral y económico que hemos mencionado hasta ahora, existe toda una colección de leyes que se han venido desarrollando en los últimos años en nuestro país, que sitúan a las directrices internacionales de accesibilidad web al nivel de norma de obligatorio cumplimiento para los portales y productos web de las administraciones públicas, los subvencionados con dinero público y para otro conjunto de portales proveedores de servicios de la administración o que cumplan con unas determinadas características.

3.1. Estadísticas sobre personas ciegas y deficientes visuales

La Organización Mundial de la Salud (OMS) estima que alrededor de 285 millones de personas en el mundo sufren algún tipo de discapacidad visual, de las cuales 39 millones son ciegas y 246 millones presentan baja visión[6]. El 82% de las personas ciegas en el mundo tienen 50 años o más. La edad asociada a la discapacidad, pone aún en mayor riesgo de exclusión a estas personas. Según datos del *Census's survey of income and program participation* (SIPP, 1999)[7] norteamericano, las personas con algún tipo de discapacidad visual son mucho menos propensas a utilizar ordenadores que las personas sin problemas de visión. En aquel momento, sólo el 21% de los ciudadanos mayores de 15 años con problemas visuales decían tener acceso a Internet, y únicamente el 13% de la misma población afirmó utilizar el ordenador habitualmente (Gerber, 2003). El 23% de la población sin ningún tipo de discapacidad afirmaba no haber usado nunca un ordenador, frente al 70% de los discapacitados visuales. Las discapacidades visuales han sido identificadas como una de las cuatro causas que más contribuyen a la pérdida de independencia entre las personas de tercera edad estadounidenses (Alliance for Aging Research, 1999)[8].

En España, la cifra total de discapacitados asciende a algo más de 3.800.000 personas. Según se desprende de los datos recogidos en la *Encuesta de discapacidad, autonomía personal y situaciones de dependencia* de 2008[9], el 21% de las personas discapacitadas presenta algún tipo de discapacidad visual. Las discapacidades visuales son las terceras más frecuentes por detrás de aquellas que afectan a los huesos y articulaciones (39,3%) y las del oído (23,8%).

[6] OMS. "Ceguera y discapacidad visual". *Nota descriptiva*. nº 282 (oct. 2013). <http://www.who.int/mediacentre/factsheets/fs282/es/>. [Consulta: 31/05/2014].

[7] *Census's survey of income and program participation*. United States Census Bureau. <http://www.census.gov/programs-surveys/sipp/>. [Consulta: 31/05/2014].

[8] *Alliance for Aging Research*. <http://agingresearch.org/>. [Consulta: 31/05/2014].

[9] *Encuesta sobre discapacidades, autonomía personal y situaciones de dependencia*. Madrid: Instituto Nacional de Estadística, 2008. <http://www.ine.es/jaxi/menu.do;jsessionid=BBC7C64324FB622FAFB69B0130F4BA4F.jaxi01?type=pcaxis&path=/t15/p418&file=inebase&L=0>. [Consulta: 31/05/2014].

Ahondando más en los datos recogidos por el Instituto Nacional de Estadística obtenemos la siguiente tabla con el número de personas en unidades de mil, según el tipo de discapacidad visual y el género en dos rangos de edad[10].

	6 a 64 años			65 o más años			Total		
Tipo de discapacidad visual	Hombres	Mujeres	Total	Hombres	Mujeres	Total	Hombres	Mujeres	Total
Percibir cualquier imagen	8,8	6,6	15,4	15,8	27,2	42,9	24,6	33,7	58,3
Tareas visuales de detalle	83,3	105	188,3	153,6	331,6	485,2	236,9	436,7	673,6
Tareas visuales de conjunto	90,1	110,3	200,4	149,9	311,8	461,7	239,9	422,2	662,1
Otros problemas de visión	65,6	62	127,6	75,2	154,6	229,9	140,8	216,6	357,4

Tabla 1. Discapacitados visuales en España según el tipo de discapacidad y el género en dos rangos de edad.

Según los datos del INE, alrededor de 794.000 personas mayores de 6 años en España sufren algún tipo de discapacidad visual. Lo que supone una tasa de 18,82 personas por cada 1.000 habitantes[11].

En Cataluña, el Instituto de Estadística de Cataluña (IDESCAT) a partir de los datos de la encuesta de 2008 del INE, permite conocer los datos referidos a nuestro territorio[12][13].

[10] Una misma persona puede encontrarse en más de un grupo.

[11] "Encuesta de discapacidad, autonomía personal y situaciones de dependencia (EDAD). Año 2008". *Notas de prensa*. <http://www.ine.es/prensa/np524.pdf>. [Consulta: 31/05/2014].

[12] Encuestas sobre discapacidades. Instituto de Estadística de Cataluña, 2012. <http://www.idescat.cat/es/societat/discapacitats.html>. [Consulta: 31/05/2014].

[13] Datos en unidades de mil.

	6 a 64 años			65 o más años			Total		
Tipo de discapacidad visual	Hombres	Mujeres	Total	Hombres	Mujeres	Total	Hombres	Mujeres	Total
Percibir cualquier imagen	1,3	0,9	2,2	1,3	3,3	4,6	2,6	4,2	6,8
Tareas visuales de detalle	8,2	13,5	21,7	18,7	40,9	59,6	26,9	54,4	81,3
Tareas visuales de conjunto	8,9	12,6	21,5	19,8	46,8	66,6	28,7	59,4	88,1
Otros problemas de visión	8,8	6,8	15,6	10	20,5	30,5	18,8	27,3	46,1

Tabla 1. Discapacitados visuales en Catalunya según el tipo de discapacidad y el género en dos rangos de edad.

Los discapacitados visuales suponen en Cataluña un total de 123.700 personas (6 años o más) según datos del IDESCAT. Lo que supone una tasa de 6,3 personas por cada mil habitantes para el tramo de 6 a 64 años y una tasa de 75,9 habitantes por cada mil en el caso de las personas de 65 años o más.

3.2. Legislación española sobre accesibilidad

El siguiente apartado recoge la principal producción legislativa de nuestro país en el campo de la accesibilidad web, así como sus principales antecedentes o normas relacionadas.

3.2.1. Antecedentes

El primer antecedente a la legislación española sobre accesibilidad en el web lo encontramos en la *Constitución Española* de 1978[14]. En la Carta magna española queda recogida la obligación de los poderes públicos de "promover las condiciones para que la libertad y la igualdad del individuo y de los grupos en que se integra sean reales y efectivas; remover los obstáculos que impidan o dificulten su plenitud y facilitar la participación de todos los ciudadanos en la vida política, económica, cultural y social" (artículo 9.2). También se destaca

[14] <http://www.boe.es/buscar/pdf/1978/BOE-A-1978-31229-consolidado.pdf>. [Consulta: 31/05/2014].

la igualdad de todos los ciudadanos "ante la ley, sin que pueda prevalecer discriminación alguna por razón de nacimiento, raza, sexo, religión, opinión o cualquier otra condición o circunstancia personal o social" (artículo 14). Más específicamente, el artículo 49 insta a los poderes públicos a realizar "una política de previsión, tratamiento, rehabilitación e integración de los disminuidos físicos, sensoriales y psíquicos a los que prestarán la atención especializada que requieran y los ampararán especialmente para el disfrute de los derechos que este Título otorga a todos los ciudadanos". El artículo 49 de la Constitución supuso el primer escalón legislativo para la integración social de los discapacitados, al hacer obligatoria una política de previsión, tratamiento, rehabilitación e integración para este colectivo.

En cumplimiento del mandato constitucional dictado por el artículo 49 antes citado, se publicó la *Ley 13/1982, de 7 de abril, de integración social de los minusválidos*[15], también conocida como LISMI. Un texto cuyos principios generales se basaron en garantizar la realización personal y la total integración social de las personas discapacitadas. La *Ley 13/1982* se mantuvo vigente hasta el 4 de Diciembre de 2013, cuando fue derogada por la entrada en vigor del *Real Decreto Legislativo 1/2013, de 29 de noviembre, por el que se aprueba el texto refundido de la Ley general de derechos de las personas con discapacidad y de su inclusión social*[16].

La *Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico*[17] (LSSI), fijó por primera vez la obligación de que los portales web de las administraciones públicas españolas fueran accesibles, aunque sin determinar un nivel requerido, ni especificar en qué consiste una página web accesible. También abría la posibilidad a que las administraciones reclamasen diseños accesibles a los webs que financiaban y a sus prestadores de servicios y fabricantes de equipos y software. Todo ello en la disposición adicional quinta que establecía como fecha límite para su aplicación el 31 de diciembre de 2005.

En 2003, la LIONDAU o *Ley 51/2003, de 2 diciembre, de igualdad de oportunidades, no discriminación y accesibilidad universal de las personas con discapacidad*[18], en su artículo 10 reguló las condiciones básicas de accesibilidad y no discriminación con el objetivo de alcanzar el máximo nivel de igualdad de oportunidades a todos los ciudadanos con independencia de sus discapacidades. La disposición final séptima de la LIONDAU fijó un plazo de 2 años desde su entrada en vigor, para que el Gobierno aprobara "unas condiciones básicas de accesibilidad y no discriminación para el acceso y utilización de las tecnologías, productos y servicios relacionados con la sociedad de la información y de cualquier medio de comunicación social, que serán obligatorias en el plazo de cuatro a seis años desde la entrada en vigor de esta ley para todos los productos y servicios nuevos, y en el plazo de ocho a diez años para todos aquellos existentes que sean susceptibles de ajustes razonables". Como en el caso de la LISMI, la LIONDAU quedó derogada por el posterior *Real Decreto Legislativo 1/2013, de 29 de*

15 <https://www.boe.es/boe/dias/1982/04/30/pdfs/A11106-11112.pdf>. [Consulta: 31/05/2014].
16 <https://www.boe.es/boe/dias/2013/12/03/pdfs/BOE-A-2013-12632.pdf>. [Consulta: 31/05/2014].
17 <http://www.boe.es/boe/dias/2002/07/12/pdfs/A25388-25403.pdf>. [Consulta: 31/05/2014].
18 <http://www.boe.es/buscar/pdf/2003/BOE-A-2003-22066-consolidado.pdf>. [Consulta: 31/05/2014].

noviembre, por el que se aprueba el Texto Refundido de la Ley General de derechos de las personas con discapacidad y de su inclusión social.

3.2.2. Legislación vigente

Con el objetivo de administrar la puesta en marcha y gestión de la LIONDAU se elaboraron dos instrumentos de planificación: el *I Plan nacional de accesibilidad 2004-2012*[19] y el *II plan de acción para las personas con discapacidad 2003-2007*[20], ambos durante el mismo año 2003. Además, con el objetivo de desarrollar y aplicar los diferentes mandatos explícitos en sus disposiciones finales, durante el periodo 2005-2007 se publicaron cinco reales decretos: *Real Decreto 424/2005, de 15 de abril, por el que se aprueba el Reglamento sobre las condiciones para la prestación de servicios de comunicaciones electrónicas, el servicio universal y la protección de los usuarios*[21] modificado por el posterior *Real Decreto 1494/2007, de 12 de noviembre, por el que se aprueba el Reglamento sobre las condiciones básicas para el acceso de las personas con discapacidad a las tecnologías, productos y servicios relacionados con la sociedad de la información y medios de comunicación social*[22], el *Real Decreto 1414/2006, de 1 de diciembre, por el que se determina la consideración de persona con discapacidad a los efectos de la Ley 51/2003 de Igualdad de Oportunidades, no discriminación y accesibilidad universal de las personas con discapacidad*[23], el *Real Decreto 1417/2006, de 1 de diciembre, por el que se establece el sistema arbitral para la resolución de quejas y reclamaciones en materia de igualdad de oportunidades, no discriminación y accesibilidad por razón de discapacidad*[24], el *Real Decreto 366/2007, de 16 de marzo, por el que se establecen las condiciones de accesibilidad y no discriminación de las personas con discapacidad en sus relaciones con la Administración General del Estado*[25] y el *Real Decreto 1494/2007, de 12 de noviembre, por el que se aprueba el Reglamento sobre las condiciones básicas para el acceso de las personas con discapacidad a las tecnologías, productos y servicios relacionados con la sociedad de la información y medios de comunicación social*[26].

El *Real Decreto 1494/2007, de 12 de noviembre* supuso uno de los textos claves para alcanzar la puesta en marcha de la LIONDAU, al establecer como grado de accesibilidad aplicable a los portales web de las administraciones públicas el nivel mínimo obligatorio de cumplimiento de las prioridades 1 y 2 de la *Norma UNE 139803:2004: Aplicaciones informáticas para personas con discapacidad: requisitos de accesibilidad para contenidos en la Web*[27]. Una norma que en

[19] <http://www.sidar.org/recur/direc/legis/ipna2004_2012.pdf>. [Consulta: 31/05/2014].
[20] <http://www.sidar.org/recur/direc/legis/iipapcd2003_2007.pdf>. [Consulta: 31/05/2014].
[21] <https://www.boe.es/buscar/pdf/2005/BOE-A-2005-6970-consolidado.pdf>. [Consulta: 31/05/2014].
[22] <http://www.boe.es/boe/dias/2007/11/21/pdfs/A47567-47572.pdf>. [Consulta: 31/05/2014].
[23] <http://www.boe.es/boe/dias/2006/12/16/pdfs/A44285-44286.pdf>. [Consulta: 31/05/2014].
[24] <http://www.boe.es/boe/dias/2006/12/13/pdfs/A43718-43724.pdf>. [Consulta: 31/05/2014].
[25] <https://www.boe.es/boe/dias/2007/03/24/pdfs/A12852-12856.pdf>. [Consulta: 31/05/2014].
[26] <http://www.boe.es/boe/dias/2007/11/21/pdfs/A47567-47572.pdf>. [Consulta: 31/05/2014].
[27] <http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0032576>. [Consulta: 31/05/2014].

el año 2012 se vio actualizada[28] y que actualmente es equivalente a las *Web content accessibility guidelines* (WCAG) 2.0. En el mismo real decreto, también se instaba a las administraciones a indicar el grado de accesibilidad aplicado, la fecha de revisión y a disponer de un medio de contacto mediante el cual los usuarios pudieran formular quejas, consultas y sugerencias. El *Real Decreto 1494/2007, de 12 de noviembre*, fue modificado en 2011 por el *Real Decreto 1276/2011, de 16 de septiembre, de adaptación normativa a la Convención Internacional sobre los derechos de las personas con discapacidad*[29].

Como hemos avanzado, el *Real Decreto Legislativo 1/2013, de 29 de noviembre, por el que se aprueba el Texto Refundido de la Ley General de derechos de las personas con discapacidad y de su inclusión social*, unificó la LISMI, la LIONDAU, la *Ley 51/2003 de 2 de diciembre, de igualdad de oportunidades, no discriminación y accesibilidad universal de las personas con discapacidad* y la *Ley 49/2007, de 26 de diciembre, por la que se establece el régimen de infracciones y sanciones...* al quedar refundidas en este nuevo texto. Respecto a este real decreto, cabe destacar la nueva definición de discapacidad. Ahora, la discapacidad ya no es una cualidad propia de una persona, sino que es una situación que surge en el momento en el que interacciona con un equipo, dispositivo, página web, etc., que presenta barreras que limitan sus capacidades.

En el ámbito europeo, la *EN 301 549: Accessibility requirements suitable for public procurement of ICT products and services in Europe*[30], se ha constituido como la primera norma europea de accesibilidad para productos y servicios relacionados con las TIC. La norma se erige como un estándar europeo de requisitos funcionales para la contratación pública de productos y servicios tecnológicos accesibles. El capítulo 9 dedicado a las páginas web, hace suyos los requisitos de nivel A y AA de las WCAG 2.0. Otros capítulos relacionados son el 8 (hardware), el 10 (documentos) y el 11 (software). Por lo que respecta a los anexos, el B incorpora una interesante tabla con todos los requisitos de accesibilidad de las WCAG expresados en términos de rendimiento funcional (uso sin visión, uso con visión limitada, uso sin percepción del color, uso sin audición...), distinguiendo además entre relaciones primarias y secundarias.

3.2.3. Otras leyes relacionadas

- ***ORDEN PRE/1551/2003****, de 10 de junio, por la que se desarrolla la Disposición final primera del Real Decreto 209/2003, de 21 de febrero, por el que se regulan los registros y las notificaciones telemáticas, así como la utilización de medios telemáticos para la sustitución de la aportación de certificados por los ciudadanos*[31]. En la que se requiere que "el registro telemático y el servicio de notificación telemática deberán cumplir los

[28] <http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0049614>. [Consulta: 31/05/2014].
[29] <http://www.boe.es/boe/dias/2011/09/17/pdfs/BOE-A-2011-14812.pdf>. [Consulta: 31/05/2014].
[30] <http://www.etsi.org/deliver/etsi_en/301500_301599/301549/01.01.01_60/en_301549v010101p.pdf>. [Consulta: 31/05/2014].
[31] <http://www.boe.es/boe/dias/2003/06/13/pdfs/A22890-22893.pdf>. [Consulta: 31/05/2014].

requerimientos en materia de accesibilidad establecidos por la Iniciativa para una Web Accesible (WAI) del Consorcio World Wide Web y en particular las especificaciones de la Recomendación de 5 de mayo de 1999 sobre Pautas de Accesibilidad del Contenido en la Web, versión 1.0, en su nivel AA".

- ***Ley 59/2003**, de 19 de diciembre, de firma electrónica*[32]. "Los servicios, procesos, procedimientos y dispositivos de firma electrónica deberán ser plenamente accesibles a las personas con discapacidad y de la tercera edad, las cuales no podrán ser en ningún caso discriminadas en el ejercicio de los derechos y facultades reconocidos en esta ley por causas basadas en razones de discapacidad o edad avanzada".

- ***Ley Orgánica 4/2007**, de 12 de abril, por la que se modifica la Ley Orgánica 6/2001, de 21 de diciembre, de Universidades*[33]. En la que se obliga a las universidades a ofrecer espacios virtuales, así como servicios, procedimientos y el suministro de información accesibles.

- ***Ley 11/2007**, de 22 de junio, de acceso electrónico de los ciudadanos a los servicios públicos*[34]. En la que se insta a ofrecer servicios electrónicos y la publicación en las sedes electrónicas de informaciones, servicios y transacciones en los términos establecidos por la normativa vigente en la materia.

- ***Ley 49/2007**, de 26 de diciembre, por la que se establece el régimen de infracciones y sanciones en materia de igualdad de oportunidades, no discriminación y accesibilidad universal de las personas con discapacidad*[35]. Derogada en 2013 por el *Real Decreto Legislativo 1/2013, de 29 de noviembre, por el que se aprueba el Texto Refundido de la Ley General de derechos de las personas con discapacidad y de su inclusión social*[36]. Conviene destacar que el abono de la correspondiente multa no exime del cumplimiento de la LIONDAU.

- ***Ley 56/2007**, de 28 de diciembre, de medidas de impulso de la sociedad de la información*[37]. En la que con fecha de 31 de diciembre de 2008 se insta, no sólo a las páginas de la administración pública y empresas que gestionen servicios públicos a cumplir la *Norma UNE 139803*, sino también a una serie de empresas con "especial trascendencia económica" con más de 100 trabajadores o una facturación superior a los 6 millones de euros, como: entidades bancarias, aseguradoras, agencias de viajes y de transporte, suministradoras de servicios básicos (agua, luz y electricidad), etc.

32 <http://www.boe.es/boe/dias/2003/12/20/pdfs/A45329-45343.pdf>. [Consulta: 31/05/2014].
33 <http://www.boe.es/boe/dias/2007/04/13/pdfs/A16241-16260.pdf>. [Consulta: 31/05/2014].
34 <http://www.boe.es/boe/dias/2007/06/23/pdfs/A27150-27166.pdf>. [Consulta: 31/05/2014].
35 <http://www.boe.es/boe/dias/2007/12/27/pdfs/A53278-53284.pdf>. [Consulta: 31/05/2014].
36 <https://www.boe.es/boe/dias/2013/12/03/pdfs/BOE-A-2013-12632.pdf>. [Consulta: 31/05/2014].
37 <https://www.boe.es/boe/dias/2007/12/29/pdfs/A53701-53719.pdf>. [Consulta: 31/05/2014].

- ***Ley 7/2010**, de 31 de marzo, general de la comunicación audiovisual*[38].

- ***Ley 26/2011**, de 1 de agosto, de adaptación normativa a la Convención internacional sobre los derechos de las personas con discapacidad*[39]. Que modifica la *Ley 34/2002*, incluyendo la obligación de que las redes sociales (desarrolladas por entidades cuyo volumen anual de operaciones sea mayor de 6 millones de euros) sean accesibles.

- ***Real Decreto 1276/2011**, de 16 de septiembre, de adaptación normativa a la Convención internacional sobre los derechos de las personas con discapacidad*[40]. Que adecua la regulación reglamentaria vigente en materia de discapacidad a las directrices de la Convención Internacional sobre los Derechos de las Personas con Discapacidad, en la línea marcada por la *Ley 26/2011*. También se designa al Comité Español de Representantes de Personas con Discapacidad (CERMI) "como mecanismo para promover, proteger y supervisar la aplicación en España del citado Tratado internacional".

- ***Ley 19/2013**, de 9 de diciembre, de transparencia, acceso a la información pública y buen gobierno*[41]. En la que sobre la información sujeta a las obligaciones de transparencia se dice que "se establecerán los mecanismos adecuados para facilitar la accesibilidad, la interoperabilidad, la calidad y la reutilización de la información publicada así como su identificación y localización".

- ***Ley Orgánica 8/2013**, de 9 de diciembre, para la mejora de la calidad educativa*[42]. En la que a se establece la obligación de contar con "los principios de diseño para todas las personas y accesibilidad universal" en la incorporación de las TIC al sistema educativo, incluidos los sistemas virtuales de aprendizaje.

[38] <http://www.boe.es/boe/dias/2010/04/01/pdfs/BOE-A-2010-5292.pdf>. [Consulta: 31/05/2014].
[39] <http://www.boe.es/boe/dias/2011/08/02/pdfs/BOE-A-2011-13241.pdf>. [Consulta: 31/05/2014].
[40] <http://www.boe.es/boe/dias/2011/09/17/pdfs/BOE-A-2011-14812.pdf>. [Consulta: 31/05/2014].
[41] <https://www.boe.es/boe/dias/2013/12/10/pdfs/BOE-A-2013-12887.pdf>. [Consulta: 31/05/2014].
[42] <http://www.boe.es/boe/dias/2013/12/10/pdfs/BOE-A-2013-12886.pdf>. [Consulta: 31/05/2014].

4. Descripción del colectivo objetivo

4.1. Características generales

Según la ONCE (Organización Nacional de Ciegos Españoles), el 80% de la información necesaria para nuestra vida cotidiana implica el órgano de la visión[43]. Esto se debe a la predominancia de la información de carácter visual en la ejecución de la mayoría de las habilidades que poseemos, de los conocimientos que adquirimos y de las actividades que desarrollamos. Las personas ciegas y deficientes visuales son aquellas que se caracterizan por la limitación total o severa de la función visual respectivamente. La perdida de la percepción visual representa una reducción drástica en la autonomía y desarrollo de las personas que la padecen, dificultando tareas cotidianas (desplazamientos, acceso a la información...), o su inclusión y participación en la sociedad (educación, trabajo, ocio...). El déficit visual es compensado por estas personas a través de patrones auditivos, olfativos, hápticos y térmicos que, a diferencia del resto de personas, ocupan un lugar predominante en su experiencia personal.

Los avances en el área de las tecnologías de la información y la comunicación han supuesto para la mayoría de nosotros la posibilidad de disponer de nuevos servicios que progresivamente han ido mejorando la manera con la que accedemos a la información. Sin embargo, la misma tecnología que nos hace a unos la vida más fácil, puede suponer nuevas barreras y factores de exclusión para las personas con algún tipo de discapacidad. La única manera de evitarlo pasa por la inclusión de la accesibilidad como factor imprescindible en el diseño de esos productos y servicios generados por el avance social y tecnológico.

El usuario ciego accede al contenido web de una manera muy diferente a la que lo hacen el resto de usuarios. Mientras que las personas que no tienen el órgano de la vista afectado por ninguna discapacidad exploran y toman decisiones en base a la organización visual del contenido, el ciego que accede mediante un lector de pantalla no es capaz de analizar la totalidad de la página con tanta inmediatez. Frente a contenidos organizados jerárquicamente o al uso de zonas destacadas cromáticamente para atraer la atención hacia los elementos más importantes de la página, el contenido al que acceden los discapacitados visuales es lineal y basado en texto. El posicionamiento en la página o el diseño gráfico ni ayudan, ni dificultan de manera inherente el acceso de este colectivo a la información que contiene el sitio web. Simplemente resultan irrelevantes. Son otros factores, como el orden por programación, una sintaxis correcta o la posibilidad de saltar bloques de información, los que realmente suponen una óptima experiencia de usuario para este colectivo. El estudio de Theofanos y Redish (2003) en el que se observó el comportamiento de varios usuarios ciegos mientras realizaban una serie de tareas en sitios web del gobierno de Estados Unidos, confirma este comportamiento. Según las conclusiones de este estudio, los usuarios ciegos saltan rápidamente y de manera secuencial a través de los contenidos de la página, escuchando sólo las primeras palabras de

[43] "Discapacidad visual: aspectos generales". En: *ONCE*. <http://www.once.es/new/servicios-especializados-en-discapacidad-visual/discapacidad-visual-aspectos-generales>. [Consulta: 01/06/2014].

cada encabezado o etiqueta. En menor medida, dejan de lado la secuencia propia del contenido para navegar a través de los enlaces y encabezados detectados por el lector de pantalla, recorriendo la página de esta manera de una forma más estructurada. Power, et al. (2013) también llegan a las mismas conclusiones en un estudio en el que intentan averiguar cuáles son las principales estrategias utilizadas por usuarios con diferentes tipos de discapacidad al utilizar algunos de los principales sitios de Internet. Para ello se propuso a un grupo de usuarios con diferentes discapacidades, una serie de tareas (darse de alta en una red social, hacer *login* en una aplicación, utilizar un chat, encontrar sitios de interés cercanos, hacer comentarios en un vídeo, escuchar un programa de radio, escribir un post, etc.) en el marco de diferentes escenarios descritos con detalle, que posteriormente debían comentar en una entrevista con los técnicos que les habían observado. El análisis de los resultados se llevó a cabo a través de la identificación y agrupación de las diferentes operaciones realizadas por los usuarios para lograr cada uno de los objetivos, en lo que denominaron "secuencias de acción estratégica". Posteriormente, cada una de estas acciones estratégicas se agrupó dando lugar a siete tipos de estrategias diferentes (navegación, descubrimiento, exploración, anclaje, búsqueda de ayuda, reiniciar y aceleración de la tarea). Los usuarios ciegos fueron los que utilizaron una mayor cantidad de estrategias diferentes, siendo la exploración, definida por los autores del estudio como la extracción del significado o contexto de una pieza de contenido o de la funcionalidad de un componente interactivo, la más utilizada. Ejemplos de esta secuencia de acción son la lectura de encabezados o del texto de un enlace para decidir si el contenido que viene a continuación o la página a la que apunta ese enlace es relevante para su necesidad informativa. El descubrimiento, definido como el intento de determinar la estructura general de una página web con el objetivo de localizar las secciones relevantes, también resultó una estrategia común para este perfil de usuario.

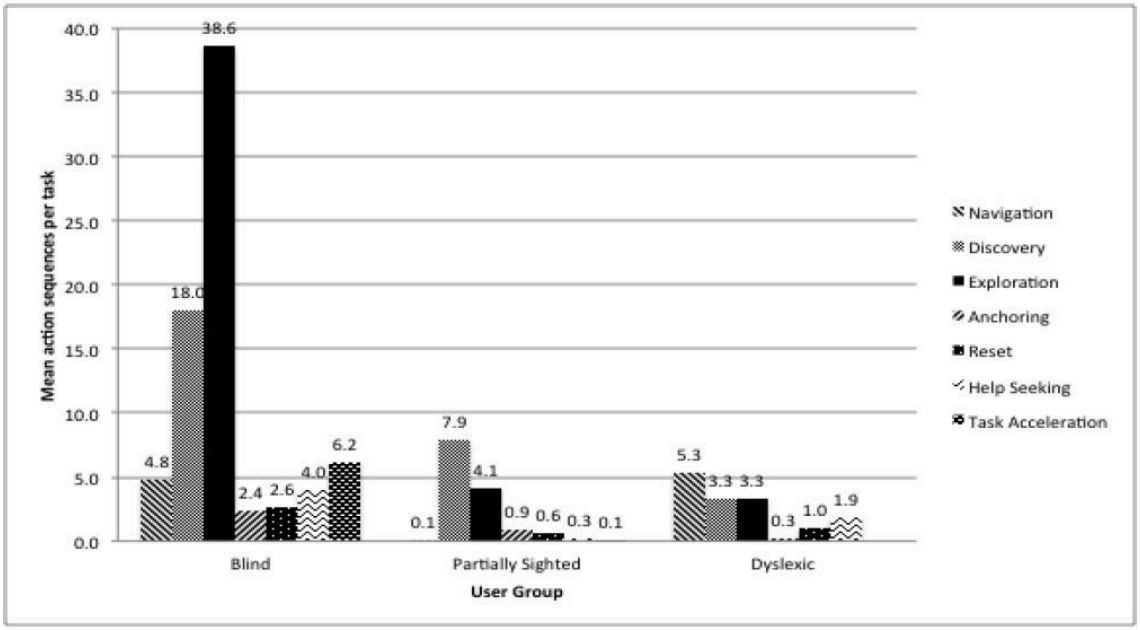


Fig. 2. Tipos de estrategias identificados en el estudio de Power, et al. (2013), por tarea para cada grupo de usuario.

El uso intensivo de enlaces, de zonas de contenido diferenciadas[44] y de encabezados tiene, en el caso de los usuarios ciegos, importantes implicaciones en la consecución de un web o aplicación accesible. El uso mayoritario de las estrategias de exploración y de descubrimiento denota la necesidad que presentan estos usuarios de construir modelos mentales de cada una de las secciones de una página para, a continuación, integrarlos en un modelo global que define la estructura general de esa página. A pesar de que los lectores de pantalla apoyan, en cierta medida, ambas estrategias (exploración y descubrimiento), si la estructura de las páginas es pobre o incoherente, también lo será el resultado del análisis realizado por el usuario, haciendo total o parcialmente incomprensible el contenido para él.

