Universidad Zaragoza

Trabajo Fin de Grado

Desarrollo de una aplicación web para la navegación por entornos tridimensionales basados en imágenes panorámicas georreferenciadas y nubes de puntos capturadas con LIDAR

Autor

Lukas Gedvilas

Director

Rubén Béjar Hernández

Escuela de Ingeniería y Arquitectura

Septiembre de 2014

Desarrollo de una aplicación web para la navegación por entornos tridimensionales basados en imágenes panorámicas georreferenciadas y nubes de puntos capturadas con LIDAR

Resumen

Este trabajo consiste en el desarrollo de una aplicación web que permita la navegación por entornos tridimensionales basados en imágenes panorámicas georreferenciadas y, además, la interacción con el entorno aprovechando la información de las nubes de puntos capturadas con la tecnología LIDAR para poder ofrecer funciones tales como la posibilidad de toma de medidas del mundo real o anotar datos sobre objetos concretos visualizados por el sistema. Todo ello sin la necesidad de instalar ningún complemento en el navegador para disfrutar de todas las funciones.

La funcionalidad descrita es de especial interés para el análisis detallado de localizaciones concretas para, por ejemplo, proyectos de construcción de carreteras, edificios, etc.

Al comienzo de la realización del trabajo, se ha llevado a cabo un análisis de sistemas que ofrezcan una funcionalidad similar. Tras la realización de dicho análisis, se ha procedido a documentar todos los requisitos que debe satisfacer el sistema, así como sus casos de uso.

A continuación, se ha realizado el proceso de familiarización con el contexto del proyecto. Por una parte, se ha profundizado en el tema de las nubes de puntos, sobre todo buscando librerías y herramientas de código libre para su manipulación. Y por otra, se han adquirido los conocimientos básicos sobre *three.js*, una librería para el desarrollo de aplicaciones 3D en el navegador web, siguiendo sus tutoriales.

Seguidamente, se ha procedido a realizar el diseño arquitectónico del sistema. Se han identificado 2 alternativas para el diseño: implementar la funcionalidad que aprovecha las nubes de puntos en el servidor y que el cliente acceda a ella mediante servicios web o, en cambio, implementar dicha funcionalidad enteramente en el cliente. Se ha procedido a la implementación del primer prototipo basado en la primera alternativa de diseño. Los resultados del prototipo no han sido satisfactorios en cuanto al rendimiento y precisión, así que se ha rediseñado el sistema para optar por la segunda alternativa. Entonces, se ha procedido a implementar un prototipo de la aplicación de consola para el procesamiento de nubes de puntos y un prototipo del cliente que utilice los ficheros generados por la aplicación de consola anteriormente mencionada para visualizar la nube de puntos y que el usuario pueda seleccionar puntos de dicha nube, además de la primera versión de la función para poder medir la distancia entre dos puntos. A continuación, se ha implementado un prototipo del servidor web que ofrezca al cliente toda la información necesaria sobre los escenarios del sistema mediante servicios web y se ha modificado el prototipo del cliente para que aproveche dichos servicios web. Y, finalmente, se ha implementado la versión final del cliente con toda la funcionalidad necesaria para satisfacer los requisitos del sistema.

Al final de la implementación de cada prototipo, se han realizado pruebas de sistema para comprobar si toda la funcionalidad ofrecida por el sistema se puede desempeñar correctamente.

Índice general

1 Introducción

Esta sección consiste en una introducción al contexto del proyecto, así como mi motivación para su realización.

1.1 Contexto del proyecto

1.1.1 Colaboración con IAAA

He realizado este proyecto siendo colaborador del Grupo de Sistemas de Información Avanzados (IAAA), un grupo de I+D adscrito al Instituto de Investigación en Ingeniería de Aragón de la Universidad de Zaragoza. Más concretamente, en su área de Sistemas de Información Geográfica (SIG).

1.1.2 Contexto tecnológico

Es importante entender que para que sea posible construir el sistema objetivo de este trabajo, hay ciertas tecnologías que deben estar necesariamente presentes.

Como el sistema debe permitir la navegación por entornos tridimensionales, es necesario tomar fotografías de todos los entornos que se quieran ofrecer en el sistema. Para ofrecer el mayor grado de similitud con el entorno real, las imágenes capturadas deben ser panorámicas y, por lo tanto, hechas por cámaras especiales. Además, dichas imágenes deben ser georreferenciadas, es decir, para cada imagen tomada es necesario saber su información geográfica, obtenida gracias al sistema de posicionamiento global (GPS). De esta forma, las imágenes pueden ser correctamente posicionadas dentro de su entorno.

Además, como el sistema no se limita a simplemente ofrecer la navegación por el entorno, sino que adicionalmente se debe poder interactuar con el entorno mediante funcionalidades como la toma de medidas del mundo real representado por el sistema, es necesario capturar información complementaria a las imágenes de los entornos. Esta información adicional se captura con LIDAR (Light Detection and Ranging) [1], una tecnología que permite determinar la distancia desde un emisor láser a un objeto o superficie utilizando un haz láser pulsado y, gracias a ello, producir una nube de puntos de un entorno.

Las nubes de puntos son ficheros, normalmente generados con tecnología LIDAR, que almacenan un conjunto de puntos de un entorno. La información almacenada por cada punto debe contener, como mínimo, las coordenadas de dicho punto. Adicionalmente, se puede almacenar información adicional como la información de color o la intensidad, entre otras muchas propiedades. Hay varios formatos en los que pueden ser almacenadas las nubes de puntos pero en este proyecto se ha utilizado el formato LAS [2]. La Figura 1 muestra la nube de puntos con la que se han realizado las pruebas del proyecto.

Normalmente, las tecnologías descritas en este apartado se suelen juntar en un vehículo (tanto terrestre como aéreo) y se hace una pasada por el entorno deseado para capturar toda su información (las imágenes georreferenciadas y la nube de puntos). La Figura 2 muestra el equipo, que se coloca sobre el techo de un coche, que incluye toda la tecnología anteriormente descrita.

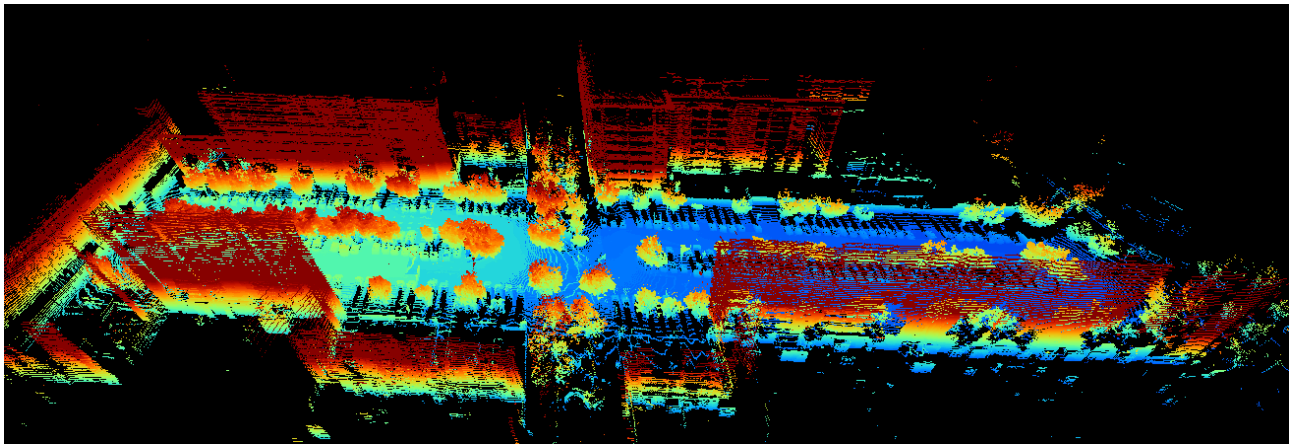

Figura 1 Captura de pantalla de la visualización de una nube de puntos, concretamente la que se ha utilizado en el proyecto, con la herramienta SAGA GIS

Figura 2 Equipo colocado encima de un coche y que incluye un sensor LIDAR, GPS y cámara panorámica

1.2 Motivación

Unas de las principales razones por las que he elegido realizar este proyecto es el hecho de que se base en tecnologías como LIDAR, que últimamente se está popularizando cada vez más y veo mucho potencial en esta tecnología que, por ejemplo, es crítica para que en el futuro cercano podamos disfrutar de los coches autónomos.

Además, me gusta exprimir las posibilidades que ofrecen los navegadores web modernos, permitiendo hacer aplicaciones web que se acercan cada vez más a las aplicaciones nativas, tanto en funcionalidad como en experiencia de usuario y rendimiento.

1.3 Estructura del documento

El documento está estructurado en secciones correspondientes a la principal documentación de un proyecto, es decir, la documentación del problema, del diseño de la solución, la implementación y sus conclusiones. Además, al final del documento se encuentra la bibliografía, los anexos, los índices de figuras y tablas, y el glosario.

La sección Problema(ver pág. 4) contiene la descripción del problema, la documentación de los requisitos funcionales y no funcionales del sistema y los casos de uso del sistema.

La sección Diseño de la solución(ver pág. 8) incluye la documentación del diseño arquitectónico del sistema que se ha ido realizando y mejorando en las sucesivas iteraciones del desarrollo del proyecto.

La sección Implementación(ver pág. 17) incluye los detalles de la implementación de los componentes del sistema, junto con el software y tecnologías usadas para la implementación de ellos.

En la sección Gestión del proyecto(ver pág. 21) se encuentra la documentación de la gestión del proyecto. Ésta se compone de la documentación del modelo del proceso elegido, la gestión de la configuración, la planificación y el coste total del proyecto.

En Conclusiones(ver pág. 26) se exponen las conclusiones obtenidas una vez realizado el proyecto.

A partir de la página 28 se encuentran los anexos de la memoria que contienen detalles adicionales del proyecto, tales como los manuales de usuario, manual de instalación del sistema, el prototipo de la interfaz gráfica de usuario, la documentación de los servicios web del sistema y detalles de las pruebas realizadas.

En el Glosario(ver pág. 49) se encuentran las definiciones de los términos más importantes empleados en el documento.

2 Problema

En esta sección se va a presentar la descripción del problema, la documentación de los requisitos funcionales y no funcionales del sistema, y los casos de uso del sistema.

En el Anexo I. Prototipo de la interfaz gráfica del usuario (ver pág. 28) se puede consultar el prototipo de la interfaz gráfica del usuario que se había documentado en fases tempranas del proyecto.

2.1 Descripción

Este trabajo consiste en el desarrollo de una aplicación web que permita la navegación por entornos tridimensionales basados en imágenes panorámicas georreferenciadas y, además, la interacción con el entorno aprovechando la información de las nubes de puntos capturadas con la tecnología LIDAR.

En cuanto a la funcionalidad que se quiere que ofrezca el sistema final, por una parte tendrá que ofrecer la funcionalidad básica de todo sistema de navegación tridimensional, es decir, el desplazamiento por el entorno y el cambio de orientación del usuario para observar el entorno desde cualquier perspectiva. Pero por otra parte, y esta es la máxima prioridad del proyecto, se tendrá que permitir la realización de toma de medidas (tales como la posición de un punto, la distancia entre 2 puntos, un área o un volumen) del mundo real representado por el sistema. Además, el usuario también deberá poder ver una representación de la nube de puntos superpuesta a las imágenes panorámicas y podrá marcar la posición de un objeto real junto con la información que quiera aportar.

Además, toda la funcionalidad anteriormente descrita se deberá implementar de manera que el sistema final se pueda usar perfectamente sin la necesidad de instalar ningún complemento en el navegador web. Para ello, se deberán utilizar estándares web como el WebGL [3].

La funcionalidad descrita es de especial interés para el análisis detallado de localizaciones concretas para, por ejemplo, proyectos de construcción de carreteras, edificios, etc. Un arquitecto podría emplear este sistema para analizar profundamente un sitio concreto para saber las restricciones que tendrá que aplicar a la hora de diseñar, por ejemplo, un edificio. Todo ello sin la necesidad de desplazarse físicamente a dicho sitio.

2.2 Requisitos

A continuación se exponen todos los requisitos que el sistema debe satisfacer. En la Tabla 1 están documentados todos los requisitos funcionales del sistema y en la Tabla 2 están los requisitos no funcionales.

Requisitos funcionales

RF-1	La aplicación debe permitir la visualización de vistas panorámicas de 360º con control de la orientación.
RF-2	La aplicación debe permitir la navegación entre las distintas fotografías panorámicas de un escenario.
RF-3	La aplicación debe permitir cargar un escenario que será formado por un conjunto de panoramas relacionados.
RF-4	La aplicación debe mostrar en todo momento la información de la orientación del usuario dentro de la vista panorámica que se está mostrando.
RF-5	La aplicación debe permitir tomar medidas de distancias entre puntos, áreas y volúmenes, además de mostrar la posición de un punto, del mundo real representado en las vistas panorámicas.
RF-6	La aplicación debe permitir la representación de la nube de puntos de la escena, superpuesta a la vista panorámica.
RF-7	La aplicación debe permitir marcar la posición real (en coordenadas en algún sistema de proyección cartográfico) de objetos que aparezcan en las vistas panorámicas (siempre que hayan sido detectados por el sensor de láser) y guardar estas posiciones junto a algo de información adicional que proporcione el usuario.
RF-8	La aplicación debe permitir visualizar los objetos que han sido guardados en el RF-7.
RF-9	La aplicación debe indicar las posiciones de las distintas fotografías panorámicas disponibles y próximas a la posición actual dentro del escenario actual.
RF-10	La aplicación podría permitir mostrar otros datos 3D sobre el panorama si se puede hacer utilizando la misma tecnología desarrollada para el RF-6(prioridad baja).

Tabla 1 Requisitos funcionales del sistema

Requisitos no funcionales

RNF-1	La aplicación deberá poderse ejecutar en los navegadores web populares(al menos Chrome y Firefox).
RNF-2	La aplicación deberá ser intuitiva.
RNF-3	La aplicación estará construida de manera que se pueda integrar fácilmente con otros componentes de aplicaciones web.

Tabla 2 Requisitos no funcionales del sistema

2.3 Casos de uso

En este apartado se encuentra el modelado de los requisitos del sistema con casos de uso. Los casos de uso describen las posibles interacciones que pueden realizar los distintos actores con el sistema. La Figura 3 muestra el diagrama de casos de uso del sistema, que muestra las interacciones anteriormente mencionadas de una manera gráfica e intuitiva.

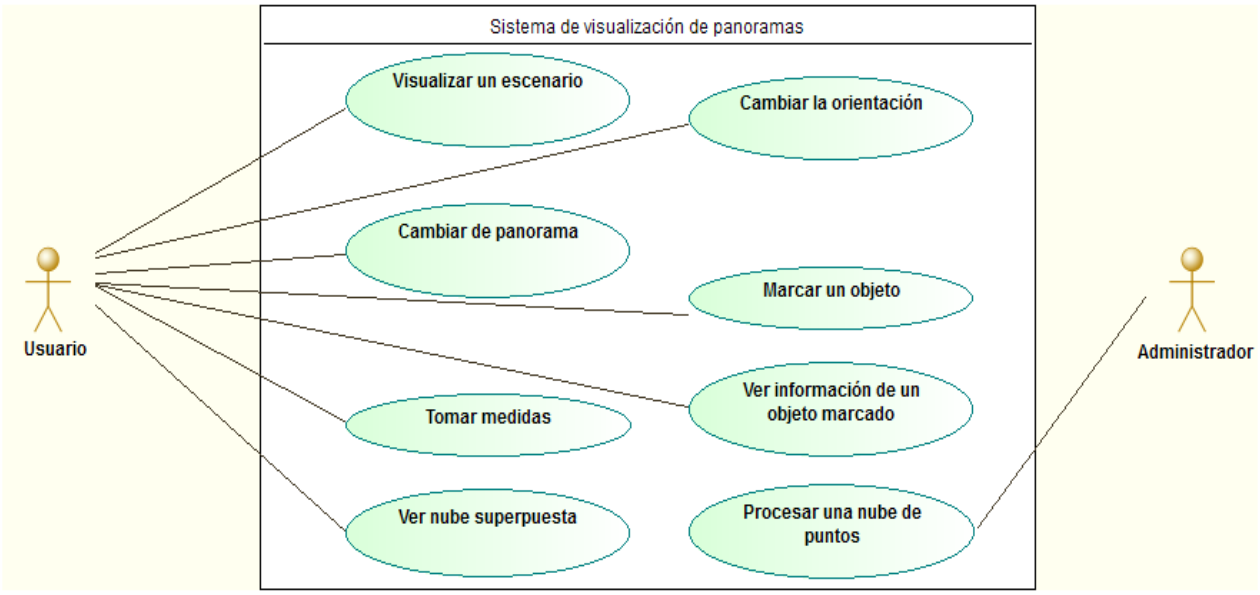

Figura 3 Diagrama de casos de uso del sistema

El modelo de casos de uso contiene a 2 actores principales del sistema: el usuario y el administrador. El actor usuario puede interactuar con los casos de uso “Visualizar un escenario”, “Cambiar la orientación”, “Cambiar de panorama”, “Tomar medidas”, “Ver nube superpuesta”, “Marcar un objeto” y “Ver información de un objeto marcado”.

