ACE 33

Electronic offprint Separata electrónica

HERRAMIENTA DE VISUALIZACIÓN DE RUTAS ACCESIBLES EN ESPACIOS URBANOS UTILIZANDO TECNOLOGÍA HTML5

Francesc Valls Dalmau y Josep Roca Cladera

Cómo citar este artículo: VALLS, F. y ROCA, J. *Herramienta de visualización de rutas accesibles en espacios urbanos utilizando tecnología HTML5* [en línea] Fecha de consulta: dd-mm-aa. En: ACE: Architecture, City and Environment = Arquitectura, Ciudad y Entorno, 11 (33): 251-264, 2017. DOI: 10.5821/ace.11.33.5138. ISSN: 1886-4805.

ACE Architecture, City, and Environment Arquitectura, Ciudad y Entorno

Journal of the Centre of Land Policy and Valuations | Polytechnic University of Catalonia and of The Thematic Network Architecture, City and Environment

ACE 33

Electronic offprint Separata electrónica

USING HTML5 TECHNOLOGY TO BUILD AN ONLINE VISUALIZATION TOOL FOR ACCESSIBLE ROUTING IN URBAN SPACES

Key words: Accessibility; Web application; Itineraries; Geolocation

Structured abstract

Despite the location and route planning capacity of mobile devices, there are significant shortcomings regarding route planning according to the needs of people with accessibility difficulties.

This paper discusses the development of a web application to visualize optimal routes for these users using HTML5 technology with open source tools. The development strategy iterated several cycles of coding and assessment, allowing the adjustment of functionality and usability in the early stages.

The result is compatible with desktop and mobile platforms because of the adoption of web standards, such as the <canvas> element and the Geolocation API. Through the web application, easily embeddable in a webpage, users can select a point in a map (using the mouse or touchscreen) and the most suitable accessible route from that point to a fixed element of interest in a map is shown and (optionally) overlaid on the Google Maps cartography.

Additionally, if the device is on the area shown on the map, its approximate position is displayed. The resulting application simplifies both the planning of the route and the on-site orientation, and is also valuable as a visual validation tool during the development of the optimal route model.

ACE Architecture, City, and Environment
Arquitectura, Ciudad y Entorno

Journal of the Centre of Land Policy and Valuations | Polytechnic University of Catalonia and of The Thematic Network Architecture, City and Environment

HERRAMIENTA DE VISUALIZACIÓN DE RUTAS ACCESIBLES EN ESPACIOS URBANOS UTILIZANDO TECNOLOGÍA HTML5

VALLS DALMAU, Francesc[1]
ROCA CLADERA, Josep[2]

Remisión inicial: 19-06-2016 **Remisión final:** 16-02-2017

Palabras clave: Accesibilidad, Aplicación Web, Itinerarios, Geolocalización

Resumen estructurado

Pese a la capacidad de los dispositivos móviles de localización y planificación de itinerarios, existe un importante vacío en cuanto a la planificación de rutas adecuadas a las necesidades de las personas con dificultades de accesibilidad.

En este artículo se describe el desarrollo de una aplicación web para visualizar las rutas óptimas para usuarios de estas características utilizando tecnología HTML5 con herramientas de código abierto. La estrategia de desarrollo de la aplicación consistió en la iteración de varios ciclos de programación y valoración, lo que permitió ajustar la funcionalidad y la usabilidad en fases muy tempranas.

El resultado es compatible con las plataformas móviles y de escritorio por la utilización de elementos web estándar, como el elemento *<canvas>* y la API de Geolocalización. A través de la aplicación web, fácilmente imbebible en cualquier página, los usuarios pueden seleccionar un punto del mapa (con el ratón o en una pantalla táctil) y se muestra la ruta accesible más adecuada desde este punto a un elemento de interés en un mapa, con la posibilidad de superponerla a la cartografía de Google Maps.

Adicionalmente, si el dispositivo se encuentra en el área correspondiente al mapa, se muestra una indicación de su ubicación estimada. La aplicación resultante facilita tanto la planificación de la ruta como la orientación in-situ, y resulta valiosa también como herramienta de validación visual durante el desarrollo del modelo de cálculo de rutas.

[1] M.Sc. Arquitecto. Estudiante del Doctorado en Gestión y Valoración Urbana y Arquitectónica e Investigador en Formación del Departamento de Representación Arquitectónica, RA, de la Universidad Politécnica de Cataluña, UPC. Correo electrónico: francesc.valls@upc.edu
[2] Dr. Arquitecto. Catedrático del Departamento de Tecnología de la Arquitectura, TA, de la Universidad Politécnica de Cataluña, UPC. Director del Centro de Política de Suelo y Valoraciones, CPSV. Correo electrónico: josep.roca@upc.edu

1. Introducción

La orientación espacial en entornos que no son familiares es un problema complejo que requiere de un enfoque multidisciplinar (Gibson, 2009). En el caso de entornos urbanos es especialmente complejo puesto que las estrategias de búsqueda son diferentes en función de la escala del espacio, debido a que en espacios de mayores dimensiones el coste de desplazamiento es significativo para distancias mayores (Pingel y Schinazi, 2014).