4.2. Ayudas técnicas

Las principales ayudas técnicas que utiliza el colectivo de ciegos para acceder a la información a través de un ordenador o dispositivo móvil son el lector de pantalla, la línea y anotador braille y el teclado estándar.

El lector de pantalla es un software que identifica e interpreta el contenido textual que se muestra en pantalla para representarlo posteriormente mediante un sintetizador de texto a voz (TTS, o *text to speech*) o una salida braille. Generalmente ofrecen atajos de teclado (*keyboard shortcuts*) permitiendo a sus usuarios prescindir del uso del ratón, un periférico que no resulta adecuado a sus características. Los lectores de pantalla permiten a sus usuarios, navegar a través del contenido web de diferentes maneras. Según la situación o el tipo de web, es posible indicar al lector de pantalla que lea todo el contenido disponible, que vaya leyendo de línea en línea, o incluso marcar el ritmo de lectura mediante alguna combinación de teclas (generalmente el tabulador) con la que es posible navegar entre elementos (enlaces, encabezados...).

Las principales limitaciones de los lectores de pantalla tienen que ver con la descripción de las imágenes, así como con el resto de contenidos no textuales, la organización de la información en la página cuando depende del diseño visual, o el acceso a la información contenida en tablas complejas. Se trata de tres limitaciones que pueden ser compensadas siguiendo recomendaciones y estándares como por ejemplo, el uso del atributo "alt" para describir el contenido de las imágenes, pero que en ningún caso pueden sustituir la experiencia de ver la imagen.

El usuario que accede mediante un lector de pantalla escucha en primer lugar el título de la página y, a continuación, el resto de los elementos presentes en el nodo según su orden de aparición en el código fuente. Los programas más avanzados permiten a los usuarios saber donde empiezan los diferentes elementos (listas, tablas, etc.), navegar por las tablas, acceder a los enlaces presentes en el contenido por orden alfabético, etc. De manera que la navegación, aunque lineal, presenta gracias a estas aplicaciones diferentes posibilidades de desplazamiento por el contenido. Una condición *sine qua non* para que los usuarios sean capaces de explotar al máximo las posibilidades de estas aplicaciones, es la correcta utilización de los diferentes

[44] Ver en el apartado 7 (WAI-ARIA) el uso de regiones.

elementos, etiquetas y atributos de los lenguajes de marcas. Una característica común en todos los lectores de pantalla es su complejidad de uso, mayor cuantas más funcionalidades presenta el software. La mayoría de usuarios de lectores de pantalla sólo utilizan una pequeña parte de las funcionalidades, combinaciones de teclas o posibilidades que ofrecen estos programas.

JAWS[45] de Freedom Scientific es el lector de pantalla promovido por la ONCE y el más utilizado en España. Se trata del lector de pantalla que año tras año aparece como primera opción entre los usuarios participantes en la encuesta anual de WebAIM. En la última de estas encuestas, realizada en enero de 2014, JAWS se posicionó como el lector de pantalla utilizado como primera opción por el 50% de usuarios, mientras que un 64% de usuarios reconocieron utilizarlo de manera habitual junto con otras alternativas. Realmente, se trata de uno de los más completos y con mayores funcionalidades.

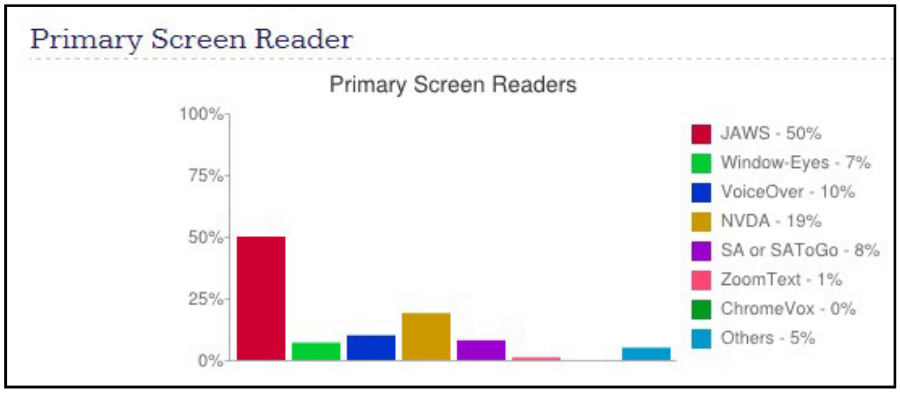


Fig. 3. JAWS es el principal lector de pantalla para el 50% de los participantes de la última encuesta de WebAIM.

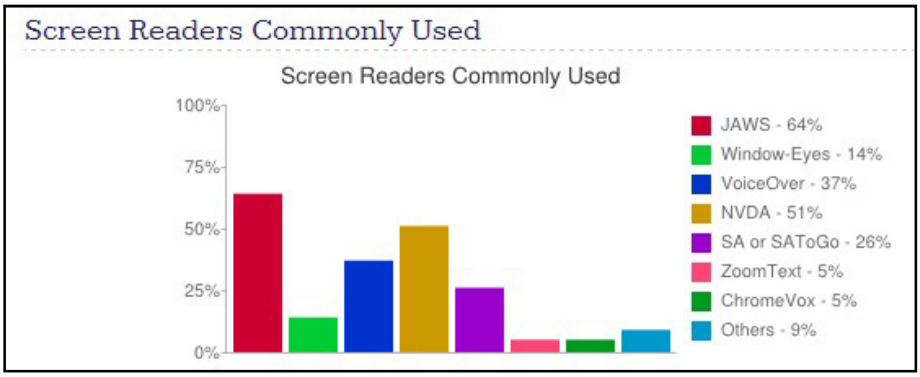


Fig. 4. JAWS es junto a VoiceOver y NVDA el lector de pantalla más utilizado en combinación con otros.

[45] *JAWS for Windows screen reading software*. http://www.freedomscientific.com/products/fs/jaws-product-page.asp>. [Consulta: 31/05/2014].

Otras alternativas según el sistema operativo son Window-Eyes[46], NVDA (NonVisual Desktop Access)[47], SAToGo (System Access to Go)[48] y Dolphin SuperNova[49] en el caso de Windows, Orca[50] para usuarios de Linux, y VoiceOver[51] en el caso de usuarios de MacOS. También se utilizan, aunque en menor medida, las extensiones de navegador Chrome Vox[52] y FireVox[53].

Una línea braille es un dispositivo electrónico que permite la salida del contenido textual presente en un dispositivo en forma de código braille. Las líneas braille muestran de forma táctil el texto que los lectores de pantalla procesarían mediante el sintetizador de voz. Como alternativa a los lectores de pantalla y su representación en forma de audio del contenido, resultan útiles tanto a las personas ciegas, como al colectivo sordo-ciego. El anotador Braille es un dispositivo con teclado Braille que permite al usuario la entrada de datos mediante el sistema Braille.

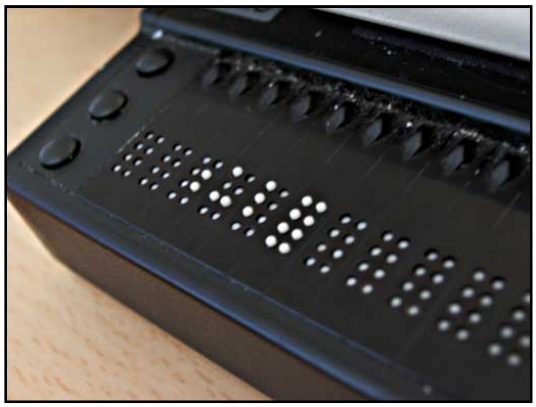


Fig. 5. Línea Braille. Fuente: *Wikimedia Commons*.

El teclado estándar es para las personas ciegas el principal periférico de entrada de datos y medio de navegación en el web. En el contexto actual, pocas páginas resultan accesibles por completo mediante el teclado. Esto se debe a diferentes razones como por ejemplo a la dependencia del ratón para ejecutar determinados eventos de JavaScript o a problemas relacionados con la imposibilidad de acceder a determinados elementos de la página mediante el teclado. Es decir, elementos que jamás reciben el foco y que por lo tanto necesitan ser seleccionados con el ratón para poder interactuar con ellos.

[46] *Window-eyes*. <http://www.windoweyesforoffice.com/>.
[47] *NV Access*. <http://www.nvaccess.org/>.
[48] *System Access to Go*. <http://www.satogo.com/>.
[49] *SuperNova*. <http://www.yourdolphin.com/productdetail.asp?id=1>.
[50] *Orca*. <https://wiki.gnome.org/action/show/Projects/Orca>.
[51] *VoiceOver para OS X*. <http://www.apple.com/es/accessibility/osx/voiceover/>.
[52] *Introducing ChromeVox*. <http://www.chromevox.com/>.
[53] *Fire Vox.* <http://www.firevox.clcworld.net/>.

4.3. Los discapacitados visuales y el acceso a la información geográfica/espacial

La independencia en la movilidad no sólo implica la capacidad de desplazarse de un punto a otro, sino que, en una visión más amplia, también supone el hecho de saber dónde estamos, dónde queremos ir y cómo podemos llegar a un punto determinado (Carreiras; Codina, 1993). La capacidad de orientación y desplazamiento en un entorno determinado requiere la presencia de una representación interna del espacio. Esta representación nos permite generar un mapa cognitivo del entorno que nos ayuda a planificar y ejecutar acciones como el desplazamiento (Downs, 1977) . Este mapa cognitivo se adquiere mediante el contacto directo con el entorno, o a través de sistemas auxiliares como mapas o descripciones verbales.

Diferentes autores han intentado sintetizar la manera en que las personas conceptualizamos el entorno geográfico. Golledge (1995) identifica cuatro inputs necesarios (identidad, localización, magnitud y tiempo) a partir de los cuales pueden derivar conceptos espaciales simples o más complejos.

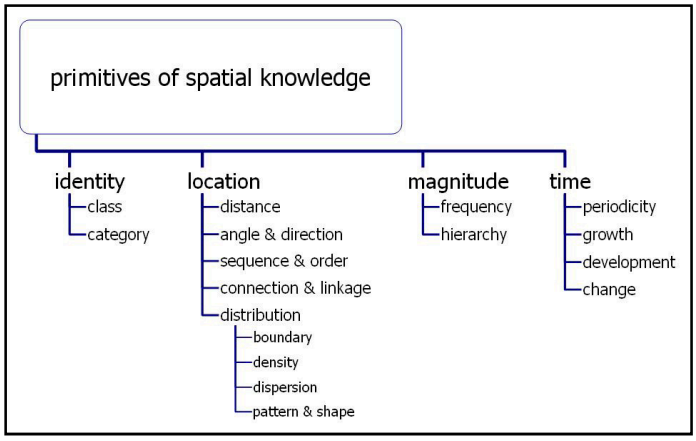


Fig. 6. Inputs necesarios para conceptualizar el entorno espacial según Golledge (1995).

Existe un importante debate en la literatura científica acerca de la capacidad de la personas ciegas congénitas (de nacimiento) para comprender el espacio (Carreiras; Codina, 1993). Algunos autores defienden la incapacidad de las personas con ceguera congénita para desarrollar la comprensión espacial, al no haber experimentado jamás los diferentes procesos de percepción necesarios (por ejemplo, el de visión) para comprender espacios bidimensionales y tridimensionales, escalas, jerarquías, etc. (Golledge, 1993). Otros, en cambio, sugieren que la capacidad de las personas ciegas para poseer representaciones espaciales es prácticamente análoga a la de los videntes (Kerr, 1983; Strelow,1985). Actualmente, la mayoría de autores sostienen la capacidad de estas personas para comprender y manipular mentalmente conceptos espaciales. No obstante, al basarse sus inputs en señales acústicas y hápticas, su nivel de comprensión resulta inferior al de las

personas que basan en la visión su capacidad para recopilar la información de carácter geográfico (Spencer, 1989). De esta manera, mediante sistemas basados en el lenguaje es posible retener relaciones espaciales dentro de codificaciones proposicionales (A está antes que B, A está más lejos de C que de B). Un sistema de información geográfica basado en la lógica proposicional podría permitir un razonamiento espacial muy sofisticado, incluyendo deducciones del tipo si sabemos que A está antes que B y que C está después de B, la ruta para llegar a C desde A pasa por B. En los últimos años se ha trabajado en dos líneas diferentes para desarrollar métodos de comunicación espacial de carácter no visual. La primera de ellas consiste en desarrollar sistemas basados en el tacto capaces de transmitir relaciones espaciales relativas. La segunda línea de investigación consiste en desarrollar sistemas basados en el lenguaje. La mayoría de las ayudas técnicas sólo buscan añadir señales ambientales a la información ya obtenida a través del mismo oído, el tacto o el olfato. Éstas van desde los simples bastones blancos, que sirven de guía a los discapacitados visuales para desplazarse de manera autónoma por la vía pública, a otras herramientas algo más sofisticadas como los *hooples*[54] o los bastones láser. A las tradicionales tecnologías de asistencia se han venido a sumar en los últimos años los llamados bastones blancos inteligentes (*smart cane*). Se trata de dispositivos que a diferencia de los bastones blancos tradicionales presentan diversas mejoras como la detección de objetos a partir de un conjunto de sensores, que permiten detectar no sólo los obstáculos por debajo de la cintura, sino también aquellos más altos como ramas, extintores, salientes, etc. De manera que si el sensor detecta algún tipo de obstáculo, el dispositivo envía una señal en forma de vibración a la muñeca del usuario. Además, también permiten adaptarse a las características físicas de cada usuario (altura, anchura de hombros, etc.)[55].

Fig. 7. Prototipo de bastón blanco inteligente desarrollado por la Universidad Miguel Hernández de Elche en colaboración con la ONCE.

[54] "Hoople (mobility aid)". En: *Wikipedia: the free encyclopedia*. Última modificación: 17 April 2012. <https://en.wikipedia.org/wiki/Hoople_(mobility_aid)>. [Consulta: 31/05/2014].

[55] El prototipo desarrollado por la Universidad Miguel Hernández de Elche en colaboración con la ONCE es un ejemplo de este tipo de tecnología <http://comunicacion.umh.es/2013/07/01/investigadores-de-la-umh-presentan-un-nuevo-baston-electronico-inteligente-para-ayudar-a-las-personas-ciegas-a-detectar-obstaculos/>. [Consulta: 31/05/2014].

Además de los *smart cane*, se han desarrollado otros dispositivos que vienen a complementar y mejorar las técnicas habituales de rastreo mediante el bastón blanco. Algunos ejemplos destacados son el Ultra-Body-Guard, CASBlip, o las iGlasses, tres dispositivos que en la línea del ejemplo anterior, mejoran la detección de obstáculos por parte de sus usuarios, especialmente aquellos que quedan por encima de la cintura o a la altura de la cabeza[56]. No obstante, ninguna de estas ayudas es capaz de proporcionar información contextual lejos de la zona de acción de la herramienta, y tampoco ningún tipo de información relativa a la orientación para su uso en la planificación de viajes (Loomis; Golledge; Klatzy, 1998). En los últimos años el desarrollo de la tecnología ha permitido la aparición de sistemas capaces de proporcionar información auxiliar a escalas más amplias a partir de bases de datos espaciales y sistemas de información geográfica (SIG). Podemos diferenciar dos grandes grupos: los que proporcionan ayudas a la orientación y la movilidad (por ejemplo, sistemas de orientación personal del tipo *talking maps*), y los que se centran en la ayuda al aprendizaje (como el sistema *NOMAD system audio-tactile graphics*). Los dispositivos de la primera categoría tienden a utilizar interfaces de lenguaje, mientras que los de la segunda son sistemas híbridos (audio y táctil). Uno de los *gadgets* más recientes en este ámbito y un buen ejemplo de convergencia de sistemas es Blind Maps[57], un dispositivo para iPhone que aprovecha la capacidad de Siri[58] para interpretar las instrucciones del usuario y las indicaciones de Google Maps para ofrecer como respuesta indicaciones de dirección mediante un patrón similar al Braille.

Fig. 8. Blind Maps para iPhone.

En algunos ámbitos como el de los museos se está aplicando eficazmente la tecnología RFID (*Radio Frequency IDentification*) para marcar puntos de interés, geolocalizar piezas museísticas, marcar rutas o activar automáticamente las audio guías. RFID se trata

[56] Para saber más sobre estas tecnologías, ver el informe del proyecto ARGUS sobre el estado del arte en el campo de la tecnología y estándares. <http://goo.gl/BCd6AV>. [Consulta: 31/05/2014].

[57] *Blindmaps*. <http://www.rubenvandervleuten.com/blindmaps.html>. [Consulta: 31/05/2014].

[58] "Siri". En: *Wikipedia: la enciclopedia libre*. Última modificación: 9 abr 2014. <http://es.wikipedia.org/wiki/Siri>. [Consulta: 31/05/2014].

propiamente de una tecnología de identificación usada mayoritariamente con fines de seguridad. No obstante, la información de potencia de las etiquetas de radio frecuencia también se puede utilizar como métrica de localización. Se trata de una tecnología interesante que también se puede aplicar como ayuda a la navegación *indoor* para personas ciegas.

Todas estas tecnologías no se han pensado para sustituir a los bastones blancos o a otras ayudas similares, sino que son herramientas complementarias. Mientras que los bastones proporcionan orientación local inmediata en un entorno limitado, el sistema de orientación personal se encarga de proporcionar macro información sobre el entorno, ayudando a la orientación, la toma de decisiones espacial, o la planificación de rutas.

5. Panorama del software y aplicaciones relacionados con la información geográfica

5.1. Sistemas de información geográfica

Burrough (1986) define los SIG como las herramientas capaces de capturar, manipular, visualizar, consultar y analizar información de carácter espacial. La definición que encontramos en *Wikipedia*[59] ofrece una visión más transversal de los SIG y además nos aproxima a su principal función. Para la enciclopedia un SIG es una integración organizada de hardware, software y datos geográficos diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión geográfica. Por último, Rodríguez y Olivella (2011) añaden el componente humano, al incluir a las personas entre los elementos indispensables en la definición de estos sistemas junto al hardware, el software y los datos.

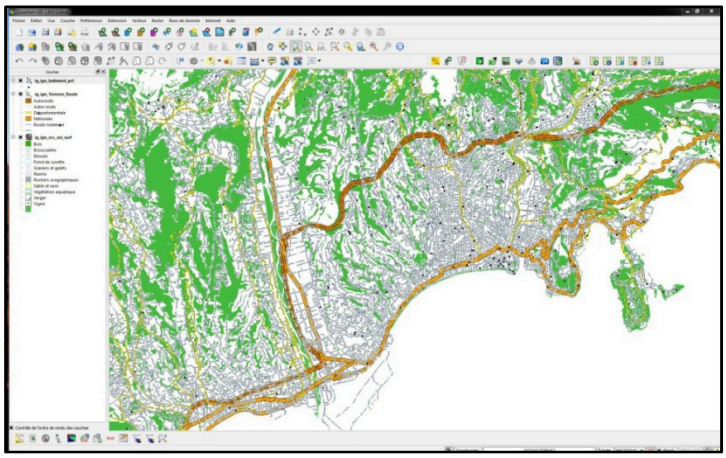


Fig. 9. Interfaz del SIG Quantum GIS.

Se trata de aplicaciones complejas y caras pensadas para las grandes agencias nacionales. No obstante, como en prácticamente todos los ámbitos, también existen algunas alternativas de software libre como Quantum GIS[60] o GRASS GIS[61] un software desarrollado por el Cuerpo de Ingenieros del Ejército de los EEUU, junto a diversas agencias federales, universidades y empresas privadas.

[59] "Sistema de información geográfica". En: *Wikipedia: la enciclopedia libre*. Última actualización: 19 abr. 2014. <http://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n_geogr%C3%A1fica>. [Consulta: 31/05/2014].
[60] *Quantum GIS*. <http://www.qgis.org/es>. [Consulta: 31/05/2014].
[61] *GRASS GIS*. <http://grass.osgeo.org/>. [Consulta: 31/05/2014].

5.2. Bases de datos geográficos

Una base de datos geográficos es una colección de datos organizados de tal manera que sirvan a las aplicaciones de sistemas de información geográfica, y permitan el almacenamiento estructurado de los datos de acuerdo a criterios espaciales, tipos de consultas y gestión de información geográfica. Son, pues, una parte indispensable del los SIG al encargarse del almacenamiento de la información que éstos manipulan. Un ejemplo de base de datos geográficos lo encontramos en Geonames[62], una base de datos gratuita, distribuida bajo licencia Creative Commons Reconocimiento que ha alcanzado una cierta notoriedad en los últimos años por su integración con diversos proyectos de la web semántica. En el ámbito público, cada país o territorio dispone de sus propias bases de datos geográficos gestionadas por sus institutos geográficos o cartográficos. Diferentes agencias o empresas privadas también almacenan y distribuyen este tipo de bases de datos.

Dentro de una base de datos geográficos podemos diferenciar entre tres tipos de entidades: los datos alfanuméricos, los datos vectoriales y los datos raster (Botella, 2011). Los datos alfanuméricos contienen información referente, por ejemplo, a la población de una ciudad. Dentro de los datos vectoriales encontramos los objetos puntuales situados en un punto concreto del espacio (un edificio, un árbol, etc.), los objetos lineales que se distribuyen sobre un territorio (un río, una carretera, etc.) y los objetos área que ocupan un espacio determinado en un territorio (el mar, un bosque, una parcela, etc.). Por lo que respecta a los datos raster, podemos diferenciar entre coberturas e imágenes raster. Las primeras son información geoespacial que representa fenómenos que varían en el espacio (calidad del aire, cantidad de nitratos del suelo, etc.), para las cuales en cada punto del territorio hay un valor diferente. Las imágenes raster son imágenes o fotografías aéreas que se sitúan sobre el territorio.

5.3. Mapas digitales

En inglés *web mapping* y traducido generalmente al español como cartografía web o mapeo web, en este trabajo nos referiremos a ellos como mapas digitales. Se trata de servicios basados en APIs o integrados en SIG que permiten a sus usuarios crear mapas digitales interactivos o aplicaciones complejas basadas en la convergencia de diferentes servicios e información de carácter geográfico. Los tres proyectos más representativos en esta línea son Google Maps, OpenStreetMap y Bing Maps.

5.3.1. Google Maps

Google Maps es un servicio en línea de Google que ofrece diferentes funcionalidades a partir de un servidor de aplicaciones de mapas web desplazables (*slippy map*). Se trata de un servicio privado de acceso gratuito que dispone, entre otras funcionalidades, de la posibilidad de visualizar fotografías por satélite de todo el mundo, obtener la ruta entre diferentes ubicaciones, o incluso imágenes a pie de calle de diferentes puntos del mundo (Google Street View). Como el resto de aplicaciones de Google, Google Maps se ha acabado integrando con

[62] *GeoNames*. <http://www.geonames.org/>. [Consulta: 31/05/2014].

otros servicios de la compañía. De esta manera, desde 2005 forma parte de Google Local[63], un servicio orientado a encontrar negocios locales y compartir con el resto de usuarios nuestra opinión sobre los mismos. A esta integración se le ha de sumar la de los perfiles o páginas de Google+ de las empresas y la de diferentes herramientas de análisis, medición, optimización o marketing, como Google Analytics, Google Webmaster Tools o Google Adwords.

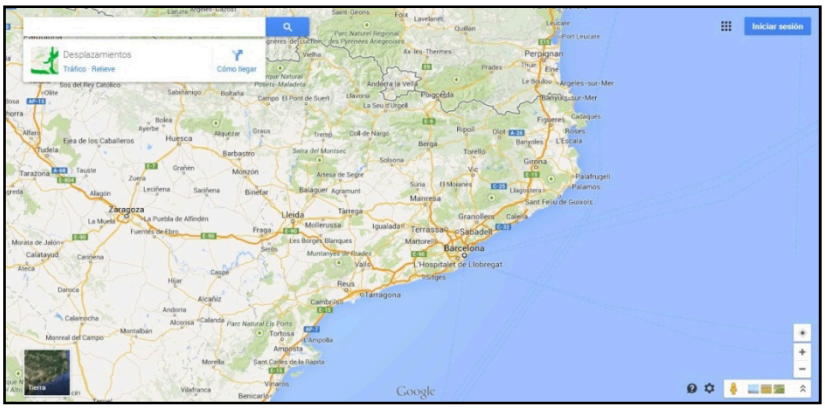


Fig. 10. Interfaz de Google Maps.

Paralelamente al servicio disponible en línea, en 2005 Google liberó su API basada en AJAX y JavaScript del servicio, permitiendo a terceros desarrolladores la creación de aplicaciones basadas en los servicios de Google Maps. Actualmente, la API se encuentra en su versión 3.

Hasta la versión 2 de la API, era necesario utilizar una clave API para poder crear mapas o aplicaciones a partir de esta interfaz de programación. Se trata de un código que se debe incluir dentro del *script* mediante el cual accedemos al servicio y que se puede solicitar de manera gratuita desde la consola de desarrolladores de Google[64]. El uso de este código permite a Google realizar el seguimiento del uso de las diferentes aplicaciones creadas. A los desarrolladores les permite controlar como utiliza su aplicación la API y les asegura que Google pueda ponerse en contacto con ellos si fuera necesario. A partir de la versión 3 de la API de Google Maps no es necesario utilizar una clave API para que nuestros mapas o aplicaciones funcionen correctamente. No obstante, desde Google recomiendan su utilización. La clave API se puede obtener de forma gratuita, pero presenta un determinado límite de uso "razonable". El límite se establece en 25.000 cargas de mapas al día, limitando el servicio sólo a aquellos sitios web que hayan excedido el límite durante más de 90 días consecutivos. Para aplicaciones

[63] *Google Local*. <http://www.google.com/+/learnmore/local/>. [Consulta: 31/05/2014].
[64] *Google developers console*. <https://code.google.com/apis/console>. [Consulta: 31/05/2014].

lucrativas que superen el límite gratuito existe una versión comercial de la API[65] que ofrece mayor capacidad y asistencia técnica entre otros servicios.

La API de Google Maps ofrece una interfaz de programación que permite desarrollar potentes servicios basados en la localización (*Located Based Services*). También es posible aprovecharse de otras librerías de Google, como la de Google Places[66], una librería de JavaScript que permite utilizar las mismas bases de datos que Google Maps y Google+ Local, incluyendo más de 80 millones de empresas verificadas. El servicio de cálculo de rutas se proporciona a partir de la API de matriz de distancia[67], capaz de devolver un servicio que proporciona el tiempo y la distancia de viaje para una matriz de orígenes y destinos. La información devuelta se basa en la ruta recomendada entre los puntos de partida y destino.

Son muchas las aplicaciones o proyectos que se han desarrollado a partir de la API de Google Maps en los últimos años. No obstante, la política de uso del producto basada en el cobro por la utilización de la API a sitios web generadores de mucho tráfico, ha provocado que empresas de referencia como Foursquare abandonaran Google Maps en favor de los datos de OpenStreetMap.

5.3.2. OpenStreetMap

OpenStreetMap (OSM)[68] es un proyecto iniciado en 2004 por Steve Coast y mantenido desde 2006 por la OpenStreetMap Foundation[69], que tiene como objetivo fomentar el crecimiento, desarrollo y distribución de datos geoespaciales libres. Se trata de un proyecto con una filosofía similar a la de *Wikipedia*, en el que cualquier persona puede registrarse y colaborar desinteresadamente en la creación de un mapamundi digital libre, de una manera similar a la que la Fundación Wikimedia intenta crear una enciclopedia libre. La relación entre ambos proyectos va más allá del hecho de compartir una filosofía casi idéntica. En 2009 se puso en marcha un proyecto de colaboración entre ambas fundaciones con el objetivo de interrelacionar ambos proyectos. De esta manera, se persigue ilustrar los artículos de la enciclopedia con mapas del proyecto OSM y permitir enlazar contenidos en ambos sentidos.

En enero de 2013 el proyecto contaba con cerca de un millón de usuarios registrados y en octubre de ese mismo año, la base de datos planet.osm[70] alcanzaba los 330 GB. Una buena parte de los contenidos de la base de datos proceden de las colaboraciones de sus usuarios, pero además, el proyecto ha contado con la colaboración en forma de cesión de datos por parte de diversas agencias o empresas como Automotive Navigation Data (con datos de los Países Bajos, India y China), TIGER (Topologically Integrated Geographic Encoding and

[65] *API de Google Maps for business*. <https://developers.google.com/maps/documentation/business/?hl=es>. [Consulta: 01/06/2014].
[66] *Google Places API*. <https://developers.google.com/places/?hl=es>. [Consulta: 31/05/2014].
[67] *El API de matriz de distancia de Google*. <https://developers.google.com/maps/documentation/distancematrix/?hl=es>. [Consulta: 31/05/2014].
[68] *OpenStreetMap*. <http://www.openstreetmap.org/>. [Consulta: 31/05/2014].
[69] *OpenStreetMap Foundation*. <http://wiki.osmfoundation.org/>. [Consulta: 31/05/2014].
[70] *Planet.osm*. <http://wiki.openstreetmap.org/wiki/Planet.osm>. [Consulta: 31/05/2014].

Referencing, oficina del censo de los Estados Unidos) o GeoBase.ca, iniciativa pública canadiense que donó todo su conjunto de datos de Canadá. Todos los datos permanecen bajo una licencia Open Database License[71] (ODL). La liberación de datos públicos por parte de instituciones gubernamentales con un tipo de licencia compatible con la ODL también ha permitido importar un gran conjunto de información geográfica al proyecto. Finalmente, primero Yahoo! hasta que estuvo disponible y actualmente Bing, han permitido a OSM hacer uso de sus fotografías aéreas, las cuales pueden ser utilizadas como base fotográfica sobre la que utilizar diferentes programas de edición para el proyecto.

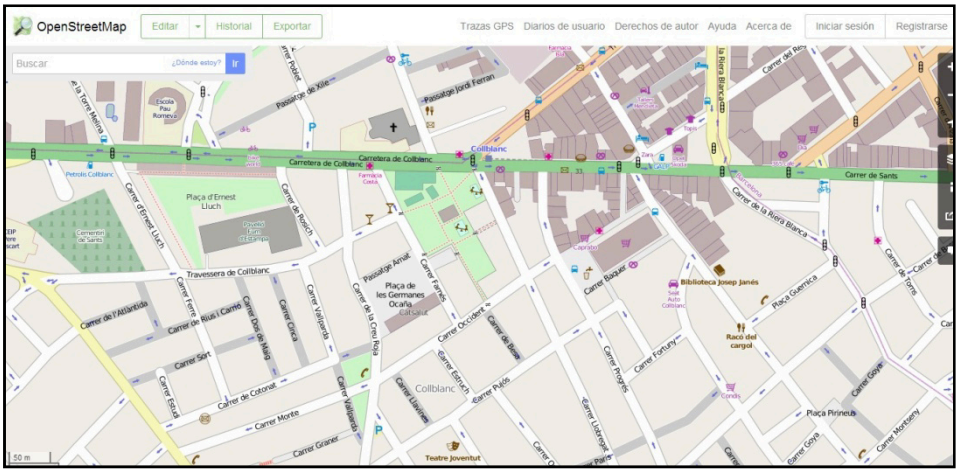


Fig. 11. Interfaz de OSM.