El caso de uso “Visualizar un escenario” le permite al usuario visualizar un escenario de los ofrecidos por el sistema. Otro caso de uso relacionado es el “Cambiar la orientación” que le permite al usuario cambiar la orientación dentro del panorama que está visualizando en ese momento, tanto en horizontal como en vertical.

El caso de uso “Cambiar de panorama” le permite al usuario navegar por el escenario que está visualizando, seleccionando el panorama deseado y cambiando la posición al del panorama elegido.

El caso de uso “Tomar medidas” le permite al usuario tomar medidas del mundo real representado por el sistema. Los tipos de medidas que el usuario le permite hacer son el de la posición de un punto, la distancia entre 2 puntos, área y volumen.

El caso de uso “Ver nube superpuesta” le permite al usuario ver la representación de la nube de puntos del escenario que está visualizando, superpuesta a las imágenes panorámicas del escenario. Esto le puede ayudar a saber que zonas del escenario son las que están recogidas en la nube de puntos y, por lo tanto, puede interaccionar con ellas.

El caso de uso “Marcar un objeto” le permite al usuario marcar una posición en el escenario junto con la información adicional que desee aportar.

El último caso de uso del actor usuario, “Ver información de un objeto marcado”, le permite al usuario seleccionar un objeto que ha sido previamente marcado con el caso de uso “Marcar un objeto” y ver la información que se aportó al marcarlo.

El actor administrador, en cambio, puede interaccionar con el caso de uso “Procesar una nube de puntos” que le permite procesar una nube de puntos asociada a un escenario, para que luego pueda ser aprovechada por el usuario.

3 Diseño de la solución

Esta sección incluye la documentación del diseño arquitectónico del sistema que se ha ido realizando y mejorando en las sucesivas iteraciones del desarrollo del proyecto.

A la hora de realizar el diseño de la arquitectura inicial, se han identificado 2 grandes alternativas. Hubo que tomar la decisión de elegir la parte del sistema a la que darle la responsabilidad de aprovechar las nubes de puntos procesadas para que el usuario pueda interactuar con el entorno. Se puede hacer tanto en la parte del cliente, con JavaScript, como en la parte del servidor con librerías como PCL (Point Cloud Library) [4]. Primero se ha intentado optar por la segunda alternativa, pero los resultados han sido insatisfactorios, por lo que el diseño final parte de la primera alternativa.

En la arquitectura final, se delega una gran cantidad de responsabilidades al cliente web, ya que se emplea una librería JavaScript muy potente, three.js [5]. Por otro lado, en el lado del servidor web se ha optado por implementar unos servicios web que abstraigan la funcionalidad del servidor con la máxima interoperabilidad, gracias al uso de estándares como JSON [6]. En definitiva, el estilo de la arquitectura del sistema es cliente-servidor, con el servidor basado en servicios web y con un cliente muy potente que también realiza muchas funciones por sí mismo.

Aparte del cliente web y el servidor web, el diseño incluye también una aplicación para el procesamiento de las nubes de puntos.

La documentación de la arquitectura del software está compuesta por 4 vistas arquitectónicas, cada una reflejando distintos aspectos del mismo sistema. Estas vistas son la vista de componentes, vista de módulos, vista de despliegue y la vista de la instalación.

La vista de componentes representa al sistema en tiempo de ejecución, mostrando como se relacionan los distintos componentes para llevar a cabo las interacciones necesarias. Además, la vista de componentes sirve para indicar la reusabilidad y la reemplazabilidad de los componentes.

La vista de módulos, por su parte, sirve para identificar las principales unidades de implementación del sistema, con las relaciones existentes entre ellos.

La vista de despliegue se utiliza para representar la arquitectura física sobre la que se despliega el software. Es decir, los dispositivos físicos sobre los que se van a desplegar los componentes software. Y, por último, la vista de instalación sirve para mostrar detalles necesarios para que el despliegue del sistema sea correcto.

En el Anexo II. Interfaz público de los servicios web (ver pág. 32) se puede consultar el interfaz público de los servicios web del sistema.

3.1 Vista de componentes

A continuación se presenta la vista de componentes del sistema, incluyendo su presentación primaria, el catálogo de los elementos junto con sus responsabilidades y las interfaces de los componentes. La *Figura 4* muestra la presentación primaria de la vista, en notación UML [7].

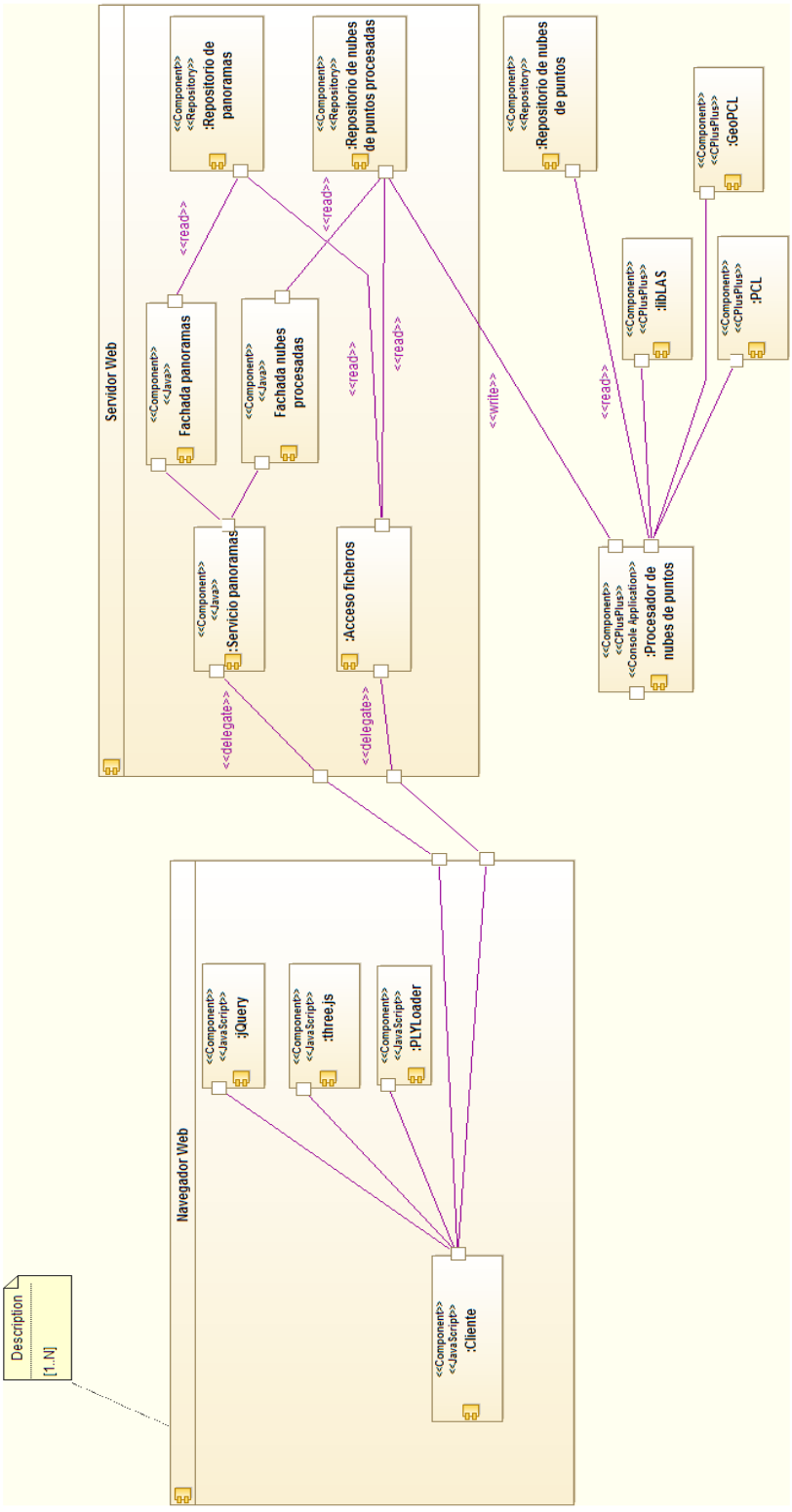

Figura 4 Vista primaria de la vista de componentes

El componente *Cliente* se encarga de ofrecer toda la funcionalidad del sistema al usuario. Para ello, descubre los escenarios existentes en el sistema, carga un escenario y lo visualiza para el usuario. Una vez visualizado un escenario, también se encarga de ofrecer las funciones de cambio de orientación del usuario en la escena virtual, mostrar información acerca de dicha orientación, desplazarse por la escena virtual, tomar medidas del mundo real representado por los panoramas, ver la nube de puntos correspondiente al escenario superpuesta a la vista panorámica y marcar objetos del mundo real junto con la información que aporte el usuario. Además, guarda los objetos marcados por el usuario en su navegador web y los visualiza, permitiéndole al usuario recuperar la información que había introducido al marcar el objeto.

El componente *three.js* es el correspondiente a la librería three.js [5], una librería JavaScript que ofrece un interfaz de alto nivel al WebGL, necesario para la visualización de los panoramas con un buen rendimiento.

El componente *jQuery* es el correspondiente a la librería jQuery [8] que reduce considerablemente el tiempo de desarrollo de componentes JavaScript, ofreciendo un interfaz de alto nivel para las operaciones básicas de JavaScript.

El componente *PLYLoader* es el responsable de cargar ficheros en formato .ply desde JavaScript.

El componente *Servicio panoramas* se encarga de ofrecerle a la aplicación cliente la información de los escenarios y sus vistas panorámicas, además de los enlaces a las imágenes panorámicas y las nubes de puntos procesadas.

El componente *Fachada panoramas* ofrece una capa de abstracción al *Repositorio de panoramas* y el componente *Fachada nubes procesadas* lo hace al *Repositorio de nubes procesadas.* Ambos componentes siguen el patrón de diseño *Façade*.

El componente *Repositorio de panoramas* es el responsable de ofrecer el acceso a los escenarios del sistema y ofrecer tanto las imágenes panorámicas como los metadatos asociados a ellas.

El componente *Repositorio de nubes de puntos* se encarga de ofrecer el acceso a las nubes de puntos, y el *Repositorio de nubes de puntos procesadas* el acceso a las nubes de puntos procesadas.

El componente *Procesador de nubes de puntos* es el responsable de cargar y procesar las nubes de puntos asociadas a cada escenario para reducir considerablemente su tamaño y guardarlas en un formato que sea aprovechable en la aplicación cliente.

PCL es el componente correspondiente a la librería PCL que es una librería de código abierto que ofrece un interfaz, en el lenguaje de programación C++, a distintas funciones de manipulación de las nubes de puntos.

El componente *libLAS* es el correspondiente a la librería libLAS [9], una librería para la manipulación de los ficheros en formato ".las", formato de los ficheros que contienen nubes de puntos obtenidas con dispositivos LIDAR.

GeoPCL es el componente asociado a la librería GeoPCL [10] que permite la manipulación de nubes de puntos en formato ".las" con la librería PCL.

3.1.1 Interfaces de los componentes

La Tabla 3 muestra la documentación de los interfaces de los distintos componentes mostrados en la vista de componentes.

Componente	**Recursos**
Servicio panoramas	+listarEscenarios() : List<String>
	+infoEscenario(String nombreEsenario) : Escenario
Fachada panoramas	+listarEscenarios() : List<String>
	+infoEscenario(String nombreEsenario) : Escenario
Fachada nubes procesadas	+getRutaNubeProcesada(String nombreEscenario) : String
Repositorio de panoramas	+obtenerFichero(String rutaFichero) : Fichero
Repositorio de nubes de puntos procesadas	+obtenerFichero(String rutaFichero) : Fichero
Procesador de nubes de puntos	+procesarNubeDePuntos(String rutaNube, String rutaNubeProcesada) : void

Tabla 3 Interfaces de los componentes

3.2 Vista de módulos

A continuación está la vista de módulos del diseño arquitectónico del sistema. La Figura 5 muestra su presentación primaria, en notación UML.

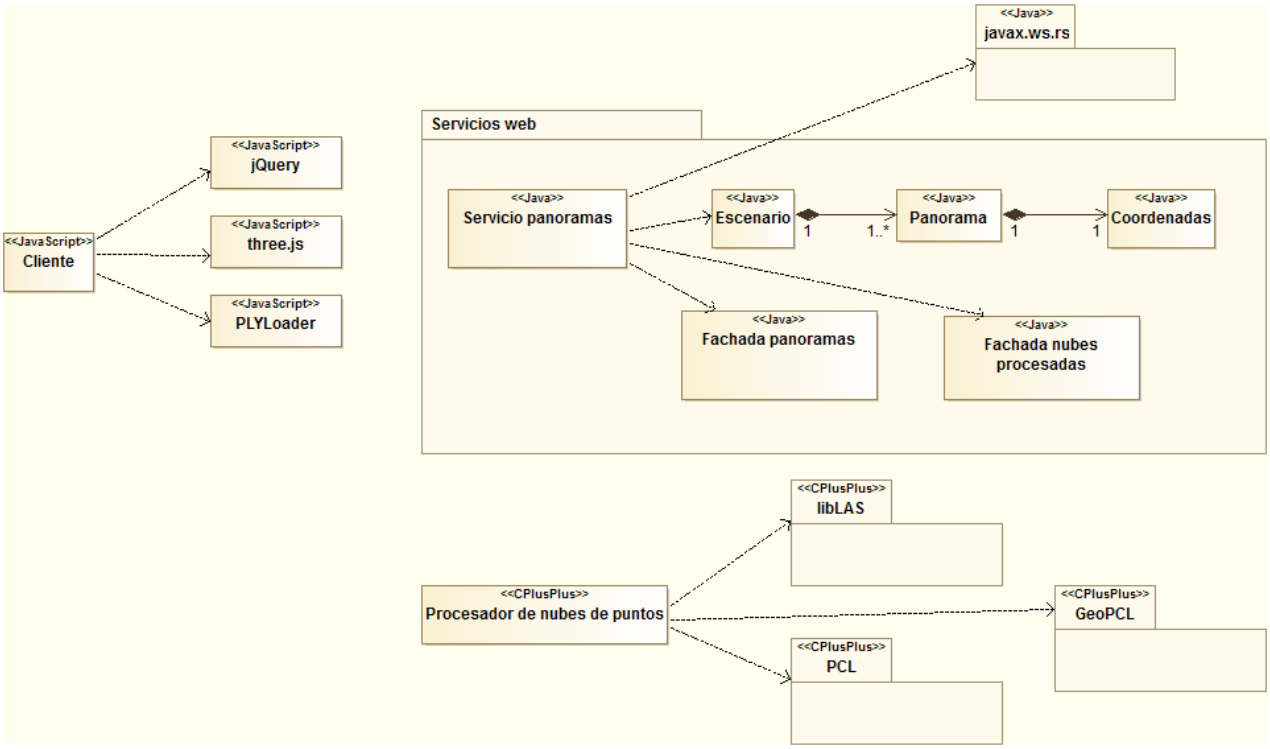

Figura 5 Presentación primaria de la vista de módulos

El módulo *Cliente* se encarga de ofrecer toda la funcionalidad del sistema al usuario. Para ello, descubre los escenarios existentes en el sistema, carga un escenario y lo visualiza para el usuario. Una vez visualizado un escenario, también se encarga de ofrecer las funciones de cambio de orientación del usuario en la escena virtual, mostrar información acerca de dicha orientación, desplazarse por la escena virtual, tomar medidas del mundo real representado por los panoramas, ver la nube de puntos correspondiente al escenario superpuesta a la vista panorámica y marcar objetos del mundo real junto con la información que aporte el usuario. Además, guarda los objetos marcados por el usuario en su navegador web y los visualiza, permitiéndole al usuario recuperar la información que había introducido al marcar el objeto.

