Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo de una plataforma de realidad aumentada para museos en Unity 3D

Autor/es

Antonio Bazco Nogueras

Director y ponente

Director: Ramón Piedrafita Moreno

Ponente: José Ramón Beltrán Blázquez

Escuela de Ingeniería y Arquitectura

2013/3014

Desarrollo de una plataforma de realidad aumentada para museos en Unity 3D

Resumen

La realidad aumentada está experimentando actualmente un gran crecimiento, desarrollándose en ámbitos como la educación, la medicina, aplicaciones industriales y militares, el turismo o el entretenimiento. Una de las aplicaciones más interesante es su uso en museos y actividades culturales, y hoy en día los principales centros ya disponen de herramientas o servicios relacionados con este fenómeno.

El objetivo principal de este TFG es la creación de una herramienta que haga uso de la realidad aumentada para aportar un valor añadido a la visita de un museo. Para ello, se basa en la tecnología de detección y seguimiento de marcadores para el correcto posicionamiento de la información digital en el entorno real. Además, utiliza un servicio remoto de almacenamiento para tener siempre disponibles los elementos que se desean mostrar, realizándose la descarga de estos cuando el usuario se encuentra en presencia del marcador.

Como propósito se encuentra el crear una aplicación que sea funcional en la mayoría de los dispositivos actuales, y que por lo tanto no requiera de unas altas especificaciones para su correcto funcionamiento, ni tenga requerimientos excesivos de memoria.

Se ha dado soporte a datos multimedia, como audio, imágenes, texto, modelos 3D, enlaces a direcciones web o vídeos –a partir de dichos enlaces– para enriquecer la experiencia al disponer de una variedad de recursos a disposición del usuario. También se ha diseñado una aplicación de gestión de los elementos depositados en la base de datos.

Índice general

Índice de figuras

MEMORIA

1. Introducción

El presente documento tiene como objetivo recoger toda la información relacionada con la realización del Trabajo Fin de Grado (TFG) titulado: *Desarrollo de una plataforma de realidad aumentada para museos en Unity 3D*.

Este trabajo pertenece a la titulación de Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación de la Universidad de Zaragoza, y ha sido realizado por el alumno Antonio Bazco Nogueras y dirigido y supervisado por Ramón Piedrafita Moreno del Departamento de Informática e Ingeniería de Sistemas (DIIS), actuando como ponente José Ramón Beltrán Blázquez.

En el primer capítulo se incluyen el contexto académico y el tecnológico de este trabajo, los objetivos y motivaciones que han llevado a su realización y la estructura en la que se organiza este documento.

1.1 Contexto académico

Este proyecto se desarrolla como trabajo fin de grado del Grado de Ingeniería en Tecnologías y Servicios de Telecomunicación, dentro del grupo de Sistemas de información Avanzados (IAAA), cuya investigación se centra en el desarrollo de sistemas de información geoespacial. Dicho grupo pertenece al Departamento de Informática e Ingeniería de Sistemas (DIIS) de la Universidad de Zaragoza.

1.2 Contexto tecnológico

Es conveniente enmarcar el trabajo en el contexto tecnológico actual, mostrando las herramientas de las cuales se sirve, para permitir una mayor visión de futuro y unas expectativas acordes a nuestro tiempo. En el Anexo B se explican detalladamente cada una de las herramientas y lenguajes utilizados en este trabajo.

Dichas herramientas son:

- ***Unity 3D***: Esta plataforma para el desarrollo de aplicaciones se ha utilizado como entorno de desarrollo y se ha aprovechado su opción de exportar proyectos a Android. Una consecuencia de realizar la aplicación mediante esta plataforma es que no se ha desarrollado en Java, el lenguaje nativo de Android, si no en C#, que es uno de los lenguajes, junto con Javascript y Boo, en los que se puede programar en Unity.

- ***Qualcomm® Vuforia™***: Esta herramienta permite incluir en un proyecto de Unity una completa API de realidad aumentada. Esta herramienta es la que añade las funcionalidades necesarias para la construcción de una aplicación de realidad aumentada, lo que la hace indispensable para el desarrollo de este trabajo.

- ***Parse***: Parse es un *Backend as a service* (BaaS) que ofrece múltiples funcionalidades en la nube. Ha sido seleccionado entre otros servicios por su facilidad de integración en Unity, y es utilizado para actuar como base de datos remota donde se almacenan los datos del museo y que son descargados cuando la aplicación lo solicite.

Además, los lenguajes de programación utilizados en este trabajo han sido C#, Javascript, HTML y CSS.

1.3 Objetivos y motivación del trabajo

La facilidad de acceso a Internet de alta velocidad que existe actualmente, principalmente en los entornos urbanos, ha potenciado de manera exponencial el desarrollo de aplicaciones y servicios para dispositivos móviles. Este crecimiento se ha debido en gran parte al éxito de servicios relacionados con las redes sociales, la interactividad y la geolocalización.

Términos como los anteriores han pasado a ser referentes en lo que a actualidad tecnológica se refiere, y en muchos ámbitos se está desarrollando un creciente interés por integrar herramientas de este tipo entre los servicios ofrecidos. De este modo, se ha generalizado el uso de la tecnología como complemento transversal a cualquier actividad, tanto de interés comercial como social.

Uno de los entornos donde más se está implantando esta tendencia es en el cultural. La búsqueda de una experiencia más dinámica y atractiva para los usuarios y visitantes ha provocado que los museos estén muy interesados en la implantación de servicios interactivos entre su oferta de contenidos.

Estas nuevas herramientas culturales establecen nuevos alicientes para el ciudadano, que además de disfrutar de una interesante experiencia cultural de manera más intuitiva y activa, puede acceder a un apoyo más completo sobre la visita –rutas, guías, relaciones…-, así como a una mayor cantidad de información de la que el centro podría mostrar en su entorno físico.

Por lo tanto, estos servicios son interesantes en el ámbito cultural porque además de atraer a diferentes sectores sociales menos interesados en este tipo de instituciones, a través de la innovación, el entretenimiento y la tecnología, permiten mejorar la experiencia cultural mediante la adición de información complementaria y la generación de nuevas posibilidades de estructuración y presentación.

Una de las herramientas más utilizadas en el entorno cultural es la Realidad Aumentada. La Realidad Aumentada permite la interacción en tiempo real de la información física presente en el museo y de la información adicional que complementa a ésta, facilitando la integración de las nuevas posibilidades a la tradicional forma de realizar esta actividad.

Además de lo anteriormente expuesto, entre las motivaciones para realizar este trabajo está el hecho de poder realizar una herramienta tecnológica que pueda ofrecer una mejora en la experiencia cultural, y haga ésta más atrayente para todos. Además, el poder desarrollar un trabajo basado en Realidad Aumentada es de gran interés, dado que es un concepto novedoso y de gran futuro, que puede ser una gran innovación en amplios sectores de nuestra sociedad.

Este trabajo tiene como objetivo la creación de una aplicación para dispositivos móviles que mejore la experiencia personal del usuario, haciendo uso de la Realidad Aumentada y del almacenamiento de datos en una base de datos remota. Dicha aplicación mostrará los elementos presentes en la sección del museo en la que se encuentre el usuario, ofreciendo información multimedia adicional de cada uno de los elementos. Esta información podrá consistir en textos, audios, imágenes, modelos 3D o enlaces web de interés.

Se ha seleccionado para su implementación el *Museo de Informática Histórica* de la Universidad de Zaragoza, localizado en el edificio Ada Byron del campus Río Ebro[i]. Para su realización se integrarán diferentes actividades como:

- Utilización del servicio Parse para la creación de la base de datos remota en la nube y la estructuración de dicha base.

- Integración de la plataforma Vuforia en Unity para la gestión de los marcadores y por lo tanto de la localización.

- Desarrollo de la aplicación en Unity3D, integrando la comunicación con la base de datos y gestionando la información recibida.

- Desarrollo de una aplicación de gestión de la base de datos, para permitir la actualización de los elementos.

[i] http://www.mih.unizar.es

1.4 Estructura del documento

Este documento consta de dos partes diferenciadas. La primera está dedicada a la Memoria en sí de este TFG, que ofrece una visión global del trabajo realizado durante el desarrollo de este trabajo, y a la cual pertenece este apartado; la segunda parte contiene Anexos que añaden y complementan la información de la Memoria.

- Parte I: Memoria
  - Introducción
  - Trabajo realizado.
  - Conclusiones y líneas futuras.
  - Bibliografía y referencias.

- Parte II: Anexos
  - Anexo A: Gestión y desarrollo del trabajo.
  - Anexo B: Contexto tecnológico y herramientas utilizadas.
  - Anexo C: Formato OBJ.
  - Anexo D: Imágenes y capturas de pantalla de la aplicación

2. Trabajo realizado

2.1 Estado del arte

Uno de los primeros pasos en la realización de este trabajo fue la búsqueda de información relacionada con él temática y tecnológicamente; el objetivo de esta búsqueda es la de adquirir un mayor conocimiento sobre la problemática que se va a abordar, para detectar posibles deficiencias o necesidades que tengan que tenerse en cuenta en el desarrollo del trabajo.

Como ya se ha comentado la realidad aumentada está teniendo un auge significativo en muy diversos ámbitos de la sociedad. Es destacable su uso en ofertas culturales. Los grandes centros museísticos, como el MoMA de Nueva York, el Museo de Londres o el Museo del Louvre de París, han implementado aplicaciones propias para ofrecer características únicas a los usuarios, complementando a los elementos físicos del museo o incluso siendo la aplicación de realidad aumentada el objeto de exposición y centro de interés de la visita (1) .

Estas nuevas opciones de interactividad mejoran las que podíamos disfrutar anteriormente en este tipo de infraestructuras, las conocidas como visitas virtuales (2). Uno de los nuevos requisitos, que es muy interesante desde el punto de vista turístico, es la necesidad de que el usuario esté localizado en una zona concreta. Esto reviste de interés porque dota al lugar turístico de un reclamo que genera un efecto llamada para el usuario. Por ello, grandes ciudades también han desarrollado servicios de realidad aumentada no localizados en un centro museístico, si no que se pueden disfrutar por todo el territorio de la ciudad. De esta manera, el turista dispone de elementos de interés durante toda su estancia. Ejemplos de estas aplicaciones son el proyecto que permite ver el antiguo muro de Berlín en todo su recorrido a través de la pantalla del móvil (3), o la aplicación Streetmuseum de Thumbspark Limited, que ofrece la visión de imágenes antiguas del lugar de Londres donde nos encontremos, invitando al usuario a situarse en la posición idónea para que la imagen real se funda con la antigua y virtual (4).

Actualmente, los *frameworks* de realidad aumentada más utilizados son:

- ***Basados en marcadores:***
    - Vuforia: Al ser soportado por Unity 3D permite la exportación de un proyecto tanto para iOS como para Android (5).
    - NyARToolKit: Nacido de un primer *framework* para ordenadores utilizando la cámara web. Se puede obtener tanto con licencia comercial como GPLv2 (6).
    - DroidAR: Aunque está basado en marcadores permite posicionar elementos en base a la posición relativa respecto del observador (7) (8).

- ***Basados en geolocalización y orientación:***
    - Mixare: De código abierto, no permite la categorización de los *POI* (9).
    - Wikitude: De licencia privativa, permite localización absoluta o relativa al usuario y se puede integrar con Vuforia para el tratamiento de imágenes (10).
    - LookAR!: Desarrollado por la Universidad Complutense de Madrid. Se basa en localización por medio señales wifi y localización inercial (11) .

- ***Con ambas funcionalidades:***
    - Layar: De licencia privativa, su estructura en capas permite al desarrollador definir contenidos (12) .
    - Junaio: Basado en geolocalización, mejora los resultados obtenidos a través de esta, sobre todo en interiores, mediante marcadores que definen su latitud, longitud y altitud; a partir de estos marcadores se puede ajustar la precisión de la situación (13) .

2.2 Análisis de requisitos

Para el desarrollo de una aplicación es necesario definir una serie de requisitos que guíen y estructuren la labor de diseño e implementación. Por ello, se han definido estos requisitos:

- Requisitos funcionales:
  - RF1: La aplicación debe ejecutarse en dispositivos móviles.
  - RF2: La aplicación debe permitir elegir idioma.
  - RF3: La aplicación debe mostrar la información relacionada con la zona del museo en la que se encuentra.
  - RF4: La aplicación debe ser capaz de conectarse a una base de datos remota para descargar la información necesaria.
  - RF5: La aplicación debe gestionar la interacción con la interfaz táctil para ofrecer una navegación intuitiva.
  - RF6: Se debe gestionar información multimedia, como audio, imágenes, modelos 3D o textos.
  - RF7: Debe ser posible la creación, actualización y eliminación de los objetos de la base de datos por el gestor de la aplicación.
  - RF8: El acceso a los elementos de la base de datos para su actualización debe poder realizarse desde cualquier lugar mediante acceso a la dirección web de una aplicación de gestión.
  - RF9: Se deben limitar las funcionalidades de usuarios anónimos para la base de datos remota.

- Requisitos no funcionales:
  - RNF1: El usuario debe utilizar la aplicación desde un dispositivo Android con cámara y con versión igual o superior a Android 2.3.1 Gingerbread
  - RNF2: El usuario debe tener acceso a internet desde su dispositivo.
  - RNF3: Se deben situar los marcadores correspondientes en los lugares adecuados.

2.3 Plataforma

Es necesario seleccionar el sistema final sobre el que se va a implementar el trabajo realizado para dispositivos móviles.

Hoy en día el mercado de los dispositivos móviles inteligentes, incluyendo *tablets* y *smartphones*, está claramente dominado por la plataforma Android, la cual continúa aumentando su cuota de mercado, como se puede ver en la Figura 2.1. Esta razón, además de la facilidad para el desarrollo y lanzamiento de aplicaciones que esta plataforma ofrece respecto a otras, hace que se haya seleccionado como plataforma destino de este trabajo. La aplicación soporta el sistema operativo Android desde la versión 2.3.1 *Gingerbread* en adelante.

Pese a ser esta la elección, el hecho de que la aplicación se haya desarrollado en Unity hace que ésta pueda implementarse para cualquiera de las plataformas que soporta. En el ámbito de los dispositivos móviles Unity permite exportar proyectos a Android, iOS, Windows Phone 8 y Blackberry. Esta característica dota de gran flexibilidad al desarrollo de aplicaciones desde Unity, haciendo posible con un mismo proyecto la creación de aplicaciones para las cuatro grandes plataformas de dispositivos móviles.

No se incluyen como plataformas de uso las antes comentadas debido a que sólo se han podido realizar pruebas en dispositivos Android, y no se ha comprobado si su funcionamiento es correcto. Pese a ello, es resaltable que la exportación a estas plataformas no requeriría de un gran esfuerzo por parte del desarrollador, si no que sería suficiente con realizar las pruebas necesarias y realizar pequeñas modificaciones para su funcionamiento estable en todas ellas.

Figura 2.1 Cuota de mercado de plataformas móviles en España (14)

2.4 Estructura general del sistema

El objetivo final de este trabajo es la creación de una aplicación de realidad aumentada para dispositivos móviles con almacenamiento de datos remoto. Por ello, se puede dividir en dos estructuras relacionadas pero separadas: La aplicación de usuario, por un lado, y, por el otro, la base de datos remota y la aplicación para su gestión.

El nombre elegido para la aplicación es ***MuseAR***, y se ha seleccionado el Museo de Informática Histórica como entorno de validación. Se ha diseñado en dos idiomas: español e inglés, de forma que pueda llegar a un mayor número de usuarios.

La aplicación se ha desarrollado utilizando el motor de videojuegos Unity3D -en su versión 4.3-, que permite la exportación multiplataforma de un mismo proyecto, además de diversas funcionalidades interesantes para la realidad aumentada. Se ha programado utilizando el lenguaje C# y se han utilizado tanto el SDK de Qualcomm® Vuforia™ como el del servicio remoto Parse.

2.4.1 Aplicación MuseAR

Como ya se ha comentado anteriormente, la aplicación para el usuario ha sido desarrollada en Unity. La estructuración de un proyecto en esta plataforma se basa en escenas, que son independientes entre sí. Cada una contiene los elementos y las relaciones entre ellos necesarias para su correcto funcionamiento.

Como se puede ver en la Figura 2.2, MuseAR solamente hace uso de dos escenas. Dichas escenas son:

- **Primera escena** - ***Intro***:
  Escena inicial de la aplicación. Es una escena muy simple, cuyo objetivo es el de presentar tanto a la aplicación como al grupo de Sistemas de información Avanzados –IAAA–. Permite elegir el idioma en el que va a funcionar la aplicación. A través de ella se accede a la segunda escena, que es la principal y la que contiene todos los elementos de navegación, conexión remota y realidad aumentada de los que hace uso el sistema.

- **Segunda escena - *MuseAR***:
  Escena que contiene el núcleo de la aplicación. Integra las funcionalidades necesarias para la correcta navegación e incluye la gestión de la descarga de datos y la fusión de la imagen real y los datos virtuales.

Centrándonos en la segunda escena, *MuseAR*, también está dividida en dos secciones principales.

- ***Módulo de AR y navegación***
  Hace uso del SDK de Vuforia para gestionar la integración de información real y virtual además de contener el menú de navegación.

- ***Módulo de conexión con el servicio remoto***
  Gestiona la comunicación con la base de datos mediante el SDK de Parse.

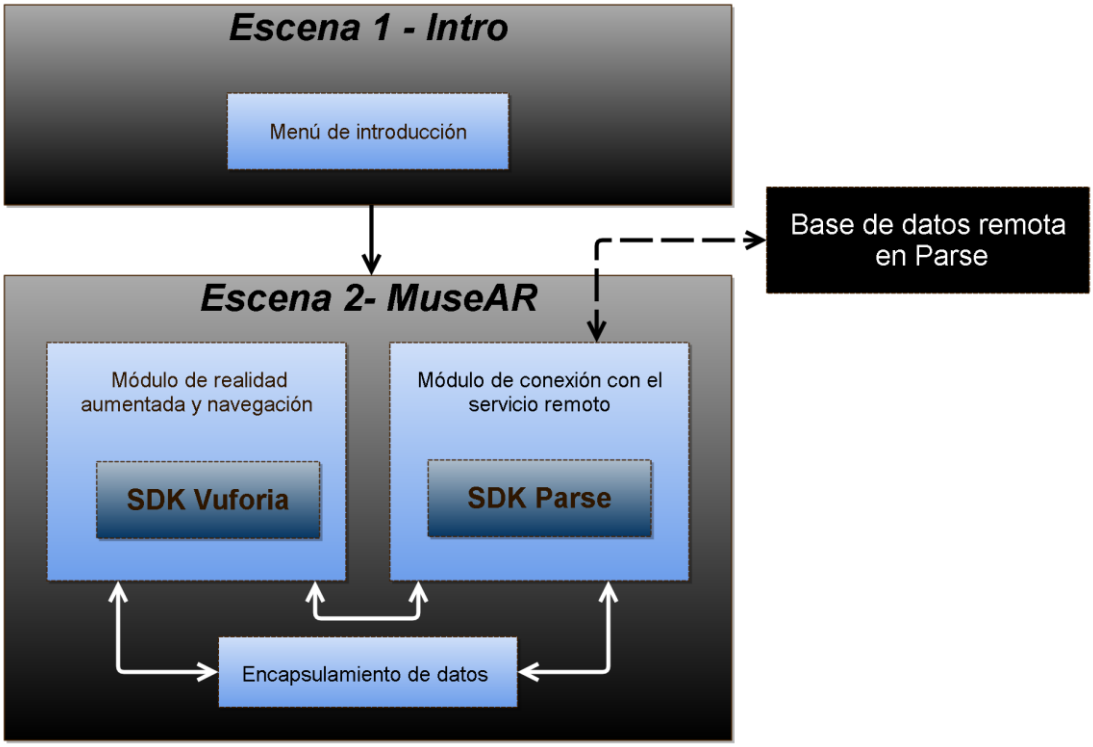

Figura 2.2 Esquema global de la aplicación

El módulo de conexión con el servicio remoto, implementado mediante la clase *Download*, abstrae al resto de la aplicación de la descarga de datos. Por lo tanto, sería posible utilizar un servicio de almacenamiento remoto distinto, ya sea mediante otros servicios de terceros o con el desarrollo de un servidor propio. Este cambio no implicaría ninguna modificación en el módulo de realidad aumentada, que es el que caracteriza realmente la aplicación, y sólo sería necesario modificar la clase encargada de la gestión de la descarga de información. A su vez, se podría modificar las características de navegación o de AR sin tener que alterar el modo de obtención de los archivos.