A diferencia de la API de Google Maps, para poder añadir mapas a una web o crear aplicaciones basadas en la API de OSM, son necesarias determinadas librerías implementadas por terceros. Las dos librerías más consolidadas y con un mayor número de desarrolladores son OpenLayers[72] y Leaflet[73], aunque existen otras alternativas, como utilizar las mismas librerías de Google Maps[74].

El cálculo de las rutas óptimas utilizando los datos de OpenStreetMap no está totalmente desarrollado, sin embargo el avance en este sentido es continuo. En numerosas regiones los datos existentes hasta la fecha todavía no son suficientemente detallados para que lleguen a ser plenamente fiables, ya que a menudo se carece de información sobre nombres de calles o información del transporte público, por ejemplo.

[71] *Open Data Commons Open Database License (ODbL)*. <http://opendatacommons.org/licenses/odbl/>.

[72] *OpenLayers: free maps for the Web*. <http://openlayers.org/>. [Consulta: 31/05/2014].

[73] *Leaflet: an open-Source JavaScript library for mobile-friendly interactive maps*. <http://leafletjs.com/>. [Consulta: 31/05/2014].

[74] En la siguiente dirección se pueden consultar todas las alternativas disponibles: <http://wiki.openstreetmap.org/wiki/Frameworks>. [Consulta: 31/05/2014].

5.3.3. Bing Maps

Bing Maps, anteriormente conocido como *Live Search Maps*, *Windows Live Maps*, *Windows Live Local* y *MSN Virtual Earth*, es a Microsoft lo que Google Maps a Google. Como en el caso de Google Maps, se trata de un servicio privado de acceso gratuito que permite a sus usuarios desplazarse a través de un mapamundi digital, obtener rutas en transporte privado, público o a pie entre dos o más puntos, visualizar fotografías por satélite o incluso imágenes a pie de calle (StreetSide[75]).

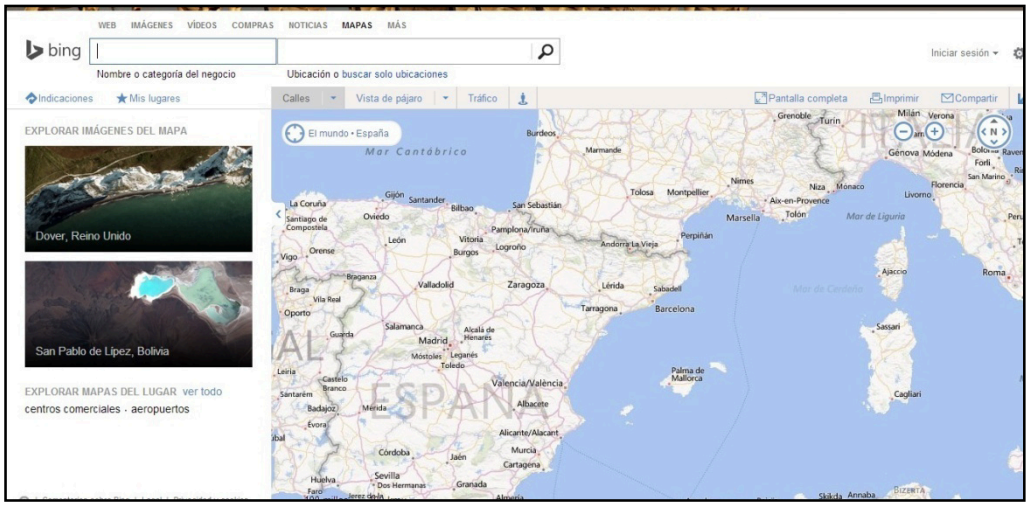


Fig. 12. Interfaz de Bing Maps.

También, como en el caso de Google Maps, dispone de manera paralela de una serie de APIs y recursos que permiten a terceros desarrolladores utilizar la tecnología de Bing Maps para crear sus propios mapas o aplicaciones. Para utilizar estas tecnologías es necesario disponer de una cuenta de desarrollador en Bing Maps Account Center[76] y solicitar una clave API[77]. El alta y la solicitud son gratuitas, pero según el tipo de aplicación que desarrollemos o de su uso previsto, es posible que debamos contratar alguno de los servicios de pago[78]. Aunque no está tan ampliamente adoptado como Google Maps, Bing Maps ofrece un conjunto de APIs equiparables a las de su competidor. La API más utilizada de Bing Maps es la basada en AJAX y

[75] *StreetSide: la experiencia más real para exploradores de cualquier lugar*. <http://www.microsoft.com/maps/es-ES/streetside.aspx>. [Consulta: 31/05/2014].

[76] *Bing Maps Account Center*. <http://www.bingmapsportal.com/>. [Consulta: 31/05/2014].

[77] "Getting a Bing Maps Key". En: *Microsoft developer network*. <http://msdn.microsoft.com/en-us/library/ff428642.aspx>. [Consulta: 31/05/2014].

[78] *Microsoft® Bing™ Maps Platform APIs' terms of use*. Last Updated: October 2013. <http://www.microsoft.com/maps/product/terms.html>. [Consulta: 31/05/2014].

JavaScript (Bing Maps AJAX Control, Version 7.0). Sin embargo, Microsoft ofrece otras APIs específicas o complementarias[79].

5.4. Servicios basados en la localización

Las búsquedas basadas en la localización son una de la funcionalidades que más expectativas despiertan entre los usuarios de Internet, especialmente entre los usuarios de dispositivos móviles (Asadi, et al., 2007). En los procesos de búsqueda y recuperación de la información habituales se devuelven al usuario documentos relevantes para su consulta a partir de la coincidencia exacta entre los términos de utilizados y el texto contenido en el documento o en sus metadatos. En el caso de las búsquedas basadas en la localización, la filosofía es intentar no sólo devolver ítems relevantes con el tema de la consulta, sino también relacionados geográficamente con la ubicación asociada al usuario. La principal dificultad para el desarrollo de esta tecnología es la falta de información geográfica codificada en las páginas web[80]. Las consultas basadas en la localización son aquellas en las que un usuario busca un determinado producto, empresa, persona o servicio en un área geográfica determinada. Este tipo de consultas se forman a partir de dos elementos: el tema de la consulta y la localización (por ejemplo, Farmacias en L'Hospitalet de Llobregat), que puede o no, ser explicita en la ecuación.

Los servicios basados en la localización (Located Based Services, LBS) son aplicaciones que ofrecen servicios personalizados en tiempo real al usuario, basándose en la localización geográfica de su dispositivo. Las operaciones básicas que permiten realizar los LBS con la información de localización son:

- **Localizar:** determinar la posición del usuario.
- **Buscar:** determinar la posición de personas, organizaciones, servicios o eventos en una determinada localización.
- **Navegar:** determinar la ruta óptima desde la localización del usuario a otra localización.
- **Consultar:** explorar las características de un lugar, una organización o un evento.
- **Encontrar:** buscar, o explorar servicios, eventos o personas cercanos a la localización del usuario.
- **Visualizar en un mapa:** ver los resultados de la búsqueda y la propia localización en un mapa, interactuar y navegar en él.

A partir de esas funciones básicas se aplican en diversos campos i situaciones (trabajo, vida personal, entretenimiento, etc.). Algunos ejemplos son diferentes servicios de información turística, localización en caso de emergencia, navegación, seguimiento de vehículos, información meteorológica local, búsqueda de servicios cercanos, búsqueda de amigos, conocidos o personas desconocidas con características afines, seguimiento de ancianos o dependientes, juegos, redes sociales o publicidad y comercio electrónico.

[79] "Getting Started with Bing Maps". En: *Microsoft developer network*. <http://msdn.microsoft.com/en-us/library/ff428643.aspx>. [Consulta: 31/05/2014].
[80] *Ibid.*

5.5. Comparativa entre Google Maps, OpenStreetMaps y Bing Maps para la creación de mapas digitales y servicios basados en la localización

Con el objetivo de encontrar la mejor alternativa tecnológica disponible para realizar un prototipo para este proyecto se incluye a continuación una tabla comparativa entre las APIs de Google Maps, OpenStreetMap y Bing Maps.

Concepto	Google Maps API v3	OpenStreetMap API v0.6	Bing Maps v7.0
Última versión	Versión 3.15 (disponible desde el 15 de noviembre de 2013)	Versión 0.6 (disponible desde el 21 de abril de 2009).	Versión 7.0 (disponible desde el 5 de diciembre de 2010)
Actualizaciones de la API	Constantes actualizaciones y mejoras requeridas tanto para el servicio <https://www.google.es/maps/> como para los clientes de Google Maps for Business.	La API no se actualiza desde hace algunos años. No obstante, continúa funcionando perfectamente.	La API no se actualiza desde hace algunos años. No obstante, continúa funcionando perfectamente.
Licencia	OEM para webs o aplicaciones cuyo acceso sea libre y gratuito. Licencia comercial de Google Maps for Bussines[81] para el resto de casos. Importante destacar que, aunque no está ejecutando esta cláusula actualmente, Google se reserva el derecho a mostrar anuncios en nuestros mapas en cualquier momento[82].	OpenStreetMap es un servicio de datos de acceso libre con licencia Creative Commons Reconocimiento-Compartir Igual 2.0 (CC BY-SA). El código fuente de las herramientas utilizadas para modificar y distribuir los datos OSM se distribuyen bajo licencia GPL. Cada librería de JavaScript relacionada con OSM tiene su propia licencia. Por ejemplo, en el caso de OpenLayers, FreeBSD.	Diferentes licencias comerciales según el tipo de proyecto (mapa en un sitio web, en una intranet privada, aplicación móvil, educativo, etc.) y según si el acceso es gratuito o de pago[83].

[81] *Maps para empresas*. <http://www.google.com/enterprise/mapsearth/>. [Consulta: 31/05/2014].

[82] Google. *Google Maps/Google Earth APIs Terms of Service* (2013). "4.3 Advertising". <https://developers.google.com/maps/terms>. [Consulta: 31/05/2014].

[83] *Bing Maps licensing options*. <http://www.microsoft.com/maps/Licensing/licensing.aspx>. [Consulta: 31/05/2014].

Concepto	Google Maps API v3	OpenStreetMap API v0.6	Bing Maps v7.0
Precio	Gratuito hasta 25.000 transacciones **diarias**.	Gratuito.	Según el tipo de licencia. Las gratuitas permiten hasta 125.000 transacciones **anuales**.
Librerías de JavaScript	La API de Google Maps incorpora su propia librería basada en AJAX y JavaScript. Existen múltiples librerías desarrolladas por terceros que facilitan el uso de la API o añaden funcionalidades o estilos propios.	Requiere el uso de una de las librerías implementadas por terceros (OpenLayers, Leaflet, Mapbox...).	La API de Bing Maps incorpora su propia librería basada en AJAX y JavaScript. Existen algunas librerías desarrolladas por terceros que facilitan el uso de la API o añaden funcionalidades o estilos propios.
Documentación	Documentación muy completa y detallada. Con muchos ejemplos de uso. Una buena parte traducida al español. El éxito de Google Maps ha provocado la publicación de millones de entradas de blog, tutoriales y varias monografías técnicas al respecto.	Documentación completa pero no tan detallada como la de Google Maps. Prácticamente no hay nada traducido al español. Existen menos publicaciones o tutoriales en Internet.	Documentación muy completa y detallada. Sólo disponible en inglés.
Servicio de rutas	Totalmente desarrollado, aunque depende de la información disponible en cada zona geográfica. En el caso de España hay una altísima cobertura[84] Las rutas a pie se encuentran en fase beta.	No se encuentra totalmente desarrollado.	Totalmente desarrollado, aunque depende de la información disponible en cada zona geográfica. En el caso de España la cobertura es alta pero no tanto como la de Google. Poca información sobre transporte público.

[84] *[Google Maps coverage]*. <http://gmaps-samples.googlecode.com/svn/trunk/mapcoverage_filtered.html>. [Consulta: 31/05/2014].

Concepto	Google Maps API v3	OpenStreetMap API v0.6	Bing Maps v7.0
Recuperación de información sobre lugares de interés	Google ofrece la librería de Google Places[85], un servicio que devuelve información sobre sitios, definidos en la API como establecimientos, ubicaciones geográficas o sitios de interés importantes mediante solicitudes HTTP. También es posible integrar información desde otros servicios y ficheros vía JSON, XML, PHP/MySQL[86], etc.	Disponemos de Xapi[87], un protocolo API de sólo lectura que permite implementar un servicio de búsqueda. También de otros métodos alternativos descritos en la sección *Databases and data access APIs*[88]. También es posible integrar información desde otros servicios y ficheros vía JSON, XML, etc. a partir de las librerías para OSM.	Microsoft ofrece Bing Maps REST Locations API[89] También es posible integrar información desde otros servicios y ficheros vía JSON, XML, etc.
Facilidad de uso	La creación de aplicaciones con la API de Google Maps, como la que nos atañe en este trabajo requiere altos conocimientos de JavaScript y otras tecnologías asociadas.	La creación de aplicaciones a partir de OSM y de alguna de sus librerías de JavaScript, como la que nos atañe en este trabajo requiere altos conocimientos de JavaScript y otras tecnologías asociadas.	La creación de aplicaciones con la API de Google Maps, como la que nos atañe en este trabajo requiere altos conocimientos de JavaScript y otras tecnologías asociadas.

Tabla 3. Comparativa entre las APIs de Google Maps, OSM y Bing Maps.

[85] "Google Places API". En: *Google developers*. <https://developers.google.com/places/documentation/?hl=es>.
[86] "Using PHP/MySQL with Google Maps". En: *Google developers*. <https://developers.google.com/maps/articles/phpsqlajax_v3?hl=es>. [Consulta: 31/05/2014].
[87] "Xapi". En: *Wiki OpenStreetMap*. <http://wiki.openstreetmap.org/wiki/Xapi>. [Consulta: 31/05/2014].
[88] "Databases and data access APIs". En: *Wiki OpenStreetMap*. <http://wiki.openstreetmap.org/wiki/Databases_and_data_access_APIs>. [Consulta: 31/05/2014].
[89] "Locations API". En: *Microsoft developer network*. <http://msdn.microsoft.com/en-us/library/ff701715.aspx>. [Consulta: 31/05/2014].

5.6. Decisión final

Para el desarrollo del prototipo de la aplicación descrito en el apartado 8 de este proyecto, he optado por el uso de las APIs de Google Maps y de Google Places. Las razones son las siguientes:

- La posibilidad de acceso a más y mejor documentación me ha facilitado tanto la creación del prototipo, como la futura creación de una versión estable.
- A pesar de tratarse de un servicio que no es completamente libre ni gratuito, las condiciones del servicio se adaptan a las necesidades del prototipo, e incluso a la creación de una futura aplicación comercial.
- El servicio de rutas de Google Maps se encuentra completamente desarrollado, a diferencia del de OSM.
- El servicio de rutas y la API de Google Places permiten crear aplicaciones de una manera relativamente más fácil de implementar, que con las tecnologías de OSM y Bing Maps.

6. Las WCAG 2.0

El W3C (World Wide Web Consortium) es un consorcio internacional que trabaja en la creación de estándares Web. Es el responsable de la creación y mantenimiento, entre otros, de los URL, el protocolo HTTP, o del lenguaje de marcas HTML. Su misión es "guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren su crecimiento futuro"[90].

La WAI (Web Accessibility Initiative), o Iniciativa para la Accesibilidad Web, es un organismo dependiente del W3C, cuyo objetivo es facilitar el acceso de las personas con discapacidad a la Web. Para ello, desarrolla estrategias, directrices y recursos para la evaluación y rediseño de webs.

Entre los diferentes recursos publicados por la WAI, destacan las *Directrices de accesibilidad para el contenido web* (*Web Content Accessibility Guidelines*, WCAG). Este documento marca las condiciones que ha de cumplir un contenido web para poder ser considerado accesible y proporciona mecanismos y estrategias para conseguirlo. Se trata del documento de referencia internacional en el campo de la accesibilidad, hasta el punto que su segunda versión, se ha llegado a convertir en una norma ISO[91].

En 2008, nueve años después de la publicación de la primera versión de las WCAG, la WAI publicó la versión 2.0 de sus directrices (WCAG 2.0). Esta nueva versión, adaptada a las tecnologías actuales, se organiza a partir de 4 principios básicos: perceptibilidad (*perceivable*), operabilidad (*operable*), comprensibilidad (*understandable*) y robustez (*robust*). A su vez, cada uno de estos principios de desarrolla a partir de diferentes directrices que proporcionan los objetivos básicos que han de guiar a los diseñadores hacia un web accesible, concretándose en una lista de criterios de conformidad. Finalmente, se establecen tres niveles de conformidad: A (el más bajo), AA y AAA (el más alto), que determinan el nivel de accesibilidad final del producto evaluado.

6.1. Las WCAG y las personas ciegas

Los cuatro principios básicos en torno a los cuales se organizan las WCAG tienen un impacto directo sobre los usuarios ciegos. Las webs, aplicaciones o documentos digitales deben ser perceptibles, operables, comprensibles y robustas para este colectivo, porque[92]:

- No son capaces de percibir (ver) información visual como gráficos, diseños, señales basadas en el color o mapas (perceptibilidad).
- Generalmente, dependen del teclado para operar y navegar a través de los contenidos (operabilidad).

[90] W3C. *Sobre el World Wide Web Consortium*. <http://www.w3c.es/Consorcio/about-w3c.html>. [Consulta: 13/10/2013].
[91] *ISO/IEC 40500:2012. Information technology, W3C Web Content Accessibility Guidelines (WCAG) 2.0.*
[92] "Visual Disabilities. Blindness". En: *WebAIM. Articles*. <http://webaim.org/articles/visual/blind>. [Consulta: 13/10/2013].

- No son capaces de comprender el contenido cuando se presenta en un orden lineal ilógico, o no pensado para ser leído palabra a palabra (comprensibilidad).
- Las tecnologías de asistencia utilizadas por este colectivo no siempre son capaces de acceder a la amplia gama de tecnologías disponibles, especialmente las más nuevas (robustez).

6.2. Las WCAG y Google Maps

Según el tipo de mapa, de los diferentes elementos y funcionalidades incorporadas, en definitiva, de la complejidad de la aplicación desarrollada mediante la API de Google Maps, resulta aplicable una mayor o menor cantidad de criterios de conformidad. A medida que añadimos nuevos elementos o funcionalidades a un mapa básico, entran en escena nuevos criterios de conformidad que se deben cumplir. Así pues, la implementación de formularios de búsqueda, listas desplegables (*combo box*), etc., requieren sus propios requisitos específicos para cumplir las directrices del W3C. En el caso de que el mapa o aplicación forme parte de una página o pantalla con más elementos, entrarán en escena otros requerimientos contemplados en las directrices (saltar bloques, etc.). Además, como veremos en el apartado 7, ciertos elementos interactivos propios de aplicaciones enriquecidas, precisan el uso de las especificaciones técnicas de WAI-ARIA.

A continuación se recogen los diferentes criterios de conformidad hasta el nivel AA, que afectan directamente al colectivo de personas ciegas y su aplicabilidad a un mapa digital o servicio basado en la localización.

6.2.1. Principio 1: Perceptible

Criterio: 1.1.1. Contenido no textual

Objetivo: El principal objetivo de este criterio es conseguir que toda la información que se transmita a través de contenido no textual pueda ser comunicada a través de una alternativa de texto. El texto es la morfología de información más manipulable (Ribera, 2009). Gracias a diferentes tecnologías o ayudas técnicas puede ser transmitido de manera eficiente a cualquier modalidad sensorial. Por ejemplo, una persona ciega puede escuchar la alternativa de texto o leerla a través de una línea braille.

Técnicas:

- Todas las imágenes y botones de imagen de los formularios tendrán un texto alternativo adecuado. Las imágenes que no transmitan contenidos, las decorativas o con el contenido ya presente como texto se ofrecerán con el texto alternativo vacío (alt="") o aplicadas como fondos de imagen CSS.
- Los botones de los formularios tendrán nombres (value) que describan su finalidad. Los elementos de los formularios tendrán etiquetas textuales (label) asociadas o, si éstas no pueden utilizarse, un título (title) descriptivo.

Nivel: A

Aplicabilidad: Sí, aplicable a los formularios de búsqueda y de obtención de rutas de la aplicación.

Ejemplos:

```
<input type="submit" name="submit" value="Buscar">
<input type="reset" name="reset" id="panelReset" value="Borrar">
```

El atributo "value" especifica el valor de los elementos "input" y "button" en los formularios. Se trata, además, de un atributo obligatorio en los inputs tipo "checkbox" y "radio".

```
<label for="modo">Medio de transporte</label>
<select id="modo" class="form-control">
      <option value="TRANSIT">En transporte público</option>
      <option value="WALKING">A pie</option>
      <option value="DRIVING">En coche</option>
</select>
```

La etiqueta "label" se debe asociar a los diferentes elementos de los formularios.

```
<input type="image" name="buscar" src="buscar.png" alt="Buscar">
```

Si utilizamos botones de imagen éstos deben tener su correspondiente texto alternativo.

Criterios: 1.2.3 y 1.2.5. Audiodescripción o alternativa multimedia / audiodescripción

Objetivo: El objetivo de estos criterios es permitir a las personas ciegas o deficientes visuales acceder al contenido visual propio de vídeos o presentaciones multimedia, a través de audiodescripciones sincronizadas.

Técnicas:

- Se debe ofrecer una descripción auditiva o textual de los vídeos grabados.

Nivel: A

Aplicabilidad: No, a no ser que se incluyan vídeos grabados.

Criterio: 1.3.1. Información y relaciones

Objetivo: Asegurar que la información y las relaciones implícitas de manera visual o auditiva no se pierdan cuando el formato de presentación cambia (al ser leído por un lector de pantalla, cambiar la hoja de estilos, etc.).

Técnicas:

- El marcado semántico se utilizará apropiadamente.
- Las etiquetas (label) textuales se asociarán con sus campos (input) correspondientes en los formularios. Los elementos de los formularios que estén relacionados se agruparán mediante fieldset/legend.

Nivel: A

Aplicabilidad: Sí, aplicable a todo el documento y especialmente a los formularios de la aplicación.

Ejemplos:

```
<form>
	<fieldset>
		<legend>Servicio de rutas</legend>
		<label for="dirSource">Origen</label>
		<input type="text" id="dirSource" class="form-control"
		placeholder="Introduce la dirección de origen"
		<label for="dirDestination">Destino</label>
		<input type="text" id="dirDestination" placeholder="Introduce la
		dirección de destino"/>
	</fieldset>
</form>
```

Relación de elementos mediante de un formulario mediante el fieldset / legend.

Criterio: 1.3.2. Orden significativo

Objetivo: Permitir a los agentes de usuario proporcionar presentaciones alternativas preservando el orden de lectura necesario para comprender el significado del contenido.

Técnicas:

- El orden de navegación (determinado por el código fuente) debe ser lógico e intuitivo.

Nivel: A

Aplicabilidad: Sí, aplicable a todo el documento.

Criterio: 1.3.3. Características sensoriales

Objetivo: Asegurar que cualquier usuario puede acceder a las instrucciones de uso del contenido, incluso aunque no sean capaces de percibir formas, tamaños, ubicaciones, etc. Las directrices no prohíben la utilización de instrucciones de uso basadas en formas, tamaños o ubicaciones (p.ej.: el botón redondo, el botón más grande o el botón de la derecha), ya que pueden ser útiles para discapacitados cognitivos. La solución es proporcionar siempre una alternativa según la necesidad.

Técnicas:

- Las instrucciones no dependerán de la forma, tamaño o ubicación visual.

Nivel: A

Aplicabilidad: Sí

Criterio: 1.4.1. Uso del color

Objetivo: El objetivo es que cualquier usuario pueda acceder a la información inherente a elementos en los cuales el color presenta un significado por sí mismo. Como en casos anteriores, las directrices no impiden el uso del color con propósitos informativos, sino que sugieren una alternativa accesible en cada caso.

Técnicas:

- El color no debe ser el único medio de transmisión del contenido o distinción de elementos visuales. Deben proporcionarse las alternativas textuales más adecuadas según el caso.

Nivel: A

Aplicabilidad: Sí, aplicable especialmente a la posible utilización del color en los marcadores para diferenciar cada uno de los tipos de lugares en el mapa o al uso de color para indicar que un enlace se abrirá en una nueva ventana.

Ejemplos:

Un problema de accesibilidad frecuente en este tipo de aplicaciones tiene que ver con el uso del color o de formas para transmitir información inherente a cada punto de interés. Es una práctica común utilizar distintos colores o formas para diferenciar entre las categorías de negocios o lugares que aparecen en el mapa en forma de marcador o POI. Si el color o la forma es la única forma mediante la cual la aplicación permite diferenciar elementos, los usuarios ciegos no serán capaces de determinar a qué categoría corresponde cada resultado. En estos

casos, debe proporcionarse una alternativa textual que permita acceder a los usuarios a este contenido.

Fig. 13. Google ofrece desde sus servidores diferentes iconos para crear los marcadores en el mapa. Si la única forma disponible para diferenciar entre, por ejemplo, un restaurante y un bar de copas es su correspondiente icono, un usuario ciego no podrá diferenciarlos.

Para evitar la confusión que puede causar la apertura de nuevas ventanas en el navegador al seguir un enlace, no basta con definir un color diferente para los enlaces que respondan de esta manera, además debemos proporcionar esa misma información de manera textual[93]

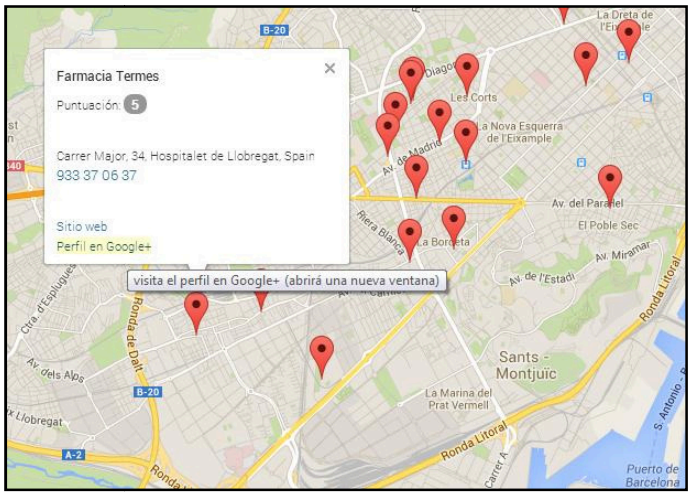


Fig. 14. En el ejemplo se informa al usuario que el objetivo del enlace es una nueva pestaña.

[93] "H83: Using the target attribute to open a new window on user request and indicating this in link text". En: *Techniques for WCAG 2.0*. <http://www.w3.org/TR/2014/NOTE-WCAG20-TECHS-20140311/H83>.

Criterio: 1.4.2. Control de sonido

Objetivo: Evitar que los usuarios no puedan escuchar la salida de voz de los lectores de pantalla debido a las interferencias de otros sonidos de fondo.

Técnicas

- Se debe ofrecer un mecanismo para poder parar, pausar, silenciar o ajustar el volumen de cualquier sonido que se reproduzca automáticamente en la página.

Nivel: A

Aplicabilidad: No, sólo si se reproducen sonidos automáticamente que puedan dificultar la comprensión del texto leído por el lector de pantalla.

6.2.2. Principio 2: Utilizable

Criterio: 2.1.1. Teclado

Objetivo: El objetivo de este criterio es asegurar que todas las funcionalidades del contenido podrán ser ejecutadas mediante una interfaz de teclado. De esta manera podrá ser operable para personas ciegas o cualquier otra que no sea capaz de utilizar periféricos como los ratones que requieren de la coordinación de ojo y mano.

Técnicas:

- Evitar el uso exclusivo de eventos de clic.
- Utilizar elementos capaces de recibir el foco del teclado.

Nivel: A

Aplicabilidad: Sí, aplicable a todo el documento.

Ejemplos:

Los mapas de Google Maps, así como los generados con otras tecnologías suelen presentar controles internos a los que no se puede acceder mediante el teclado (fig. 15). Creando controles externos capaces de acceder a las mismas funciones del mapa (zoom, movimiento, etc.), o manipulando el DOM para modificar los elementos implicados, creando otros nuevos que sí sean capaces de recibir el foco, podemos hacer que todos estos controles sean ejecutables mediante la interfaz de teclado.

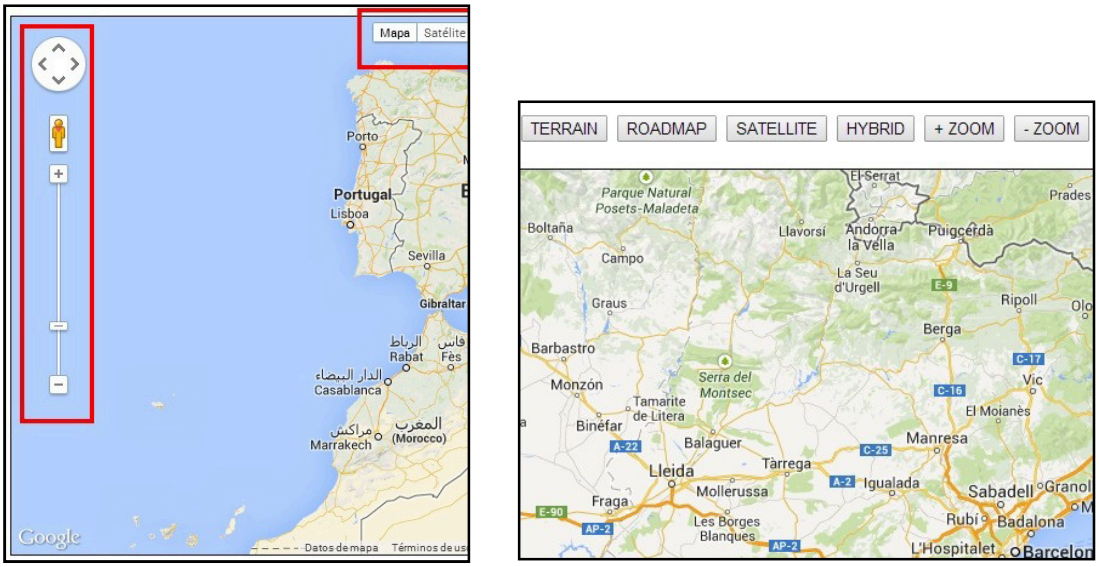


Fig. 15. A la izquierda controles internos tradicionales en un mapa creado con Google Maps. A la derecha elementos "button" creados para interactuar con el mapa.