El módulo *three.js* es un módulo externo que se corresponde con la librería three.js, una librería JavaScript que ofrece un interfaz de alto nivel al WebGL, necesaria para la visualización de los panoramas con un buen rendimiento.

Otro módulo externo, la librería jQuery, es una librería JavaScript que reduce considerablemente el tiempo de desarrollo de componentes JavaScript, ofreciendo un interfaz de alto nivel para las operaciones básicas de JavaScript.

El módulo externo *PLYLoader* es el responsable de cargar ficheros en formato .ply desde JavaScript.

El módulo *Servicio panoramas* se encarga de ofrecerle a la aplicación cliente la información de los escenarios y sus vistas panorámicas, además de los enlaces a las imágenes panorámicas y las nubes de puntos procesadas.

El módulo *Fachada panoramas* ofrece una capa de abstracción al *Repositorio de panoramas* y el módulo *Fachada nubes procesadas* lo hace al *Repositorio de nubes procesadas.* Ambos módulos siguen el patrón de diseño *Façade*.

El módulo *javax.ws.rs* es un módulo externo que ofrece un interfaz de alto nivel para la creación de servicios web. Concretamente, se usa Jersey [11], una implementación de la especificación JAX-RS [12].

La clase *Escenario* se compone de una lista de panoramas, además de un nombre y la ruta de la nube de puntos correspondiente al escenario. Es el elemento central del modelo de datos del sistema, ya que al usuario se le ofrece una lista de escenarios que explorar.

La clase Panorama se compone de una localización, una URI con la que poder acceder a la imagen panorámica y una rotación que se corresponde con la orientación horizontal de la cámara a la hora de tomar la imagen panorámica.

La clase *Coordenadas* se compone de una longitud, latitud y altitud. Es decir, un par de coordenadas esféricas y una altitud en metros.

El módulo *Procesador de nubes de puntos* es el responsable de cargar y procesar las nubes de puntos asociadas a cada escenario para reducir considerablemente su tamaño y guardarlas en un formato que sea aprovechable en la aplicación cliente.

El módulo externo *PCL* se corresponde con la librería PCL, una librería de código abierto que ofrece un interfaz, en el lenguaje de programación C++, a distintas funciones de manipulación de las nubes de puntos.

El módulo externo *libLAS* es el correspondiente a la librería libLAS, una librería para la manipulación de los ficheros en formato ".las", formato de los ficheros que contienen nubes de puntos obtenidas con dispositivos LIDAR.

GeoPCL es el módulo externo asociado a la librería GeoPCL que permite la manipulación de nubes de puntos en formato ".las" con la librería PCL.

3.3 Vista de despliegue

En esta sección se encuentra la documentación de la vista de despliegue del sistema. La Figura 6 muestra su presentación primaria, en notación UML.

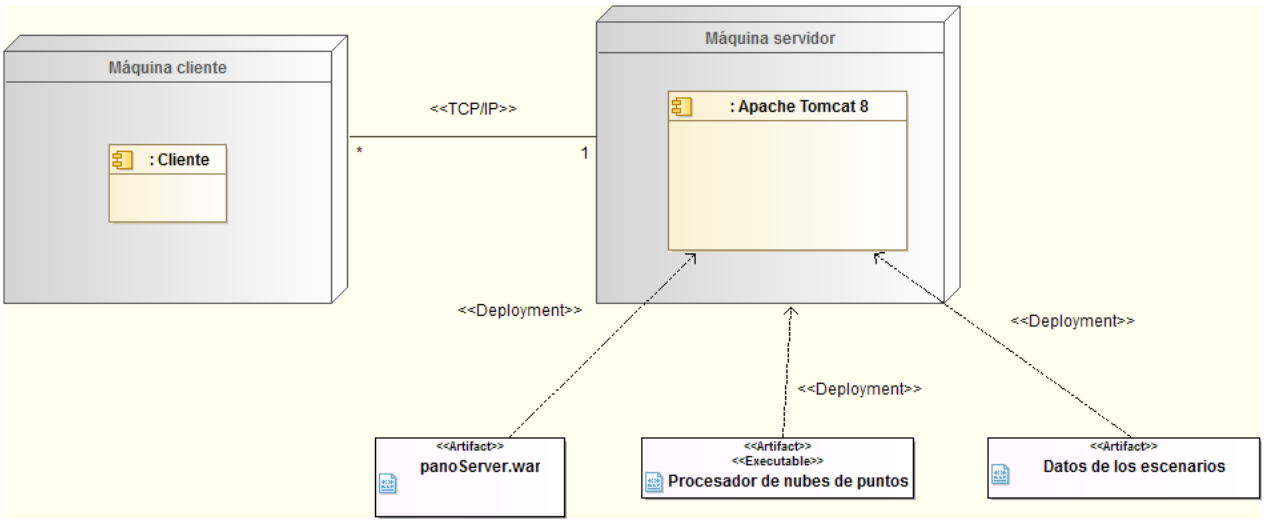

Figura 6 Presentación primaria de la vista de despliegue del sistema

Los componentes del sistema están desplegados sobre 2 nodos: *Máquina cliente* y *Máquina servidor.* Si fuera necesario, el sistema se podría desplegar sobre más nodos, ya que los componentes de tipo repositorio podrían ser desplegados en otras máquinas.

En el nodo *Máquina cliente* se despliega el componente *Cliente* que se corresponde a los componentes englobados en *Navegador web* en la Vista de componentes(pág 9).

En cambio, en el nodo *Máquina servidor* se despliega el componente *Apache Tomcat 8* que se corresponde con el servidor de aplicaciones web Apache Tomcat 8 [13], además del componente *Procesador de nubes de puntos.* En el servidor de aplicaciones se despliegan los artefactos *panoServer.war* y *Datos de los escenarios.*

El artefacto *panoServer.war* se corresponde con el despliegue del componente *Servicio panoramas* y el artefacto *Datos de los escenarios* se refiere al despliegue de los componentes de tipo repositorio, todos vistos en la Vista de componentes(pág 9).

Los detalles concretos de los artefactos anteriormente descritos están en la documentación de la vista de la instalación.

3.4 Vista de instalación

A continuación se exponen los detalles importantes que es necesario conocer para poder realizar correctamente el despliegue del sistema.

La Figura 7 muestra la vista de instalación del artefacto *panoServer.war* descrito en el apartado anterior, en notación UML.

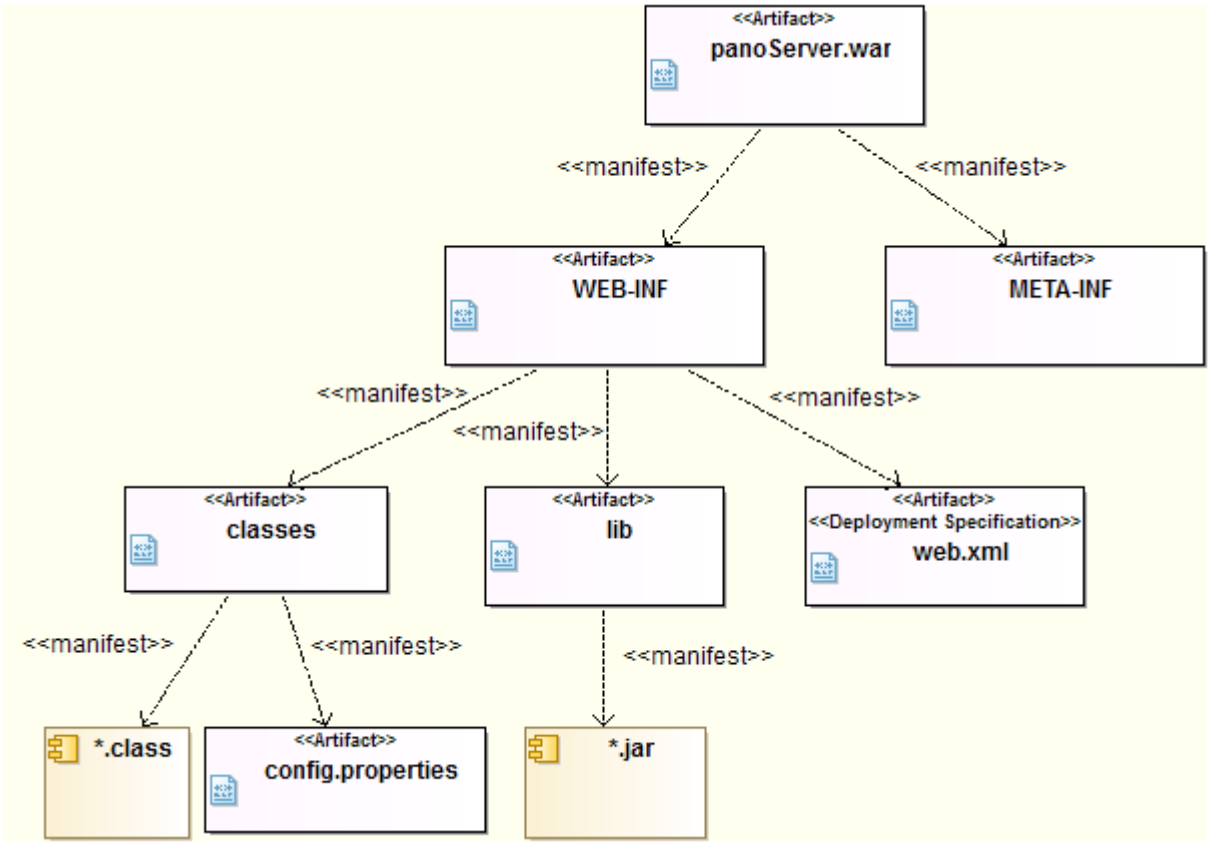

Figura 7 Vista de instalación de panoServer.war

El artefacto *panoServer.war* se subdivide en las carpetas *WEB-INF* y *META-INF.* La carpeta *META-INF* no tiene gran interés, ya que es generada automáticamente en el proceso de generación del fichero .war.

En cambio, la carpeta *WEB-INF* contiene la carpeta *classes, lib* y el fichero de especificación del despliegue *web.xml,* en el que se encuentran los detalles necesarios para el correcto funcionamiento de los servicios web.

La carpeta *classes* contiene los ficheros compilados necesarios para la ejecución de los servicios web, además del fichero *config.properties* que contiene los parámetros de configuración del sistema, permitiendo así cambiar los parámetros de configuración del sistema en vivo y sin la necesidad de recompilarlo todo.

La carpeta *lib* contiene las librerías que se usan en los servicios web, en formato .jar.

A continuación se presentan los detalles del artefacto *Datos de los escenarios,* la Figura 8 muestra su vista de instalación, en notación UML.

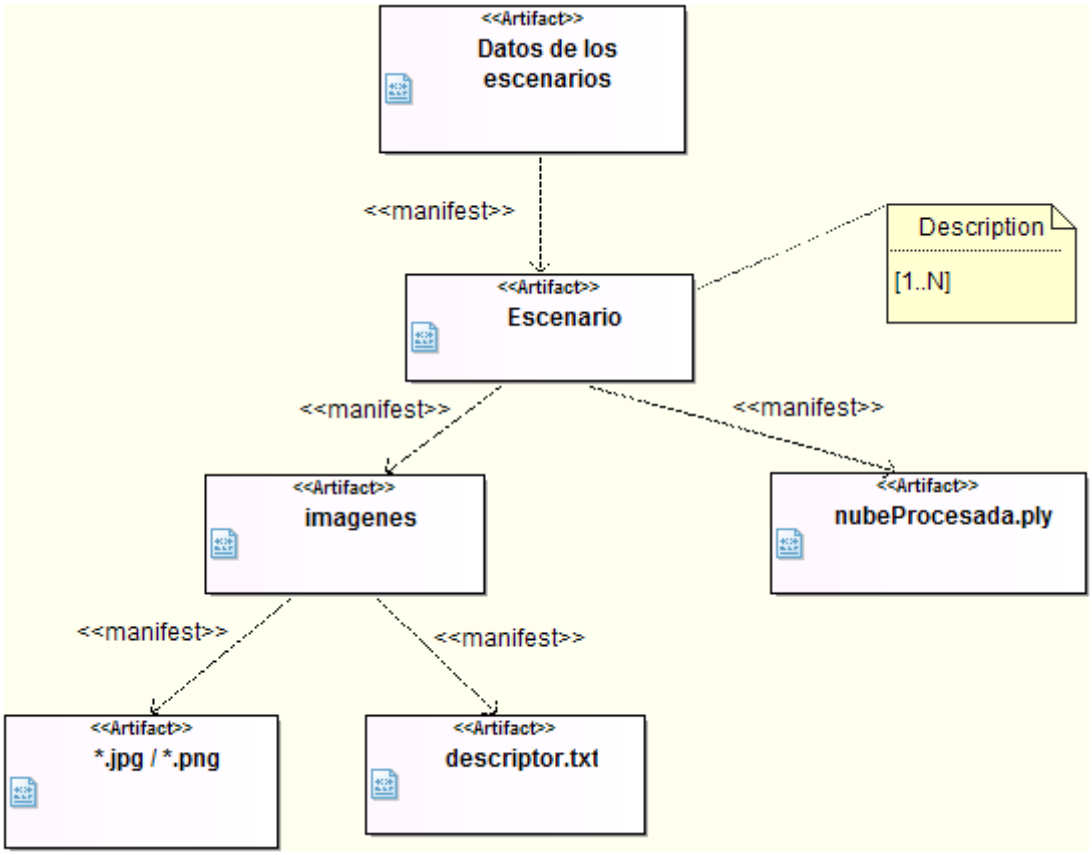

Figura 8 Vista de instalación de Datos de los escenarios

El artefacto *Datos de los escenarios* es una carpeta cuyo contenido son N carpetas, una por cada escenario ofrecido por el sistema.

Cada carpeta *Escenario* está compuesta, a su vez, por una carpeta llamada *imágenes* y un fichero con la extensión .ply, correspondiente al fichero de la nube de puntos procesada del escenario.

La carpeta *imagenes* contiene un listado de imágenes, correspondientes a las imágenes panorámicas del escenario. Y también un fichero .txt con los detalles de cada imagen panorámica.

En el Anexo IV. Manual de instalación (ver pág. 36) se puede consultar el manual de instalación del sistema.

4 Implementación

La implementación no se ha realizado de golpe, sino que se siguió un proceso incremental, un proceso en el que se han ido implementando prototipos con cada vez más funcionalidad hasta que el sistema satisfaga los requisitos establecidos en la fase de análisis.

El primer prototipo tenía como objetivo, por un lado, crear un prototipo del cliente web que pueda cargar un panorama desde un fichero local y, por otro, procesar una nube de puntos con la librería PCL y crear una aplicación de consola, basada también en la librería PCL, que utilice la nube de puntos procesada en el paso anterior y ofrezca la operación que, dada una posición y una dirección(obtenidos en el momento cuando en el cliente web se haga un clic sobre el panorama), devuelva la intersección con la nube de puntos, si la hay. Este proceso es llamado *Ray casting* [14]. Además, esta aplicación de consola se utilizó a través de un servicio web, invocado desde el cliente web, que encapsulaba la operación ofrecida por la aplicación. Una vez implementado este prototipo, los resultados han sido insatisfactorios, ya que cada invocación al servicio web tardaba varios segundos, además de que la precisión de los resultados tampoco era suficiente.

Como el primer prototipo ha resultado insatisfactorio, se ha dado retroalimentación al diseño del sistema para optar por otra alternativa. En el segundo prototipo se ha modificado el procesador de las nubes de puntos para que genere una nube de puntos procesada en un formato estándar que se pueda usar en el cliente con JavaScript, el formato PLY [15]. Además se ha modificado el prototipo del cliente web para que pueda cargar nubes de puntos en formato PLY, muestre la representación de la nube de puntos superpuesta al panorama y haga el proceso de *Ray casting* en el propio cliente web. Además, también permitía medir la distancia entre dos puntos. Los resultados han sido bastante mejores en comparación con el anterior prototipo, por lo tanto se ha probado que la decisión del cambio del diseño del sistema ha sido correcta.

A continuación, se ha implementado un prototipo del servidor web que ofrezca al cliente toda la información y recursos necesarios sobre los escenarios del sistema mediante servicios web y se ha modificado el prototipo del cliente para que aproveche dichos servicios web.

Y, finalmente, se ha implementado la versión final del cliente con toda la funcionalidad necesaria para satisfacer los requisitos del sistema.