Este problema ha sido parcialmente resuelto recientemente con los dispositivos móviles, y actualmente prácticamente todo el mundo lleva en el bolsillo un teléfono que incorpora un GPS y dispone de conexión a Internet. La actual generación de teléfonos inteligentes es capaz de ejecutar una aplicación que averigua la posición del usuario, se descarga y muestra la cartografía correspondiente a su entorno inmediato y en caso necesario indica las direcciones a seguir en un lenguaje natural, teniendo en cuenta el modo de transporte y las condiciones de movilidad en el momento de la consulta.

Sin embargo, la percepción del espacio y la capacidad de orientarse eficazmente no son las mismas para todas las personas (Allen, 1999) y en el caso de personas con algún tipo de dificultad de accesibilidad entraña algunos retos (Golledge *et al.*, 2000), no solventados satisfactoriamente por las aplicaciones estándares (del Moral y Delgado 2010).

La difusión de mapas interactivos de la accesibilidad física de los entornos urbanos a través de Internet (Queraltó y Valls 2010) conlleva una serie de ventajas para los usuarios, diseñadores y la ciudadanía en general.

- Para los usuarios con algún tipo dificultad de accesibilidad, permite por una parte planear su itinerario de manera previa al desplazamiento, anticipando las dificultades que puedan existir, buscando alternativas o descartando la visita. Por otra parte, una vez en el entorno, permite seguir un itinerario adaptado a sus necesidades.
- Para los diseñadores representa una herramienta para evaluar el grado de accesibilidad de un espacio y en caso necesario plantear alternativas de mejora.
- Para la ciudadanía es útil como elemento de difusión y concienciación acerca de las carencias de los espacios.

El objetivo de la aplicación web descrita en este artículo es generar una plataforma útil para las personas con dificultades de accesibilidad, tanto como herramienta de planificación de itinerarios como de orientación in-situ, y al mismo tiempo servir como herramienta de validación de accesibilidad para diseñadores urbanos y también difundir la necesidad de incidir en la accesibilidad de los espacios urbanos para mejorar la vida de todos sus habitantes.

2. Prototipo sobre tecnología Java

2.1 Elección de la plataforma de desarrollo

Siguiendo los principios de la filosofía de desarrollo de software "release early, release often" (Raymond, 1999; 28) se decidió realizar un prototipo inicial con una herramienta que permitiera un desarrollo rápido de la aplicación, para obtener las impresiones acerca de la usabilidad por parte de los usuarios de prueba en las fases iniciales del desarrollo e incorporar sus sugerencias lo más pronto posible y de esta manera mejorar la calidad del producto final. Esta filosofía también permitió también posponer el desarrollo de algunos algoritmos en las fases iniciales e implementarlas más adelante, una vez cercioradas la usabilidad y la aplicabilidad de la herramienta.

Para el desarrollo del prototipo se utilizó el lenguaje *NetLogo* 5.2 (Wilensky, 1999), un lenguaje de programación de código abierto[3] focalizado en el desarrollo de modelos basados en agentes. *NetLogo* es multiplataforma (Windows, macOS y Linux) gracias a utilizar la Máquina Virtual Java, aunque está desarrollado principalmente en el lenguaje de programación *Scala*.

La modelización basada en agentes aprovecha la filosofía de los lenguajes de programación orientados a objetos, puesto que en el modelo computacional cada agente es una instancia de un objeto que encapsula unos métodos y unas propiedades heredadas de su clase, pudiéndolas extender a través de subclases.

NetLogo proporciona cuatro clases de agente: tortugas (*turtles*), celdas (*patches*), conexiones (*links*) y observador (*observer*). Para el desarrollo del prototipo se aprovecharon las funcionalidades que proporcionaban las dos primeras clases:

- La posibilidad de incorporar datos espaciales provenientes de un Sistema de Información Geográfica como variables asignadas a las celdas.
- La capacidad de los agentes móviles de "sentir" el entorno, es decir, de acceder a las variables almacenadas en forma de matriz de celdas en el espacio dónde deambulaban.
- La funcionalidad heredada de la clase que permitía a los agentes simulados tener noción de dónde se encontraban en el espacio virtual y de su entorno inmediato.
- La flexibilidad de programar el comportamiento de los agentes móviles a partir de la definición de unas reglas.