Criterio: 2.1.2. Sin trampas para el foco del teclado

Objetivos: El objetivo de este criterio es conseguir que si en la página se permite mover el foco a un componente determinado usando una interfaz de teclado, también debe ser posible quitarlo de ese componente usando únicamente la interfaz de teclado. En el caso de que se requiera una combinación de teclas diferentes a las de dirección o tabulación, se debe informar al usuario del método apropiado para mover el foco.

Nivel: A

Aplicabilidad: Sí, aplicable a todo el documento.

Criterio: 2.2.1. Tiempo ajustable

Objetivos: Con este criterio se intenta asegurar que los usuarios ciegos, con baja visión o que tienen limitaciones cognitivas o físicas, dispongan del tiempo suficiente para poder interactuar con el contenido que se les presenta. De esta manera, si el contenido establece un límite de tiempo, el usuario debe poder: desactivar, ajustar y ampliar ese límite antes de que se agote.

Nivel: A

Aplicabilidad: No, en principio ninguna funcionalidad depende del tiempo.

Criterio: 2.2.2. Poner en pausa, detener, ocultar

Objetivo: Con este criterio se persigue evitar posibles distracciones provocadas por imágenes en movimiento, animaciones, etc., durante la interacción con la página web.

Nivel: A

Aplicabilidad: No, en principio el prototipo no incluye contenidos con esas características.

Criterio: 2.4.1. Saltar bloques

Objetivo: Permitir el acceso directo al contenido principal de una página web a los usuarios que navegan de manera secuencial a través de las diferentes páginas del portal, evitando, así, bloques de contenido secundario que se repiten en todos los nodos (menús, publicidad, destacados, etc.).

Técnicas:

- Agregar un enlace en la parte superior de cada página que dirija directamente al bloque de contenido principal.
- Utilizar elementos de encabezado al comienzo de cada sección de contenido.

Nivel: A

Aplicabilidad: En el caso de que exista una sección inicial en el documento que se repita en otras páginas del portal o aplicación. Es lo más probable si el mapa se incorpora a cualquier portal.

Criterio: 2.4.2. Títulos de página

Objetivo: Si cada página dispone de un título descriptivo, ayudaremos a los usuarios a encontrar el contenido que necesitan y a orientarse en él. El elemento título señala la ubicación al usuario sin que sea necesario que éste lea o interprete el contenido.

Nivel: A

Aplicabilidad: Sí

Criterio: 2.4.3. Orden del foco

Objetivo: Si se puede navegar secuencialmente por una página web y la secuencia de navegación afecta a su significado o su operación, los componentes que pueden recibir el foco lo deben hacer en un orden que preserve su significado y operabilidad.

Técnicas:

- Crear órdenes lógicos de tabulación a través de los enlaces, elementos de los formularios y objetos de las páginas[94].
- Utilizar el atributo "tabindex" cuando sea necesario alterar el orden por programación.

Nivel: A

Aplicabilidad: Sí, aplicable a todo el documento.

Criterio: 2.4.4. Propósito de los enlaces (en su contexto)

Objetivo: Ayudar a los usuarios a entender el propósito de los enlaces a partir del mismo texto del enlace. De esta manera, los usuarios que navegan a través de las listas de enlaces que proporcionan las ayudas técnicas no precisan del texto situado en el contexto del enlace para entender su propósito.

Técnicas:

- Proporcionar un texto en los enlaces suficientemente descriptivo.
- Todos los enlaces con el mismo destino deberían tener las mismas descripciones.
- Los enlaces con diferentes destinos deberían tener diferentes descripciones.

Nivel: A

Aplicabilidad: Sí, aplicable a todo el documento.

[94] En el apartado 8.2, se define una estructura lógica en el marco de un prototipo.

Ejemplo:

Fig. 16. Utilizando un texto suficientemente descriptivo para el enlace y proporcionando información adicional mediante el atributo "title" ayudamos a los usuarios a comprender el propósito de los enlaces.

Criterio: 2.4.6. Encabezados y etiquetas

Objetivos: Proporcionar encabezados y etiquetas suficientemente descriptivos e informativos con el objetivo de ayudar a los usuarios a comprender que información contiene la web.

Técnicas:

- Proporcionar un texto en los encabezados y etiquetas suficientemente descriptivo.
- Se debe evitar duplicar encabezados y etiquetas.

Nivel: AA

Aplicabilidad: Sí, aplicable a todo el documento.

6.2.3. Principio 3: Comprensible

Criterio: 3.1.1. Idioma de la página

Objetivo: Identificar por programación el idioma del contenido con el objetivo de que los agentes de usuario y las ayudas técnicas puedan representar correctamente el contenido.

Técnicas:

- Utilizar el atributo "lang" en el elemento "html".

Nivel: A

Aplicabilidad: Sí

Ejemplo:

```
<html lang="es">
```

El atributo lang en la etiqueta "html" especifica el idioma de la página. El valor del atributo se corresponde a los valores de la *ISO 639-1 language codes*.

Criterio: 3.1.2. Idioma de las partes

Objetivo: Identificar por programación el contenido escrito en varios idiomas para que los agentes de usuario y las ayudas técnicas cambien las reglas de pronunciación según el caso.

Técnicas:

- Codificar el idioma de las partes a nivel de párrafo o palabra.

Nivel: A

Aplicabilidad: Sí

Ejemplo:

```
<p lang="en">
<a hreflang="en">
```

El idioma de las diferentes partes del contenido, si es diferente al idioma establecido para la página, se puede identificar por programación utilizando en cualquier elemento el atributo "lang". De la misma manera, utilizando el atributo "hreflang", podemos definir el idioma de la página de destino a la que apunta un enlace.

Criterio: 3.2.1. Al recibir el foco

Objetivo: Asegurar que cualquier elemento que pueda recibir el foco no inicie ningún cambio en la página que pueda desorientar o confundir al usuario.

Nivel: A

Aplicabilidad: Sí

Criterio: 3.2.2. Al recibir entradas

Objetivo: Garantizar que la entrada de datos y la selección de controles en los formularios tenga efectos previsibles. En este sentido, se debe advertir al usuario con antelación de los cambios imprevistos o automáticos en la configuración de cualquier elemento de la interfaz que cause una modificación en la página.

Técnicas:

- Proporcionar información suficiente al usuario de manera previa a cualquier cambio producido en la interfaz.
- Supeditar todos los cambios a la pulsación de un botón suficientemente explicativo.

Nivel: A

Aplicabilidad: Sí

Ejemplo:

Un ejemplo de incumplimiento de este criterio lo entraríamos en el uso de eventos "onchange" en formularios como el de selección de medio de transporte para el cálculo de una ruta. Si el desplegable que permite seleccionar el medio de transporte deseado se asocia mediante este tipo de evento a la función utilizada para calcular la ruta, cada vez que se cambie su valor, se actualizarán las indicaciones en consonancia, pudiendo provocar confusión al usuario. En cambio, si la selección del medio de transporte depende no sólo de seleccionar el tipo de medio de transporte deseado, sino de ejecutar un botón con un texto suficientemente descriptivo ("Cómo llegar", "Obtén la ruta", etc.), los cambios en la interfaz serán previsibles para el usuario.

Criterio: 3.3.1. Identificación de errores

Objetivo: Asegurar que los usuarios sean avisados de que se han encontrado errores para que los puedan identificar y solucionar.

Técnicas:

- Se debe informar a los usuarios de los campos obligatorios de un formulario, de los valores, longitudes o formatos específicos para cada campo.
- Si se producen errores, deben estar claramente identificados y se permitirá al usuario acceder rápidamente a ellos para solucionarlos.

Nivel: A

Aplicabilidad: Sí, aplicable a los formularios de búsqueda y de obtención de rutas de la aplicación.

Ejemplo:

El uso del atributo "required" en un elemento "input" especifica que ese campo es obligatorio. Al ser compatible con los principales navegadores supone una manera efectiva de validar campos sin necesidad de JavaScript. Con este atributo, la validación se lleva a cabo de forma nativa desde el navegador en el lado del cliente (fig. 17).

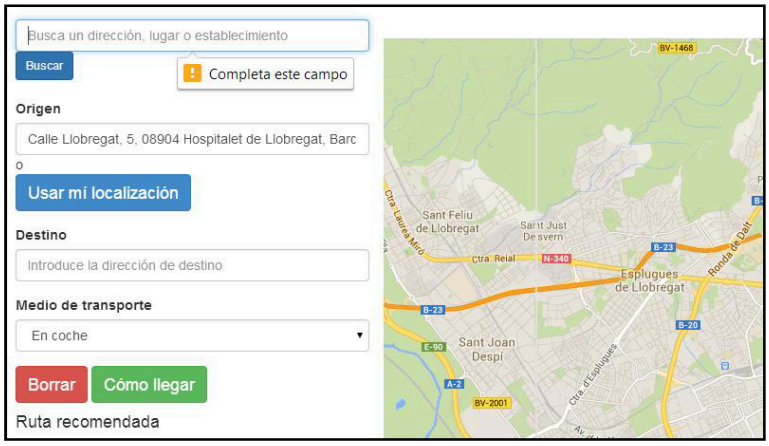


Fig. 17. Comportamiento de un elemento "input" con el atributo "required" en Google Chrome.

Criterio: 3.3.2. Etiquetas o instrucciones

Objetivo: Ayudar a los usuarios a no cometer errores en la introducción de datos, proporcionando las etiquetas, avisos o instrucciones necesarias para los elementos interactivos.

Técnicas:

- Utilizar adecuadamente las etiquetas "label" y agrupar campos mediante las etiquetas "fieldsets/legends".
- Proporcionar formatos y ejemplos del contenido esperado.
- Indicar que campos son obligatorios en un formulario.
- Utilizar WAI-ARIA para describir las acciones de un botón o asociar instrucciones a un campo del formulario (ver apartado 7).

Nivel: A

Aplicabilidad: Sí

Ejemplos:

El uso del atributo "Placeholder" en el elemento <input> permite dar pistas al usuario acerca del contenido esperado en ese campo de entrada de datos. Se puede utilizar indicando un ejemplo del contenido, o proporcionando una breve explicación (fig. 18).

```
<input id="query" type="search" placeholder="Busca una dirección, lugar o establecimiento">
```

Fig. 18. Dentro del elemento <input> se puede observar el valor del atributo "Placeholder". Un valor que el lector de pantalla leerá al usuario ciego.

Criterio: 3.3.3. Sugerencias ante errores

Objetivos: Proporcionar a los usuarios las sugerencias adecuadas para la corrección de errores en la entrada de datos.

Nivel: AA

Aplicabilidad: Sí, aplicable, por ejemplo, a los errores derivados de no tener habilitado en el navegador el seguimiento de la ubicación del usuario.

Ejemplo:

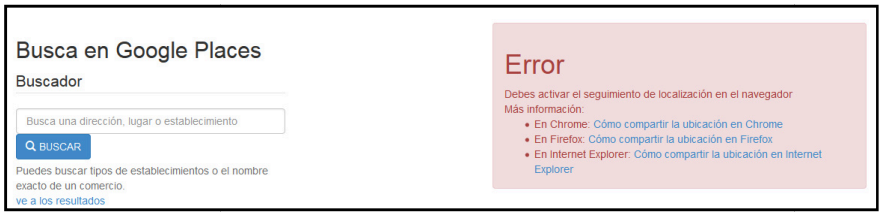


Fig. 19. Mensaje de error con ayuda adicional para el usuario.

6.2.4. Principio 4: Robustez

Criterio: 4.1.1. Análisis sintáctico

Objetivo: Asegurar que los agentes de usuario y las ayudas técnicas puedan interpretar correctamente el contenido.

Técnicas:

- Se debe evitar los errores de sintaxis HTML.

Nivel: A

Aplicabilidad: Sí, aplicable a todo el documento.

Criterio: 4.1.2. Nombre, función, valor

Objetivos: Asegurar que las ayudas técnicas puedan identificar por programación el nombre y la función de todos los componentes de la interfaz de usuario. Se debe poder configurar por programación los estados, propiedades y valores que el usuario puede ajustar.

Técnicas:

- Utilizar la sintaxis HTML de forma correcta para todos los elementos.
- Utilizar WAI-ARIA para comunicar a los lectores de pantalla información sobre la función, estados y propiedades (con sus valores) de los diferentes componentes de la interfaz (ver apartado 7).

Nivel: A

Aplicabilidad: Sí, aplicable a todo el documento.

7. WAI-ARIA

WAI-ARIA (WAI - Accessible Rich Internet Applications)[95] es una especificación del W3C que proporciona una ontología de roles, estados y propiedades que se pueden utilizar para mejorar la accesibilidad de los contenidos y aplicaciones web enriquecidas. Las *Rich Internet Applications* (RIA) son aplicaciones web ejecutadas desde un navegador, capaces de incorporar la interactividad y dinamismo característico de las aplicaciones de escritorio. En este tipo de aplicaciones, el contenido se añade, modifica o elimina sin intervención del usuario (últimas noticias en un diario, la llegada de un nuevo correo electrónico a nuestra bandeja de entrada, etc.), o como consecuencia de una acción determinada (al pulsar un botón, escoger una opción en un desplegable, etc.). Las áreas de una página web que ven modificados sus contenidos de esta manera se conocen con el nombre de *live regions* (zonas vivas). Estos nuevos contenidos pueden pasar desapercibidos para las personas ciegas o con deficiencia visual si no son marcados adecuadamente, al no ser capaz el lector de pantalla de anunciar los cambios producidos.

Los tres componentes principales de ARIA son los roles, las propiedades y los estados.

- **Roles:** Definen la función de un elemento. La mayoría de elementos HTML tienen una función predeterminada (button, form, etc.) que las tecnologías de asistencia (lectores de pantalla, etc.) son capaces de interpretar. Con ARIA, además podemos definir nuevas funciones que no están disponibles en el lenguaje HTML, o redefinir algunas de los existentes. Por ejemplo, para especificar que un elemento *form*, es en realidad un formulario de búsqueda podemos utilizar los roles de ARIA de la siguiente manera: <form role="search">

- **Propiedades:** Las propiedades nos permiten extender la capacidad semántica nativa de HTML, definiendo el significado de los elementos. Con ellas podemos definir propiedades para los elementos que no están permitidas en HTML estándar. Por ejemplo: <input aria-required="true">, permitirá a un lector de pantalla anunciar que ese campo del formulario es obligatorio.

- **Estados:** Los estados nos permiten definir la condición actual de un elemento (marcado/desmarcado, etc.). Por ejemplo: <input aria-disabled="true">, permitirá a un lector de pantalla anunciar la casilla como desmarcada.

Mediante estos componentes, WAI-ARIA nos proporciona mecanismos para transmitir a las tecnologías de asistencia información sobre la función, estados y propiedades de los diferentes elementos del DOM.

Los atributos que nos permiten identificar las *live regions* presentes en nuestra página o aplicación y definir la manera en qué se anunciarán los cambios que se produzcan en éstas son:

[95] *WAI-ARIA RDF model*. <http://www.w3.org/TR/wai-aria/rdf_model.svg>. [Consulta: 31/05/2014].

- **aria-live:** Permite identificar la *live region* y, a través de su valor, cuándo queremos que se anuncien los cambios. Los valores posibles son:
    - **aria-live="off":** Los cambios no se anuncian al usuario hasta que la región recibe el foco. Se utiliza para actualizaciones poco o nada relevantes.
    - **aria-live="polite":** Los cambios se anuncian cuando el usuario termine la tarea que está llevando a cabo (escribir en un campo, leer un contenido, etc.), sin interrumpirle.
    - **aria-live="assertive":** Los cambios se anuncian de inmediato. Se utiliza para aquellos mensajes y advertencias importantes que deben ser atendidos con prioridad (errores, resultados, etc.)
- **aria-atomic:** Permite definir si la región se anunciará por completo, o sólo las partes que se hayan visto modificadas.
    - **aria-atomic="true":** El lector de pantalla anunciará la región por completo.
    - **aria-atomic="false":** Es el valor por defecto, y anunciará sólo el nodo que ha cambiado.
- **aria-relevant:** Permite definir el tipo de actualización que se ha producido en la región. En los casos en que no se utilice este atributo, se anuncian los nodos añadidos y las modificaciones de texto.
    - **aria-relevant="additions":** Anuncia los nuevos elementos añadidos al DOM.
    - **aria-relevant="removals":** Anuncia los elementos eliminados del DOM.
    - **aria-relevant="text":** Anuncia las modificaciones en el texto.
    - **aria-relevant="all"**: Anuncia todas las modificaciones (elementos añadidos, eliminados y de texto).
- **aria-busy:** Permite detener o activar de manera temporal el aviso de una actualización, en los casos en que muchas partes de un elemento se modifiquen al mismo tiempo.

Además, mediante WAI-ARIA también podemos identificar regiones o *landmarks*, que permitirán a los usuarios de tecnologías de asistencia navegar entre ellas rápidamente. Las regiones que podemos definir con WAI-ARIA son:

- **application:** Declara que una zona no es un documento web, sino una aplicación web. De esta manera los lectores de pantalla pueden entrar en el modo específico para aplicaciones cuando la detectan.

- **banner:** La zona de la página que no tiene propiamente contenido específico de esa página, sino contenido general sobre el sitio (título, logo, etc.). Contenido que en HTML5 marcaríamos dentro del <header>.

- **complementary:** Se trata de una zona que tiene contenido complementario al contenido principal de la página, y que además sigue siendo significativa cuando se separa de éste (p.ej., la zona de artículos relacionados al final de un texto). En HTML5 estaría marcado dentro del <aside>.

- **contentinfo:** La zona de la página que contiene información complementaria sobre el documento (copyright, licencia, enlaces sobre el autor, política de cookies, etc.). Información que en HTML5 quedaría recogida en el <footer>.

- **form:** La zonas que contienen formularios, a excepción de los de búsqueda. En HTML5 lo añadiríamos a los elementos <form> o al que lo contiene.

- **main:** La región que contiene el contenido principal de la página. En HTML el elemento <main>.

- **navigation:** La región o regiones que contienen menús de navegación (listas de enlaces). En HTML5 lo tendríamos marcado como <nav>.

- **search:** Una región que contiene un conjunto de elementos pensados para crear un formulario de búsqueda. En HTML5 lo encontraríamos en el <form> correspondiente o en el que lo incluye.

Es importante destacar que WAI-ARIA sirve para mejorar, no sustituir, la sintaxis nativa disponible para cada elemento. Una sintaxis que siempre debería ser utilizada si está disponible.

7.1. WAI-ARIA y las personas ciegas

Las aplicaciones web dinámicas o enriquecidas plantean desafíos adicionales de accesibilidad a los diseñadores. Las páginas o aplicaciones web desarrolladas con AJAX son capaces de solicitar al servidor únicamente aquella información que necesitan en cada caso, sin necesidad de recargar la página por completo para mostrar aquello que el usuario precisa. Algunos ejemplos de aplicación de esta tecnología podrían ser la respuesta a una consulta o el recibimiento de un nuevo correo electrónico en la bandeja de entrada, procesos ambos que se pueden realizar en segundo plano con esta tecnología sin que el usuario lo perciba.

AJAX es una combinación de tecnologías ya existentes como HTML y CSS para el diseño de la información, el objeto XMLHttpRequest (XHR) para intercambiar datos de forma asíncrona con

el servidor de origen, el Document Object Model al que se accede mediante un lenguaje de programación (generalmente JavaScript) para mostrar e interactuar dinámicamente con la información que se presenta al usuario y XML o JSON como formatos preferentes para la transferencia de datos solicitados al servidor.

De esta manera, AJAX permite a los desarrolladores crear aplicaciones web mucho más dinámicas y ágiles al requerir menos procesamiento y menor transmisión de datos al no tener que recargar la página por completo cada vez que se producen cambios en ella. El resultado son aplicaciones más similares a las de escritorio que a las tradiciones páginas web. Los datos que el usuario envía se transmiten en segundo plano y desde el servidor mediante JavaScript se manipula la interfaz de la aplicación mostrando la nueva información de forma dinámica.

El problema con AJAX, como hemos avanzado en puntos anteriores, es su naturaleza dinámica. Mientras que en la primera versión de las WCAG, JavaScript no era bienvenido, la actual versión de las directrices del W3C sí contemplan el uso de JavaScript y AJAX, recomendando incluso estas tecnologías en diferentes técnicas para evitar la apertura de ventanas de manera no intrusiva o cambiar el estilo de una página dinámicamente, entre otras. El problema para el W3C ha dejado de ser el uso de JavaScript y se ha trasladado a la manera en cómo utilizamos esta tecnología. De esta manera, ya no es necesario proporcionar una alternativa accesible sin JavaScript para nuestras páginas o aplicaciones web, sino que lo que se espera es que el código JavaScript se accesible de manera nativa (no intrusivo, independiente del dispositivo, accesible para las tecnologías de asistencia, etc.)[96]. Sólo si utilizamos JavaScript de manera no accesible, deberemos proporcionar una alternativa al contenido sin esta tecnología[97]. Aunque la WAI desaconseja explícitamente el uso de sitios alternativos, al tratarse de soluciones no integradoras. Como dice Torres Burriel (2007)[98], "casa de dos puertas es difícil de guardar" refiriéndose además a los costes que implica esta doble estrategia. Las soluciones a los posibles problemas de accesibilidad generados por estas tecnologías han sido recogidas en dos recopilaciones de técnicas para las WCAG 2.0: *Client-side scripting techniques for WCAG 2.0*[99] y *ARIA techniques for WCAG 2.0*[100].

Para las personas ciegas y para aquellos usuarios que sufren determinadas discapacidades cognitivas, usuarios típicos de los lectores de pantalla, cuando el contenido de una página web cambia como respuesta a sus acciones, al tiempo o a determinadas actualizaciones basadas en eventos, es posible que ese nuevo contenido pase desapercibido para ellos, siendo totalmente inaccesible. Los lectores de pantalla normalmente leen de forma lineal. En el momento en que

96 En estos casos sería necesario especificar en la declaración de conformidad que nuestro web o aplicación depende de JavaScript.

97 Si no podemos ofrecer una alternativa accesible a determinadas páginas de nuestro portal o aplicaciones dependientes de JavaScript, éstas deberán quedar fuera de la declaración de conformidad.

98 Torres Burriel, Daniel. "Crónica de USID 07". *TorresBurriel.com*. 2007. <http://www.torresburriel.com/weblog/2007/05/21/cronica-de-usid-07/>. [Consulta: 31/05/2014].

99 "Client-side scripting techniques for WCAG 2.0". En: *Techniques for WCAG 2.0.* 2008. <http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/client-side-script.html>. [Consulta: 31/05/2014].

100 "ARIA Techniques for WCAG 2.0". En: *Techniques for WCAG 2.0.* 2008. <http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/aria.html>. [Consulta: 31/05/2014].

se produce un cambio en la interfaz, el usuario de un lector de pantalla puede no darse cuenta del cambio y lo más probable es que ese nuevo contenido pase desapercibido y no se lea.

La solución para que las interfaces dinámicas de las aplicaciones web enriquecidas sean accesibles pasa por alertar al usuario de los cambios acontecidos y proporcionar acceso directo a ese nuevo contenido, especificándole la prioridad o importancia según el caso. Dos técnicas que se pueden implementar gracias a WAI-ARIA. Además el uso de técnicas ARIA, también permite mejorar la accesibilidad de otros elementos como formularios, pequeños *widgets* que simulan botones, menús de navegación, etc.

7.2. WAI-ARIA y Google Maps

AJAX está presente en muchas de las funcionalidades de Google Maps. Uno de los ejemplos más claros lo encontramos en el servicio de rutas[101]. El objeto "DirectionsService" nos permite obtener diferentes rutas entre dos puntos para alguno de los cuatro medios de transporte actualmente admitidos (vehículos privados a motor, bicicleta, transporte público y a pie). El acceso al servicio de rutas se produce de forma asíncrona, al tener la API de Google Maps que realizar una llamada a un servidor externo. La solicitud se completa utilizando el objeto literal "DirectionsRequest" que contiene tanto los términos de entrada (origen y destino, medio de transporte, posibilidad de proporcionar alternativas a la ruta o no, etc.) como un método de devolución de llamada necesario para completar la solicitud y procesar los resultados. Cuando este objeto se comunica con el servicio de rutas de la API de Google Maps, éste recibe la solicitud, calcula los resultados y los devuelve mostrándolos como polilíneas que trazan una ruta en el mapa, o en forma textual, segmentadas mediante una serie de hitos dentro de un elemento . En ambos casos, la información que devuelve el servicio de rutas puede pasar desapercibida si se muestra en una zona de la página por la cual ya ha pasado el lector de pantalla y no se avisa al usuario. Se trata de un claro ejemplo de zona viva o *live region*, que se debería identificar mediante los atributos y valores correspondientes disponibles en la ontología de WAI-ARIA. Concretamente el elemento encargado de mostrar la ruta debería tener las siguientes propiedades y valores:

- **aria-live="assertive":** Anunciará los cambios producidos en la página de inmediato al usuario.
- **aria-atomic="true"**: Anunciará toda la región.
- **aria-relevant="all":** Anunciará todas las modificaciones producidas.
- Al no incluir **aria-busy**, su valor por defecto es "false".

[101] "Servicio de rutas". En: *Versión 3 del API de JavaScript de Google Maps*. <https://developers.google.com/maps/documentation/javascript/directions?hl=es>. [Consulta: 31/05/2014].

La sintaxis del elemento sería la siguiente:

```
<h2 id="ruta">Ruta recomendada</h2>
<div id="directionSteps" aria-labelledby="ruta" aria-live="assertive"
aria-atomic="true" aria-relevant="all"></div>
```

De esta manera podemos anunciarle al usuario que se ha producido un cambio en una zona dinámica de la página, en qué ha consistido el cambio y decidir cómo y cuándo queremos que le sea anunciado. Conviene destacar que en el caso de la aplicación que nos ocupa, para cumplir con el criterio de conformidad *4.1.2. Nombre, función, valor* de nivel A, de las WCAG 2.0, resulta necesario implementar esta o una técnica alternativa similar.

Otro de los elementos del prototipo que se puede beneficiar del uso de WAI-ARIA son los formularios. Un formulario consiste en una serie de elementos de entrada y de interacción (inputs, textboxes, buttons, etc.). Para asegurarnos de que las tecnologías de asistencia son capaces de interpretar la función de un elemento, con independencia de su compatibilidad en el navegador del usuario, podemos valernos de diferentes roles definidos en ARIA. Un ejemplo lo encontramos cuando queremos utilizar un elemento <input> como formulario de búsqueda. El tipo "search" para este elemento no es soportado por todos los navegadores (fig. 20)

Fig. 20. Compatibilidad del valor "search" para el atributo "type" de un elemento <input>.

Sin embargo, podemos incluir el rol "search"[102] de ARIA, en el elemento "form" o en el de su contenedor. Por ejemplo:

```
<form role="search" action="#">
      <input id="query" type="search">
      <input type="submit" name="submit" value="Buscar">
</form>
```

[102] *Accessible Rich Internet Applications (WAI-ARIA) 1.0.* "5. The Roles Model. search (role)". <http://www.w3.org/TR/wai-aria/roles#search>. [Consulta: 31/05/2014].

Para asegurarnos de que el atributo "required" funciona correctamente con el navegador Safari, así como con las versiones anteriores a Internet Explorer 11 (fig. 21), podemos definir la propiedad "required" con ARIA como se muestra a continuación.

```
<input type="text" aria-required="true">
```

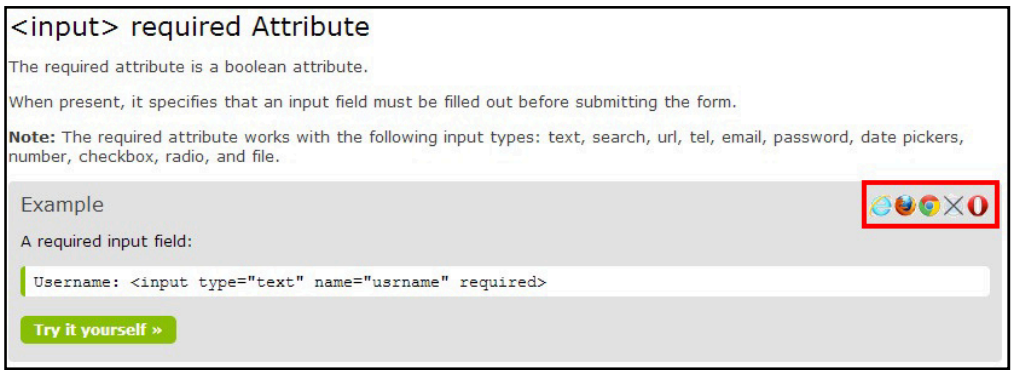


Fig. 21. Compatibilidad del atributo "required" del elemento <input>.

Carreras[103], aconseja el uso de ambos atributos para indicar que el campo es obligatorio. Además, según la misma autora, ni Chrome ni Firefox, darán la información por duplicado, mientras que Explorer sí será capaz de anunciarlo como obligatorio.

Los controles del mapa también podrían verse beneficiados del uso de WAI-ARIA. A continuación se muestra un ejemplo de implementación en un control de zoom personalizado para un mapa creado con Google Maps (fig. 22). El autor justifica la necesidad de utilizar ARIA, por no existir ningún elemento nativo en HTML para la construcción de un control con estas características. En concreto utiliza:

- El **role="slider"** para indicar que el elemento actúa como un control deslizable[104].
- **aria-orientation="vertical"** para indicar que la orientación del elemento es vertical.
- Y diferentes propiedades para el *slider* como **aria-valuemin="0"**, **aria-valuemax="17"**, **aria-valuetext="x"**, y **aria-valuenow="x"** para que el usuario pueda saber los valores mínimos y máximos permitidos, así como los que están marcados en cada momento.

[103] Carreras, Olga. " HTML5 y accesibilidad: nuevos tipos de input, atributos asociados y validación nativa". *Usable & accesible* (marzo 2014). <http://olgacarreras.blogspot.com.es/2014/03/html5-y-accesibilidad-nuevos-tipos-de.html>. [Consulta: 31/05/2014].