Durante todo el proceso de implementación, se han utilizado tecnologías multiplataforma para que el software se pueda desplegar en cualquier tipo de sistema. Además, todo el software empleado es software libre, de manera que se evita el coste que puede conllevar el uso de software propietario y el hecho de que haya una comunidad de desarrolladores detrás del software significa que el código es bastante depurado y puede ayudar en gran medida a la hora de mantener el sistema.

Como paradigma de programación, se ha tratado de maximizar el uso de la programación orientada a objetos, dado que permite una aplicación inmediata de los resultados obtenidos en las fases de análisis y diseño, gran cantidad de librerías disponibles, posibilidad de trabajar con interfaces de alto nivel, fácil mantenimiento y gran facilidad a la hora de crear servicios web.

A continuación se ofrecen los detalles concretos de la implementación de los componentes principales del sistema, es decir, el *Cliente*, el *Servidor* y la aplicación *Procesador de nubes de puntos*, junto con el software y tecnologías empleadas en cada caso. Además, al final de la sección se detallan las pruebas realizadas.

4.1 Cliente

Para la implementación del cliente web, se han utilizado tecnologías web abiertas como HTML5, JavaScript, CSS y WebGL, para que el único requisito para el correcto funcionamiento del producto final sea disponer de un navegador web moderno, sin necesidad de instalar ningún complemento adicional.

En cuanto al *WebGL*, no se ha desarrollado directamente en ese lenguaje, sino que se ha utilizado una librería JavaScript que ofrece un interfaz de alto nivel que por debajo utiliza las bondades de WebGL, por lo que se consigue que la dificultad del desarrollo no se escape de los limites pero que tampoco se pierda el rendimiento que se obtiene al emplear el aceleración por hardware de *WebGL*. La librería elegida para ello fue *three.js* [5], cuya versión usada en el proyecto es la r68, que destaca sobre el resto de las librerías de propósito similar por detalles como actualizaciones constantes y una gran comunidad activa de desarrolladores que soportan el proyecto y me han sido de gran ayuda durante el desarrollo del proyecto, a través de la plataforma de preguntas y respuestas para desarrolladores *stackoverflow.com* . También ha sido de gran ayuda que dicha librería tenga soporte nativo para las nubes de puntos y para hacer el proceso de *Ray casting* con ellas.

Cabe destacar que para desarrollar el interfaz gráfico de usuario del cliente se ha tenido que combinar el diseño 2D con el 3D, ya que, por una parte, las imágenes panorámicas, las nubes de puntos y los elementos gráficos interactivos son tridimensionales, pero por otra parte, también se deben mostrar elementos visuales en 2D, como el menú del sistema, los diálogos y la información que se muestra al usuario, como las indicaciones para ayudarle con las funciones ofrecidas o la información acerca de la orientación del usuario en el mundo virtual.

Para el desarrollo de los elementos visuales en 2D se ha utilizado la combinación típica de las aplicaciones web, es decir, HTML5 + CSS + JavaScript. Para agilizar el desarrollo del control dinámico de los elementos HTML se ha usado la librería JavaScript *jQuery*, que además ha servido para facilitar en gran medida el desarrollo de las peticiones asíncronas (*AJAX*) a los servicios web.

Además, para mejorar el aspecto del menú del sistema se ha utilizado *Metro UI CSS* [16], un framework CSS/JavaScript para el interfaz gráfico de usuario de aplicaciones web.

En el Anexo V. Manuales de usuario (ver pág. 37) se pueden consultar los manuales de usuario para las distintas funciones ofrecidas por el cliente web.

4.2 Servidor

La parte del servidor del sistema se ha implementado íntegramente en el lenguaje de programación *Java 1.7* [17] por ser un lenguaje multiplataforma, orientado a objetos y de alto nivel.

Para el desarrollo de los servicios web, se ha empleado *JAX-RS* [12], un API para la creación de servicios web RESTful en Java. En concreto, se ha utilizado *Jersey* [11], una implementación de referencia y de código abierto de la especificación *JAX-RS* [12]. De esta forma, los servicios web se han podido desarrollar con anotaciones intuitivas, que hacen que el desarrollo de los servicios web sea muy similar al desarrollo de métodos locales. Además, el código de los servicios web es muy fácil de mantener.

Se ha hecho uso de un fichero de configuración que se consulta en todo momento por los servicios web. De esta forma, se permite su modificación “en caliente”, es decir, sin la necesidad de recompilar el software tras cualquier modificación, para alterar el comportamiento del sistema.

Para abstraer el acceso a los recursos del sistema desde los servicios web, se ha seguido el patrón *Façade* en los componentes *Fachada panoramas* y *Fachada nubes procesadas*. De esta forma, la parte del sistema que accede a los recursos es altamente modificable, permitiendo ajustarse a las necesidades futuras con una facilidad razonable, por ejemplo, en el caso de que sea necesario alojar los recursos del sistema en una máquina distinta a la que aloja los servicios web o incluso que estén distribuidos en varias máquinas.

Para facilitar la gestión de las dependencias del código Java, se ha empleado la herramienta *Maven* [18]. Esta herramienta permite enumerar las dependencias del proyecto y automatiza su descarga y configuración.

4.3 Procesador de nubes de puntos

Para la implementación de la aplicación para el procesamiento de las nubes de puntos, se ha utilizado la librería *PCL*, una librería de código abierto, multiplataforma, con un interfaz C++ de alto nivel y con una gran comunidad activa de desarrolladores. Además, el código de la librería está probado con pruebas unitarias, por lo que tiene una alta fiabilidad.

La librería *PCL* ofrece un gran abanico de módulos que realizan diversas funciones de procesamiento de nubes de puntos, entre ellas filtrado de los datos, *Ray casting*, segmentación o reconocimiento de superficies. La principal función de esta librería empleada en el proyecto fue la de filtrar la nube de puntos de entrada para reducir considerablemente su densidad y, por lo tanto, el tamaño del fichero de la nube de puntos. El tamaño de los ficheros de las nubes de puntos procesadas es muy importante, ya que van a tener que ser descargados a la hora de utilizar la aplicación web y el coste temporal es un aspecto muy importante para una buena usabilidad del sistema. Según la documentación oficial de la librería, para reducir la densidad de la nube de puntos se divide el espacio 3D ocupado por la nube de puntos en vóxeles (cubos 3D) y, entonces, todos los puntos que se encuentren en un vóxel son aproximados con su centroide, es decir, del conjunto de los puntos ubicados en cada vóxel, solo se queda el centroide del espacio formado por dichos puntos.

El potencial de la librería *PCL* se podría haber exprimido más, pero las restricciones temporales del proyecto lo ha impedido. En el caso de que en el futuro se quiera hacer, se podría trabajar en la segmentación de la nube de puntos para reconocer las estructuras de la nube de puntos como edificios o suelo, y poder manejarlos de forma distinta al resto de la nube de puntos. También se podría trabajar en la reducción del ruido en la nube de puntos, ya que normalmente los dispositivos *LIDAR* generan bastante ruido.

4.4 Pruebas

Las pruebas han acompañado al desarrollo en todas las iteraciones del proceso. Cada vez que se implementaba un prototipo del sistema, éste ha tenido que pasar por pruebas de sistema para verificar que cumple con sus requisitos y solo entonces se procedía con la siguiente iteración para desarrollar otro prototipo.

Cabe decir que las pruebas se han realizado con una nube de puntos compuesta por casi 10 millones de puntos y cuyo tamaño alcanzaba casi 315MB. Una vez procesada dicha nube de puntos con el procesador de nubes de puntos que se ha implementado en el proyecto, el tamaño resultante se redujo hasta aproximadamente 5MB, aunque si fuera necesario, se podría reducir más, debido a que la librería utilizada para el proceso de reducción del tamaño permite reducir el tamaño todo lo que se quiera, pero hay que tener en cuenta que cuanto más se reduce el tamaño de la nube de puntos, más perdida de precisión se produce.

También es importante detallar las especificaciones técnicas del equipo con el que se han ido realizando las pruebas. La Tabla 4 muestra dichas especificaciones.

Componente	**Descripción técnica**
Procesador	Intel Core i5-2430M 2.4GHz
Memoria operativa	6GB DDR3 PC3-10700
Sistema operativo	Microsoft Windows 7 Home Premium x64
Tarjeta gráfica	Intel HD Graphics 3000

Tabla 4 Especificaciones técnicas del equipo de pruebas

En el Anexo III. Pruebas de los servicios web (ver pág. 33) se pueden consultar las pruebas realizadas a los servicios web del sistema.

5 Gestión del proyecto

En esta sección se encuentra la documentación de la gestión del proyecto. Ésta se compone de la documentación del modelo del proceso elegido, la gestión de la configuración, la planificación y el coste total del proyecto.

5.1 Modelo del proceso

El modelo del proceso elegido para este proyecto ha sido el desarrollo iterativo e incremental, basado en prototipos. Este tipo de modelo de proceso presenta varias ventajas como resultados en fases tempranas, de modo que los fallos se pueden detectar pronto y poder reaccionar antes de que sea tarde, su facilidad a la hora de adaptarse a las circunstancias, su flexibilidad y el hecho de que los prototipos que son resultado de cada iteración son totalmente funcionales y se pueden pasar por un proceso de pruebas.

Cada iteración del proceso comprende tanto la fase de análisis, como diseño e implementación. Al final de la iteración, cuando se ha finalizado la implementación del prototipo, se ha procedido a realizar pruebas de sistema para verificar que el prototipo cumple con sus criterios de aceptación.

Al final se ha realizado un total de 4 iteraciones hasta llegar a la versión final del sistema que satisfacía todos los requisitos del sistema que se habían planteado.

5.2 Gestión de la configuración

Para conseguir un buen nivel de control sobre los elementos más importantes del proyecto, es decir, toda la documentación generada durante el desarrollo del proyecto, así como el código fuente del mismo, se ha empleado un sistema de control de versiones. En concreto, se ha utilizado un repositorio *Subversion* [19] en un servidor interno de la Universidad de Zaragoza, en concreto, el servidor de IAAA, de modo que el director del trabajo también tuvo acceso al repositorio en todo momento. Un sistema de control de versiones facilita la especificación de los cambios realizados entre las distintas versiones de un mismo fichero. Además sirve como un sistema de copias de seguridad, ya que permite recuperar cualquier versión anterior de los elementos del repositorio.

En cuanto a las versiones de las tecnologías empleadas para el desarrollo del proyecto, la Tabla 5 enumera todas las tecnologías utilizadas junto a la versión usada.

Nombre de la tecnología	**Versión**
Cliente	
three.js	r68
jQuery	2.1.1
Metro UI CSS	2.0.23
Servidor	
Java	1.7
Jersey	1.18.1
Procesador de nubes de puntos	
PCL	1.6

Tabla 5 Versiones de las tecnologías empleadas

A continuación se detallan las distintas herramientas que se han utilizado para el desarrollo del proyecto junto con su propósito:

- Entornos de desarrollo: Eclipse, Microsoft Visual C++ 2010 Express y Chrome Developer Tools
- Ofimática: LibreOffice y Microsoft Office
- Control de versiones: TortoiseSVN
- Visualización de nubes de puntos: SAGA GIS y CloudCompare
- Creación de la documentación UML: Modelio

5.3 Planificación

A continuación se presenta la planificación del proyecto. La Figura 9 muestra el diagrama de Gantt del proyecto.

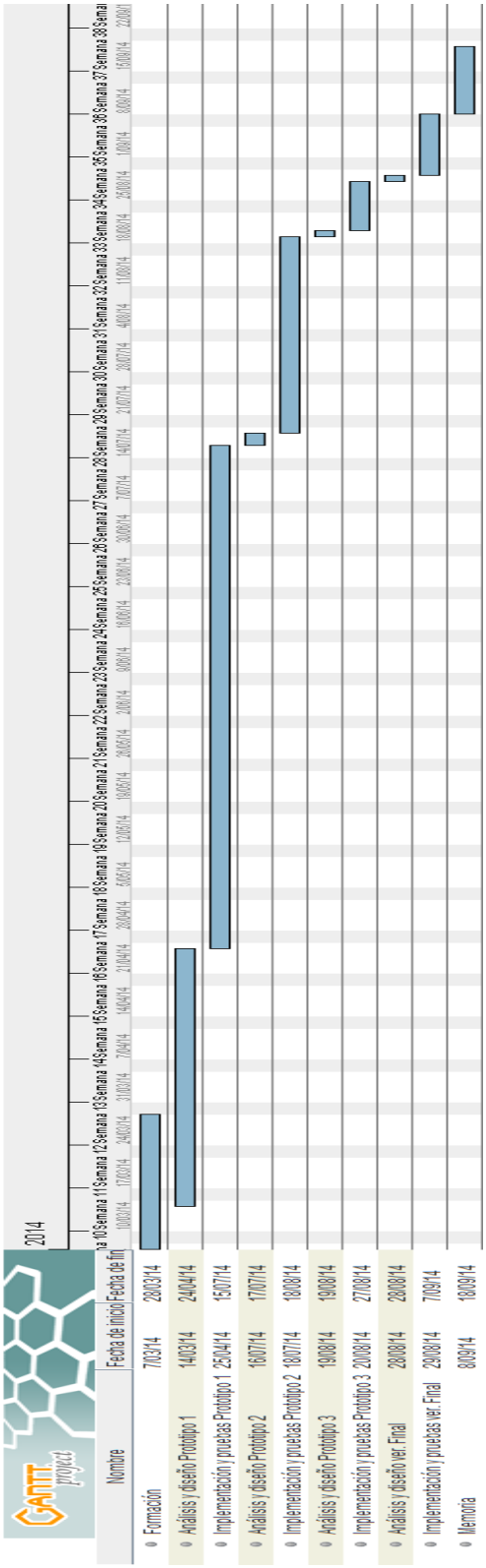

Figura 9 Diagrama de Gantt del proyecto

Como se puede observar en el diagrama mostrado anteriormente, hay dos tareas que destacan sobre el resto en cuanto a la longitud del periodo que comprenden. Dichas tareas son *Análisis y diseño Prototipo 1* e *Implementación y pruebas Prototipo 1*.

La razón por la que el periodo de la tarea *Análisis y diseño Prototipo 1* es tan alto es que al principio del proyecto ha sido necesaria la formación en las tecnologías a usar, además de que el análisis y diseño de la primera iteración siempre requiere de más esfuerzo que en el resto de las iteraciones, ya que es necesario sentar unas bases del proyecto con vistas al futuro.

Por otra parte, la tarea *Implementación y pruebas Prototipo 1* se ha tenido que prolongar tanto por otras razones. Una de ellas es porque la implementación ha sido muy costosa, ya que no se conseguían los resultados esperados. La otra razón es que al final del cuatrimestre ha aumentado mi carga de trabajo en las asignaturas matriculadas y se ha tomado la decisión de replanificar el proyecto para finalizarlo en Septiembre en vez de Junio. Cabe decir que en el mes de Junio, el proyecto ha sido dejado en segundo plano para retomarlo en Julio.

5.4 Coste total

Todo el esfuerzo dedicado al desarrollo del proyecto se ha ido recopilando en unas hojas de cálculo, en concreto, las que se usan para la recopilación de esfuerzos en el IAAA. Las distintas actividades del proyecto se han clasificado y se han establecido unos códigos de los esfuerzos.

En la Tabla 6 se encuentra el coste total del proyecto repartido entre los esfuerzos dedicados al desarrollo, incluyendo todas las fases, de la parte cliente del sistema, la parte del servidor (que incluye también la aplicación de procesamiento de nubes de datos) y la memoria del trabajo.

Actividad	**Coste**
Cliente	160h
Servidor	114h
Memoria	53h
TOTAL	**327h**

Tabla 6 El coste total del proyecto, clasificado en Cliente, Servidor y Memoria

Para ver el coste total del proyecto desde otra perspectiva, la Tabla 7 muestra el coste total repartido entre el coste que ha supuesto el desarrollo de los 4 prototipos, la formación y la memoria del trabajo.

Actividad	**Coste**
Prototipo 1	117h
Prototipo 2	76h
Prototipo 3	18h
Prototipo 4	55h
Formación	8h
Memoria del Trabajo	53h
TOTAL	**327h**

Tabla 7 El coste total del proyecto, clasificado por los prototipos, la formación y la memoria

Como se puede observar, el primer prototipo es el que más tiempo ha costado en desarrollarse. Eso es debido a que al principio del proyecto ha habido un periodo de adaptación a las nuevas tecnologías usadas y, además, los resultados han sido bastante insatisfactorios y se han gastado muchos esfuerzos en intentar mejorar el sistema hasta que se ha llegado a la decisión de cambio del diseño arquitectónico del sistema para el siguiente prototipo.