Además, el uso de un servicio de almacenamiento externo hace posible que, si se deseara hacer uso de esta aplicación para otro museo, bastara con introducir en la base de datos los elementos necesarios a través de la aplicación de gestión.

Estas características pretenden dotar al proyecto de cierta flexibilidad en el uso final, posibilitando su utilización en casos diferentes al que se ha elegido durante su desarrollo.

Otra característica general de esta escena es la definición de la clase *POI* para encapsular el modelo de datos que tenemos. Cada instancia de esta clase contiene la información multimedia de uno de los objetos reales que se pueden encontrar en el museo. Además, se hace uso de la clase *Language* para gestionar el uso del idioma seleccionado por el usuario.

Cada una de las partes aquí expuestas son explicadas con mayor detalle en los apartados posteriores.

2.4.2 Base de datos y aplicación de gestión

Una de las limitaciones de los dispositivos móviles – *smartphones* – es su capacidad de almacenamiento. Dado que la aplicación se ha diseñado para este tipo de dispositivos, la elección del modo en el que se almacenan los datos es muy importante.

El almacenamiento interno de los datos dentro de la aplicación sería inviable. Primero, porque la cantidad de datos puede llegar a ser considerable, dependiendo del tamaño de museo, y esto haría la aplicación muy pesada, pudiendo superar ésta el espacio de almacenamiento disponible. Además, requeriría la actualización de la aplicación cada vez que se modificara la información del museo o se quisieran añadir o eliminar más datos.

Por ello, se ha elegido el uso de una base de datos remota. El uso extendido de internet de alta velocidad en los dispositivos móviles, así como la existencia de redes Wi-Fi en la práctica totalidad de centros culturales hace que esta elección sea la más adecuada. Los datos sólo se descargarán cuando sea necesario – es decir, cuando el usuario entre en la zona del museo correspondiente – y serán eliminados al finalizar la visita, permitiendo un gran ahorro y una mejora de eficiencia en el almacenamiento de datos.

Una vez establecida la elección de una base de datos remota, se debe elegir cómo se va a configurar ésta. Se ha seleccionado un servicio de almacenamiento en la nube ya que ofrece un servicio completo y seguro y evita la necesidad de crear un servidor propio e integrar una base de datos, lo que conllevaría una dedicación de tiempo considerable y una

cantidad de complicaciones o errores que de esta manera quedan relegados al buen funcionamiento del servicio.

Por ello, se ha seleccionado Parse para el almacenamiento de los datos. No es el único servicio de este tipo, pero se ha seleccionado entre otros motivos por la existencia de soporte en una gran variedad de plataformas y su gratuidad para aplicaciones con poco volumen de almacenamiento y tráfico.

Para gestionar la información almacenada en la base de datos, de forma que se puedan añadir, actualizar o eliminar los elementos existentes en ella, se ha desarrollado una aplicación de gestión de dicha base de datos. A esta aplicación se puede acceder mediante la conexión a la página web correspondiente, y requiere de la contraseña del administrador. En el apartado 2.9 se explica con mayor detalle esta aplicación y en el Anexo D se pueden ver imágenes de ésta.

Figura 3 Vista parcial de la aplicación de gestión

Tanto la base de datos como la aplicación de gestión tienen como entidad básica de estructuración el *Point of Interest –POI–*, elemento que engloba datos multimedia relativos a un elemento real del museo.

2.5 Diseño de la base de datos y el tipo de datos

A la hora de construir la aplicación es necesario primero estructurar correctamente la base de datos y definir la organización de datos a utilizar de acuerdo a tus necesidades.

Una estructura acorde a los requerimientos facilita la implementación posterior. En este sentido, se ha definido la clase *POIList* para encapsular los datos obtenidos de la base de datos, construyendo ambas entidades de forma paralela.

2.5.1 Clases *POI yPOIList*

En Realidad Aumentada, un POI, o *Point Of Interest,* hace referencia a cada uno de los elementos del entorno real que tienen una entidad virtual asociada. Estas entidades suelen aportar normalmente información adicional sobre el elemento real.

En este trabajo, un POI hará referencia a cada uno de los elementos museísticos sobre los que se tiene datos de interés en la base de datos.

Para encapsular dichas entidades se ha definido la clase *POI*, la cual es una clase interna de *POIList*. *Ésta* define la colección de elementos disponibles de la sección en la que nos encontremos, facilitando la gestión y manipulación de estos al agruparlos en un solo objeto. *POIList* es una clase contenedor de *POI*, creada para simplificar la manipulación de la lista de elementos.

Su diagrama de clase se puede observar en la imagen siguiente –Figura 2.5–. *visiblePOIs*, *maxDistanceToRender*, *clearDownloadList* y *sortByDistance* son campos y métodos implementados para el caso en el que la aplicación funcione con geolocalización.

Dentro de esta clase está definida la clase raíz, que define la estructura de los POI. Como podemos ver en la Figura 2.4, posee campos privados para almacenar cada uno de los tipos de datos que se pueden dar, así como propiedades públicas para acceder a dichos campos, permitiendo

su lectura pero no su modificación, y también almacena los campos disponibles para la posterior presentación en pantalla.

Esta clase nos permite un uso más funcional e intuitivo de los elementos del museo en la propia aplicación.

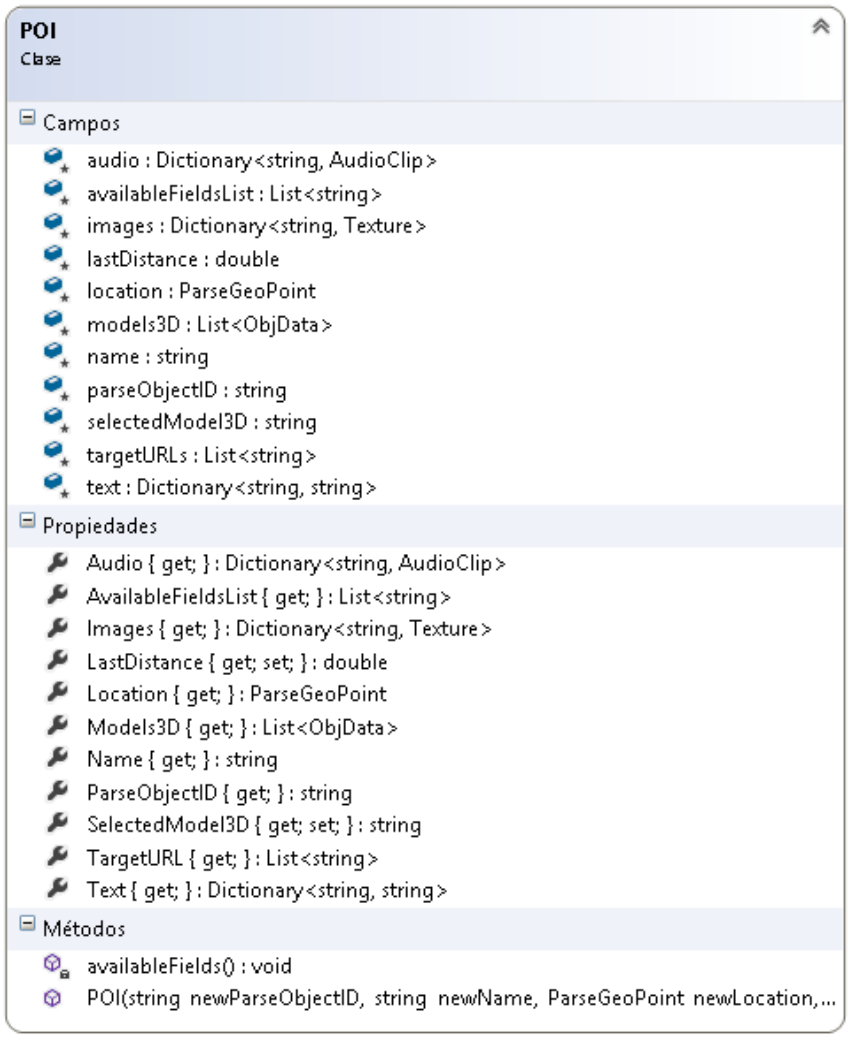

Figura 2.4 Propiedades y métodos de la clase POI

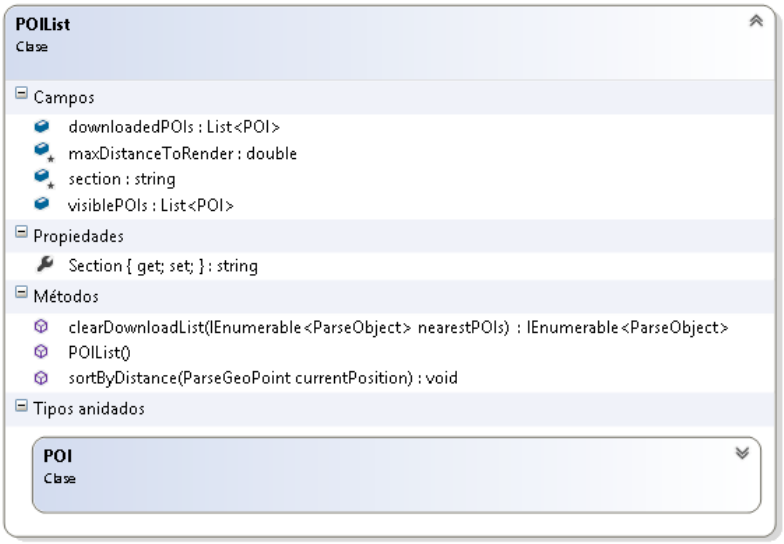

Figura 2.5 Propiedades y métodos de la clase POIList

2.5.2 Estructura de la base de datos

Una vez estructurada la clase contenedora de los elementos, se ha de construir el esquema de la base de datos de manera similar, facilitando la actualización y petición de elementos.

El servicio Parse permite la creación de clases personalizadas. Para nuestra aplicación, se han definido seis clases con sus respectivos campos.

Todas las clases contienen campos generales, como son los que existen por defecto en Parse –*objectId*, *user*, *createdAt*, *ACL*, *updatedAt*– y otros añadidos para su utilización en la aplicación, como *name* o *done*.

Cuatro de esas clases – *Audio*, *Images*, *Textures* y *Text* – son simples contenedores de ficheros de audio (.wav) , imágenes (.png o .jpg), texturas (.png, .jpg o .tga) y texto (.txt), respectivamente. La clase *Model3D* contiene un fichero .obj, su correspondiente fichero de materiales .mtl y un puntero a las texturas de dicho modelo.

Y, por último, la clase POI es la más compleja, y hace uso de las otras clases para almacenar la información multimedia. Consta de un puntero a elementos de otra clase por cada tipo de datos, así como campos para

almacenar los enlaces a páginas web relacionadas, la localización del elemento y la sección a la que pertenece.

Esta última clase es la que se usa para simplificar la descarga de archivos en ejecución, ya que permite la obtención de múltiples archivos a través de un solo elemento.

Como se puede observar, existe un paralelismo entre la estructura de la base de datos y de la clase POI, requisito indispensable para una correcta implementación de la comunicación entre la aplicación y el servicio de almacenamiento remoto.

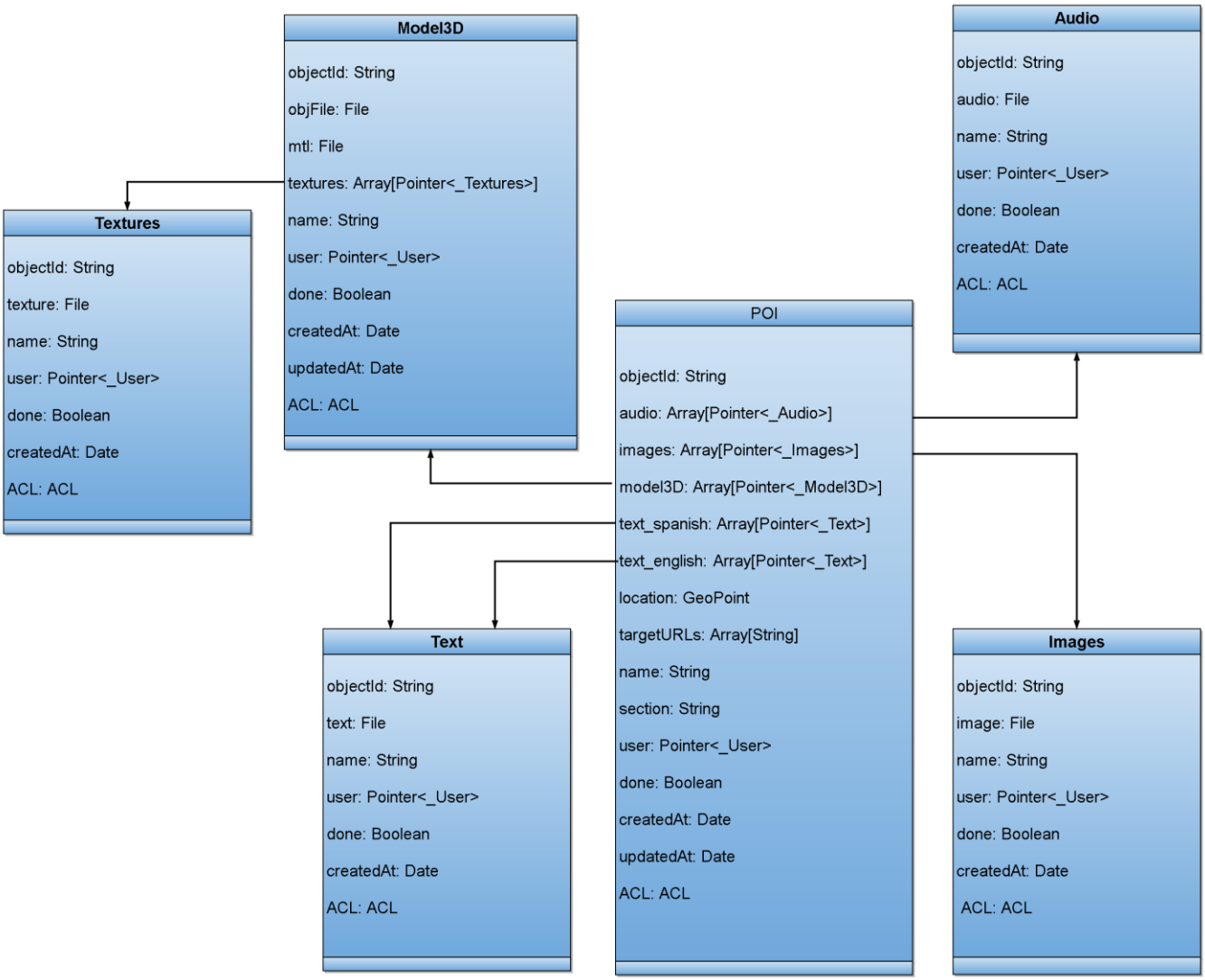

Figura 2.6 Diagrama de clases de la base de datos.

2.6 Diseño de la comunicación con la base de datos

Dado que la aplicación no contiene la información que debe mostrar al usuario y la debe obtener a través de su conexión a Internet cada vez que éste se desplace por su entorno, uno de los puntos más importantes en cuanto a eficiencia y rendimiento de la aplicación será la gestión de la descarga de datos.

2.6.1 Parse SDK

Una de las razones para seleccionar este servicio es la existencia de un SDK para Unity3D. El servicio de Parse se integra en el proyecto de Unity a través de un fichero .dll.

Solamente es necesario crear un objeto, asociarle el script y asignar tanto la *.NET key* como el *application ID* , que son claves relacionadas con tu servicio remoto, al script para tener la funcionalidad completa de Parse con tu aplicación. La necesidad de incluir las claves en el código fuente no introduce problemas de seguridad como veremos en el apartado correspondiente -2.10.2 -.

La API .NET nos permite realizar peticiones, gestionar usuarios, descargar ficheros y subirlos, etcétera. En definitiva, nos da un servicio completo de conexión con la base de datos alojada en los servidores de Parse, servicio que será utilizado en la clase *Download* para gestionar la descarga de ficheros.

2.6.2 Módulo de descarga

Dentro del proyecto en Unity, el módulo encargado de la descarga de los datos está implementado en la clase *Download*. Esta clase, junto con *ObjReader*, abstrae al resto de la aplicación de la gestión de la base de datos.

Esta clase posee las funciones necesarias para la obtención de los datos correspondientes según el estado de la aplicación. Además, permite dos modos de funcionamiento, dependiendo del modo de localización que tenga la aplicación.

El primero de ellos, el cual ha sido utilizado en este proyecto, se basa en la descarga de datos en función de la sección del museo en la que se encuentre el usuario. Es decir, cuando el sistema detecte que se encuentra en una determinada sección, procederá a la descarga de los *POIs* pertenecientes a esta. Esta solución es la más adecuada para localización por marcadores.

El segundo de los dos permite la descarga de los elementos más cercanos a la posición actual del usuario. Esta forma de funcionamiento es la óptima para una localización basada en coordenadas, como podría ser la localización en interiores.

Ambos modos poseen diferencias significativas. En el basado en la sección o marcadores, todos los datos de una zona determinada se descargan en cuanto se detecta la entrada en la sección; así pues, mientras no se cambie de zona no se requerirán nuevas descargas y el dispositivo no consumirá recursos continuamente en la conexión a la red. En el basado en geolocalización, por el contrario, los datos serán descargados a medida que el usuario vaya desplazándose y nuevos elementos entren en el área de interés, permitiendo una descarga de datos más gradual y progresiva.

En el primer caso, una de las desventajas es la espera inicial a la que el usuario es sometido, ya que dependiendo del ancho de banda disponible y de la cantidad de datos a descargar en esa sección, podría incrementarse considerablemente el tiempo de obtención de datos. Una solución simple para este problema sería la subsectorización del centro, dividiendo cada sección en varias zonas de descarga, provocando que los tiempos de espera se redujeran al fraccionar la descarga de elementos.

Para el segundo caso, el principal impedimento es la necesidad de poseer datos en tiempo real de la localización tanto del usuario como de los elementos a mostrar. Para conocer la posición del usuario sería necesario un sistema de geolocalización. Dado que los sistemas satelitales, como GPS, no funcionan con la suficiente precisión dentro de edificios, sería necesaria la implementación de un sistema de localización en interiores –basándose por ejemplo en la red Wi-Fi del centro –.

Para los dos modos de funcionamiento se ha gestionado el estado de la aplicación para no descargar elementos que ya han sido almacenados, mejorando la eficiencia y reduciendo un excesivo consumo de datos para el dispositivo. Es decir, si el usuario regresara a una sección o zona ya visitada, no se realizaría de nuevo la descarga de los mismos elementos, sino que se obtendrían de la propia aplicación que los tiene ya guardados.

El esquema de esta clase de puede ver en la Figura 2.7. Como se puede observar, contiene dos métodos públicos, *findNearestPOIs()* y *findSectionPOIs()*, que son los que implementan los dos modos de descarga antes comentados. Además, éstos hacen uso de los métodos privados para gestionar la descarga de cada uno de los tipos de datos. El lenguaje seleccionado por el usuario se tiene en cuenta a la hora de descargar los archivos de texto, obteniendo solamente los archivos del idioma correspondiente. El parámetro de salida de los métodos públicos es un objeto *POIList*, que contiene toda la información relativa a los elementos de interés.