[104] Podemos ver otro ejemplo de control con este rol en este control creado por Paciello Group: <http://files.paciellogroup.com/blogmisc/ARIA/slider/>. [Consulta: 31/05/2014].

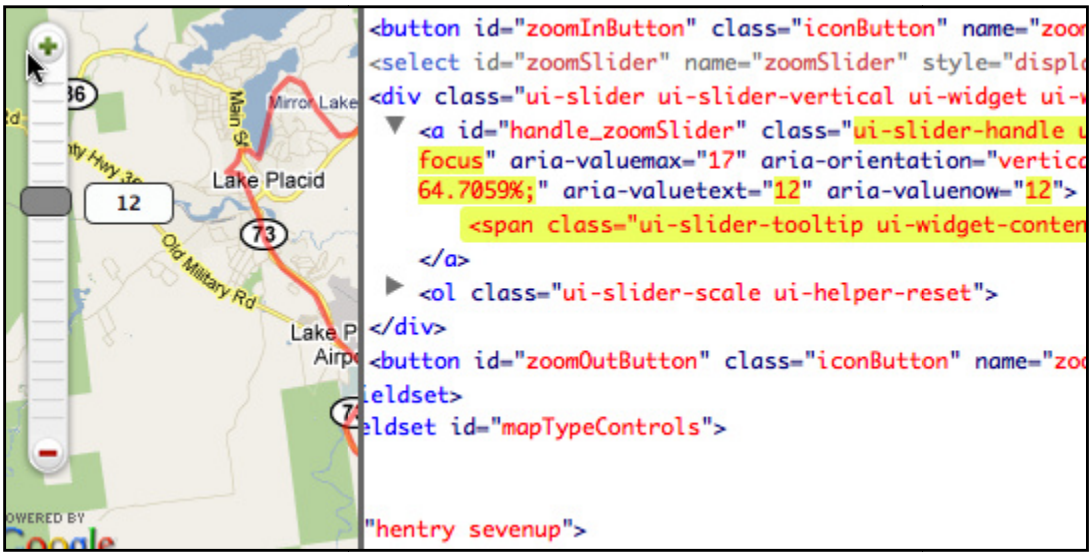


Fig. 22. aria-valuetext="x", and aria-valuenow="x" destacados en el valor 12 vistos desde Firebug.

La identificación y sugerencia ante errores, recogida en las WCAG en los criterios 3.3.1 y 3.3.3, tienen en WAI-ARIA técnicas suficientes para ser implementadas con éxito. Concretamente, utilizando el role="alertdialog" podemos crear una notificación como la que continúa:

```
<div role="alertdialog" aria-labelledby="alerta" aria-
describedby="textoAlerta">
      <h1 id="alerta">Error</h1>
      <p id="textoAlerta">Debes activar el seguimiento de localización
      en el  navegador.</p>
</div>.
```

En el ejemplo, el rol "alertdialog" se utiliza para notificar al usuario información urgente que requiere inmediatamente su atención. El atributo "aria-labelledby" da al rol "alertdialog" un nombre accesible, mientras que "aria-describedby" proporciona una referencia al texto de la alerta. Otra posibilidad consiste en utilizar el rol "alert" en la región dinámica que aparecerá con el/los mensajes de error. Por ejemplo:

```
<div role="alert" aria-atomic="true">
      <h1>Error</h1>
      <p>Debes activar el seguimiento de localización en el
      navegador.</p>
</div>.
```

En el caso de que alguna de las funcionalidades del mapa requiriera unas instrucciones adicionales para ser comprendidas, el atributo "aria-describedby" también nos permite asociarlas como se muestra a continuación:

```
<input type="text" aria-describedby="instrucciones">
<p id="instrucciones">Aquí el texto con las instrucciones.</p>
```

8. Entrevista/test con usuario

Como hemos visto en los dos apartados anteriores, actualmente disponemos de unas directrices para la accesibilidad web que nos guían en el diseño de productos digitales accesibles. En la misma línea, la especificación WAI-ARIA nos proporciona una serie de recursos en vistas a mejorar la accesibilidad de las aplicaciones enriquecidas. También hemos visto como en el ámbito normativo estas directrices se han incluido en la legislación nacional y se han convertido en normas ISO y AENOR. Todo esto está contribuyendo a que la accesibilidad se incorpore como requisito en los pliegos de condiciones de muchos proyectos web. En este contexto, algunos autores han puesto en cuestión la aceptación de la accesibilidad como el simple cumplimiento de una normativa específica, apostando por la incorporación del resto de métodos característicos del diseño centrado en el usuario en el proceso de desarrollo de un producto accesible (Sloan; Health; Hamilton, 2006). Entre estos métodos destaca el test de usuarios, una prueba que ayuda al diseñador a recuperar la perspectiva perdida después de las muchas horas dedicadas a un proyecto (Hassan; Ortega, 2009) y le permite ver de primera mano cómo interactúan los usuarios con el producto.

En este sentido, durante la elaboración del trabajo mantuve una entrevista con Santi Moese, tiflotécnico de la ONCE, con el objetivo de observar de primera mano la manera de interactuar de un usuario ciego con un mapa digital y un servicio de rutas. El encuentro tuvo como principal objetivo comprender cuáles eran las funcionalidades típicas de estos servicios que resultaban más útiles a su perfil y cuáles, a pesar de intuirse útiles, no resultaban accesibles. Además, preparé diversos prototipos que el usuario probó, ayudándome a comprender que elementos son capaces de detectar los lectores de pantalla y cuáles no, o cuál es el mejor orden para los elementos en la interfaz entre otros aspectos. La sesión resultó de gran ayuda y un complemento perfecto a las pruebas con ChromeVox y JAWS que he ido realizando a lo largo del prototipado de la aplicación. Además de este primer encuentro, el usuario ha revisado el prototipo en diferentes fases con el objetivo de ver la evolución de los cambios realizados.

A continuación se exponen las principales averiguaciones obtenidas como resultado de la entrevista y del *feedback* proporcionado por el usuario.

8.1. Averiguaciones y conclusiones

El usuario constató la utilidad de este tipo de aplicaciones tanto como sistema de posicionamiento y guía en la calle, como para la planificación de rutas desde un PC de escritorio. Si bien el usuario también había trabajado con otras aplicaciones como *Vull anar* de Transports Metropolitans de Barcelona[105] o el buscador de rutas de *Vía Michelín*[106], reconoció utilizar preferentemente Google Maps por delante del resto de alternativas. Esto se debe a que, a pesar de no ser un servicio totalmente accesible, si se trata de una aplicación mediante la cual la mayoría de usuarios ciegos obtienen un mayor nivel de satisfacción frente al resto de

[105] Disponible en: <http://www.tmb.cat/ca/vull-anar>. [Consulta: 26/05/2014].
[106] Disponible en: <http://www.viamichelin.es/web/Itinerarios>. [Consulta: 27/05/2014].

sus competidores. En este sentido, la última versión de Google Maps ha mejorado mucho, incorporando entre otros aspectos, nuevos atributos ARIA o botones que si reciben el foco (fig. 23).

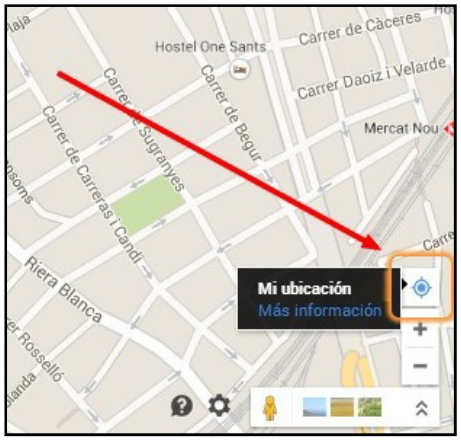


Fig. 23. Nueva interfaz de Google Maps en la que se observa uno de los controles recibiendo el foco del teclado.

Durante la sesión se propusieron diferentes tareas al usuario que tuvo que realizar en diferentes prototipos preparados para la ocasión. El conjunto de tareas propuestas fueron:

- El acceso a los controles típicos de Google Maps (cambio de tipo de mapa, zoom, etc.) mediante un grupo de controles externos ejecutables mediante teclado.
- El acceso a los controles internos del mapa modificados mediante JavaScript para recibir el foco.
- El acceso a un conjunto de marcadores y a sus respectivas ventanas emergentes con información adicional mediante una serie de enlaces que, a través de eventos de teclado, permitían situar al usuario sobre ellos.
- El uso de un servicio de rutas creado a medida para obtener las indicaciones de cómo llegar de un punto a otro de la ciudad.

El usuario fue capaz de utilizar tanto los controles externos creados con elementos <button>, como los elementos internos modificados mediante JavaScript para recibir el foco. En ambos casos, a pesar de que el lector de pantalla anunciaba los valores asociados a cada uno de los botones, el usuario no recibía la realimentación necesaria para comprender el alcance de la acción que había llevado a cabo. Es importante destacar que las acciones que permiten cambiar el tipo de mapa (satélite, callejero, híbrido, etc.), mover su centro o aumentar o disminuir el zoom, no son demasiado útiles para el usuario ciego si no tienen asociadas determinadas respuestas o funcionalidades como podría ser, por ejemplo, la actualización de los puntos de interés según el centro del mapa. El usuario tampoco fue capaz de diferenciar las dos áreas de la interfaz (zona de botones y mapa). En este sentido, la estructuración de la página en dos áreas diferenciadas mediante encabezados sería la solución.

Una de las principales dudas con respecto a la búsqueda de lugares en Google Maps era la capacidad del lector de pantalla para acceder a la información asociada a cada uno de los marcadores que ofrece el sistema en forma de ventana emergente al pulsar sobre ellos. Se trata de un contenido que es totalmente personalizable para los desarrolladores que crean aplicaciones mediante esta API y que por lo tanto resulta útil para ofrecer los detalles de cada punto de interés. Evidentemente, el usuario no puede acceder a ellos de la manera tradicional, es decir, pulsando sobre el marcador para abrir la ventana emergente, al no ser capaz de ejecutar eventos de ratón. Como solución se preparó una interfaz que permitía activar la información mediante el teclado. El resultado fue positivo, permitiendo al usuario tanto abrir la ventana emergente, como acceder a su contenido mediante el lector de pantalla.

Fig. 24. Prototipo preparado para el test en el que se observa una lista externa que activa la apertura de cada uno de los InfoWindow (bocadillos en forma de ventana emergente).

Por lo que respecta al servicio de rutas, las funciones de autocompletado en las búsquedas realizadas sobre Google Maps fueron muy bien valoradas por el usuario. Se trata de una funcionalidad que devuelve dinámicamente una lista de opciones de búsqueda que concuerdan con los términos que se van introduciendo en la ecuación de búsqueda. Esto permite al usuario evitar errores de tecleado, así como conocer la forma exacta con la que consta una dirección o lugar en el índice de Google Maps.

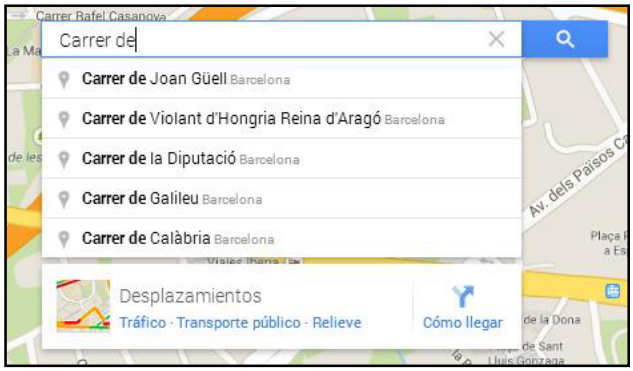


Fig. 25. Función de autocompletado en Google Maps.

El usuario fue capaz de utilizar el servicio de rutas para obtener las indicaciones entre el punto de origen y el punto de destino seleccionado. No obstante, el sistema no informó al usuario del nuevo elemento generado en la interfaz con las indicaciones correspondientes (fig. 27 y 28). Su bagaje y experiencia con interfaces similares y en el uso de JAWS, le permitió recurrir a la función de cursor virtual para actualizar la pantalla[107] y detectar los cambios. A pesar de que el mismo usuario confesó que en otras ocasiones JAWS si le había anunciado ese tipo de cambios, y que quizá en este caso se debió a problemas con la versión instalada en el PC en el que se realizó el test, resulta necesario utilizar algún tipo de tecnología que permita a cualquier usuario, con independencia del lector de pantalla que utilice o de su nivel de conocimientos sobre este tipo de software, comprender los cambios producidos en la interfaz. En este sentido, la solución que se implementó en siguientes versiones del prototipo fue la definición de "zonas vivas" mediante WAI-ARIA para anunciar los cambios generados por la aplicación. En posteriores pruebas, el usuario si fue avisado por el lector de pantalla cuando la aplicación añadía nuevos contenidos a la interfaz.

Con respecto a las diferentes opciones disponibles como medio de transporte, el usuario valoró positivamente la posibilidad de seleccionar no sólo recorridos a pie, sino también en transporte público, o incluso en transporte privado (coche, etc.), este último para planificar por ejemplo, rutas para la familia.

Las teclas de acceso rápido o atajos también fueron valoradas positivamente por el usuario, aunque no las consideró como una funcionalidad totalmente necesaria. Sí valoró positivamente la posibilidad de saltar bloques de contenido y la correcta estructuración de los contenidos en secciones. Con respecto a las teclas de acceso rápido, destacó la necesidad de que no se solaparan con otras combinaciones ya utilizadas por el lector de pantalla JAWS o el navegador.

[107] Actualizar la pantalla no equivale a refrescarla (*refresh*). La opción de refrescar vuelve a cargar el contenido, mientras que la de actualizar permite al lector de pantalla descubrir los cambios producidos en la interfaz. Si el usuario hubiera vuelto a cargar el contenido, las indicaciones habrían desaparecido.

Por lo que respecta al orden de navegación y la ubicación de los elementos en la interfaz, el usuario declaró que se trataba de una solución mucho más intuitiva y clara que la de Google Maps, que al contrario resultaba "muy caótica" según su parecer. Sin quitar méritos al trabajo realizado, como por ejemplo a la ordenación de algunos de los elementos de la interfaz, si conviene destacar que esto era previsible, dada la simplificación de la interfaz propuesta frente a la de la aplicación de Google. No obstante, en muchas ocasiones es precisamente la simplificación de la interfaz lo que hace accesibles y más usables las aplicaciones o páginas web, no sólo a este perfil de usuario, sino a la mayoría de personas.

Con respecto al orden de navegación el usuario también destacó la molestia que suponen los enlaces generados automáticamente por la API de Google Maps en la parte inferior del mapa (fig. 26). Unos enlaces que no aportan ningún valor añadido a la aplicación, pero por los que el usuario pasa al navegar por la página.

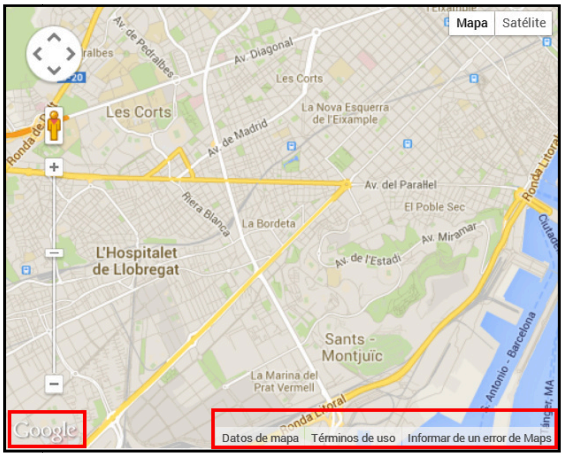


Fig. 26. En rojo, enlaces generados por defecto por la API de Google Maps.

La eliminación o alteración de estos enlaces en cualquier forma, supone un incumplimiento del punto 8.5 de la licencia de uso de Google Maps[108]. De esta manera, no es posible según los términos de la licencia de Google, ocultarlos mediante CSS o modificar el DOM para añadir un atributo "tabindex" negativo a estos enlaces para que a pesar de poder recibir el foco, no se pueda navegar hacia ellos mediante tabulación.

Uno de los principales problemas destacados por el usuario con respecto a las rutas ofrecidas por Google Maps tiene que ver con la poca especificidad en las indicaciones a pie hasta llegar a los puntos en los que el usuario debe coger los diferentes transportes públicos. Mientras que en las indicaciones de una ruta a pie se ofrece al usuario información suficiente en cada uno de

[108] 8.5 Proprietary Rights Notices. You agree that you will not remove, obscure, or alter any proprietary rights notices (including copyright and trademark notices, Terms of Use links, or Brand Features) that may be affixed to or provided through the Service. Where such notices are not affixed within the Service, you agree to display such notices according to the Maps APIs Documentation.

los pasos (fig. 27), en las indicaciones "mixtas" necesarias en la ruta con transporte público, las indicaciones a pie resultan insuficientes (fig. 28), obligándole a preguntar a otras personas para poder llegar a su destino, o a realizar una nueva búsqueda a pie entre la parada del transporte público y el punto de destino.

Fig. 27. Las indicaciones a pie resultan suficientemente específicas.

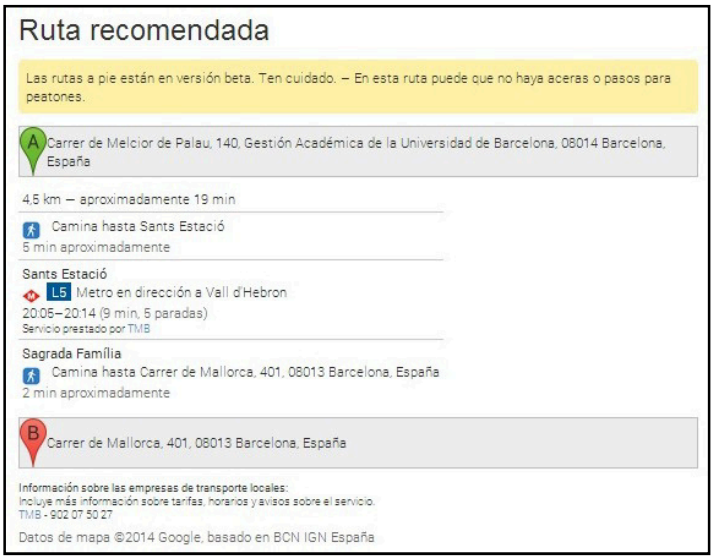


Fig. 28. Indicaciones mixtas (transporte público y a pie) para la misma ruta que la figura anterior. Las indicaciones a pie entre el punto de origen y la parada de metro de salida y las de la parada de metro de llegada al punto final de destino no son suficientemente específicas.

Se trata de un déficit de difícil solución. Algunas posibles soluciones podrían incluir la división de las rutas en varias partes o la inclusión de hitos en la ruta (las paradas de metro, el punto de destino, etc.), obligando así a las respuestas proporcionadas por el servicio a indicar la manera de llegar a éstos. En cualquier caso supone un reto considerable.

9. Prueba de concepto

La prueba de concepto que continúa consiste en la implementación de una aplicación accesible basada en las APIs de Google Maps y Google Places y en la API de geolocalización de HTML5[109] que dispone de las siguientes funcionalidades:

- Geolocaliza al usuario.
- Realiza un proceso de geocodificación inversa y devuelve al usuario la dirección formateada (calle, número, ciudad...) a partir de las coordenadas obtenidas en el proceso anterior.
- Permite realizar búsquedas de lugares, empresas o entidades sobre la base de datos de Google Places, devolviendo al usuario en forma de marcadores sobre el mapa y de lista, los resultados más relevantes (aquellos que concuerdan con los términos de la consulta y se encuentran a una determinada distancia).
- Devuelve al usuario la ruta (a pie, en transporte público o en coche) entre dos puntos especificados, el primero de los cuales puede ser la localización actual del usuario.

El objetivo de la prueba de concepto es verificar que se trata de un proyecto susceptible de ser explotado de una manera útil.

9.1. Especificaciones técnicas y análisis de referentes

9.1.1. Elementos necesarios y carga de la API

En este apartado se explican los fundamentos o elementos básicos necesarios para empezar a trabajar con la API de Google Maps: los ficheros HMTL, CSS i JS, así como la referencia necesaria y los parámetros de la API de este servicio de Google.

Para crear un mapa con la API de Google Maps, en primer lugar debemos crear un documento HTML con una estructura mínima similar a la reproducida a continuación (fig. 28).

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8">
        <link type="text/css" href="css/estilo.css" rel="stylesheet" media="all" />
        <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
        <title>GoogleMaps APIv3</title>
    </head>
    <body>
        <div id="mapa"></div>
    </body>
</html>
```

Fig. 28. Estructura mínima del documento HTML.

[109] "HTML5 Geolocation". En: *W3schools.com.* <http://www.w3schools.com/html/html5_geolocation.asp>. [Consulta: 31/05/2014].

En el ejemplo anterior y en los siguientes, utilizaremos HTML5. El primero de los elementos es la declaración de tipo de documento (DOCTYPE) y se utiliza para explicarle al navegador cómo interpretar la página. El segundo de los elementos es la declaración de idioma (<html lang="es">) en el elemento HTML, de obligatoria inclusión para cumplir con el criterio *3.1.1. Idioma de la página*, de las WCAG. El tercero de los elementos es la etiqueta <head>. Esta sección de la página contiene una serie de etiquetas muy importantes. En primer lugar encontramos la etiqueta <meta charset> que especifica qué tipo de codificación de caracteres estamos utilizando. En este caso utilizaremos UTF-8, ya que incluye caracteres especiales para todos los idiomas. A continuación podemos observar la etiqueta <title>, que establece el título de la página web. Ambos son elementos obligatorios que debemos incluir en nuestro código para poder validar el documento según los requisitos del W3C y las WCAG (criterio *2.4.2. Títulos de página*).

Para fijar el tamaño del mapa debemos establecer el estilo del elemento que contendrá el mapa. El tamaño de este contenedor (*width* y *height*) define las dimensiones del mapa. Una buena práctica consiste en mantener separados el código HTML y el CSS, por lo tanto, crearemos otro archivo con extensión .css llamado "estilo.css". Además, para mantener una estructura ordenada, crearemos una carpeta con el nombre "css" en la que guardaremos las diferentes hojas de estilo que vayamos creando. Una vez creada la hoja de estilos y definidas las propiedades del objeto, es necesario hacer una referencia en el archivo HTML apuntando al archivo CSS. Esta referencia se hace con el elemento <link> dentro de la sección <head> (línea 5, fig. 29).

La API de Google Maps se encuentra alojada en los servidores de Google. Para poder cargarla debemos hacer una referencia desde nuestro archivo HTML hacia el lugar en el que se encuentra. La referencia también se ha de incluir en la sección <head> del documento y debe ser indicada mediante un elemento <script>. Este elemento tiene dos atributos. El primero es el tipo de script que deseamos utilizar y el otro es el URL que apunta a la API (línea 6, fig. 29). Dentro de la referencia a la API encontramos diferentes parámetros opcionales que podemos alterar según nuestras necesidades:

- **sensor:** La API de Google Maps requiere que se indique si la aplicación que estamos creando utiliza algún tipo de sensor, por ejemplo, para determinar la ubicación del usuario a través de un localizador de GPS. Los posibles valores son "true" y "false". Para crear la aplicación que aquí nos ocupa utilizaremos el valor "true".

- **language:** La API de Google Maps intentará determinar automáticamente qué idioma utiliza la interfaz del usuario. No obstante, también podemos indicarlo en el <script> que hemos utilizado para referenciar la API. Para hacerlo, debemos añadir el parámetro (opcional) "&language=codigo_del_idioma" al final de la cadena.

- **region:** Al cargar la API de Google Maps desde maps.googleapis.com, se aplica un sesgo por defecto hacia los Estados Unidos. Si deseamos modificar este comportamiento predeterminado, lo podemos hacer mediante la adición de un

nuevo parámetro a la etiqueta <script>: "region". Los valores válidos para este parámetro son los de la *ISO 3166-1 alpha-2 code*[110]. El valor para España es "ES".

– **libraries:** Lo utilizamos para cargar librerías o bibliotecas adicionales. En este caso la de Google Places (&libraries=places).

```
1  <!DOCTYPE html>
2  <html lang="es">
3      <head>
4          <meta charset="utf-8">
5          <link type="text/css" href="css/estilo.css" rel="stylesheet" media="all" />
6          <script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?&sensor=true&
               language=es&region=ES&libraries=places"></script>
7          <title>GoogleMaps APIv3</title>
8      </head>
9      <body>
10         <div id="mapa"></div>
11     </body>
12 </html>
```

Fig. 29. Referencia a la API de Google Maps con los parámetros necesarios.

Ahora que ya hemos hecho la referencia a la hoja de estilos y a la API de Google Maps desde nuestro fichero HTML, ya podemos empezar a escribir código JavaScript para inicializar nuestro mapa. De la misma manera que con las hojas de estilo, también es una buena práctica separar el código JavaScript de nuestro HTML. Así que crearemos un fichero llamado "mapa.js" y lo guardaremos en una carpeta con nombre "js".

Una vez creada esa estructura, debemos referenciar el fichero "mapa.js" desde nuestra página HTML. Para ello, utilizaremos de nuevo la etiqueta <script>, pero esta vez referenciando un fichero ubicado en nuestro PC o servidor (línea 11, fig. 30).

```
1  <!DOCTYPE html>
2  <html lang="es">
3      <head>
4          <meta charset="utf-8">
5          <link type="text/css" href="css/estilo.css" rel="stylesheet" media="all" />
6          <script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?&sensor=true&
               language=es&region=ES&libraries=places"></script>
7          <title>GoogleMaps APIv3</title>
8      </head>
9      <body>
10         <div id="mapa"></div>
11         <script type="text/javascript" src="js/mapa.js"></script>
12     </body>
13 </html>
```

Fig. 30. Referencia al fichero JavaScript que contendrá la mayor parte del código de la aplicación (línea 11).

[110] "ISO 3166-1 alpha-2". En: *Wikipedia: the free encyclopedia.* Last modified: 12 May 2014. <http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2>. [Consulta: 31/05/2014].

La posición de esta nueva etiqueta no es trivial. Debe estar ubicada a continuación de la referencia a la API de Google Maps y no antes. De esta manera nos aseguramos que la API de Google Maps sea cargada antes de que la página HTML intente usarla.

Para facilitar el trabajo de programación utilizaremos jQuery[111]. jQuery es una librería de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el DOM, manejar eventos, desarrollar animaciones y agregar interacción mediante AJAX a páginas web. jQuery, al igual que otras librerías, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. Es decir, con las funciones propias de esta librería se logran grandes resultados en menos tiempo y espacio. Además, se trata de una librería con un altísimo nivel de adopción, lo que asegura su futuro mantenimiento.

Para utilizar jQuery debemos hacer una referencia desde nuestro archivo HTML hacia el lugar en el que se encuentra la librería (en un directorio en nuestro servidor o en Internet). La referencia se puede incluir en la sección <head> o en el <body> del documento mediante un elemento <script> (línea 6, fig. 31). En cualquier caso, es conveniente cargarla antes que el fichero JavaScript con el código de la aplicación.

```
1  <!DOCTYPE html>
2  <html lang="es">
3      <head>
4          <meta charset="utf-8">
5          <link type="text/css" href="css/estilo.css" rel="stylesheet" media="all" />
6          <script type="text/javascript" src="js/jquery.min.js"></script>
7          <script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?&sensor=true&
               language=es&region=ES&libraries=places"></script>
8          <title>GoogleMaps APIv3</title>
9      </head>
10     <body>
11         <div id="mapa"></div>
12         <script type="text/javascript" src="js/mapa.js"></script>
13     </body>
14 </html>
```

Fig. 31. Referencia al fichero jQuery (línea 6).

9.1.2. Inicializar un mapa

Llegados a este punto, inicializaremos un mapa dentro del que anteriormente habíamos creado en nuestra página HTML. Generalmente para crear la referencia desde nuestro fichero JavaScript se utiliza el método "document.getElementById()". Este método busca el ID de un elemento HTML (en este caso el id="mapa") y devuelve una referencia a ese mismo elemento. Es importante tener en cuenta que un mismo ID sólo puede ser utilizado una vez por página. En el caso de que no se pueda encontrar el ID indicado, el valor devuelto será "null" o lo que es lo mismo, nada. Posteriormente debemos pasar esa información al objeto "Map" (new

[111] *jQuery: write less, do more.* <http://jquery.com/>. [Consulta: 31/05/2014].

google.maps.Map(mapa, MapOptions);), junto con las propiedades del mapa que veremos en el siguiente apartado.

```
window.onload = function() {

  // Creando la referencia al DIV
  var mapDiv = document.getElementById('mapa');

  // Definiendo las coordenadas para el centro del mapa
  var catalunya = new google.maps.LatLng(41.652393,1.691895);

  // Creando un objeto literal con las propiedades del mapa
  var mapOptions = {
    center: catalunya,
    zoom: 8,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };

  // Creando el mapa
  var mapa = new google.maps.Map(mapDiv, mapOptions);
}
```

Fig. 32. Ejemplo de uso tradicional del método "document.getElementById()" y de cómo pasar las propiedades del mapa al objeto "google.maps.Map".

En nuestro caso, definiremos las propiedades del objeto "google.maps.Map" dentro del objeto mismo (fig. 33).

```
loc = new google.maps.LatLng(coords.latitude, coords.longitude);
map = new google.maps.Map(document.getElementById("map_canvas"), {
    mapTypeId: google.maps.MapTypeId.ROADMAP,
    center: loc,
    zoom: 13,
    mapTypeControl: true,
    mapTypeControlOptions: {
        style: google.maps.MapTypeControlStyle.DEFAULT,
        position: google.maps.ControlPosition.TOP_RIGHT
    },
    panControl: true,
    panControlOptions: {
        position: google.maps.ControlPosition.RIGHT_TOP
    },
    zoomControl: true,
    zoomControlOptions: {
        style: google.maps.ZoomControlStyle.LARGE,
        position: google.maps.ControlPosition.RIGHT_TOP
    },
    scaleControl: true,
    streetViewControl: true,
    overviewMapControl: true
});
```

Fig. 33. Las propiedades del mapa (línea 19-38) se definen a continuación de la referencia al ID en el que se encuentra el contenedor del mapa. Nótese que el centro se ha obtenido previamente (se explica más adelante).

9.1.3. Propiedades del mapa

El objeto "MapOptions"[112] es necesario y define las propiedades de visualización del mapa. Para utilizarlo, normalmente se crea una variable llamada "MapOptions" con las diferentes propiedades necesarias para que el mapa funcione, para pasarlas posteriormente al objeto "Map". En este caso, como hemos visto en el punto anterior, las propiedades de este objeto las hemos indicado directamente en el objeto "Map" (líneas 19-38, fig. 33).

Para que un mapa funcione se han de incluir, como mínimo, las tres propiedades siguientes:

- **center:** Define el centro del mapa mediante unas coordenadas.

- **zoom:** Define el nivel inicial de zoom del mapa. Debe ser un número comprendido entre 1 (zoom mínimo) y 23 (zoom máximo).

- **mapTypeId:** Define el tipo de mapa que será cargado inicialmente. Los diferentes tipos de mapas los encontramos en el objeto "google.maps.MapTypeId". Por ejemplo, google.maps.MapTypeId.ROADMAP o google.maps.MapTypeId.SATELLITE.

Las coordenadas para la propiedad "center" pueden facilitarse directamente como valor de la propiedad, o pueden estar almacenadas en alguna variable que hayamos definido anteriormente. También pueden obtenerse como resultado de un proceso de geolocalización (fig. 34).