6 Conclusiones

Una de las conclusiones que se puede sacar una vez terminado el proyecto es que la decisión de elegir un modelo del proceso del proyecto iterativo e incremental, basado en prototipos, ha sido muy satisfactoria. Dicho modelo del proceso ha permitido adaptarse a las circunstancias con facilidad, tener resultados presentables en fases tempranas y obtener prototipos cada uno mejor que el anterior.

El sistema ha podido satisfacer todos los requisitos que se habían impuesto al principio del proyecto, ya que se pueden realizar todas las funciones pensadas para el sistema y con una buena usabilidad.

La ayuda recibida por parte del director del proyecto ha sido muy importante, sobre todo en el tema de análisis y diseño del sistema.

La experiencia obtenida en el desarrollo del proyecto ha sido muy positiva. Por un lado, se ha mejorado la habilidad de generar documentación de software. Además, he tenido que desarrollar una parte del sistema en el lenguaje de programación C++, casi desconocido para mi antes de este proyecto, lo que es muy ventajoso para mi futuro profesional, dada la alta popularidad de dicho lenguaje de programación. Pero lo que más he aprendido durante el proyecto es manejarme en el desarrollo 3D, sobretodo el desarrollo 3D para la web con WebGL. Dado que las plataformas móviles son cada vez más potentes y populares, el desarrollo 3D para la web tiene un gran futuro. El hecho de que el proyecto se base en gran medida en las nubes de puntos también es muy positivo, debido a que dicha tecnología se está popularizando con rapidez en los últimos años y cada vez tiene más aplicaciones.

Y es que al fin y al cabo mi principal objetivo para el proyecto fue aprender y familiarizarme con nuevas tecnologías que me sean útiles en el futuro personal y creo que mi elección fue correcta.

En cuanto al trabajo futuro del proyecto, hay varios aspectos en los que se podría trabajar para mejorar el sistema. Uno de ellos es aprovechar más el potencial ofrecido por la librería PCL utilizada para el procesamiento de las nubes de puntos. Se podría estudiar más las posibilidades que ofrece y aplicar dicho conocimiento para realizar mejores filtrados de las nubes de puntos, eliminando el ruido generado durante la generación de la nube con el dispositivo LIDAR o, incluso, detectar las estructuras geométricas presentes en las ubicaciones donde se estén generando las nubes para luego aprovechar esa información a la hora de realizar las funciones que ofrece el sistema y mejorar su calidad. Otro aspecto en el que se podría trabajar es ofrecer un menú de configuración en la aplicación web para que, por ejemplo, el usuario pueda configurar los colores de los objetos interactivos. También se podría añadir una ventana de información adicional en la aplicación web que muestre la posición del usuario vista desde arriba en un mapa bidimensional, para que el usuario se pueda situar mejor en el escenario.

7 Bibliografía

1. LIDAR. [En línea] http://en.wikipedia.org/wiki/Lidar.

2. LAS. *LASer (LAS) File Format.* [En línea] http://www.asprs.org/Committee-General/LASer-LAS-File-Format-Exchange-Activities.html.

3. WebGL. *OpenGL ES 2.0 for the Web.* [En línea] https://www.khronos.org/webgl/.

4. PCL. *Point Cloud Library.* [En línea] http://pointclouds.org/.

5. three.js. [En línea] http://threejs.org/.

6. JSON. *JavaScript Object Notation.* [En línea]

7. UML. *Unified Modeling Language.* [En línea] http://www.uml.org/.

8. jQuery. [En línea] http://jquery.com/.

9. libLAS. [En línea] http://www.liblas.org/.

10. GeoPCL. [En línea] https://github.com/chambbj/GeoPCL.

11. Jersey. [En línea] https://jersey.java.net/.

12. JAX-RS. *Java API for RESTful Services.* [En línea] https://jax-rs-spec.java.net/.

13. Apache Tomcat. [En línea] http://tomcat.apache.org/.

14. Ray casting. [En línea] http://en.wikipedia.org/wiki/Ray_casting.

15. PLY. [En línea] http://en.wikipedia.org/wiki/PLY_(file_format).

16. Metro UI CSS. [En línea]

17. Java. [En línea] https://www.java.com/.

18. Apache Maven. [En línea] http://maven.apache.org/.

19. Apache Subversion. [En línea] https://subversion.apache.org/.

Anexo I. Prototipo de la interfaz gráfica del usuario

En fases tempranas del proyecto, se ha elaborado un prototipo de la interfaz grafica del usuario para la aplicación web.

La Figura 10 muestra el prototipo de la pantalla inicial.

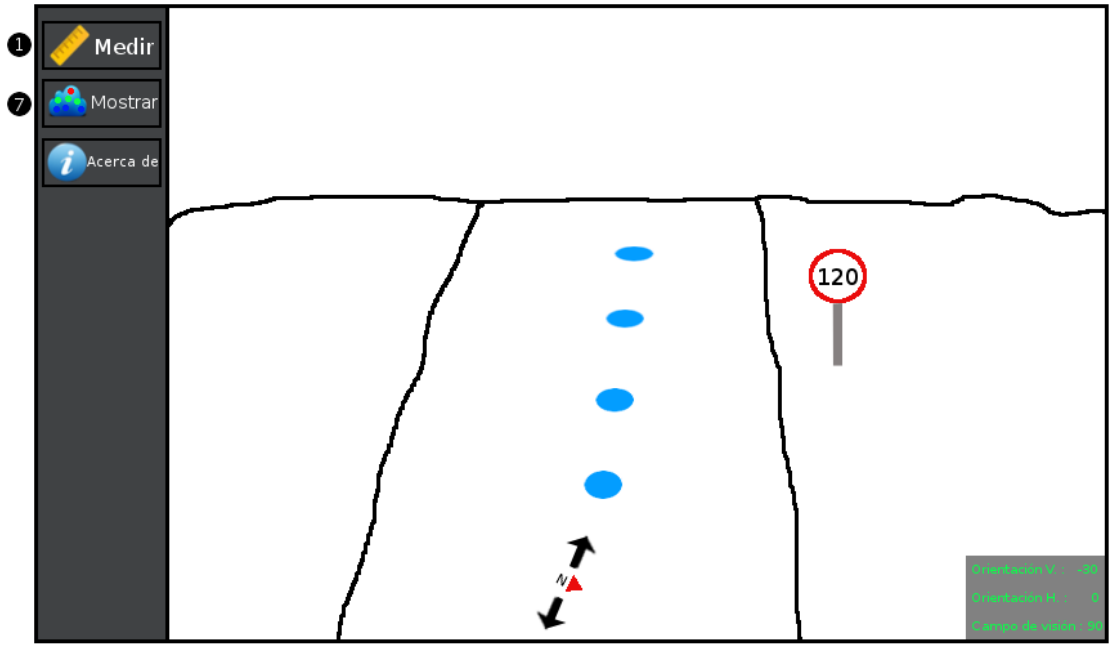

Figura 10 Prototipo de la pantalla inicial de la aplicación web

La Figura 11 muestra el estado del sistema al seleccionar la opción “Medir” en el menú principal.

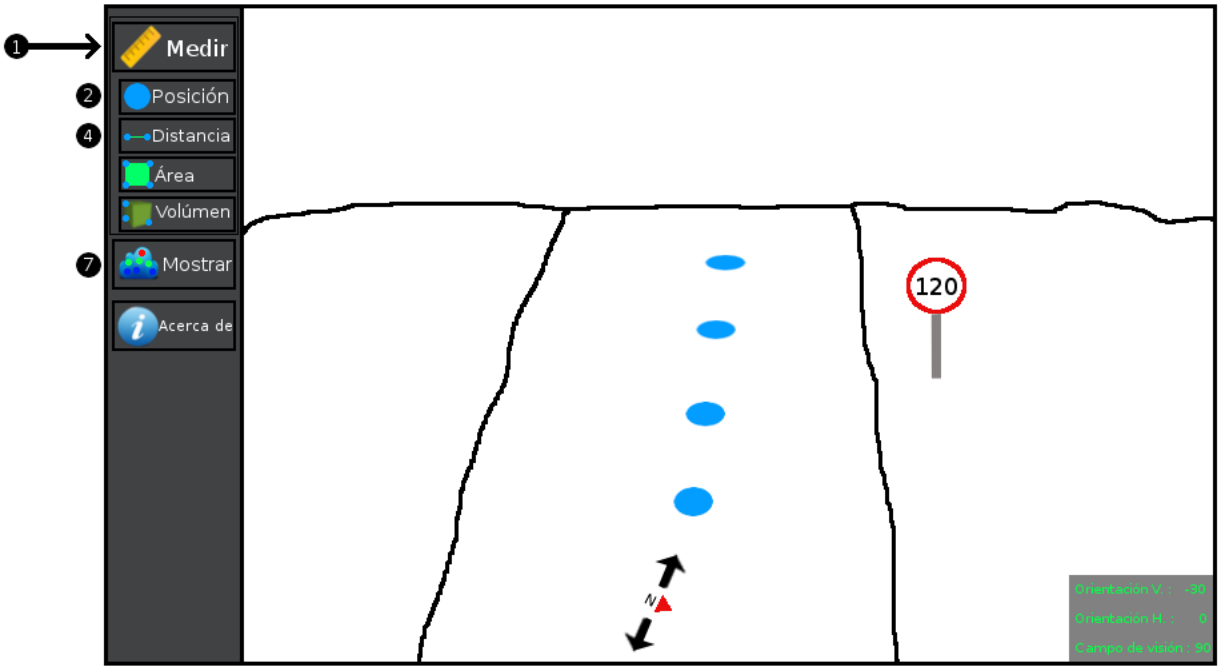

Figura 11 Prototipo del menú expandido de la aplicación web

La Figura 12 muestra el estado del sistema al seleccionar la opción “Posición”.

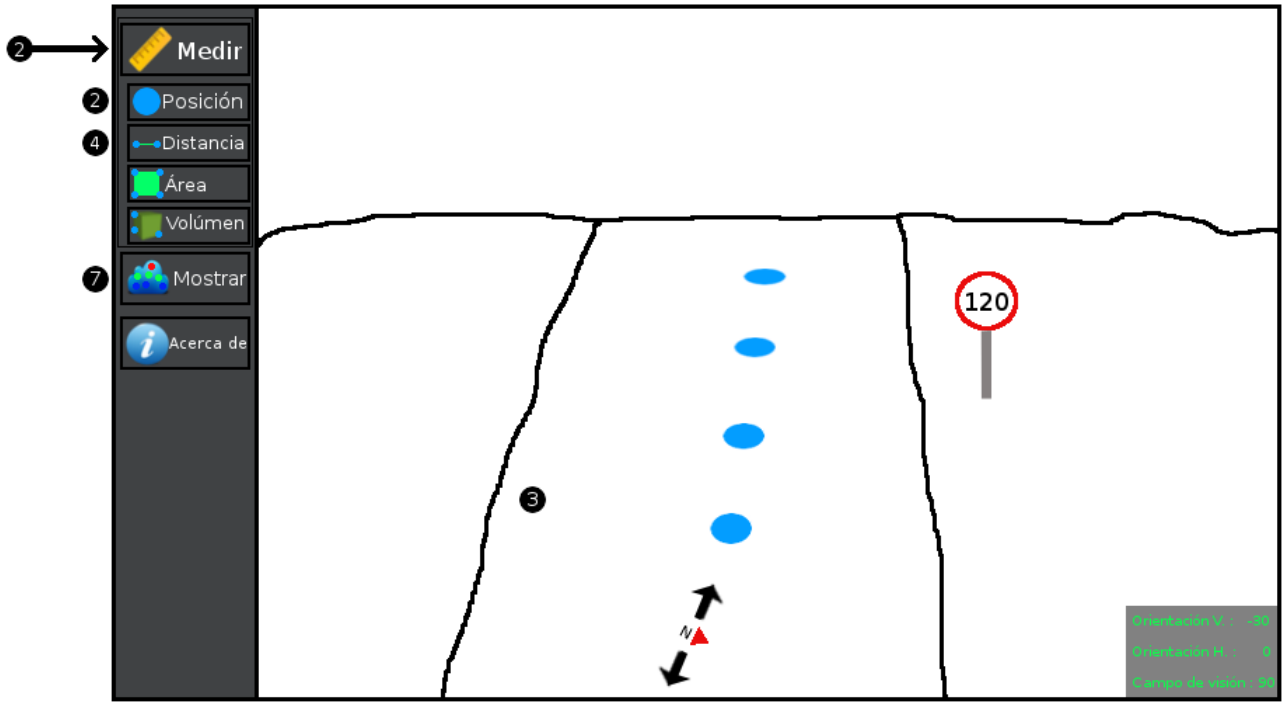

Figura 12 Prototipo GUI, opción "Posición" elegida

La Figura 13 muestra el prototipo de la visualización del resultado de la opción del menú “Posición”.

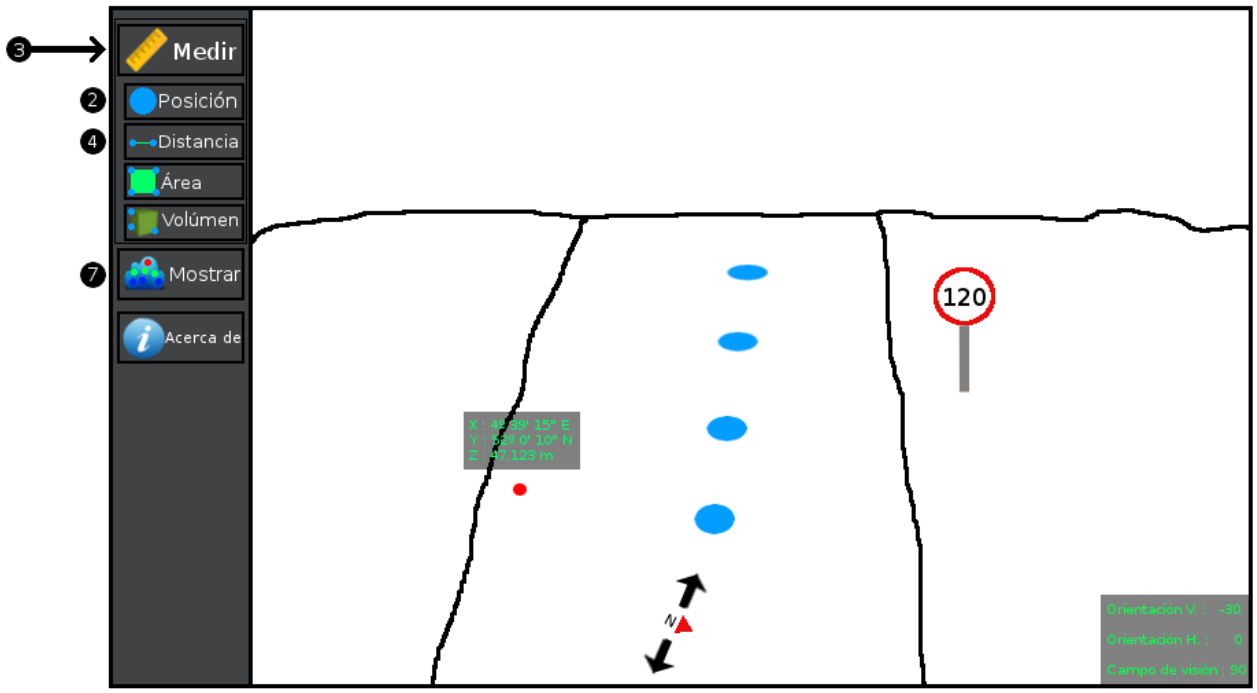

Figura 13 Prototipo GUI, resultado opción "Posición"

La Figura 14 muestra el estado del prototipo tras seleccionar la opción “Distancia”.