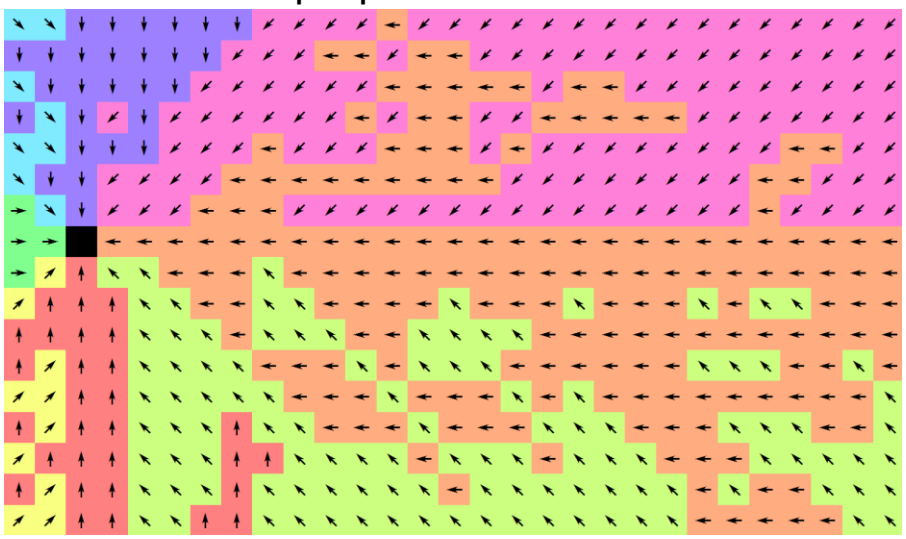
Otras ventajas adicionales proporcionadas por *NetLogo* en las fases iniciales de desarrollo del prototipo fueron consecuencia de la incorporación de un entorno de desarrollo integrado[4] que facilitaba tanto la construcción de las distintas versiones de interfaz de usuario como la creación de gráficos en tiempo real a partir del desarrollo de la ejecución del programa. Finalmente, la compacidad del código, así como la posibilidad de modificar a las instrucciones del programa y comprobar el resultado de los ajustes sin necesidad de complicación aceleró enormemente el desarrollo y permitió disponer de un prototipo rápidamente, así como realizar los ajustes necesarios de una manera ágil.

[3] Durante el desarrollo de proyecto se contribuyó con un "parche" para uno de los modelos de ejemplo de NetLogo, disponible en GitHub en https://github.com/NetLogo/models/pull/103 (en el momento de escribir este artículo)
[4] En inglés, Integrated Development Environment (IDE)

2.2 Datos utilizados

El recorrido a mostrar por parte de los agentes se obtuvo a partir de un modelo realizado por el equipo de escáner láser terrestre del Laboratorio de Modelización Virtual de la Ciudad de la Escuela de Arquitectura de Barcelona (Universidad Politécnica de Cataluña) a partir del cálculo de las rutas óptimas en una superficie de coste generada a partir de distintos parámetros de accesibilidad con el programa *ArcGIS*. (ESRI) La herramienta utilizada producía como resultado una retícula dónde cada píxel almacenaba la dirección del siguiente píxel al que el agente debería desplazarse (**Figura 1**), de manera que para cualquier píxel era posible seguir una cadena de instrucciones que llevaba de forma determinista al destino con el mínimo coste posible.

Figura 1. **Ejemplo de campo vectorial con el recorrido que lleva de forma determinista al destino desde cualquier punto minimizando el coste acumulado**



Fuente: Elaboración propia a partir de datos proporcionados por el equipo del Laboratorio de Modelización Virtual de la Ciudad (Escuela de Arquitectura del Barcelona).

Estas direcciones estaban codificadas con números del 0 a 8, siendo 0 el destino y los números 1 al 8 las direcciones en una vecindad de *Moore* (**Figura 2**), en el sentido de las agujas del reloj, con el 1 situado en el origen de rotación según la convención matemática (Este) en lugar de la geográfica (Norte).

Figura 2. **Equivalencia de codificación de orientaciones: códigos provenientes de ArcGIS (izquierda), geográfica (centro) y grados sexagesimales (derecha)**

6	7	8
5	0	1
4	3	2

N-O	N	N-E
O	X	E
S-O	S	S-E

315	360	45
270	-1	90
225	180	135

Fuente: Elaboración propia.

2.3 *Incorporación y transformación de los datos*

La codificación se transformó a grados sexagesimales,[5] obteniendo el resto de la división euclidiana, multiplicando el resultado por 45 (una octava parte del total de grados de una rotación completa) y añadiendo 45 grados para trasladar el cero a la posición Norte, como se muestra en la Ecuación 1, siguiente.

Ecuación 1. **Conversión de códigos a ángulos**

$$dirección\ (grados) = \left(1 + \ código - 8\ \left\lfloor \frac{código}{8} \right\rfloor\right) \times \frac{360}{8}$$

Fuente: Elaboración propia

Se creó un atributo para los agentes tipo celda que almacenaría la dirección en la que se encontraba el píxel al que un agente que se moviera minimizando el total acumulado en la superficie de coste debería dirigirse a continuación. Esta información, almacenada como el color de una imagen, se traspasó a este atributo para cada uno de los píxeles de la rejilla,[6] excepto para dos tipos:

- Uno de los píxeles almacenaba un valor reservado para denotar que era el destino y que el agente se debía detener al llegar a él
- Los píxeles por dónde no se puede circular (por ejemplo, por estar dentro de un edificio) se codificaron como no válidos