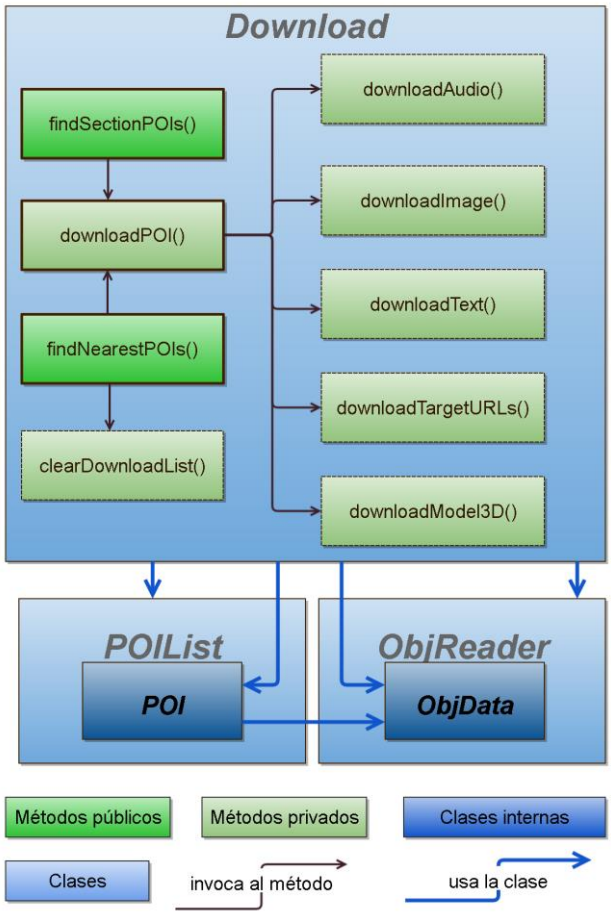

Figura 2.7 Esquema de métodos de la clase Download

2.6.3 Problemática de la descarga de modelos 3D

Una de las problemáticas a la hora de gestionar los datos se debe a la característica multimedia de estos. Además de gestionar la descarga de todos los elementos correspondientes a la posición del usuario, se debe realizar la gestión de los formatos de los ficheros para un correcto funcionamiento.

Así pues, los diferentes tipos de datos deberán gestionarse de manera independiente. Para algunos tipos, como por ejemplo los textos o las URL, no ha de realizarse ningún paso extra ya que son almacenados como *Strings*.

Para almacenar correctamente en la aplicación los datos de audio o las imágenes se hace uso de objetos propios de Unity, como son las clases *Texture* y *AudioClip*, que poseen funcionalidades para la utilización de materiales gráficos y sonoros respectivamente. Estas clases facilitan la manipulación de estos tipos de datos, tanto en su representación como en el acceso a sus características propias.

Finalmente, los modelos 3D son el tipo de datos cuya integración en tiempo de ejecución conlleva mayores dificultades.

Dado que existen diferentes formatos para los modelos gráficos en 3D, para este proyecto se ha considerado que la aplicación solo aceptara uno de ellos. Esta decisión se debe a que para cada uno de los formatos sería necesario un proceso totalmente diferente de conversión desde el fichero de texto en el que suelen estar definidos a la clase *GameObject* que es la que encapsula los elementos físicos en Unity, lo cual es inviable para la longitud de este trabajo.

Por ello, se ha elegido el formato .OBJ, desarrollado por Wavefront Technologies, libre de definición de geometrías y que es aceptado casi universalmente. En el Anexo C se explica con más detalle este formato.

Pese a ser uno de los tipos de representación de modelos 3D más extendido, no existe una transformación directa entre el fichero de texto en el que está definido el objeto y la clase *GameObject*, por lo que es necesario implementar dicha conversión. Debido a la

complejidad de este proceso se ha utilizado un *Asset* disponible en la *Asset Store* de la web de Unity.

Este *Asset* es *ObjReader*, y está desarrollado por *Starscene Software* (15). La versión utilizada en este proyecto es la 2.2.1, y la licencia permite su uso y modificación pero no su redistribución.

ObjReader es una herramienta para importar en tiempo real modelos 3D en formato .OBJ. Permite cargar los archivos desde el propio proyecto, desde una localización externa a él y desde la web. El último caso es el que interesa para esta aplicación, pero debido a la utilización de Parse como servicio de almacenamiento remoto, ha sido necesaria la modificación del código para un correcto funcionamiento.

Pese a que incorpora un fichero .dll para su integración en el proyecto de Unity, no ha sido posible utilizarlo debido a los cambios que como hemos dicho antes hemos realizado. Por ello, se ha utilizado el código fuente modificado.

Las modificaciones realizadas se deben a la forma de conectarse a la base de datos, debido a que la opción de descarga vía web implementada en el *Asset* hacía uso de la URL del fichero, obteniendo a partir de la referencia existente en el .OBJ la dirección del fichero .MTL, y de igual manera las direcciones de las texturas a partir del .MTL. El problema radica en que Parse no permite el acceso por URL de esta manera.

Para solventarlo, es necesario realizar una petición a la base de datos del objeto y los campos de este –en el caso de los modelos 3D los principales campos serían los ficheros .OBJ y .MTL y un vector de punteros a la Texturas asociadas–.

Por lo tanto se ha modificado todo lo referente a la descarga de los archivos, no así la conversión de los ficheros al objeto 3D.

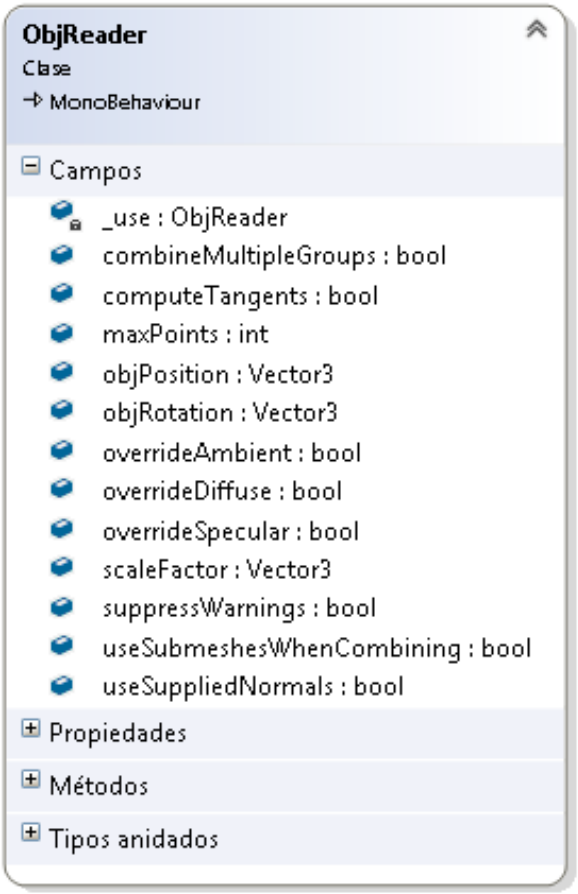

Figura 2.8 Campos de la clase ObjReader

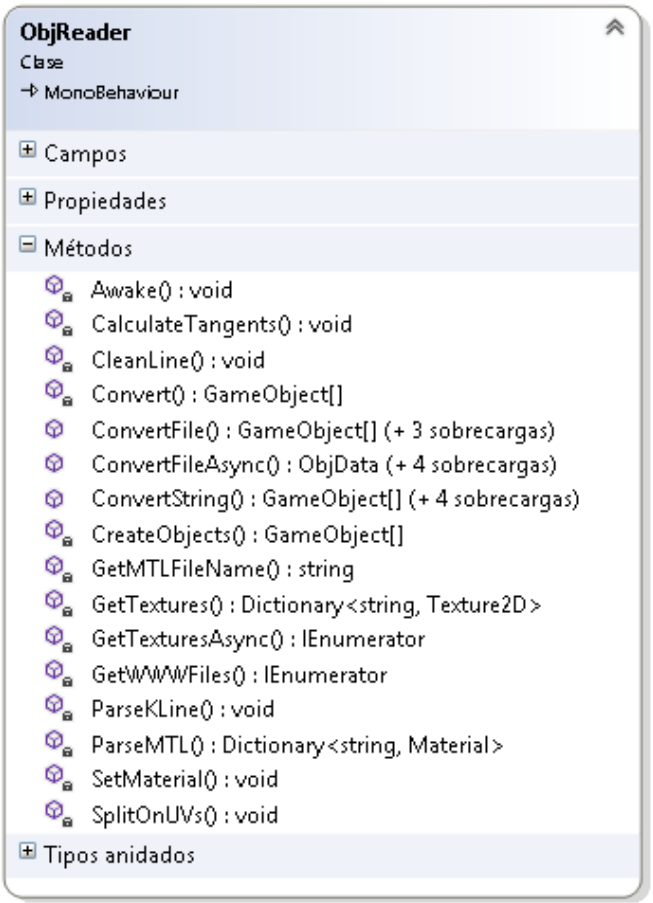

Figura 2.9 Métodos de la clase ObjReader

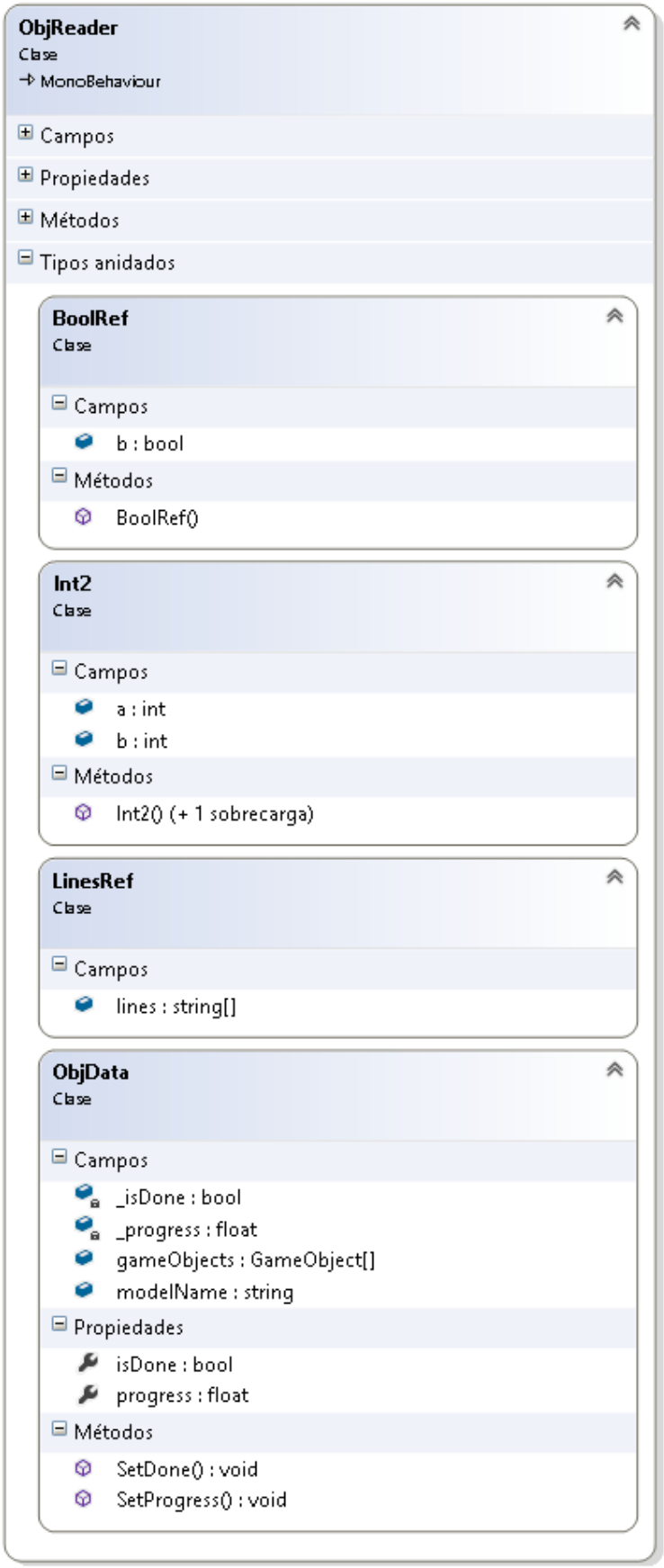

Figura 2.10 Clases anidadas de la clase ObjReader

2.7 Diseño del módulo de realidad aumentada

La aplicación debe ser capaz de superponer los datos virtuales correspondientes sobre la imagen real que ofrece la cámara del dispositivo. Además, debe ser capaz de detectar los marcadores de las diferentes secciones y actuar en consecuencia.

Para la implementación de estas características asociadas a la Realidad Aumentada se ha hecho uso del SDK para Unity Vuforia™, desarrollado por Qualcomm®. Esta herramienta nos permite tener acceso a una completa librería de realidad aumentada, aunque para los objetivos de este trabajo no es necesario utilizar la todas sus funcionalidades; su uso se centra en la detección de los marcadores, para saber que se ha entrado en una determinada sección, y en su seguimiento para mostrar los modelos 3D asociados. Además, es utilizada también para la integración de la imagen real y la virtual.

2.7.1 Vuforia

En la Figura 2.11 podemos observar el esquema general de la API de Vuforia. Permite realizar la detección de diferentes tipos de marcadores, entre ellos, imágenes, texto o formas tridimensionales. Se ha elegido que los marcadores sean imágenes dada su sencillez y su idoneidad en el entorno en el que se desarrollará el programa.

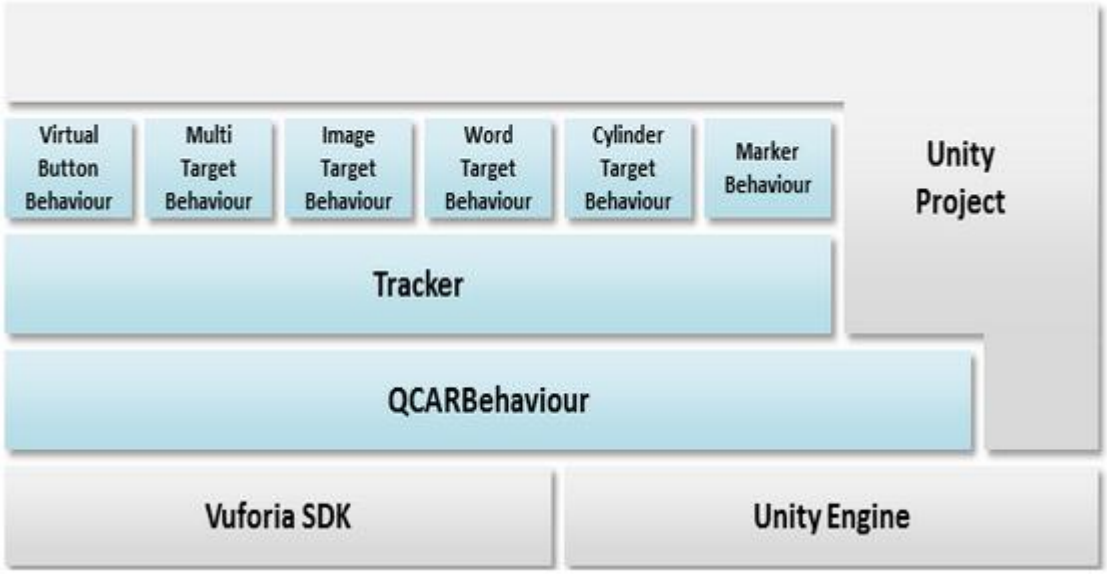

Figura 2.11 Diagrama de bloques de la API de Vuforia

Con más detalle, en la Figura 2.12 se describe el diagrama de funcionamiento del SDK. A partir de la imagen obtenida de la cámara se realiza la detección de objetivos –*trackers*- que estén integrados en la base de datos de objetivos, que puede estar incluida en la aplicación o hacer uso del servicio *cloud* que ofrece Vuforia para el almacenamiento y gestión de marcadores. Tras ello, se notifica a la aplicación los resultados obtenidos y así la lógica de ésta puede actuar en consonancia al estado correspondiente. Finalmente se integra la imagen inicial de la cámara con las representaciones gráficas que sea necesario añadir.

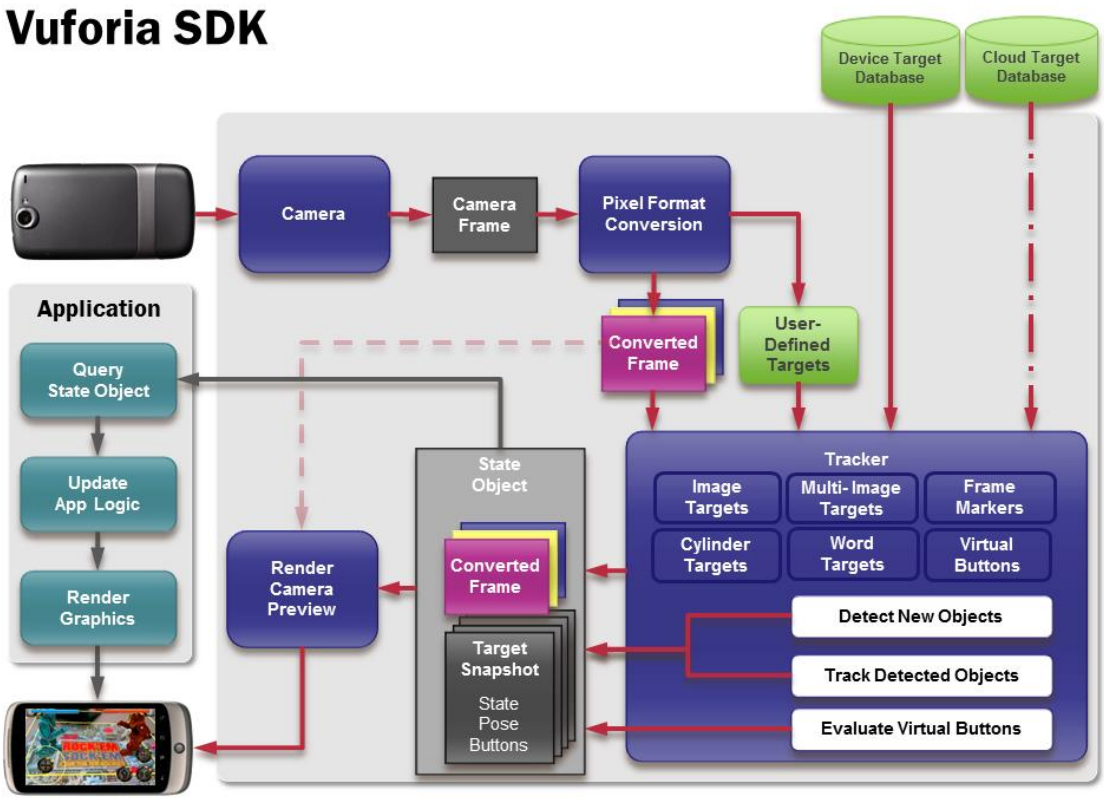

Figura 2.12 Diagrama de funcionamiento del SDK de Vuforia

2.7.2 Estructura de clases

Como ya se ha dicho anteriormente, la aplicación ha sido desarrollada en la plataforma Unity y mediante el lenguaje C#. Unity nos ofrece un entorno de desarrollo basado en un entorno virtual, en el que la entidad básica es el *GameObject,* y los scripts que dictan el comportamiento y la lógica del sistema deben ser asociados a una de estas entidades.

El SDK de Vuforia posee clases y *GameObjects* propios para la gestión de la realidad aumentada. Por ejemplo, el objeto *ARCamera* es utilizado para representar los objetos virtuales definidos en la escena de Unity 3D, mientras que *BackgroundCamera,* se encarga de situar la imagen de la cámara del dispositivo como fondo. Puesto que el renderizado en Unity se realiza por capas, cada una de las dos cámaras virtuales se encarga de mostrar diferentes capas, lo que posibilita la independencia de estas.

Mediante estos dos objetos gestionamos la integración de la imagen real y la información virtual, por lo que son la base de la realidad aumentada desarrollada en este programa.

El otro tipo de objeto proporcionado por Vuforia que vamos a necesitar es el *ImageTarget,* que contiene los componentes y scripts necesarios para la detección y el seguimiento del marcador. Será necesario incluir en la escena de Unity tantos *ImageTarget* como marcadores deseemos. En el caso que nos atañe, para el Museo de Informática Histórica de la Universidad de Zaragoza, serán necesarios cinco objetos, uno por cada sección existente en el museo.

Además de estos objetos, en la escena se ha de incluir también una *Directional Light,* para dotar de luz a los objetos que se van a representar, y un *GameObject* vacío, para incluir el SDK de Parse, mediante el script *ParseInitializeBehaviour* y las dos claves de aplicación necesarias *(Application ID* y *Dotnet Key*).

El resto de objetos necesarios se crean en tiempo de ejecución, como por ejemplo los planos donde representar las imágenes o el

botón de reproducir audio, y los *GameObject* que contendrán los modelos 3D.

No se requieren demasiados objetos porque una gran parte de la aplicación se realiza desde la interfaz gráfica de Unity, como veremos en el apartado 2.9.

Centrándonos en los *ImageTarget*, estos tienen asociados dos scripts propios, *MusearImageTargetBehaviour* y *MyTrackableEventHandler*, que gestionan su funcionamiento ante cambios de estado. Además, se les ha asociado un script que contiene la clase *ObjReader* para permitir la correcta obtención de los modelos 3D, como hemos visto en el apartado 2.6.3.