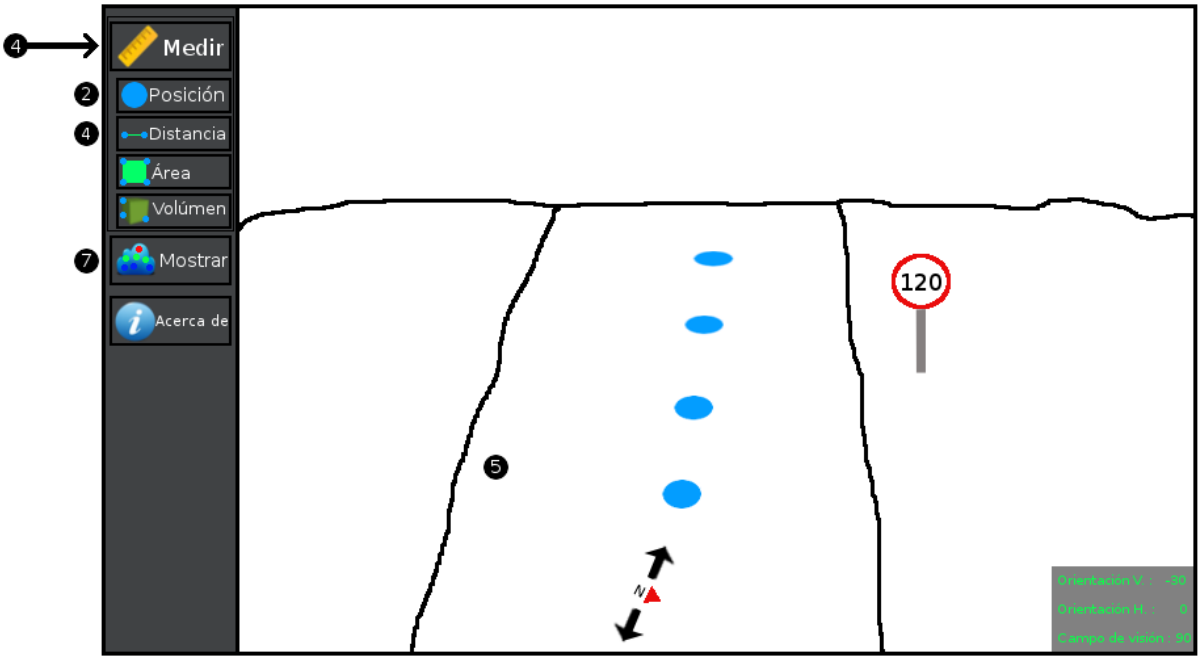

Figura 14 Prototipo GUI, opción "Distancia" elegida

La Figura 15 muestra el estado del sistema tras haber seleccionado el primer punto para medir la distancia.

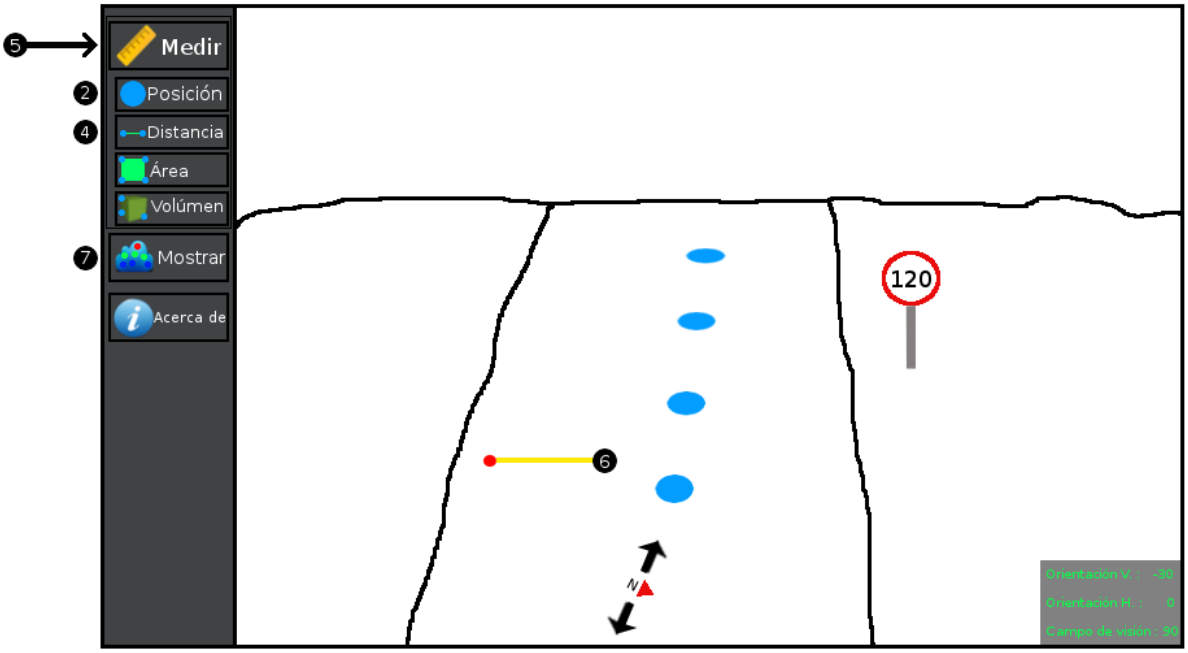

Figura 15 Prototipo GUI, seleccionado primer punto de la medición de distancia

La Figura 16 muestra el prototipo de visualización del resultado de la opción del menú “Distancia”, tras seleccionar el segundo punto.

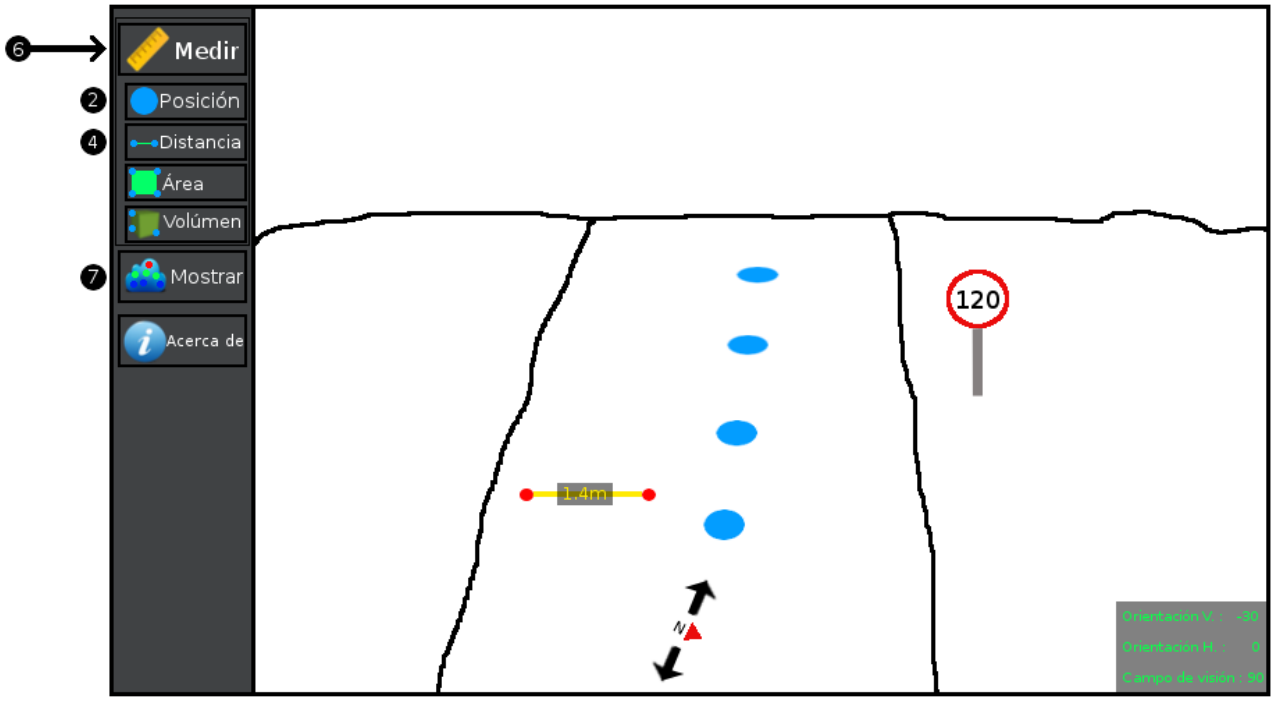

Figura 16 Prototipo GUI, resultado de la medición de distancia

La Figura 17 muestra el estado del sistema tras haber seleccionado la opción “Mostrar” del menú principal, es decir, la visualización de la superposición de la nube de puntos a la imagen panorámica.

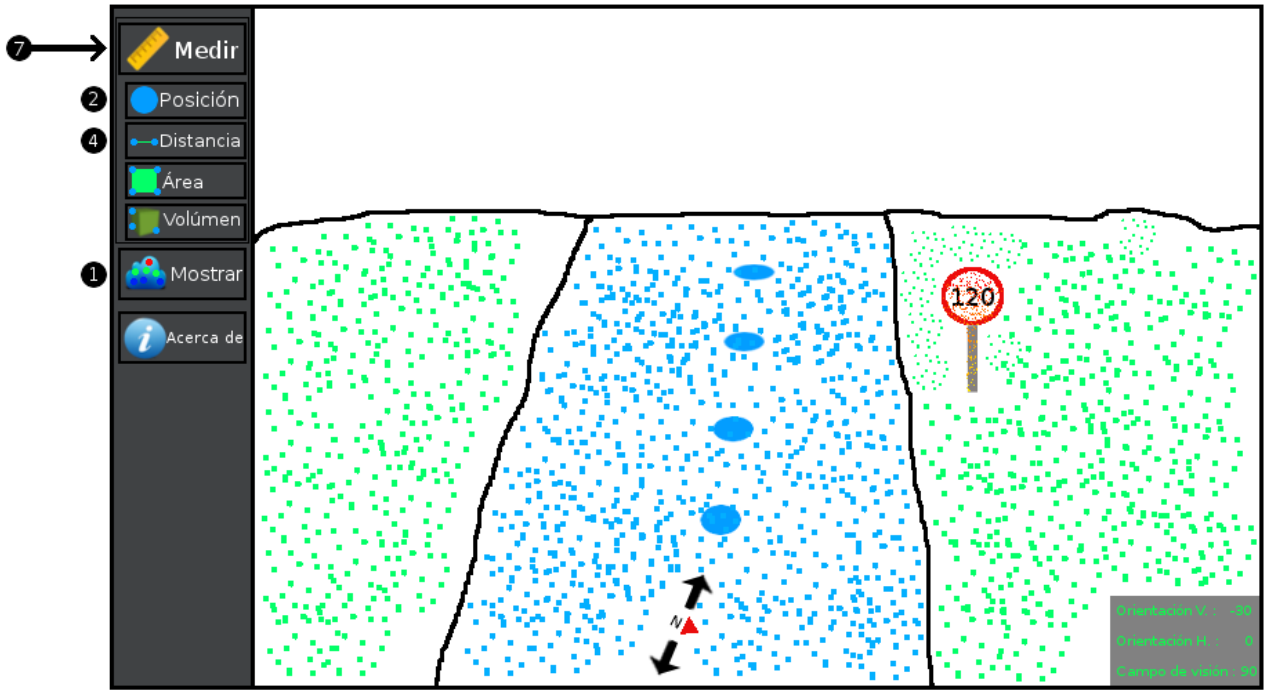

Figura 17 Prototipo GUI, visualización de la nube de puntos superpuesta al panorama

Anexo II. Interfaz público de los servicios web

A continuación se detalla la documentación del interfaz público de los servicios web que ofrece el sistema, que son *listarEscenarios* e *infoEscenario*.

Cabe decir que las rutas indicadas para los servicios web son relativas a la ruta de la aplicación en el servidor de aplicaciones.

listarEscenarios

Ruta de petición: /api/panoramas/escenarios

Tipo de petición HTTP: GET

Formato de la respuesta: JSON

Contenido de la respuesta: un vector con la lista de los nombres de los escenarios disponibles en el sistema

Significado de los estados de las respuestas:

- 200: todo correcto
- 500: ha ocurrido un error interno del servidor

infoEscenario

Ruta de petición: /api/panoramas/escenarios/{nombreEscenario}

Tipo de petición HTTP: GET

Parámetros de entrada:

- nombreEscenario: parámetro codificado en la ruta de la petición y que se corresponde con el nombre del escenario, tal y como se haya devuelto por el servicio *listarEscenarios*, cuya información se quiere consultar

Formato de la respuesta: JSON

Contenido de la respuesta: toda la información del escenario consultado, con la misma estructura que la explicada en la Vista de módulos (ver pág. 11).

Significado de los estados de las respuestas:

- 200: todo correcto
- 404: el sistema no contiene información del escenario cuyo nombre sea el proporcionado
- 500: ha ocurrido un error interno del servidor

Anexo III. Pruebas de los servicios web

A continuación se presentan las pruebas realizadas para verificar el correcto funcionamiento de los servicios web.

listarEscenarios

Petición: GET http://localhost:8080/TFG-LUKAS/api/panoramas/escenarios

Estado de la respuesta: 200 OK

Respuesta:


```
[
   "Actur"
]
```


Como en el momento de la prueba el único escenario ofrecido por el sistema se llamaba *Actur*, la prueba ha sido satisfactoria.

infoEscenario del escenario *Actur*

Petición: GET http://localhost:8080/TFG-LUKAS/api/panoramas/escenarios/Actur

Estado de la respuesta: 200 OK

Respuesta:


```
{
   "nombre":"Actur",
   "listaPanoramas":[
      {
         "localizacion":{
            "longitud":-0.89056594597054,
            "latitud":41.66596793588908,
            "altitud":245.59585656337126
         },
         "rotacion":66.4841822157959,
         "uri":"http://localhost:8080/Datos_Panoramas/Actur/imagenes/ascii_pano
ramas000000.jpg"
      },
      {
         "localizacion":{
            "longitud":-0.890521568026894,
            "latitud":41.66598069502869,
            "altitud":245.57505512674186
         },
         "rotacion":71.16410413659236,
         "uri":"http://localhost:8080/Datos_Panoramas/Actur/imagenes/ascii_pano
ramas000001.jpg"
```

```
      },
      {
         "localizacion":{
            "longitud":-0.890475722518124,
            "latitud":41.66599186636729,
            "altitud":245.54625338827046
         },
         "rotacion":72.82817843820517,
         "uri":"http://localhost:8080/Datos_Panoramas/Actur/imagenes/ascii_pano
ramas000002.jpg"
      },
      {…}
      {
         "localizacion":{
            "longitud":-0.890684309752793,
            "latitud":41.66591630841251,
            "altitud":248.89795614096155
         },
         "rotacion":64.4427520129386,
         "uri":"http://localhost:8080/Datos_Panoramas/Actur/imagenes/ascii_pano
ramas000168.jpg"
      },
      {
         "localizacion":{
            "longitud":-0.890641091683451,
            "latitud":41.66593224458264,
            "altitud":248.88423203648122
         },
         "rotacion":64.23339556601923,
         "uri":"http://localhost:8080/Datos_Panoramas/Actur/imagenes/ascii_pano
ramas000169.jpg"
      }
   ],
   "rutaNubeDePuntos":"http://localhost:8080/Datos_Panoramas/Actur/actur.ply"
}
```


Se ha sustituido gran parte de la respuesta por {…} para evitar que ocupe 15 paginas, además, la parte omitida no aporta demasiado ya que la respuesta contiene muchos datos muy similares entre sí.

Al verificar la respuesta, resulta que es correcta, ya que el número de los panoramas devueltos es el mismo que el que tiene el escenario solicitado, es decir, *Actur*, y los enlaces contenidos en la respuesta son funcionales.

infoEscenario de un escenario inexistente, por ejemplo, ABC

Petición: GET http://localhost:8080/TFG-LUKAS/api/panoramas/escenarios/ABC

Estado de la respuesta: 404 Not Found

Como el estado de la respuesta fue un error HTTP 404 Not Found, se ha verificado que el servicio web se comporta correctamente a la hora de intentar recuperar la información de un escenario que no existe en el sistema.

Anexo IV. Manual de instalación

1 Servidor

La instalación del servidor es bastante simple. Se debe disponer de un servidor de aplicaciones, por ejemplo el Apache Tomcat 8.

Antes de proceder a desplegar la aplicación del servidor, se debe modificar el fichero de configuración del proyecto Java en la ruta relativa dentro del proyecto Java *src/main/resources/config.properties*. Se debe modificar el valor de la propiedad rutaEscenarios con la ruta absoluta a la carpeta con los datos de los escenarios (para ver detalles sobre la estructura de dicha carpeta, consultar la Vista de instalación en la pág. 15).

Una vez localizado el directorio de las aplicaciones del servidor de aplicaciones, la carpeta *webapps* en el caso de Tomcat 8, se debe exportar en dicho directorio el proyecto Java del servidor en formato .war y la carpeta con los datos de los escenarios.

Finalmente, se debe arrancar el servidor de aplicaciones y el servidor debería estar listo para ser usado.