Estos atributos no eran visibles, puesto que el color de cada píxel se almacenaba independientemente en otra variable, de manera que se podía mostrar al usuario imagen que se deseara.[7]

2.4 *Definición de las reglas*

A continuación, se definió la lógica que debían seguir los agentes para llegar a su destino siguiendo las instrucciones almacenadas en la retícula de píxeles, dejando una trayectoria al moverse[8] una vez iniciada la ejecución:

1. Al pulsar con el ratón dentro del mapa, si el cursor se encontraba en una zona caminable, se iniciaba en dibujo de la trayectoria (reinicializando el mapa y por lo tanto borrando las trayectorias anteriores si existían); en caso contrario, no se realizaba ninguna acción.
2. A continuación, el agente entraba dentro de un bucle donde primero leía (de la celda donde estaba situado) la variable que almacenaba la dirección donde se encontraba el píxel al que debía desplazarse, para desplazarse a este a continuación.
3. En condiciones normales, la condición de salida de este bucle se producía cuando el agente leía el valor reservado que le indicaba que se había alcanzado el destino, momento en el que el programa pasaba a esperar una nueva acción por parte del usuario, para proceder a reanudar la ejecución desde el primer paso.

[5] Python: heading = (code % 8 + 1) * 360 / 8; NetLogo: heading = (code mod 8 + 1) * 360 / 8
[6] Un total de 501.501 (1001x501) píxeles
[7] Para este primer prototipo se utilizó primero un mapa de pendientes y posteriormente una ortofoto en la versión applet para web.
[8] Siendo NetLogo de una variante del lenguaje Logo, los agentes (tortugas) disponen de un "rotulador virtual" que puede trazar una línea cuando se mueven

4. En el caso de que durante este bucle se produjera un clic en un punto válido del mapa, se interrumpía el bucle y se reiniciaba la ejecución desde el primer paso.

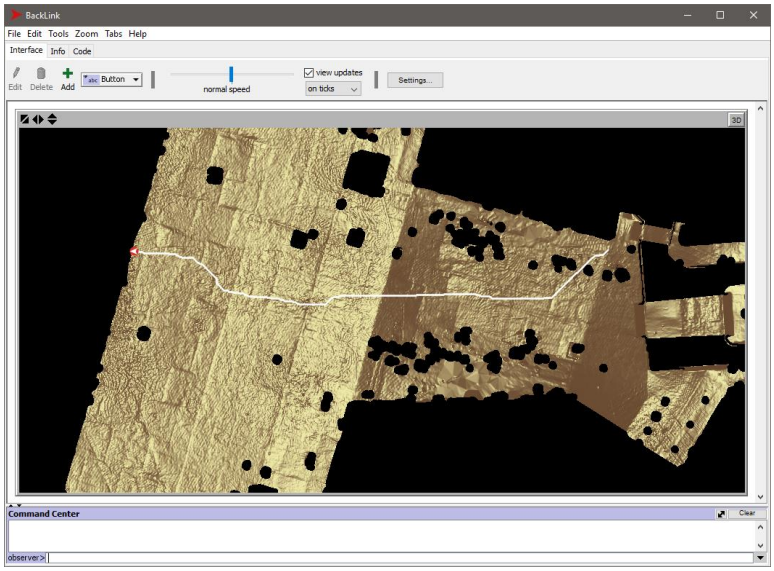
2.5 Prototipo funcional

El programa resultante (**Figura 3**) resultó una valiosa herramienta para explorar los resultados del modelo, permitiendo visualizar de manera intuitiva los itinerarios desde distintos puntos críticos, y ajustar los parámetros del modelo cuando fuese necesario.

Esta primera aproximación, permitió también refinar el interfaz de usuario en una fase muy temprana para que la aplicación fuera fácilmente entendible y trasmitiera la información necesaria de manera eficaz.

Aunque la tecnología de *applets* Java (pequeñas aplicaciones que se ejecutan en una Máquina Virtual Java y se visualizan dentro de la ventana de un navegador) está en desuso, principalmente por los problemas de seguridad que entrañan y la falta de soporte en dispositivos móviles, la versión 5.3 de *NetLogo*[9] permite todavía incrustar la aplicación en una página web,[10] siempre que configure el servidor correctamente,[11] se acompañe de unos archivos de soporte[12], y se incremente la memoria disponible para *applets* si la aplicación es compleja.[13]

Figura 3. **Captura de pantalla de prototipo una vez dibujada una trayectoria (los puntos no válidos aparecen en negro)**



Fuente: Elaboración propia a partir de datos proporcionados por el equipo del Laboratorio de Modelización Virtual de la Ciudad (Escuela de Arquitectura del Barcelona).