No es necesario asociar las clases *Language*, *Download* y *POIList* a ningún objeto ya que no son clases derivadas de *MonoBehaviour,* que es la clase básica de la que heredan el resto de scripts. Esta clase implementa una serie de métodos como son, entre muchos otros, *Awake()* y *Start(),* que son invocados cuando el objeto es creado, o *Update()*, *FixedUpdate()* y *OnGUI*(), que lo hacen en cada *frame*. De los dos scripts que se han nombrado anteriormente, el primero, *MyTrackableEventHandler,* gestiona la respuesta automática a los cambios de estado de los marcadores; por el contrario, *MusearImageTargetBehaviour*, contiene la interacción con el usuario dentro de cada una de las secciones:

- ***MyTrackableEventHandler***

  Se encarga de detectar cambios de estado en la detección del marcador asociado a este *ImageTarget*: Cuando el marcador cambia a "detectado" o "trackeado", el método *OnTrackableStateChanged()* invoca a *OnTrackingFound*(), mientras que cuando este es perdido es invocado *OnTrackingLost().* El segundo método anula la representación de los modelos 3D, mientras que *OnTrackingFound*() se encarga de lanzar la descarga de ficheros en una corrutina si es la primera vez que se detecta este marcador, y de deshabilitar los elementos de otros *targets* para no superponer información de diferentes secciones y mostrar solo el menú de navegación de la actual.

- ***MusearImageTargetBehaviour***

  Este script es el más importante para el funcionamiento de la aplicación ya que contiene toda la lógica y la interfaz gráfica de ésta. *MusearImageTargetBehaviour* es una clase derivada de *ImageTargetAbstractBehaviour*, que es la interfaz definida por Vuforia. Define todo el comportamiento de la aplicación una vez ha sido detectado el marcador asociado.

En la Figura 2.13 podemos ver un esquema parcial de funcionamiento automático de la aplicación sin intervención del usuario. *OnTrackingFound()* invoca al método para descargar los datos y según la sección en la que nos encontremos habilita o deshabilita las instancias de *MusearImageTargetBehaviour*(). A su vez, este último script ejecuta el comportamiento correspondiente de la aplicación según el estado en el que se encuentre. En el apartado 2.9 se explica en detalle el diagrama de estados y las funciones de la interfaz gráfica, que en este esquema se han obviado.

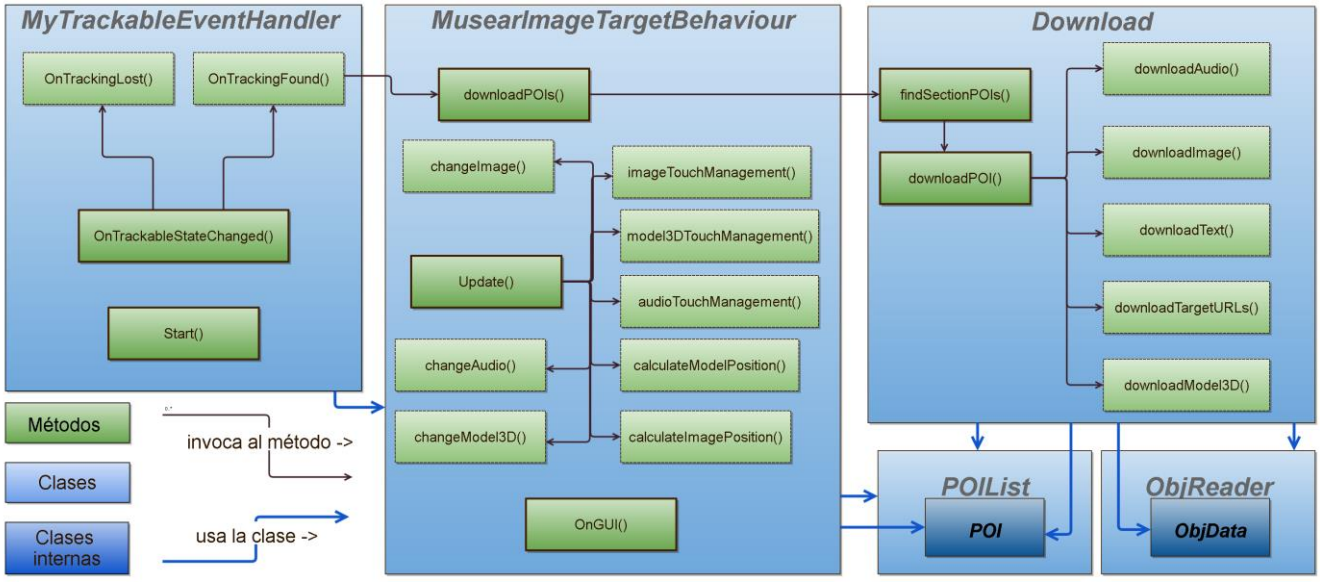

Figura 2.13 Diagrama de relación entre las diferentes clases y métodos

Además de las clases hasta ahora expuestas, se ha definido la clase *Language,* que es una clase contenedor de todo mensaje textual que deba aparecer en la aplicación, implementada para facilitar el cambio de lenguaje de la aplicación; además contiene los métodos necesarios para la modificación del idioma.

2.8 Diseño de la navegación

Una de las características más importantes de cara al usuario, tras las funcionalidades ofrecidas, es la navegación y la interfaz gráfica, que son los elementos del sistema con los que el usuario interactúa. Se ha querido dotar a esta parte del trabajo de sencillez dado que el objetivo, y por lo tanto lo que debe centrar la atención, es el contenido que muestra. Por esta razón la interfaz gráfica se estructurado de manera que ofrezca las todas las funcionalidades posibles ocupando el menor espacio para resaltar la información ofrecida.

Todas las cadenas de texto que se utilizan para la navegación están almacenadas en la clase *Language* como variables estáticas. Esta clase permite el cambio de idioma de toda la aplicación sin necesidad de modificar el resto de las clases. Añadir un idioma nuevo sería muy sencillo, ya que solo sería necesario añadir la traducción de los elementos ya definidos y crear un botón en el menú de selección de idioma para habilitarlo. En esta memoria todas las opciones y botones se definen por su nombre en español.

A excepción de la primera escena, todos los elementos gráficos están definidos en la clase *MusearImageTargetBehaviour.* Esta clase contiene el diagrama de estados gestiona la interacción con el usuario.

A continuación se va a describir las dos escenas que integran la aplicación desde un punto de vista dinámico y gráfico, mientras que en el Anexo D se encuentran las capturas de pantalla de la aplicación para ofrecer un ejemplo gráfico sobre los esquemas aquí presentados. Desde este apartado se referencia a las imágenes alojadas en dicho anexo, el cual se encuentra al final de este documento.

2.8.1 Primera escena: *Intro*

Esta escena, como ya se ha comentado antes, es muy simple, ya que sirve de introducción y presentación de la aplicación.

Mientras se inicia la aplicación y se carga la escena se muestra una imagen con el nombre de la aplicación y el logotipo de la Universidad de Zaragoza (Figura 2.14). La escena consta de una imagen de fondo que contiene el nombre de la aplicación, MuseAR, y el logo del grupo IAAA de la Universidad de Zaragoza (Figura 2.15). Sobre este fondo se presenta superpuesto el menú de inicio de la aplicación. En él existen cuatro opciones:

- "**Comenzar la visita al museo**"

  Esta opción hace que se inicie la segunda escena, donde el usuario podrá disfrutar de las funcionalidades de esta aplicación.

- **"Cambiar idioma"**

  Permite elegir entre español e inglés como idioma de la aplicación; hace uso de la clase *Language* para asignar los *strings* correspondientes al idioma seleccionado. Por defecto, la aplicación se muestra en español (Figura D.4).

- "**IAAA**"

  Pasa a mostrar un breve texto donde se describe la labor del grupo de Sistemas de Información Avanzados y su contacto (Figura D.3).

- "**Acerca de**"

  Muestra la información básica de la aplicación y el contacto con el autor (Figura D.2).

Figura 2.14 Carga de la aplicación

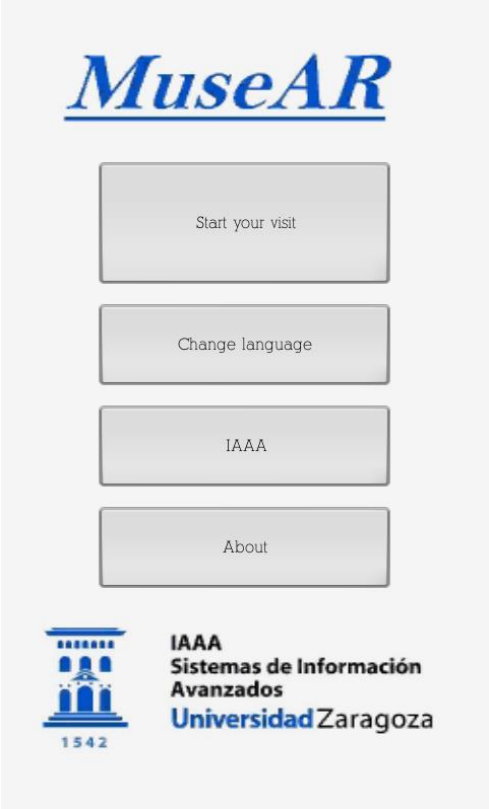

Figura 2.15 Pantalla inicial –inglés–.

2.8.2 Segunda escena: *MuseAR*

La segunda escena es considerablemente más compleja, ya que consta de un conjunto de menús y de apartados que permiten acceder a toda la información relativa a la sección en la que nos encontramos.

Tanto el tamaño de los botones como el tamaño de fuente son relativos al tamaño en píxeles de la pantalla del dispositivo. De igual manera, la sensibilidad táctil también depende de la densidad de píxeles por pulgada, todo ello para ofrecer una experiencia de usuario satisfactoria sean cuales sean las especificaciones del dispositivo.

Esta escena se inicia con un mensaje de bienvenida (Figura D.5) en el que se invita a localizar uno de los marcadores del museo para acceder a la información adicional. En segundo plano, la aplicación realiza la búsqueda y la detección de los marcadores. Cuando se localiza uno de ellos, se pasa a descargar los archivos de la base de datos (Figura D.6) y, cuando se han obtenido, se entra en el menú de navegación de la sección.

Durante todo momento la búsqueda de marcadores continúa activa mientras se maneja la aplicación. De esta manera, cuando un nuevo marcador es detectado –por ejemplo, si el usuario se ha desplazado a otra sección–, se inicia el proceso de validación de estado. Esto es, si es la primera vez que se localiza el marcador se repiten los pasos descritos para el comienzo de la aplicación y se descargan los datos correspondientes; si no es la primera vez, simplemente se cambia el menú de la anterior sección por el de la nueva automáticamente.

En la Figura 2.16 se puede ver el diagrama de estados de la aplicación. Este diagrama se repite para todas las secciones, que cuando no están activas esperan hasta que su marcador sea detectado y pasen a primer plano.

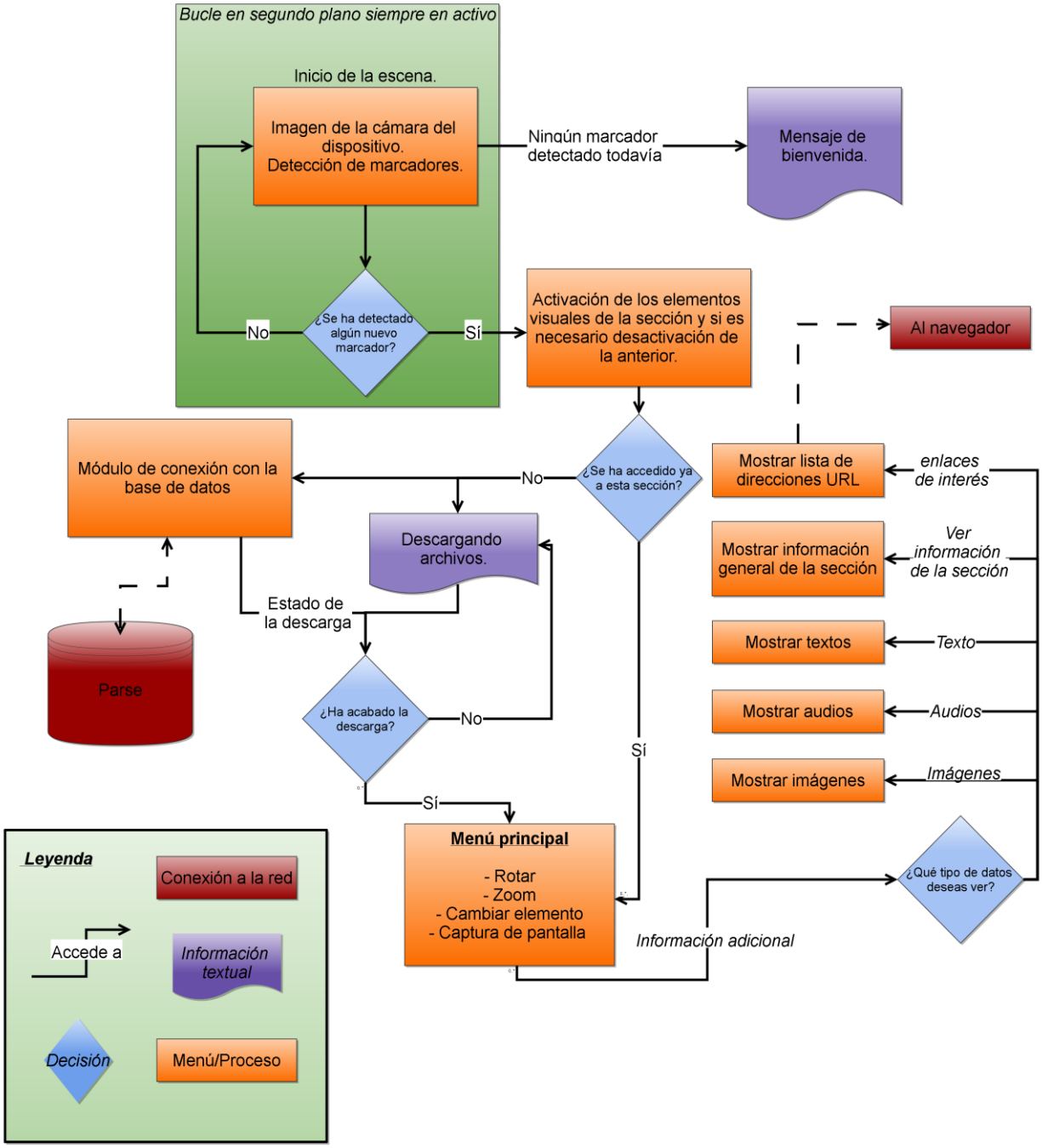

Figura 2.16 Diagrama de navegación de la escena MuseAR

Una vez se ha detectado un marcador por primera vez y se han descargado todos los elementos, la aplicación muestra el menú principal dela sección. Este menú muestra superpuesto al marcador el modelo 3D relativo a uno de los elementos de la sección, como se puede ver en la Figura 2.17. En la parte superior de la pantalla puede verse el nombre de dicho objeto, y en caso de no existir un modelo 3D asociado a este elemento se muestra un mensaje notificándolo(Figura D.10). Además,

consta de una serie de elementos gráficos para gestionar diferentes funcionalidades. Dichas funcionalidades son:

- Cambio de elemento: Se puede elegir el elemento a representar de dos formas. La primera, mediante los botones situados en la esquina inferior izquierda de la pantalla, y, la segunda, mediante el movimiento táctil horizontal. Dependiendo del sentido del movimiento se pasará al siguiente elemento o al anterior, siendo la opción más intuitiva. Esta opción permite visualizar secuencialmente todos los elementos de la sección.

- Rotar: (Figura D.9) La activación del botón que se encuentra en la esquina superior izquierda de la pantalla permite modificar la respuesta de la aplicación a los movimientos táctiles. Esta opción posibilita la rotación del modelo 3D del elemento para poder observarlo desde cualquier ángulo. Para desactivar esta opción y volver a respuesta por defecto –cambio de elemento– basta con volver a pulsar el anterior botón.

- Rotación inicial: Esta opción, relacionada con la anterior, permite situar al objeto en su rotación original, esto es, en plano respecto al marcador. Se activa mediante el botón situado en la esquina superior derecha.

- Capturar: Permite realizar una captura de pantalla de lo que se está viendo en ese instante en la aplicación. En botón se sitúa en la parte central superior de la pantalla.

- Zoom: (Figura D.8) Esta funcionalidad permite aumentar o disminuir el tamaño del objeto tridimensional que se está observando mediante entrada táctil. El uso simultáneo de dos dedos permite realizar el *zoom* mediante la variación de la distancia entre los dedos.

- Información adicional: (Figura D.11) Por último, el botón situado en la esquina inferior derecha de la pantalla abre un menú donde se muestra una lista de los tipos de datos multimedia disponibles para ese elemento. Ya que no todos los elementos deben tener información multimedia de cualquier género, sólo se muestran las opciones que contienen información. Esta criba se hace mediante la propiedad *availableFieldsList* que poseen todos los objetos de la clase *POI.* Dependiendo del tipo de información seleccionado se accederá a uno

de los cinco menús que se describen a continuación. Se puede regresar al principal mediante el botón "*Atrás*".

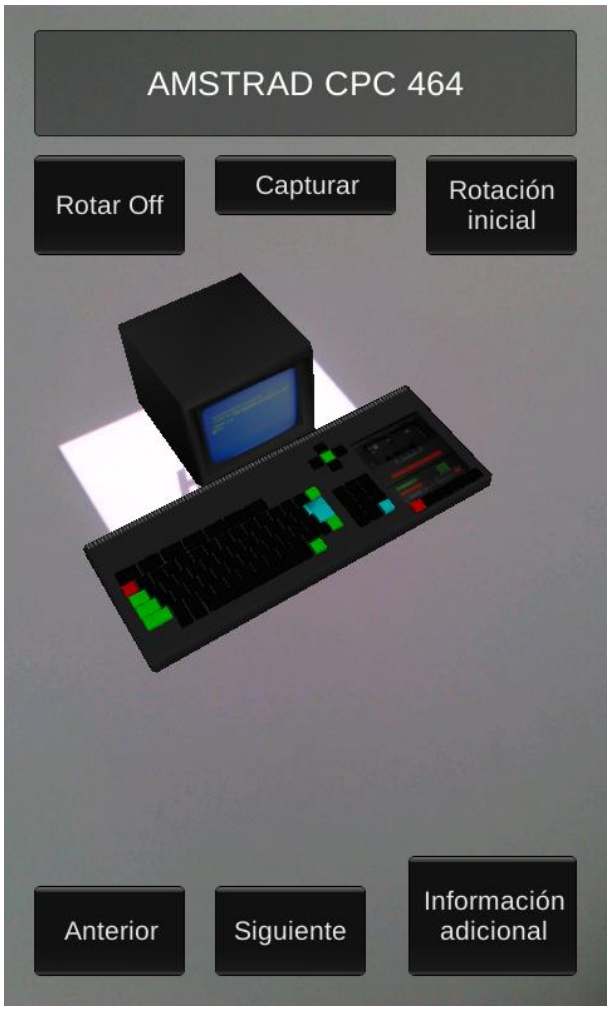

Figura 2.17 Imagen representativa de la escena MuseAR con marcador localizado

El menú que se abre al seleccionar esta última opción lleva a un menú diferente según la elección que se tome. Esos menús son:

- "Información general de la sección": (Figura D.13) Muestra información en formato textual de la sección en la que nos encontramos. Breve resumen de lo que podemos encontrar en ella.

- “Información adicional”: (Figura D.14) Muestra los archivos de texto relacionados con el elemento seleccionado. Permite cambiar entre los diferentes textos mediante dos botones, “*Siguiente*” y “*Anterior*”, y regresar al menú anterior pulsando “*Atrás*”.

- “Enlaces de interés”: (Figura D.17) Muestra una lista con los enlaces web de interés relacionados con el *POI*. Si se selecciona uno, la aplicación abre en el navegador web la página correspondiente. Pulsando el botón “*Atrás*” se regresa al menú anterior.

- “Audios”: (Figura D.12) Permite reproducir los archivos de audio asociados. Mediante la pulsación del botón que aparece en el centro de la pantalla se realiza la reproducción o pausa del archivo. En caso de existir varios elementos, se puede realizar el cambio de estos mediante los botones “*Siguiente*” y “*Anterior*” o también mediante la interfaz táctil. Si se realiza un desplazamiento horizontal con un único dedo, se pasará a mostrar el siguiente audio en el sentido del movimiento.