2. Cliente

Para que el cliente funcione correctamente, simplemente se debe sustituir el valor de la variable global *serverUri* por la ruta donde se encuentren disponibles los servicios web desplegados en el servidor.

3. Procesador de nubes de puntos

La instalación de la aplicación para el procesamiento de las nubes de puntos es la más complicada, debido a sus dependencias de librerías C++.

Primero de todo, se debe instalar la librería PCL. La forma más rápida y fácil de instalarla es descargando el instalador desde su web oficial http://pointclouds.org/downloads/. La otra opción sería descargar su código fuente y compilarlo.

Entonces, se debe instalar la librería libLAS. Para ello, se debe seguir el tutorial disponible en http://www.liblas.org/compilation.html .

A continuación, se debe instalar el programa CMake(http://www.cmake.org/) y seguir este tutorial http://pointclouds.org/documentation/tutorials/using_pcl_pcl_config.php para configurarlo para la compilación del código fuente de la aplicación del procesamiento de nubes de puntos. En el caso de un sistema Windows, se debe seguir el apartado “Using CMake gui (e.g. Windows)” de dicho tutorial para crear un proyecto del entorno de desarrollo Microsoft Visual Studio. Una vez creado dicho proyecto, se debe abrir y acabar de configurar, sobre todo asegurándose de que las dependencias de las librerías PCL y libLAS están correctamente configuradas. Adicionalmente, se debe configurar el proyecto para incluir la dependencia de la librería GeoPCL [10]. Una vez configurado correctamente el proyecto, se puede proceder a compilar el proyecto y obtener el ejecutable del procesador de nubes de puntos.

Para que el ejecutable funcione correctamente, los ficheros binarios de las librerías PCL y libLAS deben encontrarse en la variable de entorno del sistema *PATH*, para que el ejecutable pueda encontrarlos cuando se ejecute.

Anexo V. Manuales de usuario

A continuación se presentan los manuales de usuario de las principales funciones ofrecidas por el sistema.

1. Medir la distancia entre 2 puntos

Para poder medir la distancia entre 2 puntos, lo primero que se debe hacer es pulsar el botón “Medir” del menú principal. Una vez pulsado dicho botón, se va a desplegar un menú vertical en el que aparecen los distintos tipos de mediciones que permite el sistema. Entonces, se debe proceder a hacer clic sobre el botón “Distancia” en el menú desplegable que acaba de aparecer, tal y como se ve en la Figura 18.

Figura 18 Función de medir distancia, opción del menú

Una vez seleccionada la opción “Distancia”, se indica que se debe seleccionar el primer punto. Para ello, se debe hacer clic sobre el punto donde se quiere comenzar la medición. En la Figura 19 se ve el estado del sistema después de haber seleccionado el primer punto.

El último paso para obtener la distancia entre 2 puntos es seleccionar el segundo punto de la misma forma en la que seleccionado el primer punto, siendo el segundo punto el final de la línea cuya distancia se quiere medir. Una vez hecho clic sobre la posición deseada del segundo punto, el sistema dibuja la línea que une los puntos selecciones y muestra la distancia de dicha línea sobre su punto central, tal y como se ve en la Figura 20.

Una vez realizada la medición(o durante la medición) se puede hacer clic sobre el botón “Cancelar medición” para detener la medición en curso o eliminar la última medición realizada.

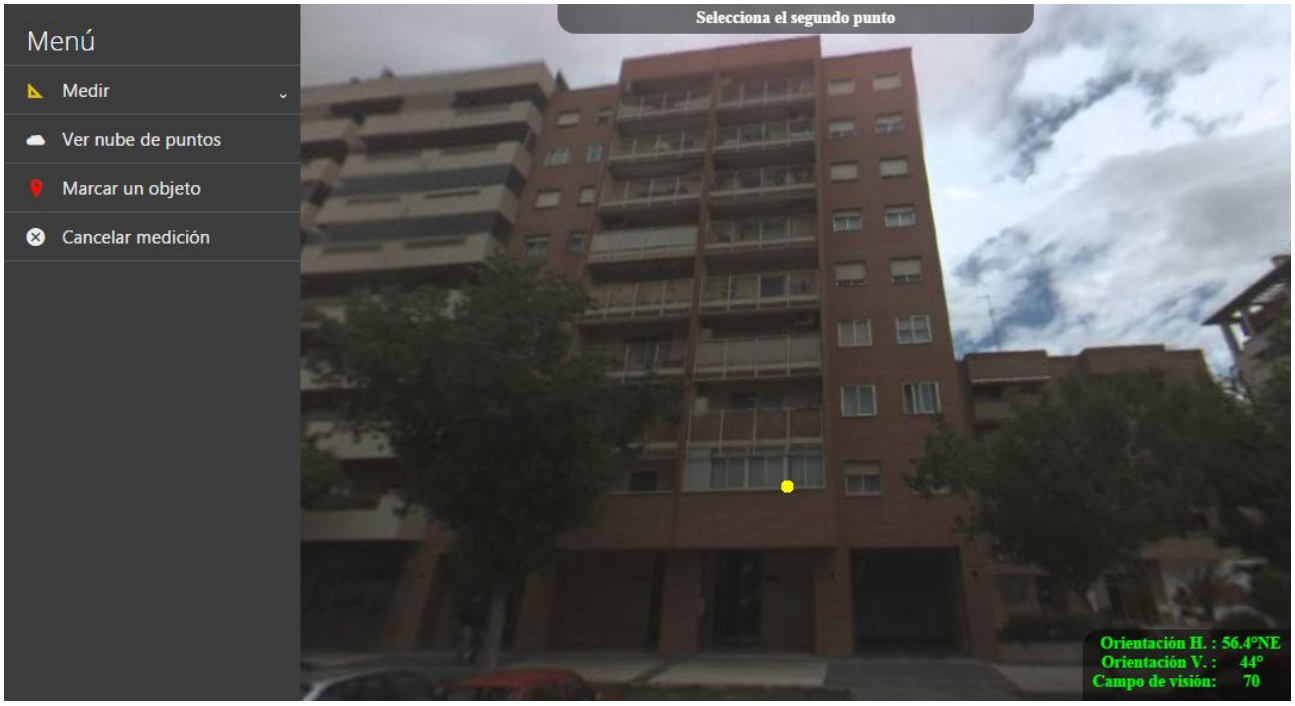

Figura 19 Función de medir distancia, primer punto seleccionado

Figura 20 Función de medir distancia, resultado

2. Obtener la posición de un punto

Para poder obtener la posición de un punto, lo primero que se debe hacer es pulsar el botón “Medir” del menú principal. Una vez pulsado dicho botón, se va a desplegar un menú vertical en el que aparecen los distintos tipos de mediciones que permite el sistema. Entonces, se debe proceder a hacer clic sobre el botón “Posición” en el menú desplegable que acaba de aparecer, tal y como se ve en la Figura 21.

Figura 21 Función de obtener posición, opción del menú

Una vez seleccionada la opción “Posición”, se indica que se debe seleccionar el punto deseado. Entonces, se debe proceder a hacer clic sobre el punto cuya posición se quiere obtener. En la Figura 22 se ve el resultado obtenido.

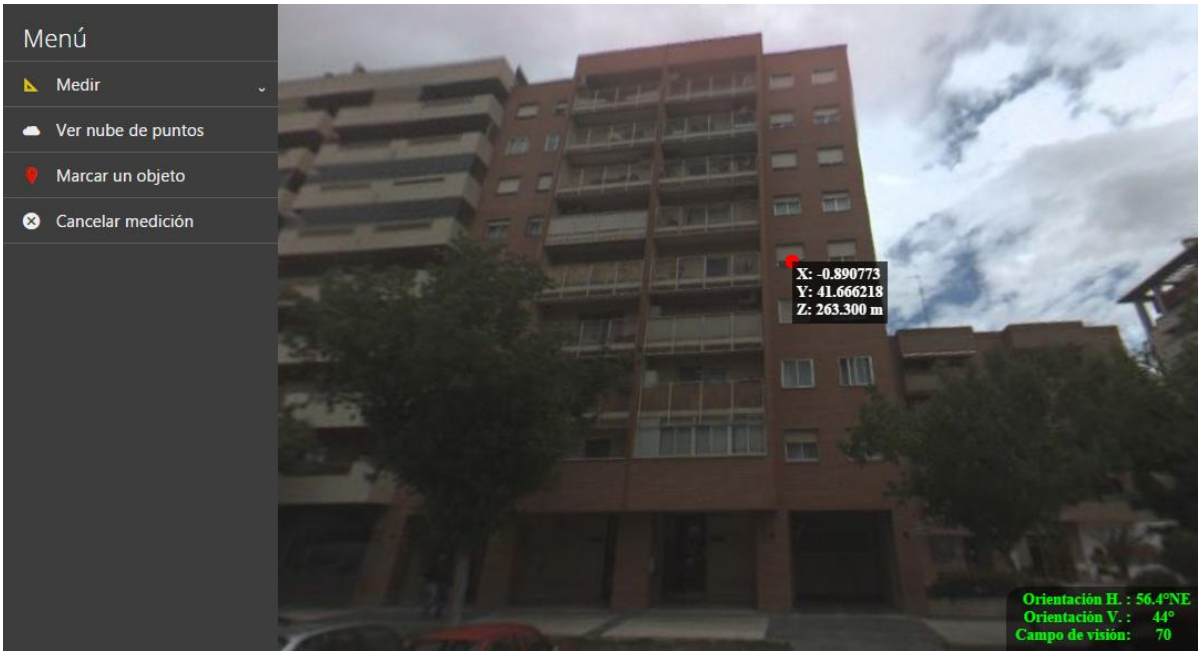

Figura 22 Función de obtener posición, resultado

3. Medir un área

Para poder medir un área, lo primero que se debe hacer es pulsar el botón “Medir” del menú principal. Una vez pulsado dicho botón, se va a desplegar un menú vertical en el que aparecen los distintos tipos de mediciones que permite el sistema. Entonces, se debe proceder a hacer clic sobre el botón “Área” en el menú desplegable que acaba de aparecer, tal y como se ve en la Figura 23.

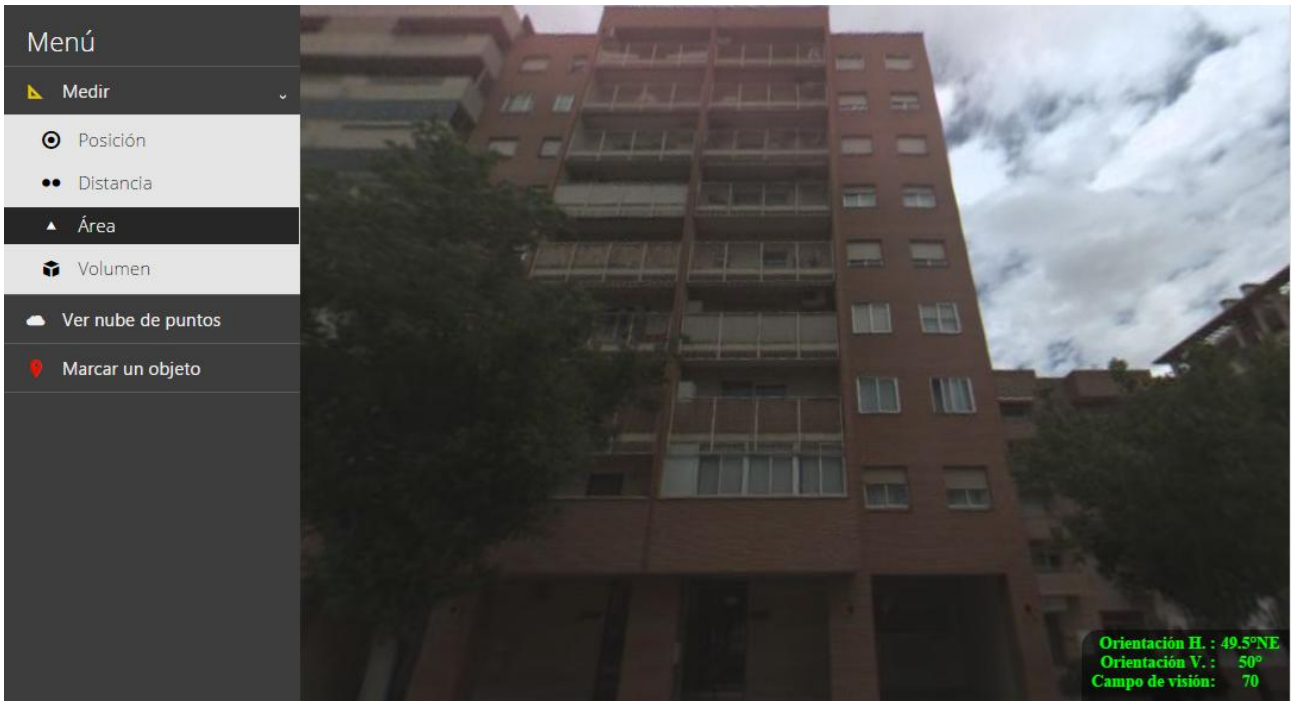

Figura 23 Función de medir área, opción del menú

Una vez seleccionada la opción “Área”, se indica que se debe seleccionar el primer punto. Para ello, se debe hacer clic sobre el punto donde se quiere comenzar la medición. En la Figura 24 se ve el estado del sistema después de haber seleccionado el primer punto. Luego se debe repetir lo mismo para seleccionar el segundo punto.

El último paso para obtener un área es seleccionar el tercer punto de la misma forma en la que fueron seleccionados los primeros puntos, de modo que los tres puntos formarán un triángulo. Una vez formado el triángulo, la aplicación dibuja dicho triángulo y muestra su área en su centro, tal y como se ve en la Figura 25.