[9] La última versión de NetLogo (6.0) no soporta la creación de applets Java, según la información consultable en https://github.com/NetLogo/NetLogo/wiki/Applets (en el momento de escribir este artículo)
[10] Ver documentación disponible en el enlace http://ccl.northwestern.edu/netlogo/5.0/docs/applet.html (en el momento de escribir este artículo)
[11] Principalmente en lo referente a los tipos MIME
[12] NetLogoLite.jar y NetLogoLite.jar.pack.gz
[13] Las instrucciones se detallan en el enlace http://www-cpsv.upc.es/RecerCaixa/JavaReg.htm (en el momento de escribir este artículo)

Pese a estas limitaciones, la distribución mediante esta tecnología permitió ampliar el alcance de la distribución del prototipo, y recabar posibles mejoras a incorporar. Sin embargo, sus limitaciones motivaron la búsqueda de otras tecnologías más modernas de desarrollo de aplicaciones para web que soportaran dispositivos móviles y no requirieran el uso de *plug-ins* como *Flash* o *applets* Java.

3. Versión definitiva utilizando estándares web

3.1 Nuevos estándares HTML5

El desarrollo del prototipo permitió comprobar el correcto funcionamiento de la lógica del programa y al mismo tiempo refinar la experiencia de usuario en las fases iniciales. Sin embargo, para conseguir la mayor difusión de los resultados, se optó por re-implementar la aplicación con tecnologías más modernas, basadas en estándares web, capaces de ser ejecutadas en múltiples plataformas y factores de forma, que garantizaran que los resultados pudieran ser consultados por un mayor número de personas, en cualquier lugar y momento, utilizando cualquier dispositivo.
El uso de tecnologías web abiertas también facilita la indexación de los contenidos[14] por parte de los buscadores (incrementando la visibilidad) y permite el uso de la aplicación en las plataformas móviles presentes y futuras, sin depender de mecanismos de distribución y actualización centralizados (*app stores*).

HTML[15] es un lenguaje que describe el contenido y estructura de una página web, mientras que su apariencia se describe generalmente con CSS[16] y los elementos interactivos o automatización se controlan mediante JavaScript. HTML5 es la más reciente versión de HTML; se trata de una tecnología abierta, de manera que es accesible por cualquier navegador basado en estándares, lo que hoy en día incluye la mayoría de dispositivos de escritorio y móviles. Las aportaciones de HTML5 son de dos tipos:

- Se trata de una nueva versión del lenguaje HTML, con nuevos elementos y atributos.
- Incluye un conjunto de nuevas tecnologías que permiten crear aplicaciones web con mayores capacidades.

Para el desarrollo de la aplicación, HTML5 aporta dos elementos clave: el elemento <*canvas*>[17] (nuevo elemento) y el acceso a las capacidades de geolocalización de los dispositivos (nueva tecnología).

3.2 El elemento <canvas>

HTML5 introdujo el elemento <*canvas*>[18] para permitir dibujar gráficos dentro de una página web de manera parecida a como se hace en una ventana de un ordenador, permitiendo

[14] Especialmente importante con el desarrollo de la llamada Web Semántica
[15] HyperText Markup Language
[16] Cascading Style Sheets
[17] Traducido al español como "lienzo"
[18] Documentación acerca del elemento disponible en https://www.w3.org/TR/html5/scripting-1.html#the-canvas-element (en el momento de escribir este artículo)

producir aplicaciones web con un a funcionalidad mucho más rica, equiparable en algunas ocasiones a una aplicación nativa. El elemento está soportado por el 97.56% de los navegadores utilizados para acceder a Internet de manera global[19] (en el momento de escribir este artículo).

El elemento <*canvas*> es un espacio rectangular dentro de una página web, con la capacidad de que su contenido (*bitmap*) sea manipulado mediante JavaScript a bajo nivel,[20] complementando al estándar SVG,[21] que proporciona funcionalidad vectorial de manera estructurada.

Mediante la rápida actualización del contenido de este elemento se pudieron realizar las animaciones y responder a las acciones del usuario de manera interactiva, proporcionando una funcionalidad equivalente a una aplicación (móvil o de escritorio), pero operando dentro de una página web.

Una página puede albergar más de uno de estos elementos y la ejecución se produce en el cliente (mediante el uso de JavaScript) por lo que la carga en el servidor es mínima, puesto que se encarga únicamente de trasmitir los datos durante la carga inicial.

3.3 Geolocalización

Otra tecnología clave utilizada por la aplicación fue la HTML5 *Geolocation API*.[22] La tecnología está soportada por el 93.21% de los navegadores utilizados para acceder a Internet de manera global[23] (en el momento de escribir este artículo) y permite que el navegador obtenga la localización del dispositivo (si se le otorgan de los permisos necesarios) a través de sus sensores.

Para comprobar la precisión de la localización utilizando esta tecnología se desarrolló una página web24 que proporcionaba distintos parámetros de localización (Tabla 1), basada en la herramienta web de Andy Gup llamada HTML5 Geolocation Demo,25 utilizando el método navigator.geolocation.watchPosition26 con la opción “enableHighAccuracy.”27 El ensayo realizaba adicionalmente una petición a Google Maps API para mostrar un mapa y una ortofoto correspondientes a la localización para poder comparar la posición real con la posición detectada (
Figura 4), siendo muchas veces el error respecto a la precisión real (puesto que era conocida por estar físicamente en el sitio) mucho menor que la estimación de error que proporcionaban los dispositivos.