- “Imágenes”: (Figuras D.15 y D.16) Muestra las imágenes relacionadas con el *POI*. En la pantalla aparece una única imagen y en caso de que existan varias asociadas se puede cambiar la imagen visible de la misma manera que en el caso anterior: Existen dos botones, “*Siguiente*” y “*Anterior*”, que permiten dicho cambio, y, además, se puede realizar mediante la interfaz táctil. También se ha implementado la opción de *zoom*, que se realiza pulsando la pantalla con dos dedos y variando la distancia entre ellos. En este menú el cambio de elemento mediante gestión táctil reviste mayor complejidad, ya que en caso de que se haya ampliado la imagen y no quepa completa en la pantalla del dispositivo, el desplazamiento táctil de un solo dedo permite el movimiento sobre la propia imagen, hasta que se alcanzan los límites de ésta y vuelve a tener la funcionalidad de cambio de elemento.

  En caso de que, en cualquier menú, una lista o un texto no quepa en la pantalla, se añade un barra vertical para desplazare por la información.

2.9 Diseño de la aplicación de gestión de la base de datos

Para el manejo y gestión de los datos almacenados es necesario desarrollar una aplicación que permita la creación, actualización y eliminación de los elementos guardados en la base de datos. Esto tiene como objetivo permitir al gestor de *MuseAR* la modificación y manipulación sencilla de los archivos relativos a su museo que desee incluir.

Se ha utilizado el servicio en la nube de Parse para poder acceder a esta aplicación de gestión desde una dirección web. Para su creación se ha utilizado HTML y JavaScript, configurando el estilo de la página con CSS.

Para su implementación se ha hecho uso del SDK para JavaScript proporcionado por Parse, y se ha tomado como punto de partida la el prototipo de aplicación que ofrece Parse en su sección de documentación. El SDK de JavaScript está basado en el *framework* Backbone.js, que permite construir aplicaciones usando JavaScript siguiendo el patrón MVC (modelo-vista-controlador). Por ello, depende de las librerías Underscore.js y jQuery.js.

Las versiones de cada uno de los recursos utilizados son:

- ***jQuery***: jquery-2.0.3.min.js Se obtiene cuando se accede a la aplicación a través de http://code.jquery.com/jquery-2.0.3.min.js

- ***Underscore***: underscore-1.1.6.js. Está almacenado en el servidor.

- ***Parse SDK***: parse-1.2.13.min.js. Se obtiene cuando se accede a la aplicación a través de http://www.parsecdn.com/js/parse-1.2.13.min.js

2.9.1 Navegación

Para estructurar la aplicación se ha tomado como referencia la configuración de la base de datos. Por ello, se han definido modelos para cada una de las clases definidas en Parse: Audio, Imagen, Texto, Textura, Modelo 3D y POI. La unidad básica tanto de estructuración como de visualización son los objetos de tipo POI, que contienen al resto de objetos como ya se explicó en la descripción de la base de

datos. En el Anexo D, apartado 2, se pueden ver diferentes capturas de pantalla de la aplicación de gestión.

Cuando se accede a la dirección web proporcionada por Parse, http://musear.parseapp.com/, se requiere al usuario que introduzca su nombre y su contraseña para identificarse. Una vez se han introducido los datos correctos, se accede a una página donde se muestra una lista con todos los objetos de tipo POI que se encuentran en la base de datos, especificado su nombre y su sección. Si se desea se puede añadir un elemento, eliminar los que se deseen o ampliar un POI en concreto para ver sus propiedades.

Al ampliar la visión de un objeto POI, se muestran una serie de botones para elegir la propiedad que se quiere visualizar. Estas propiedades son:

- Audio
- Imágenes
- Textos en español
- Textos en inglés
- Modelos 3D
- Localización
- Enlaces
- Cambiar nombre
- Cambar sección

Al seleccionar una de las cinco primeras opciones se muestra una lista con todos los objetos del tipo seleccionado asociados a ese POI. Estas listas también pueden modificarse de la misma manera, añadiendo o eliminando elementos, y también modificando cualquiera de sus campos, como el nombre y el fichero que contienen. La lista de Modelos 3D es más compleja que las anteriores, ya que el modelo de datos contiene más campos. Por ello, cada objeto de tipo Modelo 3D permite la modificación de dos ficheros, uno .obj y otro .mtl, además de permitir desplegar una lista con las texturas asociadas a éste.

Cuando se desea modificar cualquier parámetro la aplicación debe conectarse con el servicio Parse para modificar los datos de la base de datos y así mantener toda la información en estado persistente. Por ello, se notifica al usuario mediante un mensaje que la acción que ha realizado ha finalizado con éxito –o que no ha sido posible realizarla, en

caso negativo- para que continúe con trabajo. Debido al manejo de datos multimedia, que pueden alcanzar un tamaño considerable en lo que a tiempos de subida al servidor se refiere, como audio o modelos 3D, la actualización de los ficheros de este tipo puede requerir de un tiempo de espera.

Además se permite la opción de visualizar los ficheros que están almacenados, para comprobar si los datos ya registrados en el servidor son los deseados. Es posible abrir los archivos de texto, audio e imágenes que se muestran mediante el botón indicado para ello.

En la base de datos y de cara al funcionamiento de la aplicación, existe un tipo especial de POI que es gestionado de manera diferente. Estos elementos se diferencian del resto por su nombre, ya que el nombre de estos viene dado por la expresión "*sectionInfo_*" seguida del nombre de una las secciones del museo. Este tipo de elementos se ha creado para dotar a cada sección de información general y no aparecen en la lista general que se muestra en la aplicación; dicha información global se podrá acceder desde el menú de información adicional presente en cada uno de los elementos de la aplicación. No es obligatoria la existencia de este tipo de POI en cada una de las secciones, pero sí recomendable si se desea ofrecer una visión general de los elementos que podemos encontrar.

2.10 Consistencia de datos y seguridad

2.10.1 Consistencia de datos

Una de las cuestiones que se deben tratar con especial cuidado es la consistencia entre las diferentes plataformas que integran este trabajo. Es necesario que los datos almacenados en la base de datos tengan un modelo idéntico en la aplicación para dispositivos móviles, para evitar errores de lectura y manipulación de datos en la descarga de información. A su vez, también es necesario cuidar que la aplicación de gestión de la base de datos actualice y cree los elementos de la manera correcta, evitando datos inconsistentes con el funcionamiento y el estado de la aplicación.

Por ello, se han realizado diferentes mecanismos para procurar una correcta gestión de los datos, evitando errores relacionados con el formato o la estructura de las entidades utilizadas.

En la aplicación de gestión de Parse no se permite la existencia de elementos con el mismo nombre entre los que engloba un determinado POI si son objetos de la misma clase. Pese a que Parse asocia un ID único a cada uno de los objetos definidos en su seno, la configuración que se ha realizado de la aplicación de gestión hace que sea conveniente para evitar posibles incongruencias.

También se ha implementado un control de formatos para todos los ficheros que son subidos al servicio remoto. Estos se han seleccionado según un criterio práctico basado en los formatos que soporta la aplicación desarrollada en Unity. Así, los formatos permitidos para cada tipo de archivo son:

- Texto: .txt
- Imágenes: .png, .jpg, o .jpeg
- Audio: .wav
- Texturas: .png, .jpg, .jpeg o .tga
- Modelos 3D:
    - Fichero de geometría: .obj
    - Fichero de materiales: .mtl

- Enlaces: Pese a ser variables de texto –*strings*– sin formato, se comprueba que comiencen por “http://” o “https://” ya que Unity no redirecciona al navegador si el enlace empieza con otros caracteres.

En caso de haber sido seleccionado un fichero que no sea de ninguno de los tipos válidos, se notifica al usuario qué archivos son los permitidos y se cancela la subida del elemento.

El control de formato posee redundancia, ya que en la clase *Download* del proyecto desarrollado en Unity se vuelve a controlar que los elementos descargados desde la base de datos se correspondan con los formatos válidos. Esto se debe a que la aplicación es independiente de la gestión del servicio remoto, por lo que si se utilizara otro método para actualizar los datos guardados en Parse podrían producirse errores de formato.

2.10.2 Seguridad

Dado que es una aplicación de carácter público y abierta a todo el mundo, se ha de tener en cuenta el uso que se pueda hacer de ella por personas no autorizadas.

Parse posee funcionalidades interesantes a este respecto, y su adecuada configuración es una necesidad para la correcta implementación de cualquier proyecto.

Una de las características de seguridad que han sido descritas en esta memoria es la necesidad de incluir claves del servicio remoto de Parse dentro del código desarrollado. Esto es necesario tanto en la aplicación para usuario, desarrollada en Unity, en la que se ha tenido que incluir la *application ID* y la *dotnet key*, como en la aplicación de gestión, desarrollada en JavaScript, en la se incluyen la *application ID* y la *javascript key*. Pero el hecho de que estas claves sean públicas –en el caso de javascript se pueden obtener leyendo el código fuente de la página – no implica problemas de seguridad debido a que no son claves que sirvan de protección. Es la *Master Key* la clave que permite realizar acciones por encima de los mecanismos de control que se hayan establecido.

Toda comunicación con Parse se realiza mediante https y ssl, y se rechazan todas las conexiones que no son https. Para evitar el uso inadecuado –consciente o insconsciente- de ninguna aplicación cliente en este servicio, se han restringido la mayoría de las funcionalidades ofrecidas por Parse para usuarios anónimos.

Sólo existe un usuario registrado en el servicio de parse, *admin*, e iniciar sesión con dicha cuenta es la única manera posible de modificar la configuración de la base datos. Solo el usuario *admin* es capaz de crear, actualizar, o eliminar objetos; tampoco es posible crear o eliminar clases en la base de datos.

La única acción que pueden realizar los demás usuarios es la de leer los archivos presentes en la base de datos. La aplicación está destinada al ámbito cultural, el cual es uno de los pocos sectores donde la privacidad de la información almacenada no es importante. Más aún, conviene que sea accesible a todo el mundo y que tenga la mayor difusión posible.

Aun así, se ha visto necesaria la limitación de derechos disponibles al público para evitar problemas derivados de una mala configuración de seguridad, no para proteger la información almacenada, sino para proteger el funcionamiento y la integridad del servicio.

2.11 Pruebas realizadas

Durante el desarrollo de la aplicación se han utilizado diferentes dispositivos para la comprobación y corrección de la aplicación.

Desde el inicio del proyecto se ha usado el editor de Unity, que posee un simulador del funcionamiento del programa, para la comprobación de todas las funcionalidades que no requerían de un dispositivo móvil, dado que el flujo de trabajo es mucho más dinámico. Para las que sí lo necesitaban, ha sido utilizado un Huawei Y300.

Al finalizar el periodo de desarrollo, se han realizado pruebas en diferentes dispositivos para comprobar el correcto funcionamiento tanto para diferentes versiones de Android como para diferentes especificaciones, siendo las más representativas el procesador, la resolución y densidad de pantalla. La mayoría de las pruebas se han realizado con terminales de gama media-baja.

En todos los dispositivos en los que ha sido probada, y tras los ajustes necesarios durante el desarrollo, ha funcionado de manera correcta. Los dispositivos utilizados han sido:

- Huawei Ascend Y300 (16):
    - Procesador: Qualcomm MSM8225. 1,0 GHz doble núcleo.
    - RAM: 512 MB
    - Resolución: 800x480, 4'' (~233 ppi)
    - Cámara frontal: 5 Megapíxeles
    - Versión Android: Android 4.1.2 Jelly Bean

- Sony Xperia J (17):
    - Procesador: Qualcomm MSM7227A 1,0 GHz
    - RAM: 512 MB
    - Resolución: 854 x 480 píxeles, 4'' (~233 ppi)
    - Cámara frontal: 5 Megapíxeles

- LG 3 II E430 (18):
    - Procesador: Qualcomm MSM7225A 1,0 GHz
    - RAM: 512 MB
    - Resolución: 240 x 320 píxeles, 3,2'' (~ 125 ppi)
    - Cámara frontal: 3.2 Megapíxeles

- Versión Android: Android 4.1.2 Jelly Bean

- ZTE GRAND X(M) (19):
  - Procesador: dual-core 1GHz
  - RAM: 1 GB
  - Resolución: 540×960 píxeles, 4,3'' (~256 ppi)
  - Cámara frontal: 5 Megapíxeles
  - Versión Android: Android 4.0 Ice Cream Sandwich

3. Conclusiones

Partiendo del objetivo principal que era la creación de una plataforma para museos basada en realidad aumentada, se ha buscado maximizar tanto la usabilidad como la flexibilidad o la eficiencia de recursos durante el desarrollo. En esta línea se han tomado diversas decisiones; entre ellas, la elección de Unity como plataforma de desarrollo para permitir la exportación multiplataforma, la elección de una base de datos remota mediante un servicio de terceros, y la creación de una aplicación de gestión de la base de datos para facilitar dicha gestión al responsable de la aplicación en el Museo.

Al finalizar el trabajo la aplicación es plenamente funcional y está alojada en la tienda de aplicaciones del sistema Android – *Play Store* –. Pese a que se ha implementado para el Museo de Informática Histórica de Zaragoza (MIH), es posible su utilización en cualquier museo, sin más que modificar la información almacenada en la base de datos y los marcadores a utilizar para su utilización. De igual manera, aunque no se haya comprobado ni publicado, también es posible su utilización en otras plataformas de dispositivos móviles.

Puesto la descarga de los elementos a mostrar se realiza durante la ejecución de la aplicación, es necesario seleccionar dichos elementos para optimizar el tiempo de descarga. Por ello, ha de tenerse en cuenta el tamaño de los ficheros a incluir para que no se produzca un tiempo excesivo de espera. Esta razón –evitar que se produzcan tiempos de espera excesivos–, es, junto a la limitación de Unity para el manejo de videos, la razón por la que se ha elegido ofrecer la información audiovisual a través de los enlaces a páginas web. La inclusión de videos en la base de datos ralentizaría en gran manera el proceso de descarga de archivos, dado el gran peso de este tipo de ficheros en relación a los otros tipos integrados.

3.1 Líneas futuras

Dado que este trabajo está limitado a la longitud propia de un TFG, las líneas de trabajo futuras que se pueden desarrollar son amplias. Una de las características de este tipo de plataformas para museos es que normalmente son de uso único por el museo que las utiliza. Es decir, una misma aplicación no suele utilizarse en distintos museos, ya que son soluciones propias que realizan estos de forma individual y ajustándose a sus necesidades. Por ello,

en caso de que la aplicación se fuera utilizar en algún otro museo sería necesaria la colaboración del personal encargado para realizar las modificaciones o mejoras que requiriesen.

Como mejoras se podrían destacar:

- ***Gestión e integración de formatos:*** En base a lo anteriormente comentado sobre la problemática del tiempo de espera y su mejora, un paso importante sería la disminución del tamaño de los archivos mediante la conversión de formato. En este punto en tipo de datos más importante y el que ocupa un mayor espacio es el audio. Esto se debe a que para que se pueda reproducir correctamente el fichero debe estar en formato WAV, lo cual aumenta considerablemente su tamaño en comparación con otros formatos de audio. Por ello, una mejora sería la inclusión de un módulo de descompresión mp3. Unity no permite la reproducción de archivos mp3 descargados en tiempo de ejecución, por lo que sería necesario realizar la conversión del fichero descargado a wav y almacenarlo en la aplicación para su reproducción. El otro formato que puede conllevar prolongaciones en la espera es el formato de modelos 3D OBJ. En este caso, para realizar la mejora sería necesario un estudio de los diferentes formatos gráficos, para integrar en la aplicación el más eficiente en cuanto a tamaño. Además, otra opción que no requiere de modificaciones en la aplicación es la creación autónoma de modelos 3D, lo que permitiría la regulación de tamaños y calidades de dichos modelos en base a nuestras necesidades.

- ***Mejora del diseño:*** Se ha realizado el diseño de forma que fuera intuitivo e interactivo, pero como en todo desarrollo de un producto debe realizarse una comprobación de que el diseño es óptimo y gusta. Por ello, podría realizarse modificaciones en base a la opinión de los usuarios.

- ***Integración de video:*** En vista al continuo aumento del ancho de banda que se da en nuestra sociedad, el problema generado por un elevado tiempo de espera puede verse reducido en gran medida en poco tiempo. Por ello, sería interesante, para redes de alta velocidad, la inclusión de videos en la aplicación y en la base de datos.

- ***Integración de animaciones:*** Para dotar a la realidad aumentada de una mayor profundidad, sería interesante elaborar animaciones para los modelos 3D generados. Dependiendo del museo para el que se desarrolle esta característica será más importante y llamativa. Por ejemplo, para el MIH no aporta un gran cambio debido al tipo de elementos que almacena –dispositivos y máquinas–, pero para un museo de historia natural podría resultar muy interesante.

- ***Testeo y publicación en otras plataformas:*** Dada la flexibilidad de Unity en este sentid no sería muy complejo el proceso de exportación multiplataforma.

- ***Integración en redes sociales:*** Siguiendo la tendencia actual, sería interesante la inclusión de la aplicación en redes sociales de manera que los usuarios pudieran compartir la información a sus contactos, favoreciendo la difusión de esta información.

3.2 Opinión personal

El resultado final del trabajo es muy satisfactorio a nivel personal. Al partir de un desconocimiento casi total de las herramientas y recursos que se han utilizado, la elaboración de este proyecto ha sido muy enriquecedora, ya que el gran proceso de aprendizaje llevado a cabo ha sido muy gratificante. He adquirido conocimientos tanto en diferentes lenguajes de programación – C#, Javascript, html– como en diferentes plataformas y recursos –Unity, Parse, Vuforia, Backbone, jquery…–, además de aprender a gestionar el trabajo autónomo.

El interés que me suscitaba la unión de un concepto tecnológico actual como realidad aumentada con el mundo cultural ha sido un gran impulso para la consecución de este trabajo.

El uso de herramientas de este tipo en centros culturales genera un acceso a la información más dinámico, flexible e interactivo, lo que permite llegar a sectores de la población menos interesados en el saber histórico, enlazando cultura y entretenimiento. A su vez, también crea una nueva fuente de conocimiento, facilitando la integración de datos y ofreciendo una cantidad de información secundaria que puede resultar interesante pero inviable de acumular en el espacio físico del centro.

4. Bibliografía y Referencias

1. **GALINDO, Dolores.** Realidad aumentada en museos. *socialmuseum.net.* [En línea] 16 de Julio de 2012. [Citado el: 2 de Septiembre de 2014.] http://socialmuseum.net/2012/07/16/realidad-aumentada-en-museos/.

2. **Ministerio de Educación, Cultura y Deporte.** Visitas virtuales de los Museos. *mecd.gob.es.* [En línea] http://www.mecd.gob.es/cultura-mecd/areas-cultura/museos/visitas-virtuales.html.

3. **GROENHART, Maurice.** The Berlin Wall is back. *layar.com.* [En línea] 16 de Abril de 2010. [Citado el: 25 de Marzo de 2014.] https://www.layar.com/news/blog/2010/04/16/the-berlin-wall-is-back/.

4. **Thumbspark.** Museum of London: Streetmuseum. *itunes.apple.com.* [En línea] 27 de 04 de 2014. https://itunes.apple.com/es/app/museum-london-streetmuseum/id369684330?mt=8.

5. Vuforia. [En línea] https://www.vuforia.com/.

6. NyARToolKit. [En línea] http://nyatla.jp/nyartoolkit/wp/.

7. droidar.blogspot.com.es. [En línea] http://droidar.blogspot.com.es/ .

8. droidar. *https://github.com/bitstars.* [En línea] https://github.com/bitstars/droidar.

9. mixare.org. [En línea] http://www.mixare.org/.

10. wikitude.com. [En línea] http://www.wikitude.com/.

11. lookar.net. [En línea] http://www.lookar.net/.

12. layar.com. [En línea] https://www.layar.com/.

13. junaio.com. [En línea] http://www.junaio.com/.

14. **Roger.** Windows Phone a punto de superar a iOS en España. *www.eleventel.es.* [En línea] 3 de Diciembre de 2013. [Citado el: 3 de Septiembre de 2014.] http://www.eleventel.es/windows-phone-a-punto-de-superar-a-ios-en-espana/.