```
1  <!-- Definimos el centro mediante el objeto Google.maps.LatLng-->
2
3  center: new google.maps.LatLng(41.652393,1.691895),
4
5  <!-- Creamos una variable en la cual definimos el centro mediante
6  el objeto Google.maps.LatLng para reutilizar sus valores después-->
7
8  var catalunya = new google.maps.LatLng(41.652393,1.691895);
9  // [...]
10     center: catalunya,
11
12 <!-- Obtenemos la localización del usuario mediante un proceso de
13 geolocalización para utilizarla posteriormente como centro-->
14
15 function getLocation()
16   {
17   if (navigator.geolocation)
18     {
19     navigator.geolocation.getCurrentPosition(
20         function(posicion) {
21                 var coords = posicion.coords;
22                         localizacion = new google.maps.LatLng(coords.latitude, coords.longitude);
23                         // [...]
24                             center: localizacion,
```

Fig. 34. Diferentes maneras de proporcionar u obtener las coordenadas para el centro del mapa.

[112] "google.maps.MapOptions object specification". En: *Google Maps Javascript API V3 Reference*. <https://developers.google.com/maps/documentation/javascript/reference?hl=es#MapOptions>. [Consulta: 31/05/2014].

En nuestro caso, necesitamos que el centro del mapa sea la posición actual de nuestro usuario. Para ello utilizaremos la API de geolocalización de HTML5. Además, para manejar los posibles errores utilizaremos una declaración *try-catch*. El bloque "try" (fig. 35, líneas 11-32) contiene el código que debe ser ejecutado cuando cargue la página y una función que maneja los errores que se pueden producir, por ejemplo, si el usuario no acepta compartir su posición. Mientras que el bloque "catch" (fig. 35, líneas 36-38) contiene el código que se ejecutará si se produce cualquier otro tipo de error.

```
11      try {
12          if (typeof navigator.geolocation !== 'undefined') {
13              navigator.geolocation.getCurrentPosition (
14                  function(position) {
15                      var coords = position.coords;
16
17                      loc = new google.maps.LatLng(coords.latitude, coords.longitude);
18                      map = new google.maps.Map(document.getElementById("map_canvas"), {
19                          //Aquí las propiedades del mapa
20                  },
21                  function(error) {
22                      if (error.code == 1) {
23                          $('#location-details').append('<h1 id="alerta">Error</h1><p id="textoAlerta">Debes activar
24                      } else if (error.code == 2) {
25                          $('#location-details').append('<p>No es posible determinar su localizaci&oacute;n. Asegures
26
27                      } else {
28                          $('#location-details').append('<p>No es posible determinar la localizaci&oacute;n</p>');
29                      }
30                  },
31                  {enableHighAccuracy: true}
32              );
33          } else {
34              $('#location-details').append('Tu navegador no soporta geolocalización');
35          }
36      } catch (e) {
37          alert('Se ha producido un error');
38      }
```

Fig. 35. Mediante la API de Geolocalización de HTML5 podemos obtener las coordenadas necesarias para centrar el mapa en la posición actual del usuario.

Para cumplir de manera efectiva con los criterios 3.3.1 (identificación de errores) y 3.3.3 (sugerencias ante errores) de las WCAG, podemos utilizar WAI-ARIA para anunciar los mensajes de error que aparecerán en la interfaz y así asegurarnos de que el usuario es avisado por el lector de pantalla del error. Para ello podemos utilizar el método ".setAttribute()" para agregar los atributos ARIA y sus correspondientes valores al en el que se mostrará el error.

```
var alertaAria = document.querySelector('#location-details');
alertaAria.setAttribute("aria-labelledby", "alerta");
alertaAria.setAttribute("aria-describedby", "textoAlerta");
alertaAria.setAttribute("aria-live", "assertive");
alertaAria.setAttribute("aria-atomic", "true");
```

Fig. 36. Mediante el método ".setAttribute()" añadimos nuevos atributos al elemento con id "location-details" en el caso de que se produzca algún error.

9.1.4. Propiedades de control

Los mapas que podemos crear mediante la API de Google Maps disponen de diferentes elementos que permiten a los usuarios interactuar con ellos. Estos elementos se denominan controles en la jerga de Google Maps. Por defecto, estos controles se comportan de una manera predeterminada, pero es posible alterar este comportamiento accediendo a las diferentes propiedades de control.

9.1.4.1. Propiedades de control sobre la interfaz de usuario

En este apartado se describen diferentes propiedades para el control de la interfaz de usuario.

- **Botones de tipo de mapa y zoom (disableDefaultUI):** El valor por defecto de esta propiedad es "false" o lo que es lo mismo, la interfaz de usuario (UI) por defecto se muestra si nadie dice lo contrario. Al establecer esta propiedad como verdadera ("true") podemos desactivar la interfaz. Desactivar la interfaz significa dejar de mostrar el control de zoom, el control de dirección y el selector de tipo de mapa. Como veremos más adelante, también es posible desactivar la interfaz y posteriormente activar o desactivar los diferentes controles individualmente.

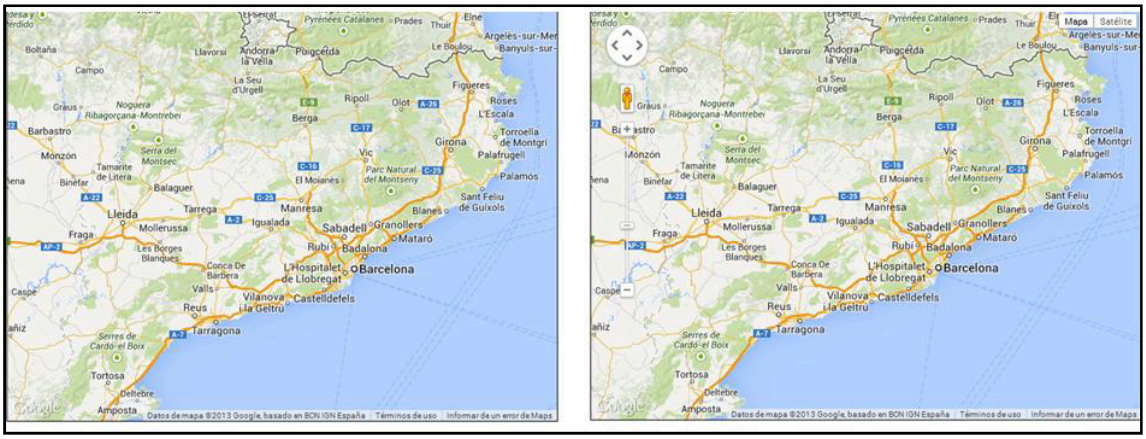


Fig. 37. A la izquierda, mapa con la interfaz de usuario desactivada. A la derecha, mapa con la interfaz de usuario clásica.

- **Botón de tipo de mapa (mapTypeControl):** Con esta propiedad se controla la aparición del "mapTypeControl". Este elemento aparece ubicado por defecto en la esquina superior derecha del mapa. Se utiliza para escoger el tipo de mapa a mostrar. El valor predeterminado es "true", es decir que si no hacemos nada el "mapTypeControl" siempre se mostrará. Si le asignamos el valor "false", dejará de mostrarse.

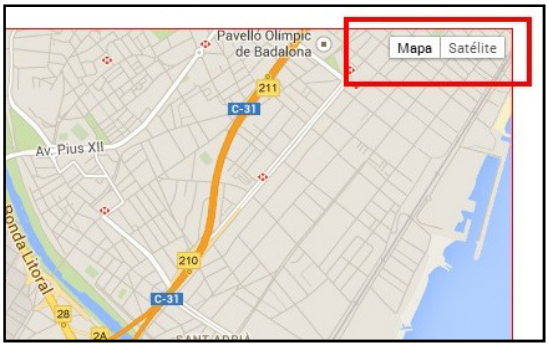


Fig. 38. En la parte superior derecha podemos observar el tradicional “MapTypeControl” de Google Maps.

- **Opciones del botón de tipo de mapa (mapTypeControlOption):** Si decidimos dejar activado el control de tipo de mapa, podemos personalizar su estilo, posición y los tipos de mapa entre los que nuestros usuarios podrán escoger. Para poder personalizar el control de tipo de mapa, es imprescindible utilizar el valor "true" para la propiedad "mapTypeControl". La propiedad "Style" determina la apariencia del objeto "mapTypeControl". Esta propiedad reside en el objeto "google.maps.MapTypeControlStyle". Los diferentes valores aplicables son:

  - **DEFAULT:** el valor DEFAULT modifica la apariencia del "mapTypeControl" según el tamaño de la pantalla y otros factores. Si la pantalla es suficientemente grande, el control se mostrará en una barra horizontal, en caso contrario, se utilizará en forma de desplegable.

  - **HORIZONTAL_BAR:** Este valor mostrará siempre nuestro "mapTypeControl" en forma de barra horizontal, por muy pequeño que sea el tamaño de la pantalla.

  - **DROPDOWN_MENU:** Esta opción muestra siempre el control en forma de desplegable.

- **Posición (google.maps.ControlPosition):** La posición por defecto del objeto "mapTypeControl" es en la parte superior derecha del mapa. Mediante la clase "google.maps.ControlPosition" es posible posicionar este control en una ubicación diferente.

Fig. 39. Posiciones disponibles en la API de Google Maps.

A pesar de poder seleccionar cualquiera de las ubicaciones mostradas en la figura anterior, no todas funcionan correctamente. Las posiciones LEFT y RIGHT se muestran en la misma zona que las posiciones TOP_LEFT y TOP_RIGHT. La posición BOTTOM_RIGHT no se muestra. Finalmente, la posición BOTTOM_LEFT no se muestra en la esquina inferior izquierda, sino que para evitar tapar el logo de Google, lo hace algo más centrada.

- **Lista de tipos de mapas (mapTypeIds):** La propiedad "mapTypeIds" controla los diferentes tipos de mapas disponibles para los usuarios. Es posible controlar que tipos de mapa aparecerán como seleccionables. Los mapas disponibles por defecto son:
  - MapTypeId.ROADMAP
  - MapTypeId.SATELLITE
  - MapTypeId.HYBRID
  - MapTypeId.TERRAIN

- **Control de zoom (zoomControl):** La herramienta de zoom se controla mediante el "zoomControl". Podemos mostrar una barra larga o un mini control más adecuado, por ejemplo, en el caso de pequeños mapas. El "zoomControl" se modifica al alterar de manera apropiada el campo "zoomControlOptions" dentro del objeto "mapOptions". El control del zoom se puede presentar según los siguientes estilos:
  - google.maps.ZoomControlStyle.SMALL
  - google.maps.ZoomControlStyle.LARGE
  - google.maps.ZoomControlStyle.DEFAULT (se adapta según el tamaño del mapa).

- **Zoom con doble clic (disableDoubleClickZoom):** Hacer doble clic dentro del mapa equivale a aumentar el zoom. Si deseamos desactivar este comportamiento por

defecto, podemos utilizar la propiedad "disableDoubleClickZoom" mediante el valor "true".

- **Zoom con rueda del ratón (scrollwheel):** Por defecto, podemos aumentar o disminuir el zoom de nuestro mapa mediante la rueda del ratón. Si queremos desactivar este comportamiento debemos establecer la propiedad "scrollwheel" como "false".

- **Control de desplazamiento (panControl):** El "panControl" es el control que nos permite mover la ventana de visualización del mapa de izquierda a derecha, de arriba a abajo y viceversa, y que aparece por defecto en la esquina superior izquierda. Mediante las "panControlOptions", podemos ubicar este control en otras posiciones.

- **Control de desplazamiento con el cursor (draggable):** Por defecto podemos movernos por el mapa arrastrando el cursor. Si por alguna razón deseamos desactivar este comportamiento debemos establecer como "false" la propiedad "draggable".

- **Teclado (keyboardShortcuts):** Esta propiedad permite activar o desactivar la posibilidad de navegar por el mapa con el teclado. Los atajos de teclado disponibles por defecto para navegar son las cuatro teclas de dirección y las teclas +/- para el zoom. El valor por defecto es true.

- **Vista a pie de calle (streetViewControl):** La propiedad "streetViewControl" permite activar y desactivar la funcionalidad *Street View* en nuestro mapa. También podemos modificar la posición de este control de la misma manera que en casos anteriores mediante la clase "google.maps.ControlPosition" del objeto "streetViewControlOptions".

```
        center: localizacion,
        zoom: 8,
        zoomControl: true,
            zoomControlOptions: {
                style: google.maps.ZoomControlStyle.SMALL
        },
        keyboardShortcuts: true,
        disableDoubleClickZoom: true,
        draggable: true,
        scrollwheel: false,
        panControl: true,
        panControlOptions: {
        position: google.maps.ControlPosition.TOP_RIGHT
        },
        streetViewControl: true,
            streetViewControlOptions: {
                position: google.maps.ControlPosition.BOTTOM_CENTER
            },
        mapTypeControl: true,
            mapTypeControlOptions: {
                style: google.maps.MapTypeControlStyle.DROPDOWN_MENU,
                position: google.maps.ControlPosition.TOP_CENTER,
                mapTypeIds: [
                    google.maps.MapTypeId.ROADMAP,
                    google.maps.MapTypeId.HYBRID
                ]
    }
```

Fig. 40. Resumen de las propiedades de control vistas.

9.1.5. Accesibilidad de los controles de la interfaz

La imposibilidad de utilizar los diferentes controles del mapa mediante el teclado provoca que la mayoría de productos creados con la API de Google Maps no sean accesibles. Para poder cumplir con el criterio "2.1.1. Teclado" de las WCAG, debemos solucionar este déficit por defecto de la aplicación. El problema radica en que los controles del mapa están creados con elementos , que no pueden recibir el foco y que aunque lo recibieran tampoco sería posible interactuar con ellos. Existen varias alternativas para solucionar este problema de accesibilidad. La primera de ellas consiste en sacar fuera del mapa los diferentes controles de la interfaz, creando nuevos elementos interactivos que sí puedan captar el foco (buttons, etc.), ubicados en el código HTML. Otra de las opciones pasa por manipular el DOM[113] mediante JavaScript para insertar en el código HTML generado por Google Maps, es decir, dentro del mismo mapa, elementos que sí capten el foco en lugar de los creados por defecto. Cabe destacar que en el proyecto que nos ocupa, teniendo en cuenta las funcionalidades del prototipo y las de los diferentes botones de la interfaz implicados (tipo de mapa, zoom, control del panorama, etc.), el uso de estos controles no interviene directamente en ninguna de las tareas que pueda llevar a cabo un usuario ciego. Por ejemplo, el hecho de cambiar del tipo de mapa *Roadmap* al tipo de mapa *Terrain*, o el hecho de aplicar más o menos zoom al mapa, resulta irrelevante para este tipo de usuario. En cambio, si en posteriores desarrollos se añaden otras funcionalidades como por ejemplo, la actualización automática de los puntos de interés a partir de la localización del centro del mapa o según el nivel de zoom determinado por el usuario, acciones ambas que dependen de que el usuario ciego sea capaz de utilizar los controles de movimiento sin el ratón, si estaría justificado este trabajo extra. No obstante, el hecho de crear controles accesibles mejora la accesibilidad general de la aplicación y permite acceder a estos controles a usuarios que sin ser ciegos, sí padecen otro tipo de discapacidades como puedes ser las motrices. Teniendo en cuenta que uno de los principios de la accesibilidad es crear aplicaciones válidas para todo tipo de colectivos (con o sin discapacidades) y que como funcionalidad podrá ser aprovechada en futuros desarrollos, se ha optado por implementarla en el prototipo.

Como hemos avanzado, existen dos formas diferentes de solucionar este problema de accesibilidad: crear elementos HTML capaces de acceder a los controles del mapa o conseguir que los controles internos del mapa reciban el foco. La primera de las opciones se puede implementar de diferentes maneras según el caso. Para crear botones mediante los cuales poder hacer un cambio de tipo de mapa, podemos asociar eventos de teclado (onkeypress) y de clic (onclick) a ese botón. Gracias a estos dos eventos tanto los usuarios de teclado, como los de ratón, podrán utilizar el control. El método "setMapTypeId()" es el encargado de llamar a los diferentes tipos de mapas disponibles (fig. 41).

[113] El DOM (*document object model*) es una interfaz de programación de aplicaciones (API) que permite manipular los objetos y atributos de la página HTML.

```
<button onkeypress="map.setMapTypeId(google.maps.MapTypeId.TERRAIN);"
onclick="map.setMapTypeId(google.maps.MapTypeId.TERRAIN);">TERRAIN</button>
```

Fig. 41. Botón en el código HTML para cambiar el tipo de mapa.

En el caso del control de zoom, necesitamos trabajar directamente sobre el fichero JS. En primer lugar debemos utilizar el método "map.getZoom();" para que la aplicación pueda determinar cuál es el nivel de zoom actual en el mapa. El siguiente paso es crear otras dos funciones a las que llamaremos "subirZoom" y "bajarZoom", que cogerán como argumento el "nivel" actual de zoom. Las variables "maszoom" y "menoszoom" almacenarán el valor resultado del método "map.getZoom()" (es decir, el zoom actual del mapa) y le sumarán el valor del argumento "nivel" de sendas funciones. Para que funcione correctamente, en el código HTML debemos indicar un valor para el argumento "nivel". Este valor determinará el incremento o disminución del nivel de zoom. Así, por ejemplo, si el valor del argumento "nivel" es 1, cada vez que pulsemos sobre el botón +Zoom o -Zoom la función se encargará se subir o bajar el nivel del zoom en un punto respecto al valor del zoom actual del mapa (fig. 42).

```
function Zoomactual() {
    return map.getZoom();
}
function subirZOOM(nivel) {
    var maszoom = Zoomactual() + nivel;
    map.setZoom(maszoom);
}
function bajarZOOM(nivel) {
    var menoszoom = Zoomactual() - nivel;
    map.setZoom(menoszoom);
}
```

Fig. 42. map.getZoom() obtiene el nivel de zoom actual, y las funciones "subirZOOM" y "bajarZOOM" suman o restan respectivamente, el valor del argumento "nivel" dinámicamente.

```
<button onkeypress="subirZOOM(1)" onclick="subirZOOM(1)">+ ZOOM</button>
<button onkeypress="bajarZOOM(1)" onclick="bajarZOOM(1)">- ZOOM</button>
```

Fig. 43. Botones para los controles de zoom en el documento HTML.

Para conseguir que los controles internos reciban el foco podemos acceder a los diferentes elementos del mapa a través del método " getElementsByTagName()", para posteriormente crear dentro de ellos elementos "button" con el método " createElement()" para el objeto "document". También podemos definir los atributos de nuestro nuevo elemento (id, style, tabindex, etc.) mediante el método "setAttribute()". Finalmente, añadimos el nuevo elemento con el método "appendChild()". Para que el mapa pueda ser cargado antes de la función, ésta debe llamarse mediante el método "setTimeout" (fig. 44).

```
setTimeout('gmaps_teclado();',500);

function gmaps_teclado() {
    var capaAccesible = 'map_canvas' // crear una capa sobre el objeto "mapa"
    var button, button_estilo = 'width:100%;height:100%;padding:2px;margin:2px;background-
    var divs = document.getElementById(capaAccesible).getElementsByTagName('div');
    for (var i = 0; i < (divs.length); i++) {
        if ( divs[i].getAttribute('title')) {
            // Crear el elemento botón
            button = document.createElement("button");
            button.setAttribute('id','boton-mapa-'+i);
            button.setAttribute('tabindex',0+i);
            button.setAttribute('value',divs[i].getAttribute('title'));

            // estilos...
            button.setAttribute('style',button_estilo); // Para la mayoría de navegadores
            button.style.cssText = button_estilo; // Para IE

            // Se añade el button
            divs[i].appendChild(button);
        }
    }
```

Fig. 44. Función para modificar el DOM y crear controles accesibles internos[114]

Fig. 45. Resultado visto a través de la consola de Chrome.

En el *Mapa accesible* de Discapnet (fig. 48) encontramos otro ejemplo diferente de mapa en el que también se han creado controles externos accesibles. En este caso para el control del zoom y el desplazamiento horizontal, vertical y en diagonal se ha utilizado una misma función (fig. 46). La función almacena en sendas variables ("centro" y "marco"), los valores de latitud/longitud del centro y los del suroeste y el noreste, mediante los métodos ".getCenter()" y ".getBounds()" del objeto "map". Mediante una estructura condicional *if*, se suma o resta

[114] Adaptado de: "Google Maps: accesibilidad de teclado y alternativa en ausencia de Javascript". *Outbook*. <http://blog.outbook.es/desarrollo-web/google-maps-accesibilidad-de-teclado-y-alternativa-en-ausencia-de-javascript>. [Consulta: 31/05/2014].

según el caso el valor asignado a la función en los respectivos botones creados en el código HTML (p. ej. "mueveMapa(2,0)") (fig. 47) al valor actual del zoom ("currentZoom").

```
function mueveMapa(x, y) {
    var centro = map.getCenter();
    var marco = map.getBounds();
    var currentZoom = map.getZoom();
    if (x == 2) {
        currentZoom++;
        map.setZoom(currentZoom);
    }
    else if (x == 3) {
        currentZoom--;
        map.setZoom(currentZoom);
    }
    else {
        sw = marco.getSouthWest();
        ne = marco.getNorthEast();
        desplazamientoVertical = (ne.lat() - sw.lat()) / 3;
        desplazamientoHorizontal = (sw.lng() - ne.lng()) / 3;

        var latlng = new google.maps.LatLng(centro.lat() + desplazamientoVertical * y, centro.lng() +
            desplazamientoHorizontal * x);
        map.setCenter(latlng);
    }
    return false;
}
```

Fig. 46. Función utilizada en Discapnet para el control de zoom y movimiento del mapa.

Para lograr el desplazamiento se almacenan en dos variables diferentes los valores del suroeste y el noreste ("sw" y "ne"), a los cuales se les aplica la siguiente fórmula "latitud/longitud del noreste, menos latitud/longitud del suroeste, entre 3 y se almacenan en otras dos variables ("desplazamientoVertical" y "desplazamientoHorizontal"). En una nueva variable ("latlng") se suman los valores del centro más los de las variables "desplazamientoVertical" y "desplazamientoHorizontal", multiplicadas por el valor asignado a la función en la etiqueta "input" correspondiente en el código HMTL ("mueveMapa (1,1)") (fig. 47).

```
// Botones para el control de zoom
<input type="button" class="ampliar" value="Ampliar" id="btnZoomIn" onclick="return mueveMapa(2, 0);">
<input type="button" class="reducir" value="Reducir" id="btnZoomOut" onclick="return mueveMapa(3, 0);">

// Uno de los botones para el movimiento
<input type="image" id="btnArribaIzquierda" src="img/esqSupIzq.png" alt="Arriba izquierda"
onclick="return mueveMapa(1, 1);" onmouseover="this.src='img/esqSupIzqSelect.png'"
onmouseout="this.src='img/esqSupIzq.png'">
```

Fig. 47. Botones en el código HTML para los controles de zoom y movimiento.

Fig. 48. Mapa con el control de movimiento, zoom y tipos de mapa accesible mediante teclado de Discapnet (http://mapaccesible.discapnet.es/).

9.1.6. Google Places: búsqueda de puntos de interés y autocompletado

La librería de Google Places permite localizar lugares disponibles en la base de datos de Google Places situados en una área especificada, por ejemplo, un radio de 5 km desde el centro del mapa. Este servicio es capaz de realizar tres tipos de búsquedas: de lugares, de texto y solicitudes de detalles de lugar. Las solicitudes de búsqueda de texto devuelven información sobre un conjunto de lugares relevantes para una cadena de texto determinada (p. ej., " librería anticuaria"). El servicio muestra una lista de lugares relevantes, pero además, la podemos combinar con una solicitud de detalles de lugar para obtener información complementaria de la ficha de Google Places de cada uno de los resultados. También es posible mostrar los resultados sobre el mapa en forma de marcador, asociando a cada uno de ellos su correspondiente "infoWindow" con la información obtenida como respuesta a la solicitud de detalles de lugar. Esa misma información también puede ser inyectada en cualquier otra parte de la página.

Las búsquedas de texto se inician con la ejecución del método "textSearch()" de "PlacesService". También se debe especificar un método de devolución de llamada para "textSearch()" que permita procesar el objeto de resultados y una respuesta a "google.maps.places.PlacesServiceStatus".

```
service = new google.maps.places.PlacesService(map);
service.textSearch(request, callback);
```

Fig. 49. Estructura de una hipotética ejecución del método "textSearch" y de la devolución de llamada (callback).

La solicitud del método "textSearch()" ha de incluir el campo obligatorio "query" al que se le debe pasar la cadena de texto de la búsqueda (p. ej., "farmacias", "hoteles hesperia", etc.). En este caso, asociamos el elemento <input> del formulario de búsqueda en el código HTML mediante el id "query". Para obtener el valor de ese elemento podemos utilizar el método ".val()" de jQuery[115] (fig. 50).

```
$('#search').submit(function(e){
    e.preventDefault();

    var query = $('#query').val();
    var request = {
        location: map.getCenter(),
        radius: '5000',
        query: query
    };

    service.textSearch(request, function(results, status, pagination){
        for(var i = 0; i < overlays.length; i++){
            overlays[i].setMap(null);
        }
        resultList.length = 0;
        overlays.length = 0;
        if (status == google.maps.places.PlacesServiceStatus.OK) {
            resultList = resultList.concat(results);
            plotResultList();
        }
    });
});
```

Fig. 50. Mediante el método ".val()" recogemos el valor del elemento <input> en el que el usuario introduce su ecuación de búsqueda.

Para que los resultados de la consulta se muestren en el mapa en forma de marcadores debemos utilizar el objeto"google.maps.Marker". El parámetro "google.maps.MarkerOptions" tiene diversas propiedades que podemos utilizar para modificar el aspecto y comportamiento del marcador. Las dos únicas propiedades obligatorias son: "position" y "map".

- **Position:** Esta propiedad define las coordenadas en las que el marcador estará ubicado. El argumento a utilizar es idéntico al del objeto google.maps.LatLng.

- **Map:** Esta propiedad es una referencia al mapa en el que deseamos añadir el marcador.

- **Title:** Muestra un texto al pasar el cursor sobre el marcador.

- **Icon:** Permite utilizar imágenes personalizadas para los iconos que representan los POI. Es posible utilizar los iconos creados por Google o una imagen propia que referenciaremos con su URL.

[115] ".val()". En: *jQuery: write less, do more*. <http://api.jquery.com/val/>. [Consulta: 31/05/2014].

Como en nuestro caso deseamos crear los marcadores a partir de la información que nos ha devuelto la consulta a Google Places en forma de *array*, debemos utilizar un bucle *for* para extraer los datos del vector (fig. 51). Durante el bucle se recorre el vector y se crea un marcador en cada nueva iteración. El valor de i se inicializa con 0, en tanto que la condición se verifica como verdadera (i es menor que el número de marcadores) se ejecuta el bloque del *for*. A continuación, el bucle pasa a su tercer argumento. En este caso el operador ++ incrementa en 1 el contenido de la variable i, y el bucle vuelve a empezar. El proceso acaba cuando i es superior al número de resultados devuelto en la búsqueda, momento en que la condición deja de ser verdadera.

```
function plotResultList(){
        var iterator = 0;
        for(var i = 0; i < resultList.length; i++){
             setTimeout(function(){
                    var lat = resultList[iterator].geometry.location.Za;
                    var lng = resultList[iterator].geometry.location.Ya;
                    var name = resultList[iterator].name;
                    var addr = resultList[iterator].formatted_address;
                    var reference = resultList[iterator].reference;


        var marker = new google.maps.Marker({
                     position: resultList[iterator].geometry.location,
                     map: map,
                     title: name,
        });
        overlays.push(marker);
        iterator++;
            });
        }
}
```

Fig. 51. Creación de los marcadores a partir de los resultados de Google Places.

Para añadir más información a los marcadores nos valdremos de los "InfoWindow" y de las solicitudes de detalles de sitios de Google Mas. Un "InfoWindow" es una especie de bocadillo que aparece al pulsar sobre el marcador en cuestión. De la misma manera que el objeto "Marker", el objeto "InfoWindow" reside en el espacio de nombres de "google.maps", y presenta un único argumento, el objeto "InfoWindowOptions". Este objeto tiene diferentes propiedades. La más importante de ellas es la propiedad "content", el valor de la cual será mostrado dentro del bocadillo. El valor de la propiedad "content" puede ser texto plano o código HTML. Una vez creado el objeto "InfoWindow", tenemos que conectarlo con su marcador para que al pulsar sobre él, el bocadillo aparezca. Para ello añadiremos un evento de clic al marcador. El método "google.maps.events.addListener()" nos permite asociar un determinado evento a un objeto dentro del DOM, y a una función que se ejecuta cuando se activa el evento. El objeto google.maps.Marker puede detectar los siguientes eventos de usuario:

- 'click'
- 'dblclick'
- 'mouseup'

- 'mousedown'
- 'mouseover'
- 'mouseout'

Este tipo de eventos de usuario no son apropiados para un usuario ciego y su uso exclusivo nos lleva a incumplir el criterio "2.1.1.Teclado" de las WCAG. Es por ese motivo que para que este tipo de usuario sea capaz de acceder al contenido del "InfoWindow" se le debe proporcionar una alternativa eficaz, como por ejemplo, acceder a los marcadores a partir de enlaces en una lista en html que al ser pulsados activen el correspondiente bocadillo.

Como se trata varios marcadores con sus respectivos "infoWindow", para que la aplicación sea capaz de determinar que bocadillo corresponde a cada marcador, crearemos un "infoWindow" global para todos los marcadores dentro del bucle.