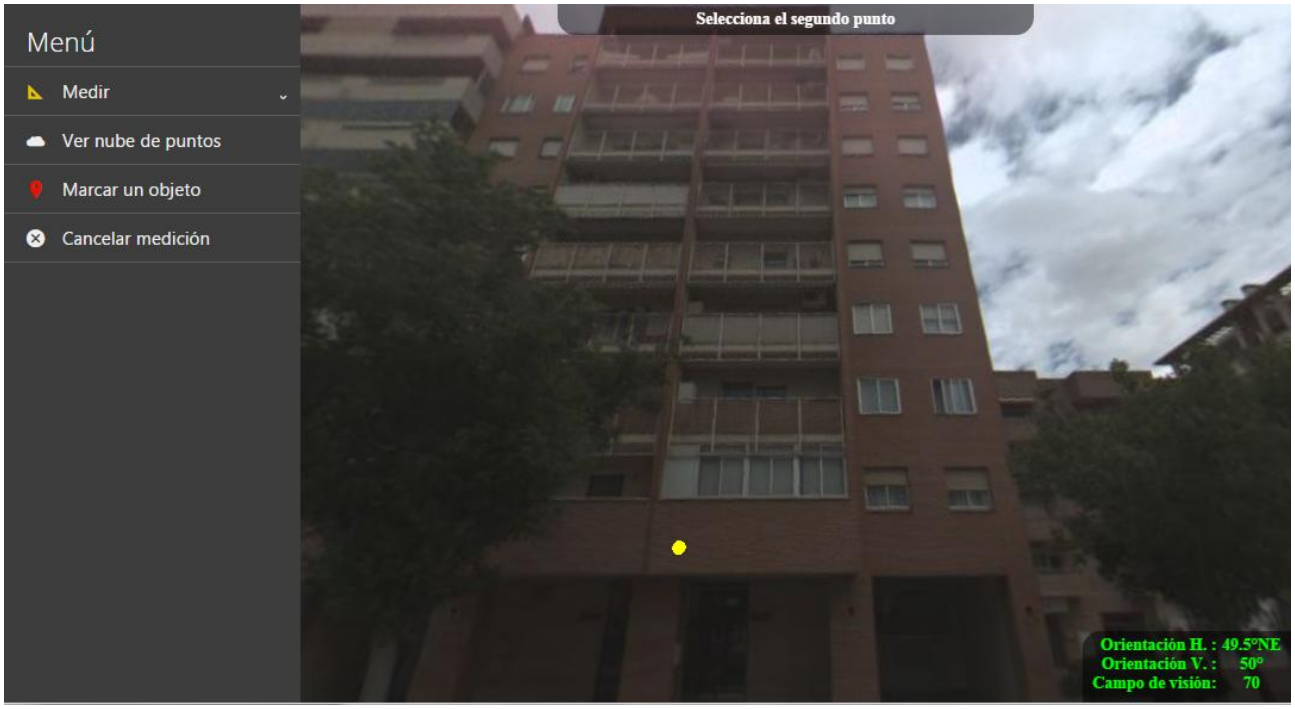

Figura 24 Función de medir área, primer punto seleccionado

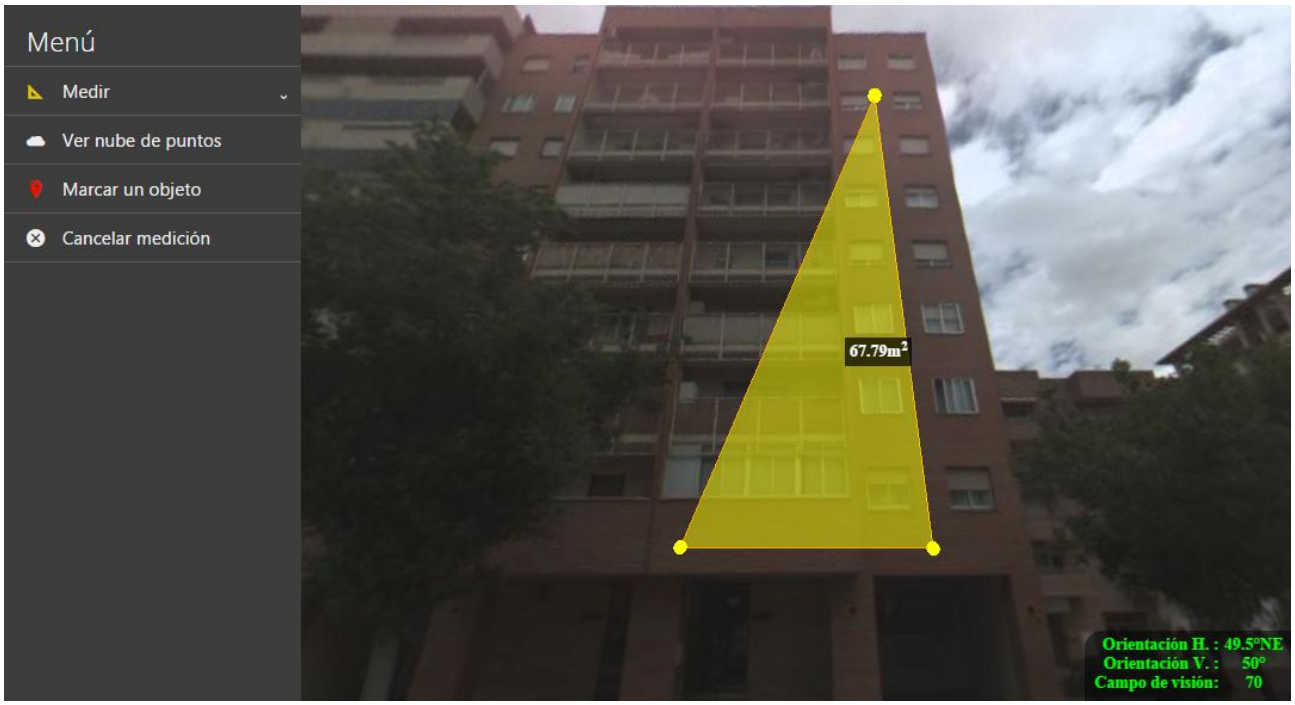

Figura 25 Función de medir área, resultado

4. Medir un volumen

Para poder medir un volumen, lo primero que se debe hacer es pulsar el botón “Medir” del menú principal. Una vez pulsado dicho botón, se va a desplegar un menú vertical en el que aparecen los distintos tipos de mediciones que permite el sistema. Entonces, se debe proceder a hacer clic sobre el botón “Volumen” en el menú desplegable que acaba de aparecer, tal y como se ve en la Figura 26.

Figura 26 Función de medir volumen, opción del menú

Una vez seleccionada la opción “Volumen”, se indica que se debe seleccionar el primer punto. Para ello, se debe hacer clic sobre el punto donde se quiere comenzar la medición. En la Figura 27 se ve el estado del sistema después de haber seleccionado el primer punto. Luego se debe repetir lo mismo para seleccionar el segundo punto.

El último paso para obtener un área es seleccionar el tercer punto de la misma forma en la que fueron seleccionados los primeros puntos, de modo que los tres puntos formarán un triángulo. Una vez formado el triángulo, la aplicación dibuja dicho triángulo y muestra su área en su centro, tal y como se ve en la Figura 28.

Figura 27 Función de medir volumen, primer punto seleccionado

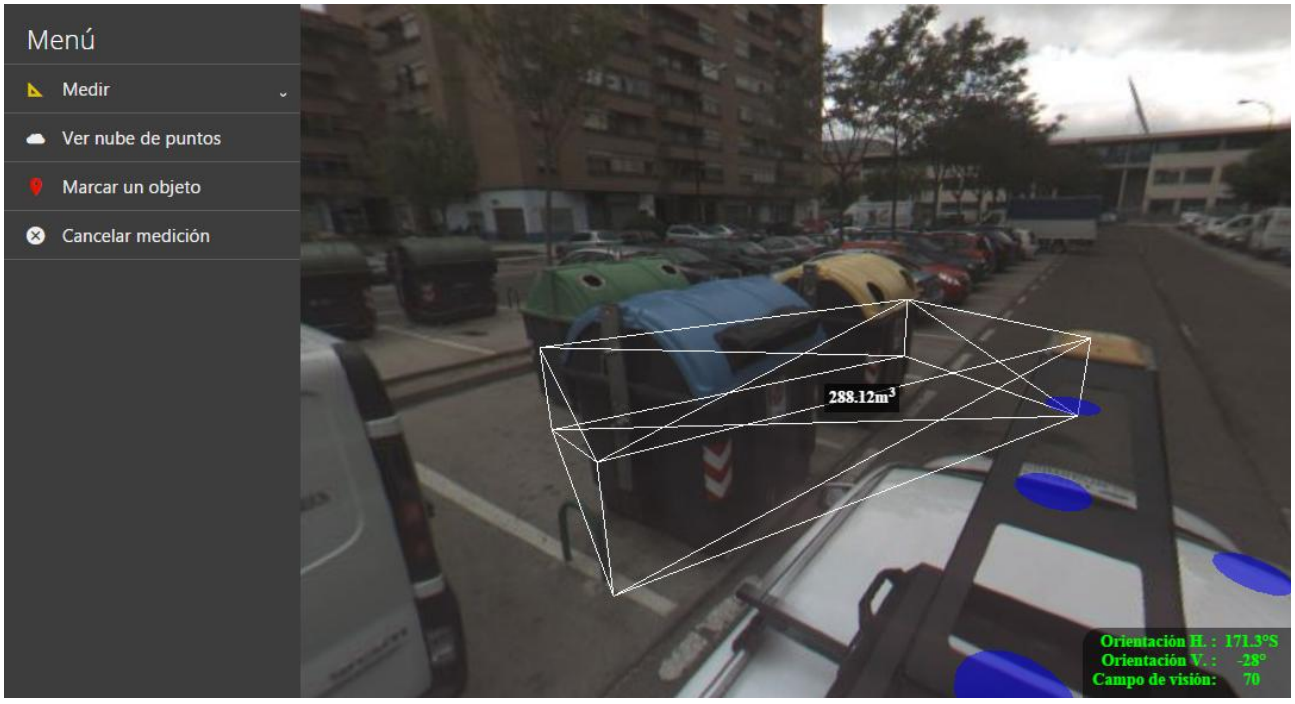

Figura 28 Función de medir volumen, resultado

5. Marcar un objeto

Para poder marcar un objeto del escenario, lo primero que se debe hacer es pulsar el botón “Marcar un objeto” del menú principal. Una vez pulsado dicho botón, se indica que se debe seleccionar la posición del objeto a marcar. Para ello, se debe proceder a hacer clic sobre el objeto a marcar. Una vez seleccionada la posición del objeto, dicha posición se marca con un punto rojo y aparece un cuadro de diálogo que solicita la información del objeto que se desea guardar en el sistema, tal y como se ve en la Figura 29.

Figura 29 Función de marcar un objeto, cuadro de diálogo

Entonces, se debe rellenar el formulario del cuadro de diálogo con la información que se quiera guardar del objeto a marcar y hacer clic sobre el botón “Marcar objeto”. Finalmente, el sistema guarda la información del objeto marcado en el navegador web del usuario y desaparece el cuadro de diálogo.

Posteriormente, cuando se quiera consultar la información de un objeto marcado, se debe hacer clic sobre la esfera roja con la que se representan todos los objetos marcados del escenario. Una vez seleccionado el objeto, su información aparece en el centro de la ventana de la aplicación, tal y como se puede ver en la Figura 30.

Figura 30 Función de marcar un objeto, viendo la información de un objeto marcado

6. Ver la nube de puntos superpuesta al panorama

Para ver la representación de la nube de puntos del escenario superpuesta a las imágenes panorámicas, simplemente se debe hacer clic sobre la opción “Ver nube de puntos” del menú principal. El resultado se puede ver en la Figura 31.

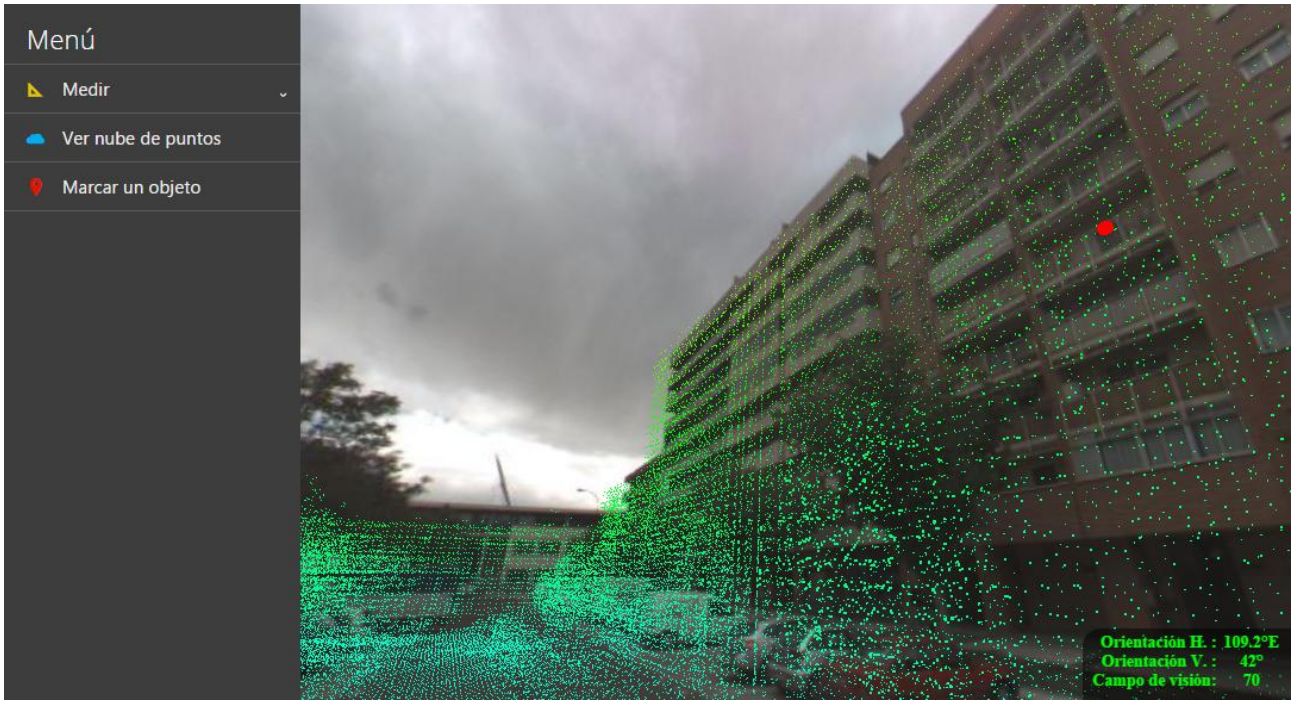

Figura 31 Función de ver la nube de puntos

7. Cambiar de panorama

Las posiciones de los panoramas más próximos a la posición actual del usuario se indican en forma de círculos azules semitransparentes, tal y como se ve en la Figura 32.

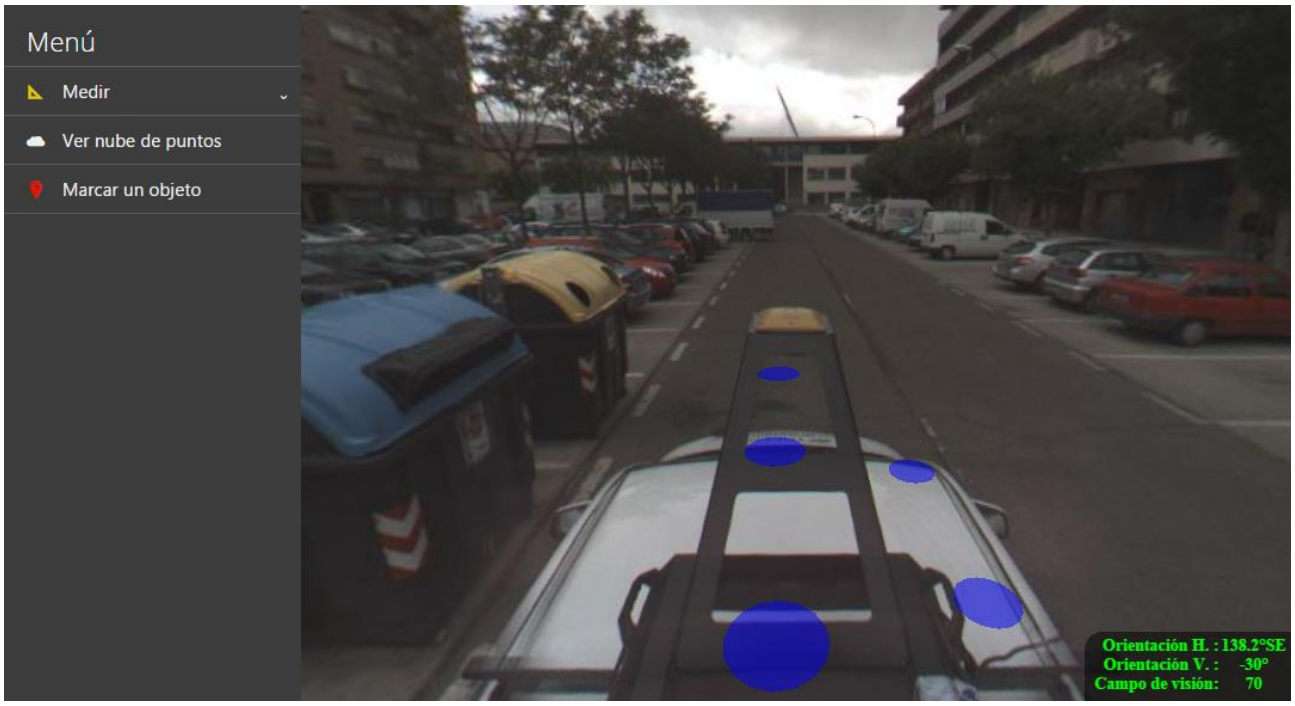

Figura 32 Círculos azules que indican la posición de los panoramas cercanos

Índice de figuras

Índice de tablas

Glosario

IAAA: el Grupo de Sistemas de Información Avanzados (IAAA), un grupo de I+D adscrito al Instituto de Investigación en Ingeniería de Aragón de la Universidad de Zaragoza.

HTTP: Hypertext Transfer Protocol es un protocolo de red, en concreto, del nivel de aplicación. Es usado ampliamente para las peticiones en la web. Se caracteriza por ser un protocolo sin estado.

CSS: Cascading Style Sheets u hojas de estilos en cascada, es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML, publicado por el World Wide Web Consortium (W3C).

GUI: Graphical User Interface, es la interfaz gráfica de usuario.

API: Application Programming Interface, es el conjunto de funciones que ofrece un sistema para ser utilizado por otro software como una caja negra, abstrayendo los detalles de su implementación.

RESTful: Representational State Transfer, es una técnica de arquitectura de software para los sistemas web que se caracteriza por ser un protocolo sin estado y con un conjunto de operaciones bien definido, entre las cuales las más importantes son GET, POST, PUT y DELETE.

AJAX: Asynchronous JavaScript And XML, es una técnica de desarrollo web que permite comunicaciones asíncronas en el cliente para poder crear aplicaciones web avanzadas y con una experiencia de usuario mejorada.