15. **Startscene Software.** *starscenesoftware.com.* [En línea] [Citado el: 28 de Marzo de 2014.] http://www.starscenesoftware.com/.

16. Huawei Ascend Y300. *gsmarena.com.* [En línea] [Citado el: 1 de Septiembre de 2014.] http://www.gsmarena.com/huawei_ascend_y300-5386.php.

17. Sony Xperia J. *www.gsmarena.com.* [En línea] [Citado el: 2 de Septiembre de 2014.] http://www.gsmarena.com/sony_xperia_j-4930.php.

18. LG 3 II E430. *www.gsmarena.com.* [En línea] [Citado el: 3 de Septiembre de 2014.] http://www.gsmarena.com/lg_optimus_l3_ii_e430-5292.php.

19. ZTE GRAND X(M). */www.smart-gsm.com.* [En línea] [Citado el: 4 de Septiembre de 2014.] http://www.smart-gsm.com/moviles/zte-grand-xm.

ANEXOS

ANEXOS

Índice general de anexos

A . Gestión y desarrollo del trabajo

A.1. Desarrollo del trabajo

Al inicio de este trabajo, las tecnologías y recursos utilizados eran en gran parte desconocidos por el alumno y por ello se comenzó con una fase de preparación y estudio previo de dichos recursos. Dado el desconocimiento inicial, se siguió un desarrollo iterativo e incremental, en el cual el trabajo se dividió en partes y se les asignó un orden de elaboración. A su vez, estas tareas se dividieron en subtareas.

Cada una de estas divisiones no se cerraban completamente ya que, como el aprendizaje también fue progresivo, se redefinían algunas de las partes de módulos anteriores o se solventaron algunos problemas que pudieron aparecer.

Las tareas principales en las que se dividió fueron, en este orden:

- Estudio previo.
- Estructuración de datos y base de datos.
- Conexión con la base de datos remota.
- Módulo de realidad aumentada / Integración de Vuforia.
- Interfaz gráfica.
- Aplicación de gestión.

Todas estas secciones a su vez estaban divididas en subtareas de diseño, implementación, integración y test.

Debido al punto de partida de desconocimiento, el aprendizaje se ha llevado a cabo durante la realización del trabajo, por lo que cada nueva tarea en la que se hacía uso de nuevas tecnologías conllevaba un nuevo proceso de adaptación y conocimiento que hacía que el progreso se ralentizara, aumentando éste conforme se avanzaba en dicha tarea.

El trabajo se comenzó el 11 de Febrero de 2014 y se finalizó el 10 de septiembre del mismo año. En la Figura A.2 se puede observar el diagrama de Gantt de este trabajo, y en la Figura A.1 se detalla el número y porcentaje de horas invertido en cada tarea. Algunas tareas se han extendido mucho más en el tiempo, debido a lo comentado anteriormente sobre el proceso de modificaciones que se llevaba a cabo conforme se iba desarrollando el proyecto y surgían nuevas necesidades o problemas.

Tarea	Horas	Porcentaje
Estudio previo / bibliográfico / Documentación	54	14,40%
Realidad Aumentada	24	6,40%
Unity 3D	30	8,00%
Conexión con la base de datos	55	14,67%
Estructuración de la base de datos	9	2,40%
Módulo de conexión con BBDD	46	12,27%
Módulo de realidad aumentada	118	31,47%
Librería de interiores	9	2,40%
Realidad aumentada y movimiento	41	10,93%
Integración de Vuforia	18	4,80%
Interfaz gráfica	50	13,33%
Aplicación de gestión	90	24,00%
Funcionalidad	72	19,20%
Formato	18	4,80%
Memoria	45	12,00%
Memoria	45	12,00%
Otros	13	3,47%
Inglés	5	1,33%
Estructura aplicación	8	2,13%
TOTAL	**375**	**100,00%**

Figura A.1 Horas invertidas en cada sección del TFG

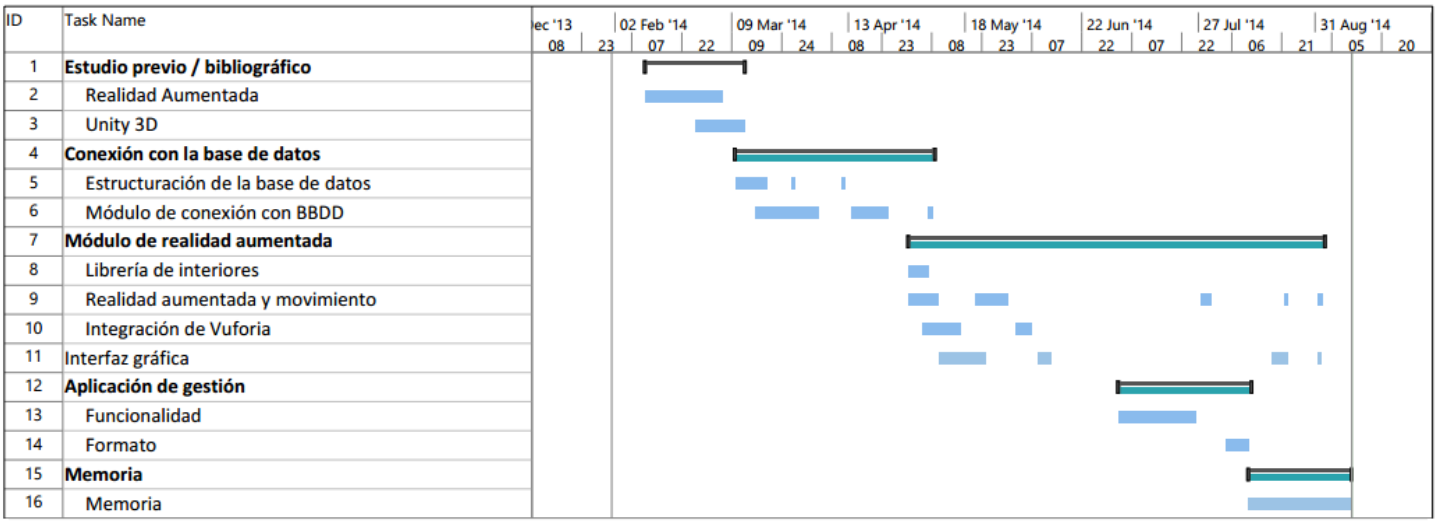

Figura A.2 Diagrama de Gantt del TFG

A.2. Recursos utilizados

Durante la elaboración de este trabajo se han utilizado diversos programas, herramientas y recursos:

- Gestión y organización del trabajo:
  - Microsoft Office Excel: Control de esfuerzos
  - Dropbox: Copias de seguridad.

- Documentación y presentación:
  - Microsoft Office Word: Elaboración de la memoria del trabajo. Gliffy.com: Creación de diagramas y esquemas.
  - Microsoft Office Powerpoint: Presentación del trabajo.
  - Microsoft Project Professional 2013: Diagrama de Gantt

- Modelado 3D
  - GIMP: Edición de texturas.
  - Blender: Manipulación de objetos 3D.
  - Polygon Cruncher: Compresión de modelos 3D.

- Elementos multimedia
  - GIMP: Edición de imágenes.
  - Audacity: Edición e Audio.
  - NotePad++: Edición de texto.

- Implementación:
    - Unity 3D: Creación de la aplicación para dispositivos móviles.
    - MonoDevelop: Desarrollo del código para Unity.
    - Firebug: Corrección de errores en aplicación de gestión.
    - NotePad++: Desarrollo del código de la aplicación de gestión.
    - RapidCSS: Desarrollo del código CSS para aplicación de gestión.

B. Contexto tecnológico y herramientas utilizadas.

B.1 Realidad Aumentada

El término *Realidad Aumentada* (AR) hace referencia a la visión combinada del entorno físico real junto con elementos virtuales, es decir, la creación de una realidad mixta en tiempo real a través de elementos tecnológicos. La principal diferencia de ésta con la realidad virtual es que la AR superpone la información virtual sobre la imagen del mundo real, mientras que la realidad virtual define un nuevo mundo que sustituye a nuestro entorno.

Basándonos en la forma en la que el sistema sincroniza y sitúa la información adicional en el entorno real, lo cual tiene especial interés en entornos móviles, existen diferentes tipos de realidad aumentada. Los principales son:

- ***Con marcadores***: El sistema hace uso de un conjunto de imágenes conocidas, las cuales reconoce dentro del entorno real y actúa según la posición de estos marcadores. Pueden utilizarse también como puntos de referencia en ausencia de herramientas de localización. El uso de marcadores está avanzando y hoy en día existe la posibilidad de definir cualquier imagen captada por la cámara como un nuevo marcador e incluso un edificio real.

- ***Geolocalización***: Se obtiene la posición actual de sistemas de geolocalización y se integra esa información en el sistema para su comportamiento óptimo. Este tipo de AR está muy ligado a los elementos móviles y con los datos de orientación del dispositivo se puede realizar una muy buena gestión de la información a mostrar en función de la posición.

B.2 Unity3D

Unity3D es un ecosistema de desarrollo de videojuegos multiplataforma, con una comunidad muy activa que debe su expansión en parte a la existencia de una versión con licencia gratuita. Esto ha permitido a desarrolladores y programadores poder acceder a una completa herramienta de renderizado y creación sin las siempre presentes barreras de tiempo y coste.

Esta herramienta permite generar contenidos para Windows, OS X, Linux, Adobe Flash Player, Xbox 360, PlayStation 3, PlayStation Vita, Wii, Wii U, iPad, iPhone, Android y Windows Phone.

El *scripting* se basa en Mono, la implementación de código abierto de .NET Framework. Se pueden utilizar como lenguajes de programación UnityScript –un lenguaje personalizado inspirado en la sintaxis ECMAScript–, C# o Boo –el cual está inspirado sintácticamente en Python–. A partir de la versión 3.0 añade una versión personalizada de *MonoDevelop* para la depuración de scripts.

B.3 Qualcomm® Vuforia™

Vuforia es una plataforma de desarrollo de software que pone a disposición de los programadores de aplicaciones móviles un motor de reconocimiento de imágenes muy potente, así como un amplio abanico de herramientas. Es totalmente compatible tanto con Android como con iOS, lo que permite a los desarrolladores portar sus aplicaciones de una plataforma a otra sin dificultad y en un plazo de tiempo mínimo.

Este completo sdk desarrollado por Qualcomm permite trabajar, además de con los dos sistemas móviles, con Unity3D de forma totalmente integrada gracias a su extensión. Dicha extensión está disponible tanto para la versión de pago del motor como para la gratuita. La conjunción de ambas herramientas permite el desarrollo de aplicaciones de realidad aumentada de forma mucho más dinámica, facilitando tanto la animación gráfica como el proceso de seguimiento de marcadores.

B.4 Parse

Parse es un *Backend as a service* (BaaS) que ofrece múltiples funcionalidades en la nube. Entre los servicios que ofrece Parse podemos destacar:

- ***Modelo de datos en la nube***: creación de tablas no-SQL en la nube y capacidades para inserción, modificación y consulta vía API.

- ***Notificaciones Push***: posibilidad de envío de notificaciones push a nuestros usuarios, previa aceptación por parte del usuario.

- ***Cloud Code***: capacidades para la ejecución de código en el servidor, muy útil para la realización de validaciones de seguridad, o procesos automáticos por cambios en los datos.

Este servicio posee SDK para múltiples plataformas, como iOS, OS X, Android, Windows Phone, Windows, Javascript, .NET, Unity, PHP y Xamarin, lo que lo convierte en una muy buena herramienta para gestionar tus datos en la nube dado su grado de flexibilidad en lo que ha plataforma nativa se refiere. Es un servicio gratuito para aplicaciones que no consuman excesivo tráfico ni almacenamiento.

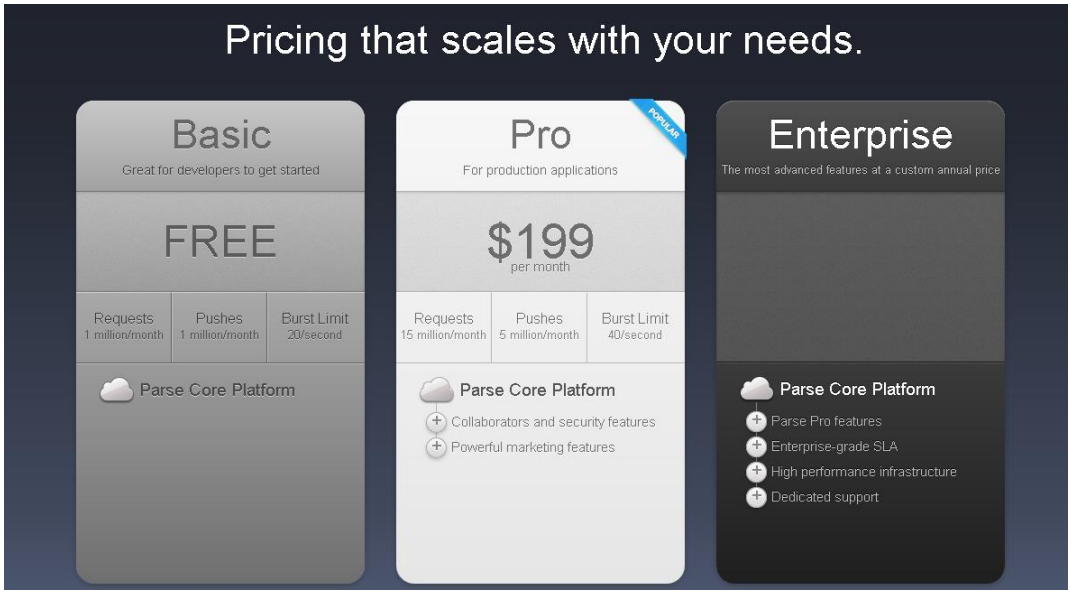

Figura B.1 Opciones de uso de Parse[ii].

[ii] https://parse.com/legacy_plans

B.5 C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270).

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Existen compiladores para C# tales como Microsoft Visual Studio, SharpDevelop, Mono o Delphi.

Entre las metas y objetivos del diseño de este lenguaje caben destacar la búsqueda de un lenguaje orientado a objetos de carácter general y simple, con la inclusión de bases de ingeniería de software como revisión de tipos de datos o límites de vectores y recolección de basura automática; también se incluyen en las metas una fácil migración del programador al nuevo lenguaje, así como portabilidad del código fuente y eficiencia en cuanto a procesado y memoria.

B.6 HTML

HTML -*HyperText Markup Language* o lenguaje de marcas de hipertexto-, hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web, como texto, imágenes, etc. Es un estándar a cargo de la W3C y es el lenguaje con el que se definen las páginas web.

El lenguaje HTML basa su filosofía de desarrollo en la referenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, etc.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De esta forma,

el contenido de la página web es integramente texto y es el navegador web el que se encarga de unir los elementos y ofrecernos la visión final de la página.

Su papel de estándar hace que HTML busque que desde cualquier navegador actualizado una página web sea interpretada de la misma manera. Sin embargo, ha ido evolucionando mediante diferentes versiones añadiendo y eliminando diferentes características en aras de facilitar el desarrollo web multiplataforma.

B.7 JavaScript

JavaScript es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase. Está basado en prototipos, es multi-paradigma, dinámico y soporta estilos de programación funcional, orientada a objetos e imperativa. Es conocido como el lenguaje de script para páginas web, pero es utilizado también en otros entornos sin navegador.

Este lenguaje fue desarrollado por Netscape, y estandarizado más tarde por ECMA - *European Computer Manufacturers Association*-. JavaScript es una implementación concreta realizada por Netscape del estándar autorizado por ECMA – denominado ECMAScript – y es usado principalmente en el lado del cliente para aplicaciones web mediante el navegador.

B.8 CSS

CSS - *Cascading Style Sheets* o Hoja de estilo en cascada- es un lenguaje utilizado para estructurar la presentación de un documento definido en HTML o XML. El estándar depende del *World Wide Web Consortium* (W3C), el cual formula las especificaciones para las hojas de estilo.

Su objetivo básico es el de separar la presentación de un documento de su estructura y lógica. Se puede incluir la información de estilos tanto en el propio documento como en otro documento externo.

Posee una sencilla sintaxis para definir los nombres de varias propiedades de estilo a partir de palabras clave y a través de los selectores se especifican los elementos que van ser modificados en el documento.

Entre las ventajas de su uso se encuentra la mejora de la accesibilidad del documento, la separación del contenido de la presentación que permite una gran flexibilidad de visualización para un mismo documento, el control centralizado de la presentación que induce una agilización del proceso, y la optimización de ancho de banda, ya que un selector puede definir el estilo de muchos elementos e incluso un mismo archivo de estilo puede ser utilizado en multitud de documentos.

Por el otro lado, las limitaciones que nos podemos encontrar en su utilización son la dificultad para el alineamiento vertical, para el cual no existen reglas evidentes o estándares, la ausencia de expresiones de cálculo numérico, la incapacidad de utilizar los selectores en orden ascendente según la jerarquía del DOM, y la imposibilidad de controlar o deshabilitar *las pseudo-clases* dinámicas –que definen estados como activo o seleccionado– desde el navegador.

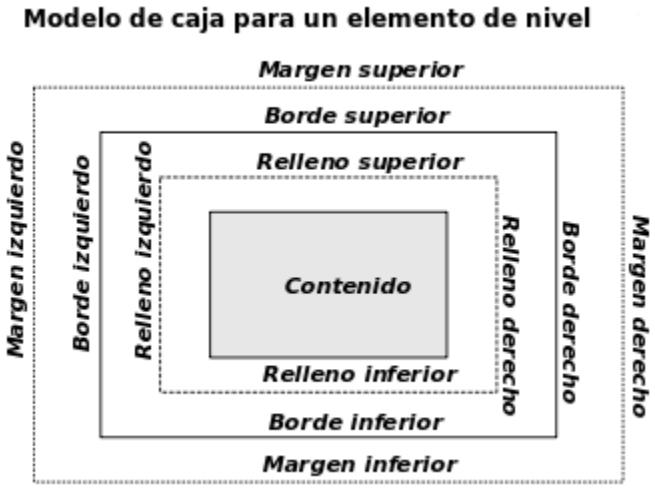

Figura B.2 Modelo de caja para un elemento de nivel

C . Formato OBJ

El formato utilizado para manejar información en 3D en este trabajo ha sido el formato creado por *Wavefront Technologies.* Este formato, con extensión .obj, ha expandido su uso en aplicaciones gráficas hasta convertirse en casi un estándar de facto que es soportado por la mayoría de los servicios.

Wavefront es una compañía de gráficos por computación que desarrolla y vende software de animación utilizado, entre otras industrias, por la cinematográfica. Hoy en día forma parte de *Silicon Graphics* (RC1).

OBJ es un formato abierto de definición de geometrías aceptado casi mundialmente. Soporta líneas, polígonos, curvas libres de formas y superficies. Líneas y polígonos son descritos en base a sus puntos, mientras que las curvas y superficies se definen a través de puntos de control y otra información dependiendo el tipo de curva. Además soporta curvas racionales como no racionales. Los vértices se almacenan en sentido contrario a las agujas del reloj por defecto, haciendo innecesaria la declaración expresa de caras normales. Las coordenadas OBJ no tienen unidades, pero los archivos OBJ pueden contener información de escala en una línea de comentario.

C.1 Estructura del fichero

Este formato no requiere de cabecera, aunque es habitual comenzarlo con una línea de cometario. Los comentarios comienzan con el símbolo #. Se pueden añadir líneas y espacios en blanco en todo el documento para mejorar la legibilidad, y cada línea escrita debe comenzar con una de las palabras clave definidas en la especificación (RC2) (RC3) y debe continuar en la misma línea con los datos relativos a esa clave.