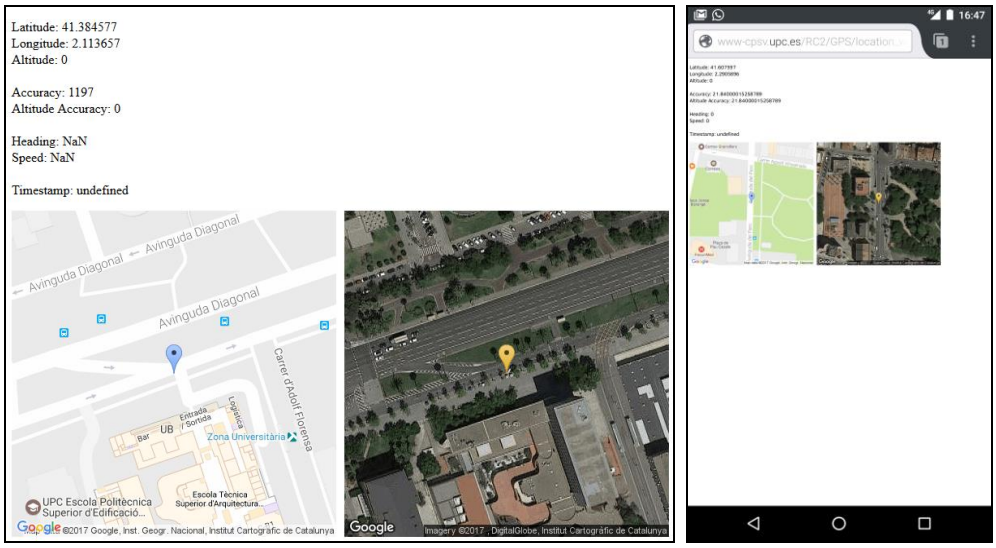
[19] Según la página http://caniuse.com/#feat=canvas (disponible en el momento de escribir este artículo)
[20] De manera parecida a las API 2D
[21] Scalable Vector Graphics
[22] Especificaciones de la Geolocation API (Application Programming Interface) disponibles en el enlace https://www.w3.org/TR/geolocation-API/ (en el momento de escribir este artículo)
[23] Según la página http://caniuse.com/#feat=geolocation (disponible en el momento de escribir este artículo)
[24] Disponible en http://www-cpsv.upc.es/RC2/GPS/location_web.html (en el momento de escribir este artículo)
[25] Disponible en http://andygup.net/samples/html5geo/ (en el momento de escribir este artículo)
[26] En lugar de navigator.geolocation.getCurrentPosition() para leer la posición de manera continua, según la información publicada en la página https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/watchPosition (en el momento de escribir este artículo)
[27] A expensas de un mayor consumo de batería en dispositivos móviles, según la información publicada en la página https://developer.mozilla.org/en-US/docs/Web/API/PositionOptions (en el momento de escribir este artículo)

Tabla 1. **Parámetros de localización disponibles en HTML5 Geolocation API y valores obtenidos en la aplicación de prueba**

Propiedad	Firefox Windows (sin GPS)	Firefox Android (con GPS)
latitude	Disponible	Disponible
longitude	Disponible	Disponible
altitude	No disponible (cero)	No disponible (cero)
accuracy	Alrededor de 1Km	Alrededor de 20m
altitudeAccuracy	No disponible (cero)	Igual valor que “accuracy”
heading	No disponible (nulo)	No disponible (cero)
speed	No disponible (nulo)	No disponible (cero)
timestamp	No disponible (nulo)	No disponible (nulo)

Fuente: Elaboración propia

Figura 4. **Página web creada para averiguar la precisión de la localización mediante HTML5 + JavaScript en ordenadores Windows (izquierda) y móviles Android (derecha), ambos utilizando Mozilla Firefox**



Fuente: Elaboración propia. Cartografía de Google Maps.

3.4 Processing

El lenguaje de programación de código abierto *Processing*[28] (Reas y Fry, 2014) está concebido para crear programas[29] ligados a las artes multimedia. Creado en 2001 por Casey Reas y Benjamin Fry, pertenecientes al Aesthetics and Computation Group del MIT Media Lab, se basa en el lenguaje Java y generalmente se ejecuta en una Máquina Virtual Java, aunque también es posible generar apps Android y ejecutables Java para Windows, macOS o Linux.

[28] Proyecto alojado en http://processing.org/ (en el momento de escribir este artículo)
[29] Los programas de Processing se denominan *Sketches*

Adicionalmente, las bibliotecas del proyecto *Procesing.js*[30] (desarrollado originalmente por John Resig) permiten ejecutar los programas de Processing en una página web sin requerir la instalación de ningún complemento[31], dentro de un elemento <*canvas*> utilizando JavaScript.