```
infoWindow = new google.maps.InfoWindow();

google.maps.event.addListener(marker, 'click', function() {
    infoWindow.close();
    var request = {
        reference: reference
        };
    service.getDetails(request, function(place, status){
        var content = "<h3>" + name + "</h3>";
        if(status == google.maps.places.PlacesServiceStatus.OK){
            if(typeof place.formatted_address !== 'undefined'){
                content += "<br><small><b>Dirección:</b> " + place.formatted_address + "</small>";
            }
            if(typeof place.formatted_phone_number !== 'undefined'){
                content += "<br><small><b>Teléfono: </b><a class='text-info' href='tel:" + place.formatted_
            }
            if(typeof place.website !== 'undefined'){
                content += "<br><small><a class='text-info' target='_blank' title='visita el web (abrirá un
            }
        }
        infoWindow.setContent(content);
        infoWindow.open(map, marker);
    });
});
```

Fig. 52. Creación de los “infoWindow” a partir de la información de Google Places.

Por lo que respecta a su contenido, nos valdremos de la información detallada que nos devuelve Google Places sobre lugares específicos a través de la propiedad "reference" y el método "getDetails()" del servicio[116]. En concreto, podemos utilizar entre otros[117]:

- **name:** Contiene el nombre del local, comercio, edificio, etc.

[116] “Detalles de lugar”. En: *Versión 3 del API de JavaScript de Google Maps*. <https://developers.google.com/maps/documentation/javascript/places?hl=es#place_details>. [Consulta: 28/05/2014]. [Consulta: 31/05/2014].

[117] “Resultados de detalles de lugar”. En: *Versión 3 del API de JavaScript de Google Maps*. <https://developers.google.com/maps/documentation/javascript/places?hl=es#place_details_results>. [Consulta: 28/05/2014].

- **types:** El tipo/s de comercio bajo los cuales se ha categorizado la ficha en Google Places. Se trata de valores de un campo controlado que son devueltos en el idioma original (p.ej., establishment, museum, store, movie_theater, pharmacy, etc.).

- **price_level:** Indica de 0 (gratis) a 4 (muy caro) el precio del sitio según la opinión de los usuarios.

- **rating:** Indica la valoración del sitio (entre 1.0 y 5.0) basada en los votos de los usuarios.

- **formatted_address**: La dirección del local o establecimiento.

- **formatted_phone_number:** El teléfono de contacto.

- **website:** La página web.

- **url:** El URL del perfil de Google +.

- Etc.

Podemos utilizar estos y/u otros datos devueltos por Google Places para asociarlos a la propiedad "content" de cada "Infowindow" (fig. 53).

Fig. 53. Ejemplo de "Infowindow" con información obtenida de Google Places.

Como hemos dicho anteriormente, en tanto que la única manera de acceder a la información contenida en los "infoWindow" es mediante alguno de los eventos de ratón aceptados por el objeto "google.maps.Marker" ('click', 'dblclick', 'mouseup', 'mousedown', 'mouseover' y 'mouseout'), se hace necesaria una alternativa que permita a los usuarios ciegos acceder a esa información. De esta manera podremos cumplir con el criterio "2.1.1. Teclado" de las WCAG.

Existen diferentes posibilidades. Una de ellas consiste en crear una lista externa al mapa con los resultados obtenidos en forma de enlaces que, mediante un evento de teclado (onkeypress) o de ratón y teclado (onclick y onkeypress) ejecuten el método ".open()" del objeto "infoWindow". Una vez abierto el "infoWindow" mediante el teclado, el usuario ciego si será capaz de navegar hasta su contenido al ser receptor del foco del teclado. Otra de las opciones consiste en ofrecer toda la información asociada a cada punto de interés fuera del mapa junto con la lista de resultados. De esta manera, el usuario tiene toda la información en la misma sección de la pantalla. Se trata de dos de las posibles alternativas que dan solución a la imposibilidad de que los marcadores creados reciban el foco del teclado.

En el caso de que tuviéramos los valores de nuestros marcadores en vectores como los que se observan en la figura 54, podríamos utilizar el método ".trigger()", para que se encargue de activar el evento "onkeypress" asociado al primer valor del vector de cada marcador (fig. 55). De esta manera al pulsar sobre cada uno de los elementos de la lista podríamos acceder a su "InfoWindow".

```
var marcadores = [
    ["marcador1","Calle Lorem ipsum, 23","41.3846241973006","2.1716089115143404"],
    ["marcador2","Calle lorem ipsum, 10-12","41.38346100200511","2.177043066978513"],
    ["marcador3","Calle lorem ipsum, 26","41.39295110192035","2.1786953077316866"]
    ];


for (var i = 0; i < marcadores.length; i++) {
markers[marcadores[i][0]] =
createMarker(new google.maps.LatLng(marcadores[i][2], marcadores[i][3]), marcadores[i][0] + "<br>" + marcadores[i][1]
    );
}
```

Fig. 54. Fragmento del código JavaScript en el que se observan unos vectores con información de cada marcador y un bucle *for* que los recorre.

```
<ul>
    <li><a href="javascript:google.maps.event.trigger(markers['marcador1'],'onkeypress');">marcador1</a></li>
    <li><a href="javascript:google.maps.event.trigger(markers['marcador2'],'onkeypress');">marcador2</a></li>
    <li><a href="javascript:google.maps.event.trigger(markers['marcador3'],'onkeypress');">marcador3</a></li>
</ul>
```

Fig. 55. En el código HTML podemos ejecutar una función de JavaScript dentro del "href" para acceder a los marcadores desde cada enlace de la lista.

En cuanto a la opción de ofrecer la información asociada a cada marcador en la misma lista, bastaría con llamar a los campos de Google Places que nos interese destacar (nombre, dirección, etc.) como hemos hecho con el contenido para el "InfoWindow". Para ello podemos utilizar la propiedad "innerHTML" sobre el elemento al que le queremos añadir el contenido. En este caso, una lista con ID "places" (fig. 56). El contenido lo habíamos almacenado previamente en las variables que se observan en la figura 51.

```
placesList = document.getElementById('places');
placesList.innerHTML += '<li>'+'<b>'+ name +'</b><br>'+ addr +'</li>';
```

Fig. 56. La propiedad "innerHTML" nos permite insertar código HTML en la página. En este caso, elementos <li> con el nombre y la dirección de cada resultado recuperado.

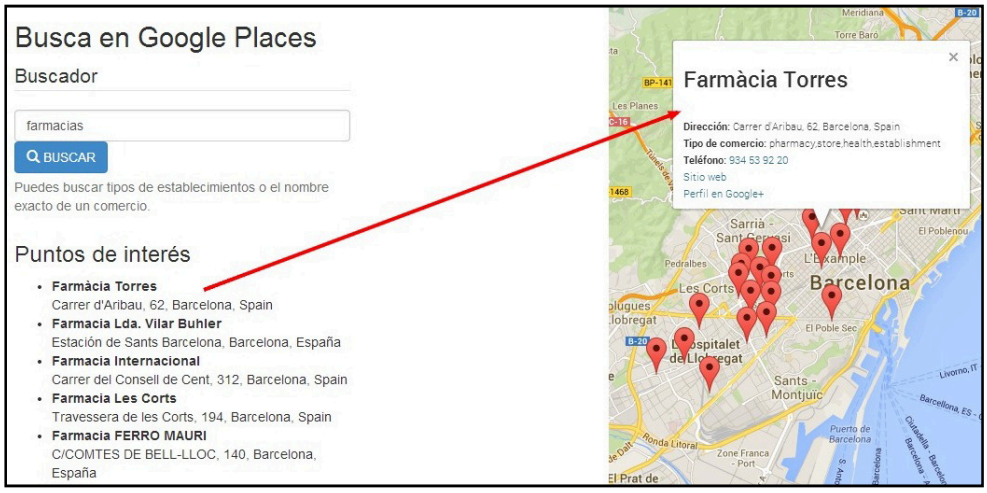


Fig. 57. Demostración en la que se devuelve al usuario información sobre los puntos de interés dentro del "InfoWindow" y en el mismo código HMTL.

La API de autocompletado de Google Places[118] es un servicio web que ofrece información sobre un sitio basada en términos de búsqueda de texto y, opcionalmente, en límites geográficos. Esta API se puede utilizar para proporcionar funciones de autocompletado para búsquedas geográficas basadas en texto mostrando sitios como empresas, direcciones y lugares de interés a medida que el usuario escribe. La función se vincula a los elementos <input> disponibles en los formularios de búsqueda y controla los caracteres que el usuario introduce, encargándose de enviar predicciones de lugares en forma de lista desplegable. En la figura 58 se puede observar el constructor de autocompletado con la referencia al elemento <input> HTML de tipo *text*, al que el servicio asociará sus resultados.

```
185   autoCompleteSetup = function() {
186       autoSrc = new google.maps.places.Autocomplete($Selectors.dirSrc[0]);
187       autoDest = new google.maps.places.Autocomplete($Selectors.dirDst[0]);
188   }, // final autoCompleteSetup
```

Fig. 58. Función para el autocompletado.

[118] "Autocomplete for addresses and search terms". En: *API de Google Maps*. <https://developers.google.com/maps/documentation/javascript/places-autocomplete>. [Consulta: 28/05/2014].

Fig. 59. Ejemplo de uso del autocompletado en la búsqueda de direcciones.

Se trata de una funcionalidad que como hemos visto en la entrevista con el usuario ciego, resulta muy útil a este colectivo, ayudándoles a evitar que cometan errores en la introducción de datos ("Criterio 3.3.2. Etiquetas o instrucciones" de las WCAG).

9.1.7. Servicio de rutas

El objeto "DirectionsService" permite obtener rutas entre dos puntos para distintos medios de transporte al comunicarse directamente con el servicio de rutas de la API de Google Maps. Para que el objeto "DirectionsService" pueda calcular la ruta, además del punto de origen y el punto de destino, debemos indicarle el medio de transporte. Actualmente, la API de Google Maps admite los siguientes medios de transporte:

- **google.maps.TravelMode.DRIVING** (predeterminado): proporciona rutas estándar para llegar en coche a través de la red de carreteras.

- **google.maps.TravelMode.BICYCLING:** solicita rutas para llegar en bicicleta a través de carriles bici y vías preferenciales para bicicletas.

- **google.maps.TravelMode.TRANSIT:** solicita rutas de transporte público.

- **google.maps.TravelMode.WALKING:** solicita rutas a pie a través de aceras y rutas peatonales (en fase beta).

La información de la ruta se ha de pasar al objeto "DirectionsService" mediante un objeto "DirectionsRequest" que debe contener el origen y el destino. Para ello tenemos que ejecutar el método ".route()". El resultado de este método devuelve el objeto "DirectionsResult", un literal que consta de un único campo: "routes[]" con un conjunto de objetos "DirectionsRoute" que muestran las rutas para llegar desde el origen al destino. Cada uno de estos objetos consta

de diversos "DirectionsStep" que son la unidad más pequeña de cada ruta e incluye cada uno de los pasos que describen como llegar al punto de destino y la distancia prevista (fig. 60).

Fig. 60. En rojo un "DirectionsStep".

Una vez tenemos el resultado, podemos acceder a la información que contiene para representarla como polilíneas que trazan la ruta sobre el mapa entre dos puntos identificados por sendos marcadores, o como una serie de descripciones textuales paso a paso dentro de un elemento (fig. 60). Para poder representar esa información debemos usar el objeto "google.maps.DirectionsRenderer". Para configurarlo utilizaremos los métodos ".setMap()" para indicar cuál es el mapa, ".setPanel()" para indicarle el dónde ha de mostrar el texto de la ruta y "setDirections()" para mostrar la respuesta dentro del .

```
directionsSetup = function() {
                    directionsService = new google.maps.DirectionsService();
                    directionsDisplay = new google.maps.DirectionsRenderer({
                    });

                    directionsDisplay.setPanel($Selectors.dirSteps[0]);
                },

directionsRender = function(source, destination) {
                    $Selectors.dirSteps.find('.indicaciones').hide();
                    directionsDisplay.setMap(map);

                    var selectedMode = document.getElementById('modo').value;
                    var request = {
                        origin: source,
                        destination: destination,
                        provideRouteAlternatives: false,
                        travelMode: google.maps.TravelMode[selectedMode]
                    };

                    directionsService.route(request, function(response, status) {
                        if (status == google.maps.DirectionsStatus.OK) {

                            directionsDisplay.setDirections(response);

                        }
                    });
                },
```

Fig. 61. Funciones para configurar el servicio de rutas.

Para permitir al usuario seleccionar el medio de transporte crearemos un cuadro desplegable que cogerá como valor una de las opciones disponibles (WALKING, DRIVE, TRANSIT). Para ello podemos crear una variable asociada al <select> de un formulario mediante el método "getElementById". Cuando el usuario seleccione una de las opciones disponibles, el valor de ese campo, definido en el atributo "value", pasará al campo "travelMode", campo obligatorio donde se especifica el medio de transporte que se utilizará al calcular las rutas.

```
<label for="modo">Medio de transporte</label>
  <select id="modo" class="form-control">
    <option aria-checked="true" value="TRANSIT">En transporte p&uacute;blico</option>
    <option value="WALKING">A pie</option>
    <option value="DRIVING">En coche</option>
  </select><br>
```

Fig. 62. Cuadro combinado para escoger el medio de transporte.

```
var selectedMode = document.getElementById('modo').value;
                    var request = {
                        origin: source,
                        destination: destination,
                        provideRouteAlternatives: false,
                        travelMode: google.maps.TravelMode[selectedMode]
                    };
```

Fig. 63. Función asociada al cuadro combinado de selección de medio de transporte.

La creación de formularios correctos, mediante el uso de etiquetas textuales <label> asociadas a los diferentes elementos del formulario y de la agrupación de los mismos mediante los elementos <fieldset>/<legend>, nos ayuda cumplir con los criterios 1.1.1., 1.3.1, 3.3.2, 4.1.1 y 4.1.2 de las WCAG.

```
<h2 id="rutas">Servicio de rutas</h2>
  <form class="form-horizontal">
    <fieldset>
      <legend>Buscador de direcciones y puntos de interés</legend>
        <div class="control-group">

          <label for="dirSource">Origen</label><br>
            <button type="button" class="btn btn-primary" id="useGPS" aria-describedby="instrucciones-geo"><span class="glyphicon
            glyphicon-screenshot"></span> Usar tu localización</button> <br><br>
            <p tabindex="0" class="help-block" id="instrucciones-geo">Pulsando sobre este botón la aplicación utilizará tu posición
            actual como punto de origen para el servicio de rutas. También puedes entrarla manualmente a través del siguiente formulario
            .</p>

            <input type="text" id="dirSource" class="form-control" placeholder="Introduce la dirección de origen" required aria-required
            ="true"/><br>

            <label for="dirDestination">Destino</label>
            <input type="text" id="dirDestination" class="form-control" placeholder="Introduce la dirección de destino" required aria-
            required="true"/><br>

            <label for="modo">Medio de transporte</label>
              <select id="modo" class="form-control">
                <option aria-checked="true" value="TRANSIT">En transporte público</option>
                <option value="WALKING">A pie</option>
                <option value="DRIVING">En coche</option>
              </select><br>

            <input type="reset" name="reset" class="btn btn-danger" id="panelReset" value="Borrar">
            <input type="submit" name="submit" class="btn btn-success" id="getDirections" value="Cómo llegar"><br><br><p><a href="#ruta"
            >Ve a la ruta</a></p>
        </div>
    </fieldset>
  </form>
```

Fig. 64. Formulario para el servicio de rutas.

La API de geolocalización de HTML5 permite compartir la localización (latitud y longitud) siempre que el usuario permita a la API rastrearla. Una vez obtenida, es posible pasar esas coordenadas a diferentes funciones en nuestro código para utilizarla, por ejemplo, para centrar el mapa en el punto en el que se encuentra el usuario, o para obtener una dirección formateada (p. ej., Carrer Melcior de Palau, 140, Barcelona, España) mediante la API de geocodificación inversa de Google. El objetivo es generar las direcciones a partir de los valores de origen y destino introducidos en sendos campos de un formulario HTML o, en su defecto, a partir de la geolocalización del usuario. Estos valores se pasarán al "DirectionsService" de la forma que se ha comentado anteriormente.

Para utilizar la posición del usuario como punto de origen asociaremos un elemento <button> a una función de geolocalización similar a la que hemos visto anteriormente (fig. 65).

```
1  $Selectors.useGPSBtn.on('click', function(e) {
2                          if (navigator.geolocation) {
3                              navigator.geolocation.getCurrentPosition(function(position) {
4                                  fetchAddress(position);
5                              });
6                          }
7                      });
```

Fig. 65. Función para geolocalizar al usuario asociada a un botón en el código HTML.

El resultado puede ser almacenado en una variable ("dirSrc") para luego utilizar la API de geocodificación para transformar las coordenadas en una dirección.

```
fetchAddress = function(p) {
                    var Position = new google.maps.LatLng(p.coords.latitude, p.coords.longitude),
                        Locater = new google.maps.Geocoder();