Las siguientes palabras clave pueden ser incluidas en un archivo OBJ. En esta lista, las palabras clave están organizadas por tipo de datos, y a cada una le acompaña una breve descripción:

- ***Datos de vértices:***

  v ---------- Vértices geométricos
  vt ---------- Vértices de texturas
  vn ---------- Vectores normales a los vértices
  vp ---------- Parámetros de espacio de los vértices / atributo de curvas y superficies
  deg -------- Ángulo
  bmat ------ Matriz de base
  step ------- Tamaño de paso

  cstype ----- Tipo de curva o superficie

- ***Elementos***:

  p ---------- Punto
  l ---------- Línea
  f ---------- Cara
  curv -------- Curva
  curv2 ------ Curva 2D
  surf -------- Superficie

- ***Declaraciones de cuerpos de curvas/superficies:***

  parm -------- Valores de parámetros
  trim -------- Círculo de recorte exterior
  hole -------- Círculo de recorte interior
  scrv -------- Curva especial
  sp -------- Punto especial
  end -------- Fin de la declaración

- ***Conectividad entre superficies libres de forma:***

  con -------- Connect

- ***Agrupaciones:***

g	---------	Nombre de grupo
s	---------	Grupo regular (*smoothing)*
mg	---------	Grupo de unión (*Merging*)
o	---------	Nombre del objeto

- ***Atributos de renderizado/representación:***

bevel	Interpolación de bisel
c_interp	Interpolación de color
d_interp	Interpolación diluida
lod	Nivel de detalle
usemtl	Nombre de material
mtllib	Librería de material
shadow_obj	Proyección de sombras
trace_obj	Trazamiento de rayos
ctech	Técnica de aproximación de curvas
stech	Técnica de aproximación de superficies

Un ejemplo simple de fichero OBJ que define un triángulo sería:

```
# Simple Wavefront file

    mtllib triangle.mtl
    o Grid
    v 0.0 0.0 0.0
    v 0.0 1.0 0.0
    v 1.0 0.0 0.0
        f 1 2 3
```

Los ficheros OBJ no contienen definiciones de color para las caras, aunque pueden contener referencias a materiales que son almacenados en una librería de materiales externa. La librería de materiales puede ser cargada usando la palabra clave *mtllib*. Estas librerías contienen los valores RGB de los materiales, así como otras características como refracción, transparencia o especularidad.

C.2 Fichero de materiales

El fichero OBJ referencia a los materiales mediante el uso de la palabra clave usemtl. Las librerías de materiales poseen un formato específico definido también por Wavefront Technologies, con extensión .MTL y en formato ASCII. Estos ficheros contienen una o varias definiciones de materiales, incluyendo cada una olor, textura y mapa de reflexión, y son aplicados a las superficies o vértices de los objetos. Se pueden asociar imágenes a materiales para que actúen como texturas.

Un fichero MTL (RC4) está usualmente organizado de la siguiente manera:

newmtl color1
    Sentencias de color e iluminación

    Sentencias de mapa de texturas

    Sentencias de mapa de reflexión

newmtl color2
    Sentencias de color e iluminación

    Sentencias de mapa de texturas

    Sentencias de mapa de reflexión

[…]

C.3 Bibliografía del Anexo C

(RC1)-. Silicon Graphics.*sgi.com.* [En línea] https://www.sgi.com

(RC2)-. Especificación OBJ. *www.martinreddy.net*. [En línea] [Citado el: 2 de Septiembre de 2014]. http://www.martinreddy.net/gfx/3d/OBJ.spec.
.

(RC3)-. Wavefront OBJ File Format Summary. *www.fileformat.info.* [En línea] [Citado el: 2 de Septiembre de 2014]
http://www.fileformat.info/format/wavefrontobj/egff.htm
.

(RC4)-. MTL material format (Lightwave, OBJ). *www.paulbourke.net*. [En línea] [Citado el: 2 de Septiembre de 2014]
http://www.paulbourke.net/dataformats/mtl/

Figura C.1 Ejemplo de modelo 3D en formato OBJ

D. Imágenes del trabajo

En este Anexo se incluyen las capturas de pantalla de la aplicación de usuario para ofrecer una visión gráfica del funcionamiento de la aplicación, así como imágenes de la aplicación de gestión de la base de datos y las imágenes que actúan como marcadores para el Museo de Informática Histórica de Zaragoza.

D.1. Aplicación MuseAR

Todas las capturas de pantalla realizadas en la aplicación se han realizado desde un Huawei Y300, de 800x480 píxeles y 4''.

D.1.1. Escena I: Intro

Figura D.1 Pantalla inicial

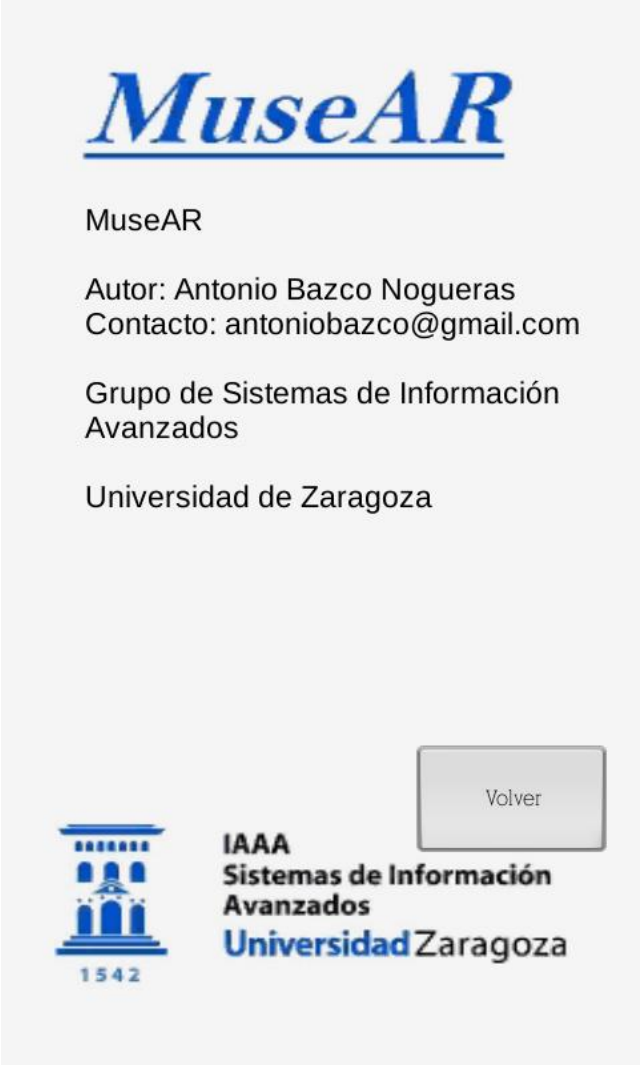

Figura D.2 Pantalla "Acerca de"

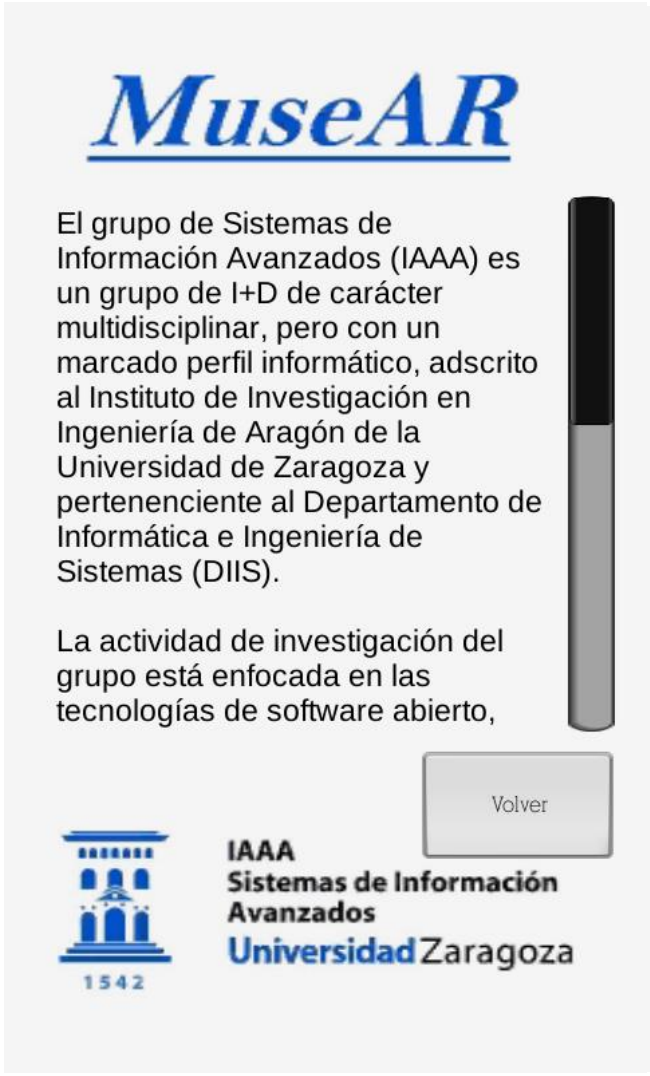

Figura D.3 Pantalla "IAAA"

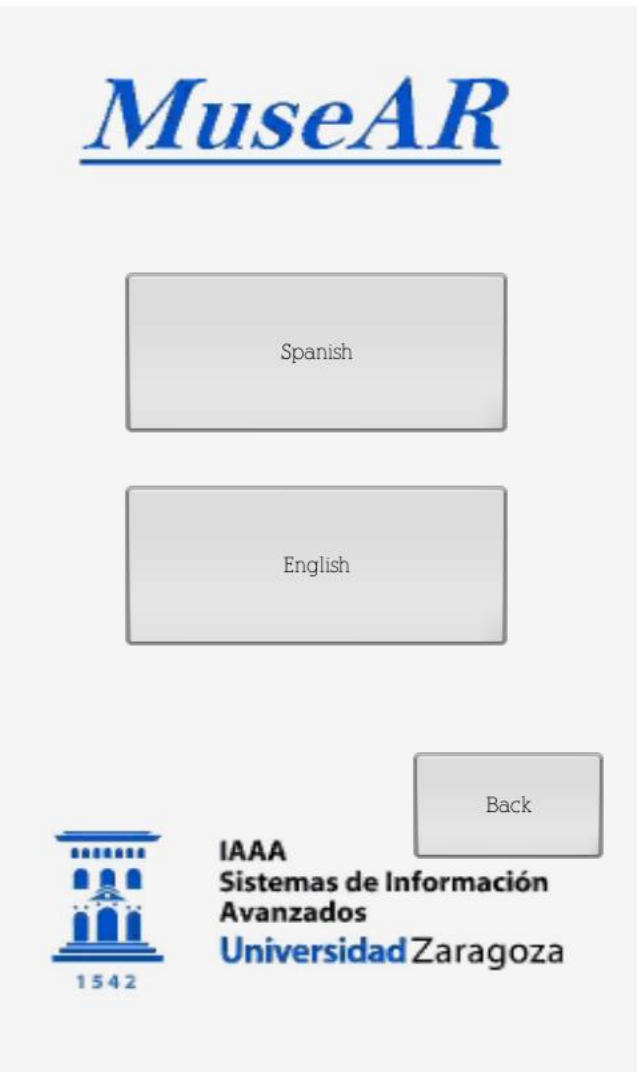

Figura D.4 Pantalla "Cambiar idioma"