Adicionalmente, Processing.js permite la comunicación entre el código de Processing y el código JavaScript de la página web[32], admitiendo tres modalidades distintas:

- Acceder a objetos JavaScript de la página desde Processing
- Mezclar JavaScript y Processing
- Acceder a Processing desde JavaScript

Esta capacidad permitió desarrollar la aplicación utilizando Processing, con las ventajas de disponer de un lenguaje con tipos estrictos y las facilidades para generar gráficos interactivos, pero al mismo tiempo acceder a la localización a través del navegador utilizando la HTML5 Geolocation API via JavaScript.

4. Resultados

4.1 Aplicación web

La aplicación web fue desarrollada utilizando Processing 3.0, a partir de las experiencias obtenidas con el prototipo inicial. Para ello fue necesario por una parte replicar la funcionalidad ofrecida por *NetLogo* y por otra incorporar algunas mejoras de interfaz de usuario y relacionadas con la localización en tiempo real.

La aplicación mostraba en una región de la página web un mapa con el área analizada, dónde aparecían los puntos para los que se había calculado una ruta en colores más saturados y mayor brillo que el resto del mapa (**Figura 5**). Sobre este mapa se actualizaba constantemente la posición del dispositivo, representada con un punto rojo en el mapa rodeado de un círculo traslúcido de color naranja, correspondiente al error de localización reportado. Este círculo servía de ayuda para ubicar al usuario en el mapa.

Si el usuario hacia pulsaba una ubicación en la zona resaltada (con el ratón en un ordenador o con el dedo en una pantalla táctil), se dibujaba la ruta correspondiente. En caso que arrastrara el ratón o el dedo, se actualizaba la traza en tiempo real[33].

Para conseguir la funcionalidad deseada se utilizaron los siguientes componentes para cada uno de los mapas interactivos:

- Un archivo HTML con la página web
- Un archivo CSS con el diseño de la página
- El motor de Processing.js

[30] Proyecto alojado en http://processingjs.org/ (en el momento de escribir este artículo)
[31] Únicamente es necesario enlazar la biblioteca processing.min.js, disponible en https://raw.github.com/processing-js/processing-js/v1.4.8/processing.min.js (en el momento de escribir este artículo)
[32] Documentación disponible en http://processingjs.org/articles/jsQuickStart.html#combineprocessingandjavascript (en el momento de escribir este artículo)
[33] Aproximadamente cada 3ms

- Un archivo JavaScript con rutinas obtener la localización y traspasarla a Processing
- Un archivo Processing con las instrucciones del programa
- Un archivo Processing de clase encapsulando la funcionalidad para interpretar la ruta
- Un archivo de imagen con el mapa
- Un archivo de imagen con la codificación de la ruta (detallado en la subsección 2.2)

Figura 5. **Captura de pantalla de la aplicación ejecutándose en una página web una vez dibujada una trayectoria**

Observación: Los puntos de partida válidos se representan con colores más saturados y mayor brillo.
Fuente: Elaboración propia a partir de datos proporcionados por el equipo del Laboratorio de Modelización Virtual de la Ciudad (Escuela de Arquitectura del Barcelona).

El código HTML de la página define el diseño de la página (enlazando con un archivo CSS), establece el área del mapa y algunos textos, y carga el motor de Processing.js y otro script cuyo cometido es obtener la localización del navegador[34] y almacenarla en una variable global, accesible por *Processing.* A partir de este momento, *Processing* toma el control de elemento *<canvas>* y lo inicializa mostrando el mapa, realizando las siguientes acciones repetidamente:

1. Actualizar posición cada cierto intervalo definido a partir de los datos que le proporciona el navegador

[34] Parte del código se encargaba del manejo de excepciones cuando la localización no estaba disponible

2. Esperar a que el usuario haga pulse o arrastre sobre un elemento resaltado del mapa, para a continuación reinicializar el mapa (borrando la ruta existente si la hay), calcular y dibujar la nueva ruta en un buffer y, una vez dibujada, superponerla en el mapa

4.2 "Mashup" con Google Maps

A partir de los comentarios algunos usuarios, que consideraron que un mapa fijo sin posibilidad de hacer zoom no era suficientemente accesible, se realizó un ensayo para integrar Google Maps dentro de la aplicación, añadiendo una pestaña que añadía la posibilidad de dibujar la ruta superpuesta a la cartografía de Google Maps (
Figura 6), con posibilidad de ver el mapa como guía urbana, como ortofoto, o en 3D.

Para ello fue necesario obtener una API Key para poder utilizar la API de Google Maps y expandir la funcionalidad de la aplicación web, que consistió principalmente en crear dos funciones en el módulo de *Processing* y enlazarlas con el código que controlaba la cartografía de Google Maps:

- Una función para convertir pares de coordenadas cartesianas a geográficas
- Una función para generar una polilínea en el formato Simple Polylines[35] de Google Maps

De esta manera, cada vez que el usuario pulsaba sobre el mapa de la aplicación web, además de generar el buffer con la ruta, se generaba paralelamente una nueva polilínea en coordenadas geográficas, que posteriormente se enviaba a la cartografía de Google Maps, dónde también se superponía un rectángulo traslúcido con el área analizada.