                    Locater.geocode({'latLng': Position}, function (results, status) {
                        if (status == google.maps.GeocoderStatus.OK) {
                            var _r = results[0];
                            $Selectors.dirSrc.val(_r.formatted_address);
                        }
                    });
                },
```

Fig. 66. Proceso de geocodificación inversa.

9.2. Wireframes

A continuación se muestra en diferentes *wireframes* de baja fidelidad, la estructura visual de la aplicación y se explica la lógica que subyace al orden de los diferentes elementos de la interfaz.

La aplicación se encuentra formada por cuatro zonas diferenciadas: el buscador de sitios, el servicio de rutas, el mapa y una última zona "viva" en la que se muestran los resultados de búsqueda y las indicaciones del servicio de rutas. Con el objetivo de facilitar la rápida navegación hacia los diferentes apartados, se ha agrupado cada una de estas zonas mediante sus correspondientes encabezamientos. Además, también se ofrecen diferentes enlaces en la parte superior de la página que permiten al usuario acceder directamente a cada uno de los apartados ("Criterio 2.4.1. Saltar bloques" de las WCAG).

El orden de navegación y el orden del foco procuran preservar la operabilidad de la aplicación. En este sentido, el usuario que navega a través del teclado pasa en primer lugar por el buscador de sitios y el servicio de rutas, para dirigirse a continuación al mapa (y a sus diferentes controles) y por último a los resultados. Desde el buscador de sitios y el servicio de rutas se ofrecen igualmente sendos enlaces directos o atajos a sus respectivos resultados que además, serán anunciados por el lector de pantalla al estar marcados como zonas vivas ARIA.

Fig. 67. Orden de navegación y atajos ofrecidos al usuario.

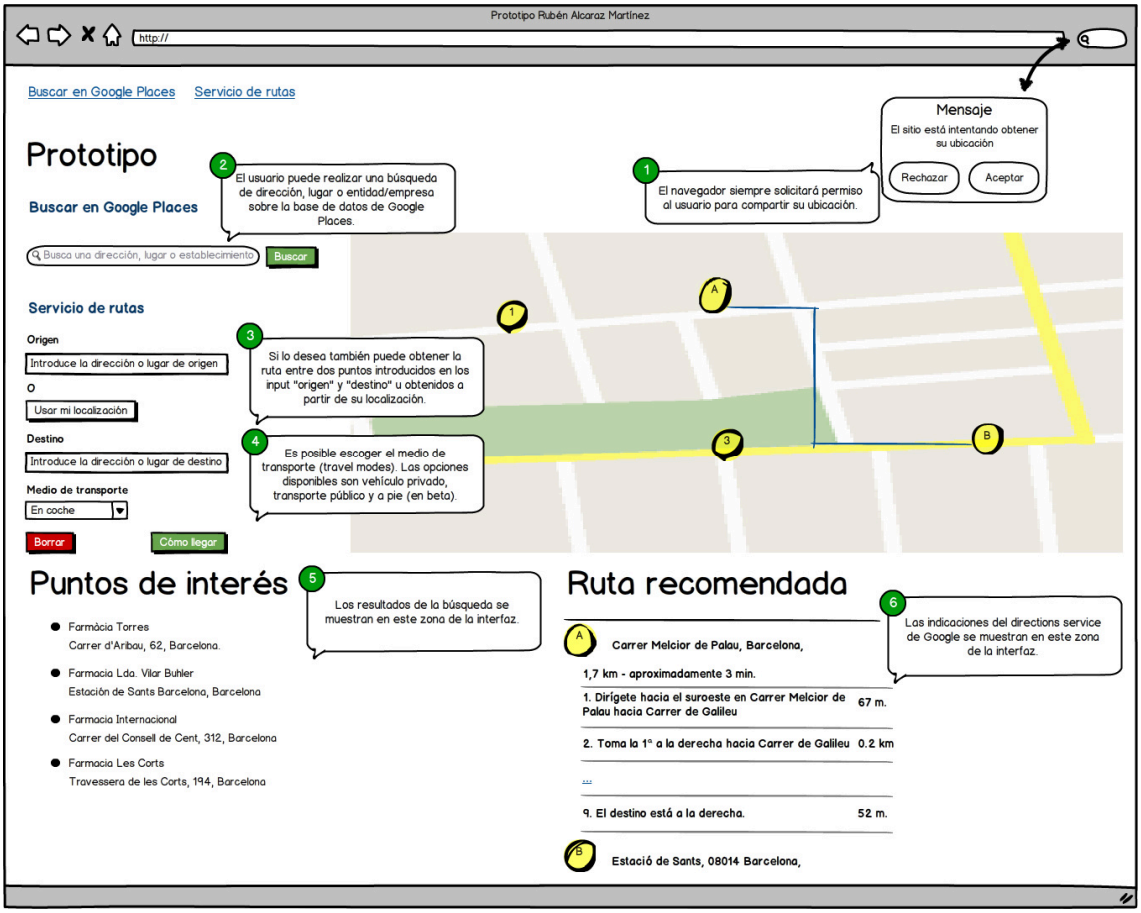


Fig. 68. *Wireframe/Storyboard* explicativo de las funcionalidades de la aplicación.

10. Planificación

Para poder llevar a cabo un proceso de planificación en un proyecto teórico como este, se ha optado por la creación de un escenario ficticio en el cual se enmarque el proyecto.

10.1. Escenario

El Ayuntamiento de un municipio de Catalunya ha desarrollado en los últimos años un proyecto de reutilización de su información pública con el objetivo de mejorar el acceso de las partes interesadas a esta información. De esta manera, el ayuntamiento pretende facilitar la reutilización de dicha información por parte de terceros, así como por los propios departamentos del ente. También promover la innovación y la creación de valor local gracias a la oportunidad de desarrollo de nuevos modelos de negocio que pueden partir del uso de la información pública. Con este objetivo se ha elaborado un catálogo de datos formado por diferentes conjuntos de datos (*datasets*) que pueden ser reutilizados por cualquier ciudadano o empresa. Dichos datos se encuentran en diferentes formatos propios de la web semántica como JSON, XML, RDF, KML o geoRSS entre otros.

La segunda fase del proyecto tiene como objetivo el desarrollo de diferentes aplicaciones capaces de consumir los *datasets* publicados y proporcionar diferentes servicios de valor añadido a la ciudadanía. La primera de estas aplicaciones que el Ayuntamiento desea incorporar a su página web es un servicio basado en la localización que permita a sus ciudadanos consultar el plano de la ciudad, buscar direcciones, obtener información acerca de los equipamientos municipales, así como de otros servicios públicos o privados de interés (farmacias, escuelas, restaurantes, etc.), disponer de un servicio de rutas entre dos o más puntos, entre otras funcionalidades. Actualmente, el Ayuntamiento dispone de un mapa digital en su portal web que hace las veces de callejero del municipio, pero que no incorpora ninguna de las funcionalidades descritas en el punto anterior.

10.2. Fases, tareas asociadas y recursos humanos necesarios

10.2.1. Recursos humanos

Para el proyecto que nos ocupa se prevé un equipo formado por los siguientes perfiles profesionales:

- **Jefe de proyecto:** Encargado de coordinar todas las fases del proyecto y a las personas implicadas. También es el responsable del cumplimiento del calendario con los recursos destinados. Se trata de un profesional con conocimientos transversales en las diferentes áreas implicadas y experto en la gestión de proyectos. Será el enlace con el cliente.
- **Informático programador:** Programador sénior. Encargado de programar la aplicación. Será necesario un profesional con conocimientos en JavaScript y en la API de Google Maps, así como en diferentes tecnologías de la web semántica (JSON, RDF, etc.).

- **Consultor de accesibilidad:** Consultor de accesibilidad sénior con conocimientos en test de usuarios y la especificación WAI-ARIA. Será el responsable de la definición de los requerimientos de accesibilidad para el nivel de conformidad especificado, de la realización de test de usuarios y de la evaluación final de la accesibilidad de la aplicación.
- **Diseñador gráfico/Maquetador web:** Responsable de la creación de los elementos gráficos, la maquetación de la aplicación y el diseño gráfico de la interfaz según la identidad corporativa del Ayuntamiento. Necesarios conocimientos en maquetación web con CSS y HTML.

10.2.2. Fases y tareas

Fase 0. Reunión de *kick off*

- **Objetivos y tareas asociadas:** En la reunión de inicio de proyecto se definirán los objetivos, el alcance y las fases del mismo, así como las funciones y responsabilidades de los diferentes miembros del equipo.
- **Perfiles implicados:** Equipo completo a excepción del maquetador.
- **Calendario:** 1 día.

Fase 1. Definición de la estrategia y estudio de usuarios

- **Objetivos y tareas asociadas:** En esta fase se deben formalizar los objetivos del proyecto. Al definir nuestros objetivos y aquello qué esperan de nosotros los ciudadanos, podremos ajustar con más precisión las decisiones posteriores. Para ello se debe realizar un análisis de la organización y un estudio de las necesidades de los usuarios. Las tareas previstas son:
    - Análisis de los objetivos de la organización.
    - Análisis de referentes.
    - Estudio de usuarios (6 entrevistas).
- **Entregables:**
    - Documento con los objetivos de la organización.
    - Informe de buenas prácticas en otros proyectos similares.
    - Informe con los resultados del estudio de usuarios.
- **Perfiles implicados:** Jefe de proyecto y programador.
- **Calendario:** 5 días.

Fase 2. Definición de los requerimientos funcionales

- **Objetivos y tareas asociadas:** El objetivo de esta fase es conseguir una base de conocimiento para el posterior desarrollo informático de la aplicación. Se deberá dar respuesta a preguntas como qué acciones podrá realizar el usuario, cómo podrá

llevarlas a cabo, qué respuestas proporcionará la aplicación, etc. La fase de definición de los requerimientos funcionales está compuesta por las siguientes tareas:

- Descripción de los requerimientos de la aplicación.
- Diagramación de los procesos (usuarios y administradores) para anticipar las relaciones que mantendrán los usuarios con la aplicación.

- **Entregables:**
  - Informe con los requerimientos funcionales de la aplicación.
  - Diagrama de flujo de los procesos.
- **Perfiles implicados:** Programador.
- **Calendario:** 2 días.

Fase 3. Definición de los requerimientos de accesibilidad y puesta en común

- **Objetivos y tareas asociadas:** Describir los requerimientos de accesibilidad necesarios para alcanzar el nivel de conformidad acordado a partir de los requerimientos funcionales propuestos. Una vez definidos se trabarán conjuntamente con el programador para implementarlos en la aplicación.
- **Entregables:**
  - Informe final de requerimientos técnicos y de accesibilidad.
- **Perfiles implicados:** Consultor de accesibilidad, programador y jefe de proyecto.
- **Calendario:** 2 días.

Fase 4. Diseño conceptual

- **Objetivos y tareas asociadas:** Diseño de la estructura de la página (*wireframe*) y de todos los elementos de la interfaz para facilitar la interacción del usuario con la aplicación. El objetivo es seleccionar aquellos elementos más adecuados para cada una de las tareas previstas y ubicarlos en la zona en la que mejor se puedan comprender y utilizar.
- **Entregables:**
  - *Wireframes* de baja fidelidad de la aplicación que muestren todas sus funcionalidades de acuerdo con los diagramas de flujo.
- **Perfiles implicados:** Consultor de accesibilidad.
- **Calendario:** 1 día.

Reunión de validación (todo el equipo). Se prevé unos días de contingencia por si se ha de corregir alguno de los entregables.

Fase 5. Desarrollo de la aplicación. Implementación técnica

- **Objetivos y tareas asociadas:** Programación de la aplicación y primeras pruebas de funcionamiento.
    - Desarrollo del primer prototipo.
    - Testeo de las funcionalidades.
- **Entregables:**
    - Prototipo.
- **Perfiles implicados:** Programador y consultor de accesibilidad.
- **Calendario:** 9 días.

Fase 6. Test con usuarios

- **Objetivos y tareas asociadas:**
    - Captación de usuarios.
    - Test con usuarios (4 usuarios en un test de observación de simulación de tareas (grabado) + entrevista).
    - Interpretación y resumen de los resultados.
- **Entregables:**
    - Informe de observaciones y recomendaciones. Contiene los resultados y recomendaciones para la mejora de la aplicación de acuerdo a lo observado en el test con usuarios.
- **Perfiles implicados:** Consultor de accesibilidad.
- **Calendario:** 12 días (margen suficiente para poder encontrar y acordar las condiciones de la colaboración con los usuarios). Realmente no se trata de 12 días a tiempo completo.

Reunión de validación (todo el equipo).

Fase 7. Aplicación de los resultados del test al prototipo.

- **Objetivos y tareas asociadas:** En el caso de que se hayan encontrado incidencias en los test de usuario, el prototipo deberá ser corregido en consecuencia.
- **Entregables:**
    - El prototipo con las incidencias corregidas.
- **Perfiles implicados:** Programador y/o consultor de accesibilidad.
- **Calendario:** 1-3 días (según la cantidad de incidencias detectadas).

Fase 8. Diseño gráfico y maquetación.

- **Objetivos y tareas asociadas:** Diseñar la apariencia gráfica del esqueleto prototipado y formalización del resultado en una guía de estilo. Las tareas previstas son:
    - Diseño de los elementos gráficos.
    - Maquetación de la aplicación.
    - Redacción de la guía de estilo.
- **Entregables:**
    - Colección de elementos gráficos (iconos, logos, botones, etc.).
    - Prototipo final.
    - Guía de estilo.
- **Perfiles implicados:** Diseñador gráfico/maquetador web.
- **Calendario:** 3 días.

Fase 9. Lanzamiento.

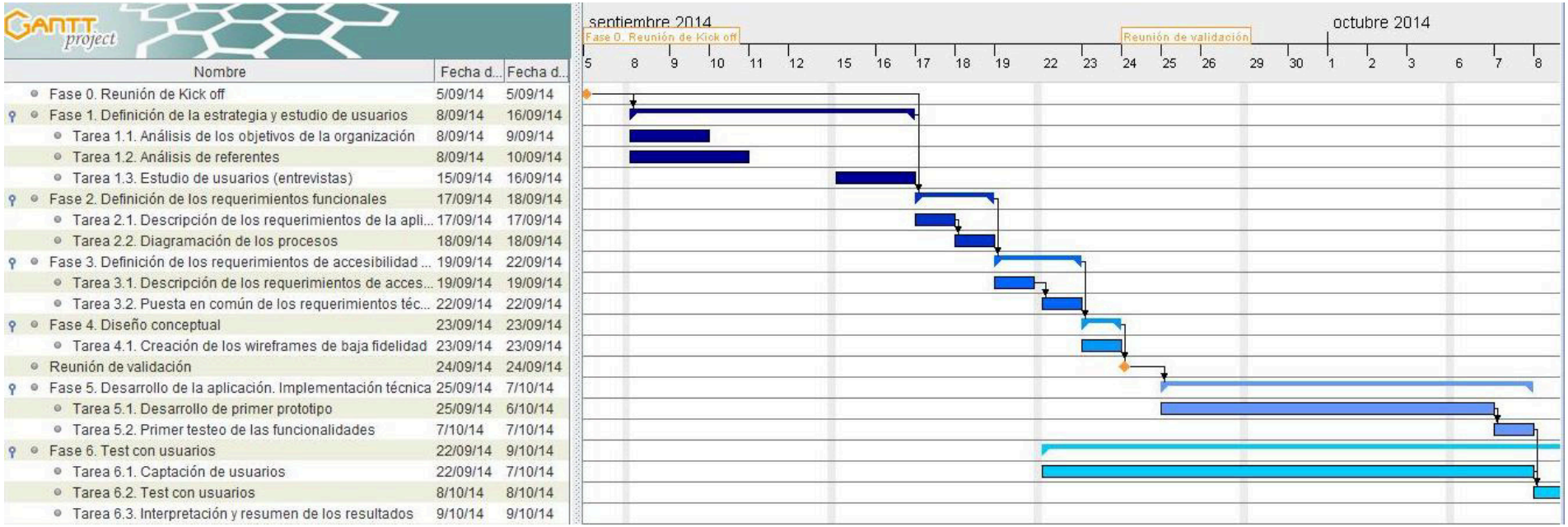
- **Objetivos y tareas asociadas:**
    - Traspaso del entorno de producción al entorno definitivo.
    - Último test de la aplicación (pruebas de estrés).
    - Lanzamiento de la aplicación.
- **Perfiles implicados:** Programador.
- **Calendario:** 3 días.

Se prevé unos días de contingencia por si se ha de corregir algún error no previsto.

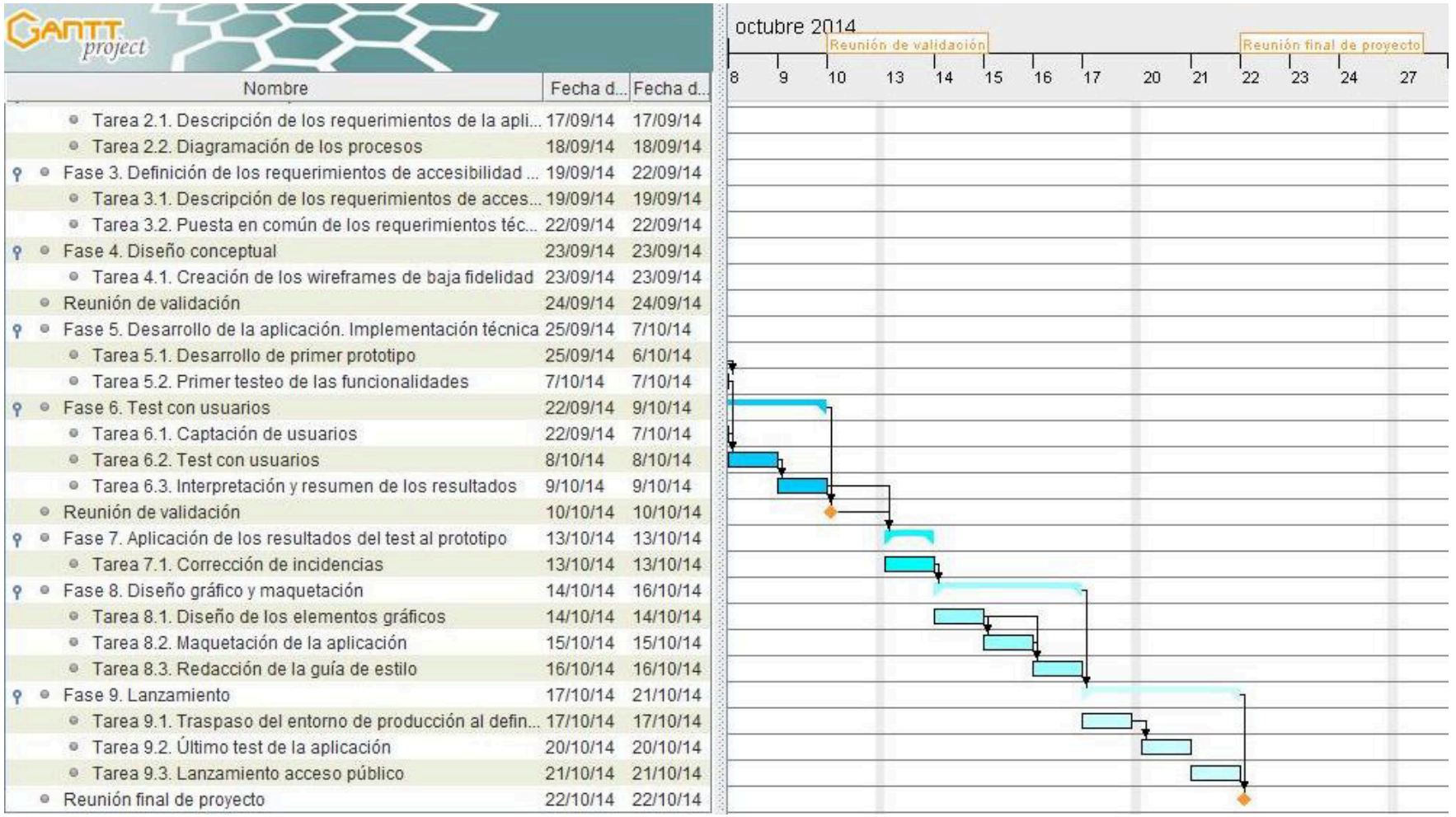
Fase 10. Reunión final de proyecto.

- **Perfiles implicados:** Todo el equipo.
- **Calendario:** 1 día.

10.3. Cronograma (diagrama de Gantt)



Nombre	Fecha d...	Fecha d...
Fase 0. Reunión de Kick off	5/09/14	5/09/14
Fase 1. Definición de la estrategia y estudio de usuarios	8/09/14	16/09/14
Tarea 1.1. Análisis de los objetivos de la organización	8/09/14	9/09/14
Tarea 1.2. Análisis de referentes	8/09/14	10/09/14
Tarea 1.3. Estudio de usuarios (entrevistas)	15/09/14	16/09/14
Fase 2. Definición de los requerimientos funcionales	17/09/14	18/09/14
Tarea 2.1. Descripción de los requerimientos de la apli...	17/09/14	17/09/14
Tarea 2.2. Diagramación de los procesos	18/09/14	18/09/14
Fase 3. Definición de los requerimientos de accesibilidad ...	19/09/14	22/09/14
Tarea 3.1. Descripción de los requerimientos de acces...	19/09/14	19/09/14
Tarea 3.2. Puesta en común de los requerimientos téc...	22/09/14	22/09/14
Fase 4. Diseño conceptual	23/09/14	23/09/14
Tarea 4.1. Creación de los wireframes de baja fidelidad	23/09/14	23/09/14
Reunión de validación	24/09/14	24/09/14
Fase 5. Desarrollo de la aplicación. Implementación técnica	25/09/14	7/10/14
Tarea 5.1. Desarrollo de primer prototipo	25/09/14	6/10/14
Tarea 5.2. Primer testeo de las funcionalidades	7/10/14	7/10/14
Fase 6. Test con usuarios	22/09/14	9/10/14
Tarea 6.1. Captación de usuarios	22/09/14	7/10/14
Tarea 6.2. Test con usuarios	8/10/14	8/10/14
Tarea 6.3. Interpretación y resumen de los resultados	9/10/14	9/10/14



Nombre	Fecha d...	Fecha d...
Tarea 2.1. Descripción de los requerimientos de la apli...	17/09/14	17/09/14
Tarea 2.2. Diagramación de los procesos	18/09/14	18/09/14
Fase 3. Definición de los requerimientos de accesibilidad ...	19/09/14	22/09/14
Tarea 3.1. Descripción de los requerimientos de acces...	19/09/14	19/09/14
Tarea 3.2. Puesta en común de los requerimientos téc...	22/09/14	22/09/14
Fase 4. Diseño conceptual	23/09/14	23/09/14
Tarea 4.1. Creación de los wireframes de baja fidelidad	23/09/14	23/09/14
Reunión de validación	24/09/14	24/09/14
Fase 5. Desarrollo de la aplicación. Implementación técnica	25/09/14	7/10/14
Tarea 5.1. Desarrollo de primer prototipo	25/09/14	6/10/14
Tarea 5.2. Primer testeo de las funcionalidades	7/10/14	7/10/14
Fase 6. Test con usuarios	22/09/14	9/10/14
Tarea 6.1. Captación de usuarios	22/09/14	7/10/14
Tarea 6.2. Test con usuarios	8/10/14	8/10/14
Tarea 6.3. Interpretación y resumen de los resultados	9/10/14	9/10/14
Reunión de validación	10/10/14	10/10/14
Fase 7. Aplicación de los resultados del test al prototipo	13/10/14	13/10/14
Tarea 7.1. Corrección de incidencias	13/10/14	13/10/14
Fase 8. Diseño gráfico y maquetación	14/10/14	16/10/14
Tarea 8.1. Diseño de los elementos gráficos	14/10/14	14/10/14
Tarea 8.2. Maquetación de la aplicación	15/10/14	15/10/14
Tarea 8.3. Redacción de la guía de estilo	16/10/14	16/10/14
Fase 9. Lanzamiento	17/10/14	21/10/14
Tarea 9.1. Traspaso del entorno de producción al defin...	17/10/14	17/10/14
Tarea 9.2. Último test de la aplicación	20/10/14	20/10/14
Tarea 9.3. Lanzamiento acceso público	21/10/14	21/10/14
Reunión final de proyecto	22/10/14	22/10/14

10.4. Presupuesto

10.4.1. Desglose de horas

Tareas	Jefe de proyecto	Programador	Consultor de accesibilidad	Maquetador
Reunión de Kick off	3 h.	2 h.	2 h.	
Tarea 1.1. Análisis de los objetivos de la organización	16 h.			
Tarea 1.2. Análisis de referentes		16 h.		
Tarea 1.3. Estudio de usuarios (entrevistas)	16 h.			
Tarea 2.1. Descripción de los requerimientos de la aplicación		8 h.		
Tarea 2.2. Diagramación de los procesos		8 h.		
Tarea 3.1. Descripción de los requerimientos de accesibilidad			8 h.	
Tarea 3.2. Puesta en común de los requerimientos técnicos	2 h.	8 h.	8 h.	
Tarea 4.1. Creación de los wireframes de baja fidelidad			8 h.	
Tarea 5.1. Desarrollo de primer prototipo		60 h.		
Tarea 5.2. Primer testeo de las funcionalidades		4 h.	4 h.	
Tarea 6.1. Captación de usuarios			12 h.	
Tarea 6.2. Test con usuarios			8 h.	
Tarea 6.3. Interpretación y resumen de los resultados			8 h.	
Reunión de validación	2 h.	2 h.	2 h.	2 h.
Tarea 7.1. Corrección de incidencias		8 h.	8 h.	
Tarea 8.1. Diseño de los elementos gráficos				8 h.
Tarea 8.2. Maquetación de la aplicación				8 h.
Tarea 8.3. Redacción de la guía de estilo				8 h.
Tarea 9.1. Traspaso del entorno de producción al definitivo		4 h.		
Tarea 9.2. Último test de la aplicación		8 h.		

Tareas	Jefe de proyecto	Programador	Consultor de accesibilidad	Maquetador
Tarea 9.3. Lanzamiento y acceso público	4 h.	4 h.	4 h.	
Reunión final de proyecto	1 h.	1 h.	1 h.	1 h.
Total	44 h.	133 h.	73 h.	27 h.
Coste/h.	30 €	15 €	30 €	15 €
Coste final	1320 €	1995 €	2190 €	405 €

Tabla 4. Desglose de horas y cálculo del coste de cada perfil implicado.

10.4.2. Coste final

Concepto	Horas	Total
Directos		
Recursos humanos		
Jefe de proyecto	42 h.	1320 €
Programador	133 h.	1995 €
Consultor de accesibilidad	77 h.	2190 €
Diseñador gráfico/maquetador	27 h.	405 €
	Total	5910 €
Contingencias	10%	591 €
Total recursos humanos		**6501 €**
Recursos materiales		
Material fungible		80 €
Disco duro SAS 1 TB		425 €
Alquiler sala y equipo para el test de usuarios (1 día 8h.)		700 €
Recompensa estudios de usuarios (10 usuarios)		300 €
Contingencias (10%)		150
Total recursos materiales		**1655 €**
Total gastos directos		**8156 €**
Indirectos		
Gastos varios (alquiler, ordenadores, luz, agua, Internet, etc.)		500 €
Total gastos indirectos		**500 €**
Total proyecto sin IVA		**8656 €**
Total con IVA		**10.393, 9 €**

Tabla 5. Presupuesto final.

11. Conclusiones y trabajo futuro

11.1. Conclusiones

Una de las principales conclusiones a la que se puede llegar tras la lectura de este trabajo es que mediante la filosofía del diseño centrado en el usuario es posible desarrollar cualquier tipo de producto o aplicación accesible con independencia de sus características iniciales. En ningún caso eso significa que desarrollar aplicaciones totalmente accesibles sea fácil. Según la cantidad y complejidad de las funcionalidades que presente el producto o de las diferentes morfologías de información presentes, hacerlo accesible será más o menos complicado. Además, la accesibilidad universal conlleva la necesidad de diseñar para una gran diversidad y heterogeneidad de necesidades de acceso, cosa que puede implicar la puesta a disposición de varias versiones o de diseños adaptables según el usuario. Este trabajo se ha centrado en un colectivo particular como es el de las personas ciegas y en un tipo de producto muy concreto como son los mapas digitales. Algunas de las soluciones previstas, como por ejemplo, el acceso total a través de una interfaz de teclado, también resultan de utilidad para otros colectivos como el de discapacitados motrices. No obstante, para resultar accesible de manera universal seguramente se debería revisar de nuevo el producto y aplicar cambios a entre otros elementos, a la tipografía utilizada, al contraste de colores o al tamaño de los elementos de la interfaz (botones, cajas de texto, etc.).

En comparación con el resto de disciplinas relacionadas con el diseño de interacción y a pesar de los esfuerzos del W3C y de otras entidades nacionales e internacionales, o incluso de la obligación por parte de las administraciones públicas de crear productos accesibles, la accesibilidad sigue siendo la "cenicienta" en el proceso de diseño de webs y aplicaciones. Los beneficios evidentes del diseño web accesible, como pueden ser el incremento de la cuota de mercado a la que puede llegar una empresa, la posibilidad de conseguir un mejor posicionamiento en motores de búsqueda como Google o Bing o la posibilidad de reutilizar más fácilmente el contenido en múltiples formatos o dispositivos entre otros muchos, no parecen suficientes como para hacer de la accesibilidad una cuestión capital como ya lo son la usabilidad o la arquitectura de la información. Las cuestiones éticas tampoco parecen, en general, un argumento suficiente para concienciar a empresas y diseñadores. No obstante, en los últimos años estamos viviendo un interés creciente en esta disciplina, y sobre todo un aumento de recursos y estándares relacionados que no hacen más que augurar un futuro prometedor a la accesibilidad web.

11.2. Conocimientos adquiridos

La realización de este trabajo final de máster empezó en julio de 2013 con el autoaprendizaje del lenguaje de programación JavaScript. Hasta el momento no conocía ningún lenguaje de programación, más allá de algunas horas dedicadas a Visual Basic en el marco de la programación de algunas macros para varias aplicaciones del paquete ofimático Microsoft Office. Si bien sigo sin dominar este lenguaje de programación, actualmente tengo las bases necesarias para realizar pequeños programas y aplicarlo, no sin límites evidentes, a pequeños proyectos que lo utilicen como el del prototipo del apartado 8 de este trabajo. La necesidad de conocer este lenguaje de programación resultaba imprescindible para poder trabajar no sólo

con la API de Google Maps, sino también con cualquiera del resto de APIs analizadas en el apartado 5. La utilidad de estos conocimientos adquiridos no se limita a su aplicación a estas tecnologías, sino que me servirán en el futuro para mejorar las funcionalidades de cualquier producto web que pueda desarrollar.

Una vez adquiridos los conocimientos mínimos de JavaScript necesarios para comprender la documentación de la API de Google Maps, me inicié en el uso de esta interfaz de programación. Un proceso que documenté, dando lugar a un tutorial de uso de esta API que en un futuro cercano pretendo compartir.

Retomar las WCAG, trabajadas en la asignatura de accesibilidad de este máster, me ha permitido profundizar aún más en unas directrices que no sólo permiten comprender las necesidades de los diferentes colectivos de personas discapacitadas, sino que además aportan como complemento una gran cantidad de técnicas en vistas a implementar los diferentes criterios de conformidad. Además, el uso de una aplicación basada en parte en AJAX me ha obligado a trabajar con WAI-ARIA para conseguir que esta aplicación enriquecida fuera accesible.

Finalmente, me he familiarizado con el uso de algunos lectores de pantalla como ChromeVox y JAWS.

En definitiva, la realización de este trabajo me ha permitido conocer y trabajar con diferentes tecnologías y estándares que seguro me ayudarán en proyectos futuros, tanto en el ámbito de la accesibilidad como en otros ámbitos relacionados.

11.3. Trabajo futuro

Con respecto al prototipo, en ningún momento el objetivo fue desarrollar una aplicación totalmente acabada. No obstante, el interés creciente por mi parte en JavaScript y la API de Google Maps, y el hecho de ir viendo como de un mapa simple logré pasar a una pequeña aplicación con mayores funcionalidades, me llevó a dedicar una importante cantidad de horas a esta parte del trabajo. En este sentido, en un futuro mi intención es seguir trabajando en la aplicación, mejorándola y añadiendo nuevas funcionalidades entre las que se encuentran:

- Dar la posibilidad a los usuarios de seleccionar uno de los puntos de interés obtenidos en los resultados de búsqueda como lugar de destino.
- La actualización automática de los resultados de búsqueda si el usuario nueve el centro del mapa.
- Añadir la función de autocompletar presente en el servicio de rutas al buscador de puntos de interés.
- Categorizar los resultados de búsqueda según el tipo de servicio o establecimiento.
- Permitir a los usuarios saltar de los resultados de búsqueda al mapa y del mapa a los resultados de búsqueda.
- Integrar datos abiertos liberados por Ayuntamientos con información sobre equipamientos y servicios municipales.
- Pensar en la adaptación total del producto al entorno móvil a través de una interfaz compatible con dispositivos móviles o la creación de una aplicación.

Bibliografía

3. Justificación

Arch, Andrew; Letourneau, Chuck (2002). *Beneficios auxiliares del diseño web accesible*. <http://www.w3.org/2003/11/benefits-es.html>. [Consulta: 26/05/2014].

Bruce I.; McKennell A.; Walker E. (1991). *Blind and partially sighted adults in Britain: the RNIB survey*. London, Her Majesty's Stationery Office. Vol. 1.

Carreras Montoto, Olga. "Accesibilidad web y SEO". En: Paz, Lorena (comp.); Malumián, Víctor (ed.). *Pioneros y hacedores: fundamentos y casos de diseño de interacción con estándares de accesibilidad y usabilidad*. Buenos Aires: Gogot, 2013. Versión en línea disponible en: <http://www.edicionesgodot.com.ar/pionerosyhacedores/>. [Consulta: 26/05/2014].

Dolson, Joseph (2012). "SEO and accessibility". En: *Accessible Web design. Blog*. <http://www.joedolson.com/articles/2012/01/seo-accessibility/>. [Consulta: 26/05/2014].

"Encuesta de discapacidad, autonomía personal y situaciones de dependencia (EDAD). Año 2008". *Notas de prensa*. <http://www.ine.es/prensa/np524.pdf>. [Consulta: 31/05/2014].

Encuesta sobre discapacidades, autonomía personal y situaciones de dependencia. Madrid: Instituto Nacional de Estadística, 2008.

García Soria, F.; Ruiz Prieto, P. (2010) " Mapas geográficos para personas ciegas y deficientes visuales". *Integración: revista sobre discapacidad visual*. Nº 57 (mayo/agosto). <http://www.once.es/new/servicios-especializados-en-discapacidad-visual/publicaciones-sobre-discapacidad-visual/nueva-estructura-revista-integracion/copy_of_numeros-publicados/numero_57/mapas-geograficos-para-personas-ciegas>. [Consulta: 26/05/2014].

Gerber, Elaine (2003). "The benefits of and barriers to computer use for individuals who are visually impaired". *Journal of visual impairment & blindness*. Vol. 97, issue 9, p. 536-550.

Nielsen, Jakob (2000). *Usabilidad, diseño de sitios web*. Madrid [etc.]: Prentice Hall.

Nielsen, Jakob (2012). "SEO and Usability". En: *Nielsen Norman Group. Articles*. <http://www.nngroup.com/articles/seo-and-usability/>. [Consulta: 26/05/2014].

Romero Zúnica, Rafael. "Usabilidad como ventaja competitiva". En: *Accesibilidad a la Red*. <http://acceso.uv.es/accesibilidad/artics/01-usab-ventaja.htm>. [Consulta: 26/05/2014].

4. Descripción del colectivo objetivo

Brabyn, J. (1995). "Orientation and navigation systems for the blind: an overview of different approaches". En: *Conference on orientation and navigation systems for blind persons*, Hatfield, 1-2 February 1995.

Carreiras, Manuel; Codina, Benito (1993). "Cognición espacial, orientación y movilidad: consideraciones sobre la ceguera". *Integración: revista sobre ceguera y deficiencia visual.* Nº 11 (feb.). <http://www.once.es/new/servicios-especializados-en-discapacidad-visual/publicaciones-sobre-discapacidad-visual/nueva-estructura-revista-integracion/copy_of_numeros-publicados/integracion-pdf/Integracion-11.pdf>. [Consulta: 26/05/2014].

"Discapacidad visual: aspectos generales". En: *ONCE*. <http://www.once.es/new/servicios-especializados-en-discapacidad-visual/discapacidad-visual-aspectos-generales>. [Consulta: 01/06/2014].

Downs, Roger M (1977). *Maps in minds: reflections on cognitive mapping*. New York [etc.]: Harper and Row. (Harper & Row series in geography).

Golledge, Reginald G. (1993). "Geography and the disabled: a survey with special reference to vision impaired and blind populations". *Transactions of the Institute of British Geographers*. New Series, Vol. 18, no. 1, p. 63-85. <http://www.jstor.org/stable/623069>. [Consulta: 26/05/2014].

Golledge, Reginald G. (1995). "Primitives of spatial knowledge". En: *Cognitive aspects of human-computer interaction for geographic information systems*. Amsterdam: Kluwer Academic Publishers, p. 29-44.

Hill Everett W., et al. (1993). "How persons with visual impairments explore novel spaces: strategies of good and poor performers". *Journal of visual Impairment and blindness*. Vol. 87, no. 8, p. 295-30 1.

Jacobson, R. Dan; Kitchin, Robert M. (1997). "GIS and people with visual impairments or blindness: exploring the potential for education, orientation, and navigation". *Transactions in GIS*. Vol. 2, issue 4. <http://people.ucalgary.ca/~rjacobso/haptic/publications/TGIS.pdf>. [Consulta: 01/06/2014].

Kerr, N.H. (1983). "The role of vision in 'visual imagery' experiments: evidence from the congenitally blind". *Journal of experimental psychology: general*. No. 112, p. 265-277.

Loomis, Jack M.; Golledge, Reginald G.; Klatzy, Roberta L. (1998). "Navigation system for the blind: auditory display modes and guidance". *Presence*. Vol. 7, no. 2 (April), p. 193-203. <http://www.geog.ucsb.edu/pgs/papers/applied1.pdf>. [Consulta: 26/05/2014].

Muñoz Sevilla, J. A.; Blocona Santos, C. (2011). "Haciendo accesibles a las personas con discapacidad visual los Sistemas de Posicionamiento Global (GPS): el Proyecto HaptiMap". *Integración: revista sobre discapacidad visual*. Nº 61 (septiembre-diciembre). <http://www.once.es/new/servicios-especializados-en-discapacidad-visual/publicaciones-sobre-discapacidad-visual/nueva-estructura-revista-integracion/copy_of_numeros-publicados/numero-61/copy2_of_accesibilidad-gps-para-personas-con-discapacidad-visual>. [Consulta: 26/05/2014].

Power, Christopher. "Navigating, discovering and exploring the web: strategies used by people with print disabilities on interactive websites". En: Kotzé, Paula, et al. (ed.). *Human-Computer Interaction, INTERACT 2013*. (Lecture notes in computer science; vol. 8117), p. 667-684.

Spencer C.; Blades M.; Morsley K. (1989). *The child in the physical environment: the development of spatial knowledge and cognition*. New York, NY: John Wiley & Sons.

Strelow, E.R. (1985). "What is needed for a theory of mobility: direct perception and cognitive maps-lessons from the blind". *Psychological review*. Vol. 92 (April), p. 226-248.

Theofanos, M.F., Redish, J. (2003). "Bridging the gap: between accessibility and usability". *Interactions*. Vol. 10, issue 6 (Nov./Dec. 2003), p. 36-51.

5. Panorama del software y aplicaciones relacionadas con la información geográfica

Asadi, Saeid, et al. (2007). "Location-based search engines tasks and capabilities: a comparative study". *Webology*. Vol. 4, nº 4. <http://www.webology.org/2007/v4n4/a48.html>. [Consulta: 26/05/2014].

Burrough P.A. (1986). "Principles of geographical information systems for land resources assessment". *Geocarto international*. Vol. 1, issue 3.

Botella Plana, Albert (2011). "Bases de datos geográficos: almacenes de datos geográficos". En: Pérez Navarro, Antoni (coor). *Introducción a los sistemas de información geográfica y geotelemática*. Barcelona: Editorial UOC.

Rodríguez Lloret, Jesús; Olivella González, Rosa (2011). "Introducción a los sistemas de información geográfica: conceptos y operaciones fundamentales". En: Pérez Navarro, Antoni (coor). *Introducción a los sistemas de información geográfica y geotelemática*. Barcelona: Editorial UOC.

6. Las WCAG 2.0

Ribera, Mireia (2009). "La nueva normativa de accesibilidad WCAG 2.0 y los documentos en Internet". *Hipertext.net*. Núm. 7 (2009). <http://www.upf.edu/hipertextnet/numero-7/wcag-2-0.html>. [Consulta: 26/05/2014].

"Visual Disabilities. Blindness". En: *WebAIM. Articles*. <http://webaim.org/articles/visual/blind>. [Consulta: 13/10/2013].

Web Content Accessibility Guidelines (WCAG) 2.0. W3C Recommendation 11 December 2008. <http://www.w3.org/TR/WCAG20/>. [Consulta: 01/06/2014].

7. WAI-ARIA

Accessible Rich Internet Applications (WAI-ARIA) 1.0. W3C Recommendation 20 March 2014. <http://www.w3.org/TR/wai-aria/>. [Consulta: 01/06/2014].

"ARIA Techniques for WCAG 2.0". En: *Techniques for WCAG 2.0*. 2008. <http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/aria.html>. [Consulta: 31/05/2014].

Carreras, Olga. " HTML5 y accesibilidad: nuevos tipos de input, atributos asociados y validación nativa". *Usable & accesible* (marzo 2014). <http://olgacarreras.blogspot.com.es/2014/03/html5-y-accesibilidad-nuevos-tipos-de.html>. [Consulta: 31/05/2014].

"Client-side scripting techniques for WCAG 2.0". En: *Techniques for WCAG 2.0*. 2008. <http://www.w3.org/TR/2008/NOTE-WCAG20-TECHS-20081211/client-side-script.html>. [Consulta: 31/05/2014].

Torres Burriel, Daniel. "Crónica de USID 07". *TorresBurriel.com*. 2007. <http://www.torresburriel.com/weblog/2007/05/21/cronica-de-usid-07/>. [Consulta: 31/05/2014].

8. Entrevista/test con usuario

Hassan Montero, Yusef; Ortega Santamaría, Sergio (2009). *Informe APEI sobre usabilidad*. Gijón: Asociación Profesional de Especialistas en Información. (Informes APEI; 3). <http://hdl.handle.net/10760/13253>. [Consulta: 02/06/2014].

Sloan, David; Health, Andy; Hamilton, Fraser (2006). "Contextual web accessibility: maximizing the benefit of accessibility guidelines". En*: International cross-disciplinary workshop on*

web accessibility (W4A). <http://opus.bath.ac.uk/402/3/w4a-2006-contextual-accessibility.pdf>. [Consulta: 02/06/2014].

9. Prueba de concepto

API de Google Maps. En: *Google developers*. Última actualización: Enero 31, 2013. <https://developers.google.com/maps/?hl=es>. [Consulta: 01/06/2014].

Flanagan, David (2011). *JavaScript : the definitive guide*. 6th ed. Sebastopol : O'Reilly.

jQuery Community Experts (ed.) (2010). *jQuery cookbook*. Sebastopol, CA; Cambridge: O'Reilly.

McFarland, David Sawyer (2012). *Javascript y jQuery*. Madrid: Anaya Multimedia.

Svennerberg, Gabriel (2010). *Beginning Google Maps API 3*. [Berkeley, CA]: Apress.

Udell, Sterling (2009). *Beginning Google Maps mashups with mapplets, KML and GeoRSS: from novice to professional*. Berkeley, CA: Apress ; New York, NY: Distributed by Springer-Verlag New York.