5. Conclusiones

El proceso iterativo de desarrollo-validación de la herramienta, permitió ajustar la funcionalidad y la usabilidad en fases muy tempranas. El uso de lenguajes de alto nivel (*NetLogo* y *Processing*, y en menor medida *JavaScript*) posibilitó disponer de prototipos funcionales (desde el punto de vista del uso y de la implementación) sin tener que abordar las complejidades antes de que el diseño final y el interfaz de usuario estuviera definido.

Para el desarrollo tanto del prototipo como de la versión final, se utilizaron únicamente herramientas de código abierto, multiplataforma y si coste. El resultado final utiliza tecnologías web estándar y por lo tanto es también compatible con plataformas de escritorio y móviles, presentes y futuras.

La tecnología web elegida, junto a una interfaz de usuario intuitiva y minimalista, hizo que el resultado final es una herramienta suficientemente flexible y abierta para ser aplicada a cualquier ámbito urbano, puesto que únicamente es necesario actualizar los ficheros de datos (imagen de fondo y modelo de rutas) para obtener un nuevo mapa funcional.

[35] Documentación en https://developers.google.com/maps/documentation/javascript/examples/polyline-simple (en el momento de escribir este artículo)

Todo ello permitió el objetivo principal de obtener una aplicación útil para los usuarios con dificultades de accesibilidad, y la máxima diseminación del modelo desarrollado. Adicionalmente, la herramienta se mostró útil para visualizar y validar visualmente el modelo desarrollado.

Figura 6. **"Mashup" de la aplicación web con Google Maps**

Fuente: Elaboración propia a partir de datos proporcionados por el equipo del Laboratorio de Modelización Virtual de la Ciudad (Escuela de Arquitectura del Barcelona). Cartografía de Google Maps.

Agradecimientos

Este artículo ha sido posible, gracias a los trabajos realizados desde el Centro de Política de Suelo y Valoraciones (CPSV) y el Laboratorio de Modelización Virtual de la Ciudad (LMVC) en el marco del *"Ciudad sin barreras. Herramienta para la avaluación y visualización de la accesibilidad al espacio público, en base a tecnologías TLS, GIS i GPS,* dirigido por el profesor Josep Roca Caldera (coautor) financiado en el marco de la Convocatoria RecerCaixa 2013, en la temática *La casa y la ciudad adaptadas a las persones con discapacidad.* Por ello los autores agradecen a RecerCaixa y ACUP el desarrollo de la convocatoria, que posibilita publicar trabajos como este, resultantes de los proyectos de investigación que se pueden realizar gracias a ella.

Bibliografía

ALLEN, G.L. *Spatial Abilities, Cognitive Maps, and Wayfinding: Bases for Individual Differences in Spatial Cognition and Behavior.* En: R.G. GOLLEDGE (ed.), *Wayfinding behavior: cognitive mapping and other spatial processes*. Baltimore: Johns Hopkins University Press, 1999. pp. 46-80. ISBN 978-0-8018-5993-9

DEL MORAL, C. y DELGADO, L. *Evaluación de los niveles de accesibilidad en los entornos patrimoniales.* En: ACE: Architecture, City and Environment, 5 (13): 41-60, 2010. Disponible en: <http://upcommons.upc.edu/handle/2099/9198>

GIBSON, D. *The Wayfinding Handbook: Information Design for Public Places*. 1st. S.l.: Princeton Architectural Press. 2009. 152 p. ISBN 1-56898-769-2

GOLLEDGE, R.G., JACOBSON, R.D., KITCHIN, R. y BLADES, M. *Cognitive Maps, Spatial Abilities, and Human Wayfinding.* En: Geographical review of Japan, Series B. 73 (2): 93-104, 2000. DOI 10.4157/grj1984b.73.93

PINGEL, T.J. y SCHINAZI, V.R. *The Relationship Between Scale and Strategy in Search-Based Wayfinding.* En: Cartographic Perspectives, 0 (77): 33-45, 2014. ISSN: 1048-9053. DOI 10.14714/CP77.1232

QUERALTÓ, P. y VALLS, F. *Herramienta de cálculo de rutas óptimas según parámetros de accesibilidad física en itinerarios urbanos.* En: ACE: Architecture, City and Environment, 5 (13): 161-184, 2010. Disponible en: <http://upcommons.upc.edu/handle/2099/9204>

RAYMOND, E.S. *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. S.l. O'Reilly Media. 1999. 258 p. ISBN 1-56592-724-9.

REAS, C. y FRY, B. *Processing: a programming handbook for visual designers and artists*. Second edition. Cambridge, Massachusetts: The MIT Press. 2014. 642 p. ISBN 978-0-262-02828-8. QA76.6 .R4138 2014

WILENSKY, U. *NetLogo.* En: Evanston (IL), USA: Center for Connected Learning and Computer-Based Modeling, Northwestern University. [En línea], 1999 [Fecha de consulta: 12 enero 2016]. Disponible en: <http://ccl.northwestern.edu/netlogo/>