D.1.2. Escena II: Musear

Figura D.5 Pantalla inicial antes de detectar un marcador

Figura D.6 Pantalla durante la descarga de datos

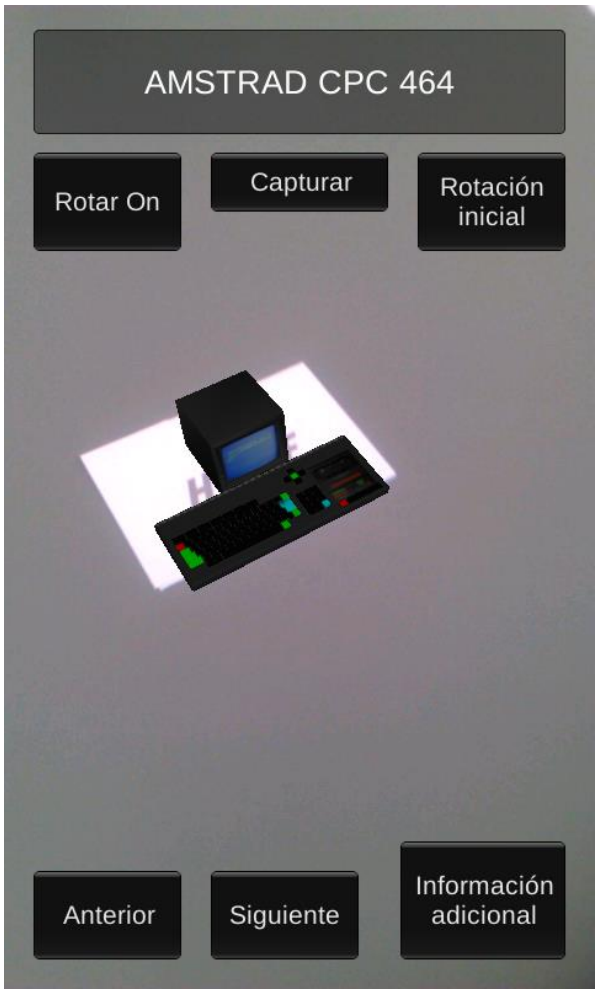

Figura D.7 Modelo 3D sobre marcador

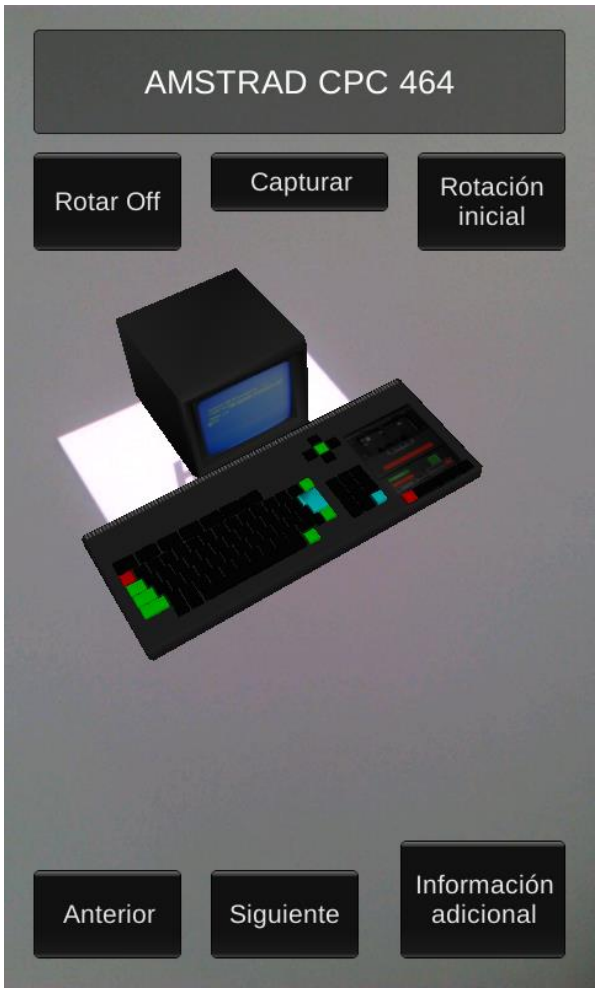

Figura D.8 Ejemplo de zoom

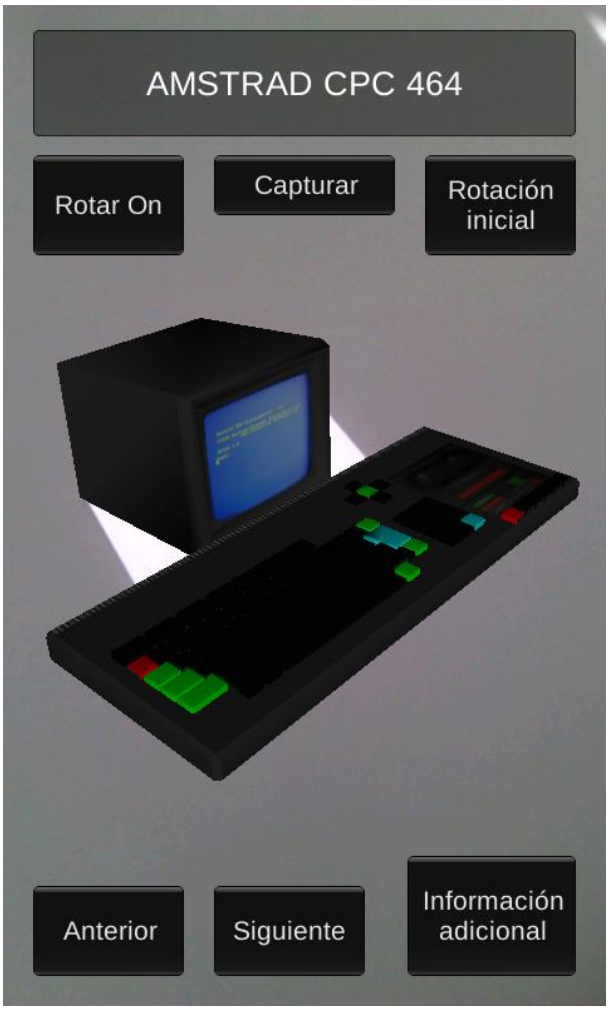

Figura D.9 Ejemplo de rotación del modelo 3D

Figura D.10 Aviso de la ausencia de modelo 3D

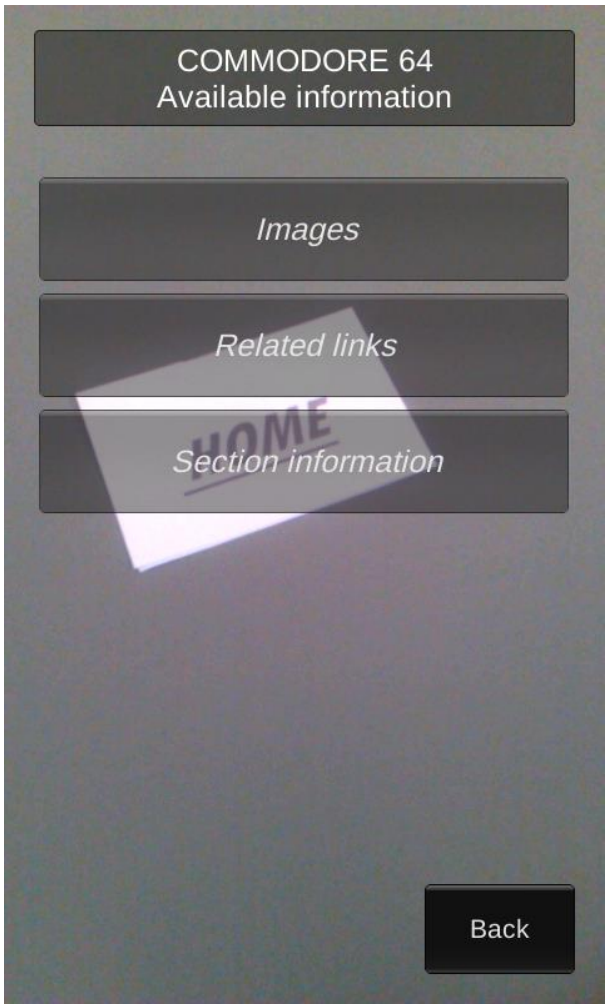

Figura D.11 Menú –inglés– de información adicional

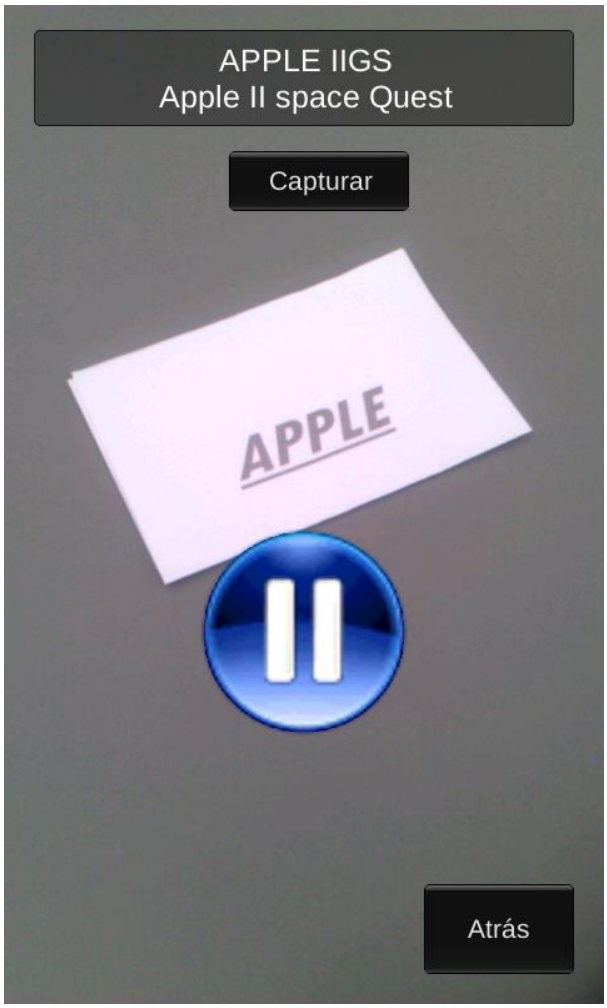

Figura D.12 Pantalla de audio

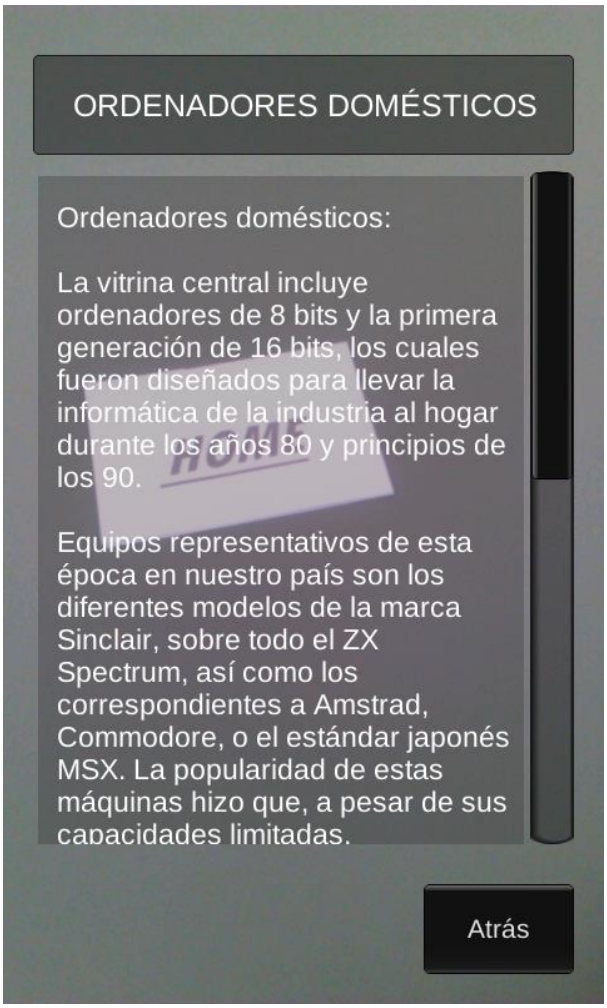

Figura D.13 Información de la sección

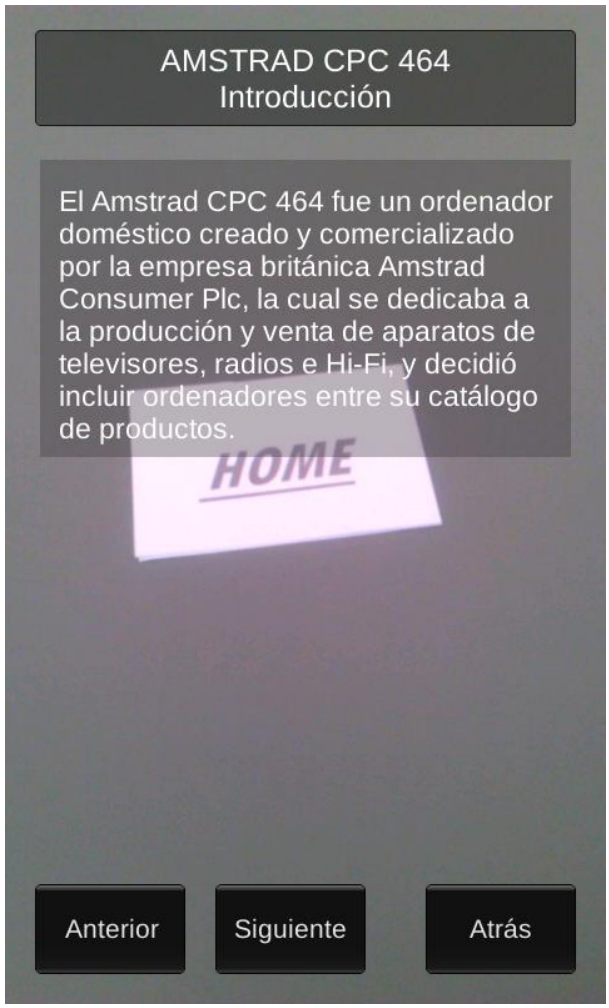

Figura D.14 Información textual del elemento

Figura D.15 Pantalla de imágenes I

Figura D.16 Pantalla de imágenes II

Figura D.17 Pantalla de enlaces de interés

D.2.Aplicación de gestión

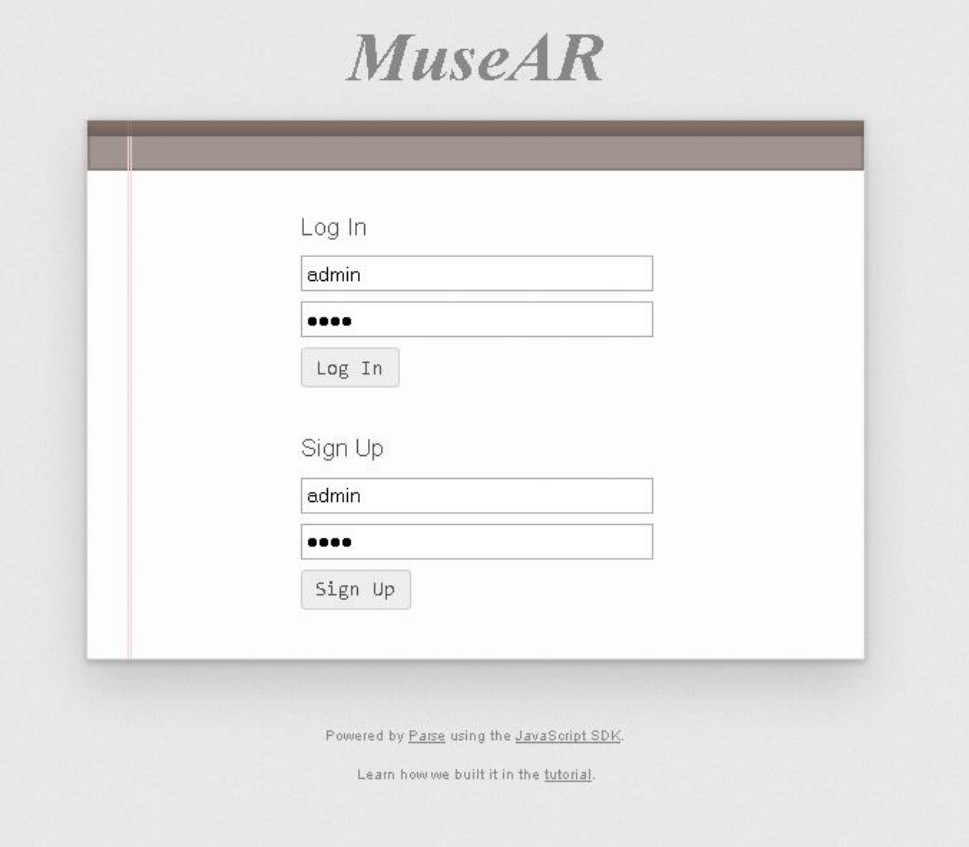

Figura D.18 Acceso al servicio

Figura D.19 Vista general de la lista de elementos

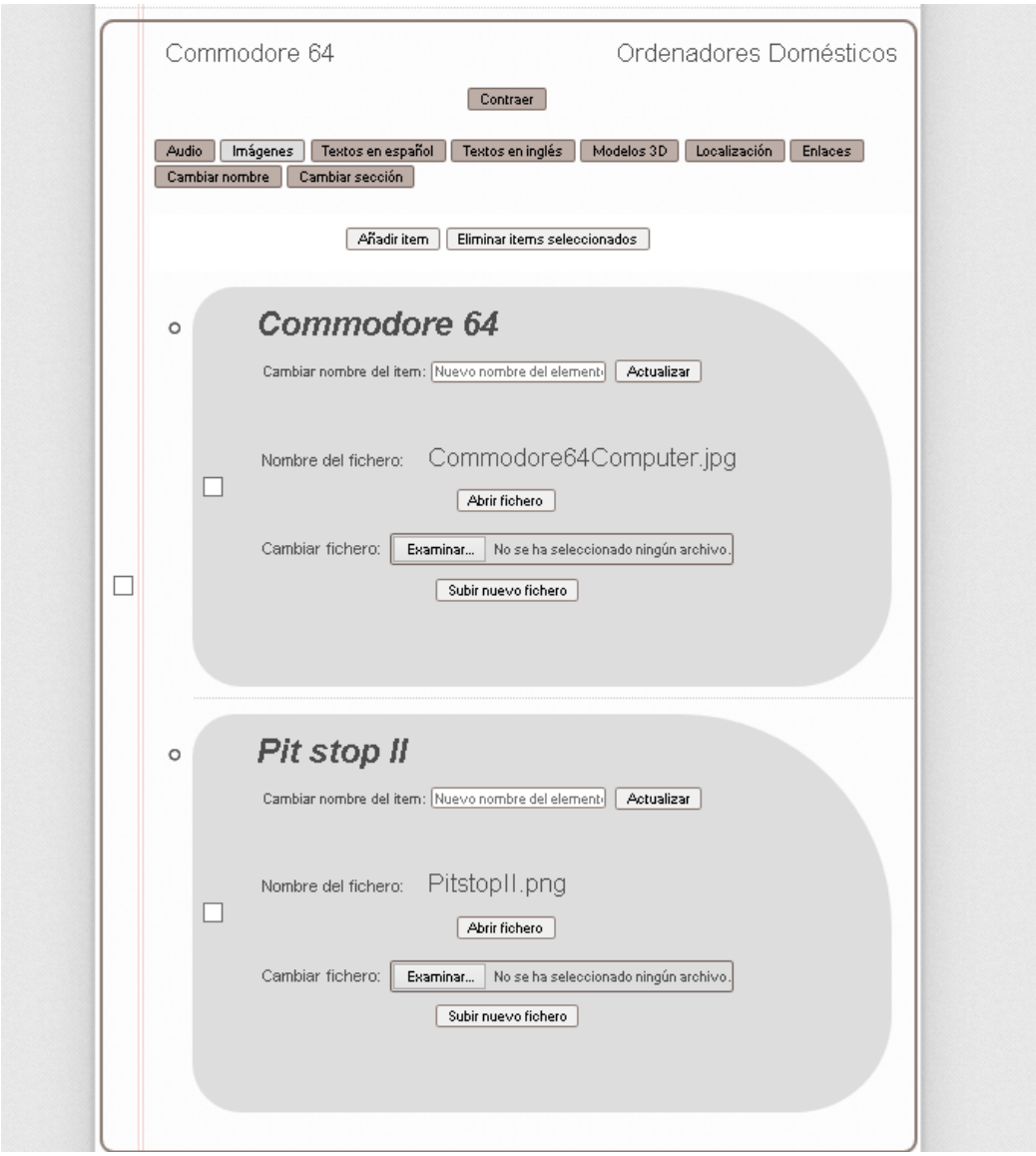

Figura D.20 Apartado de imágenes expandido

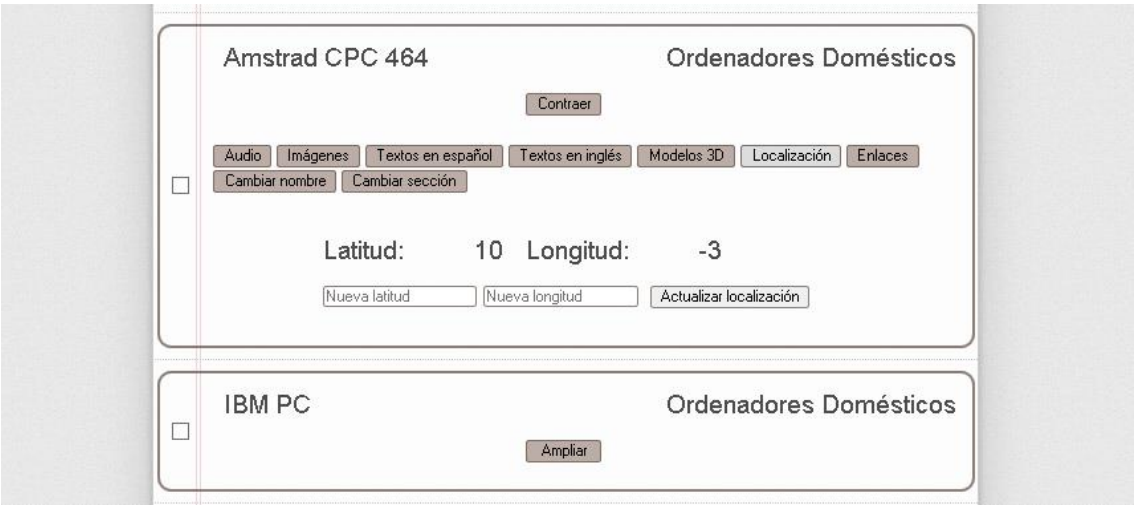

Figura D.21 Apartado de localización expandido

Figura D.22 Apartado de audio expandido

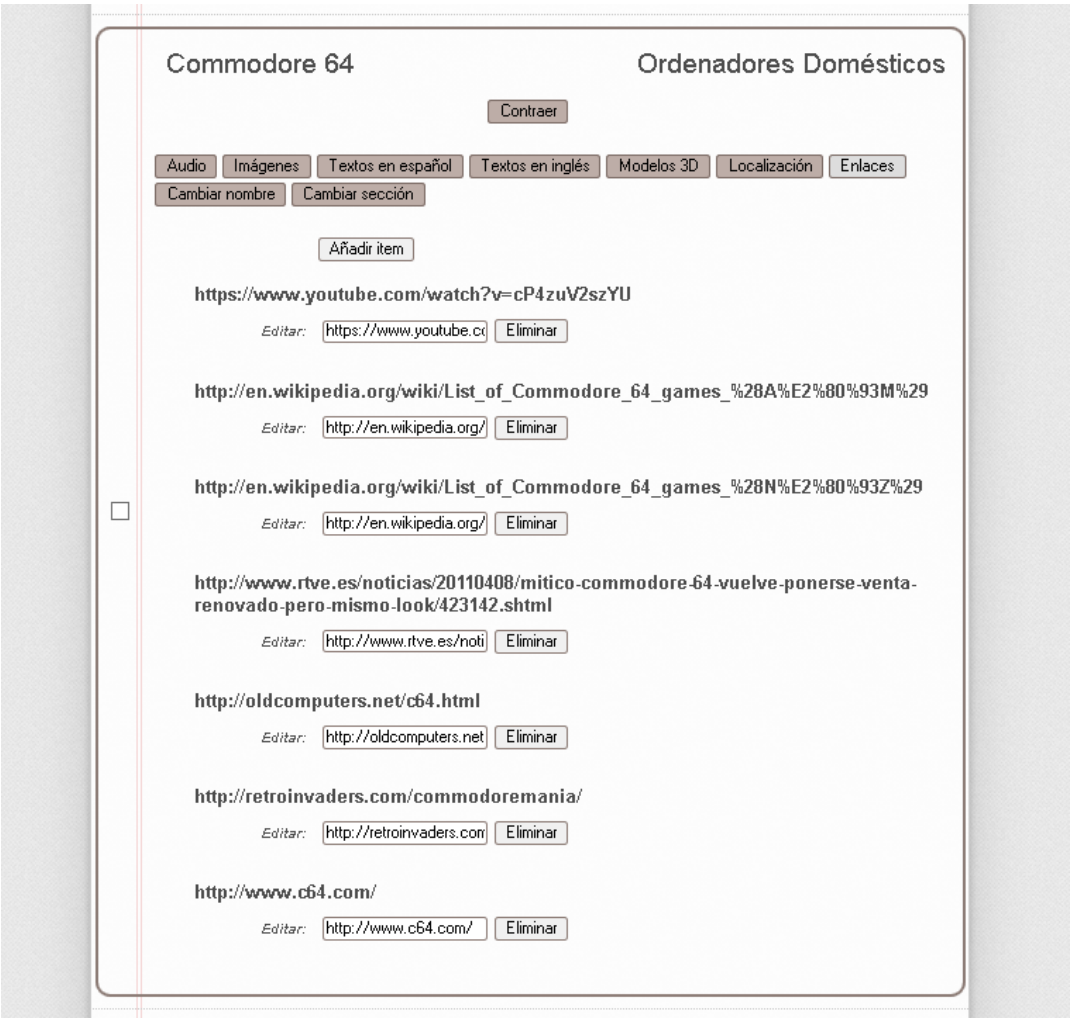

Figura D.23 Apartado de enlaces de interés expandido

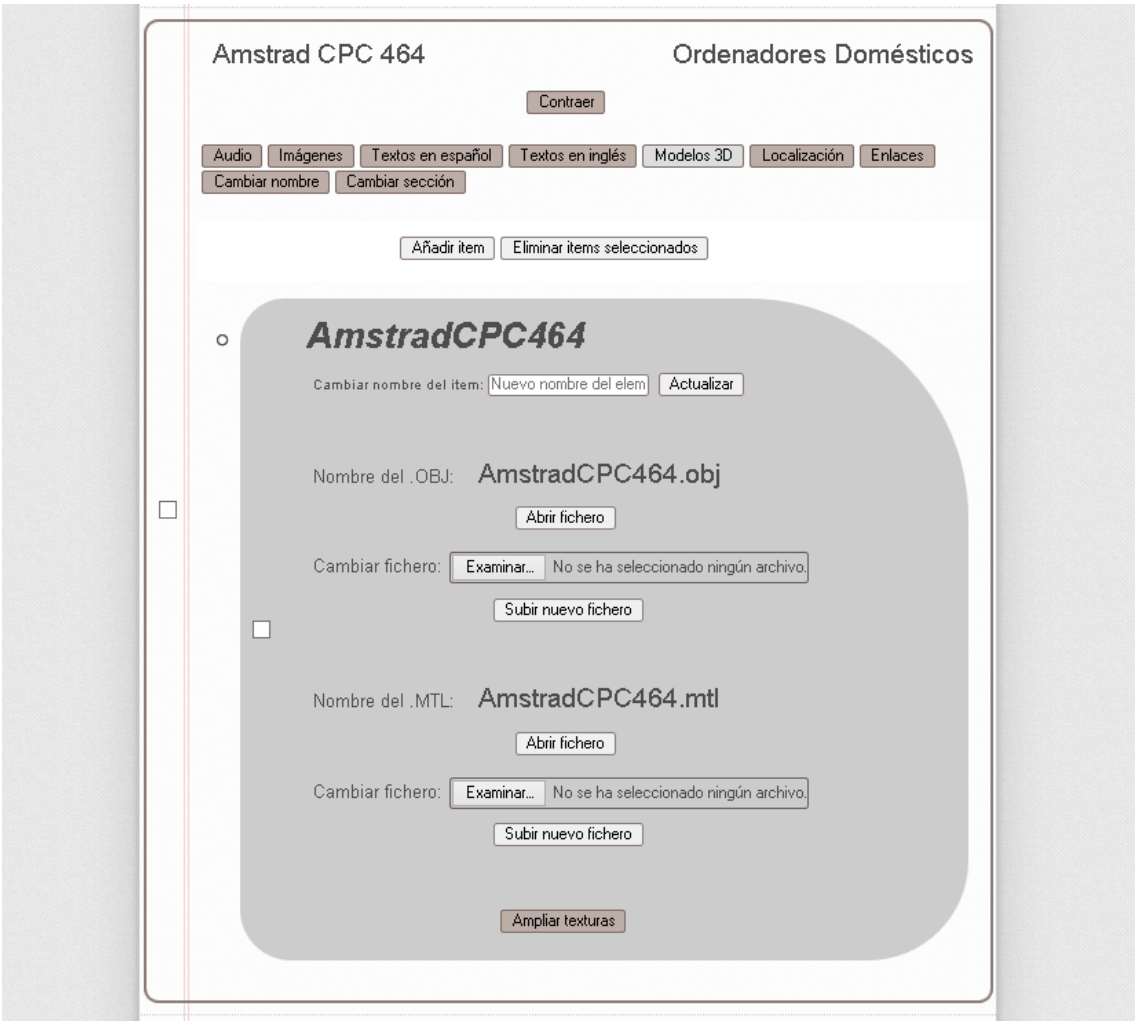

Figura D.24 Apartado de modelos 3D expandido sin abrir texturas

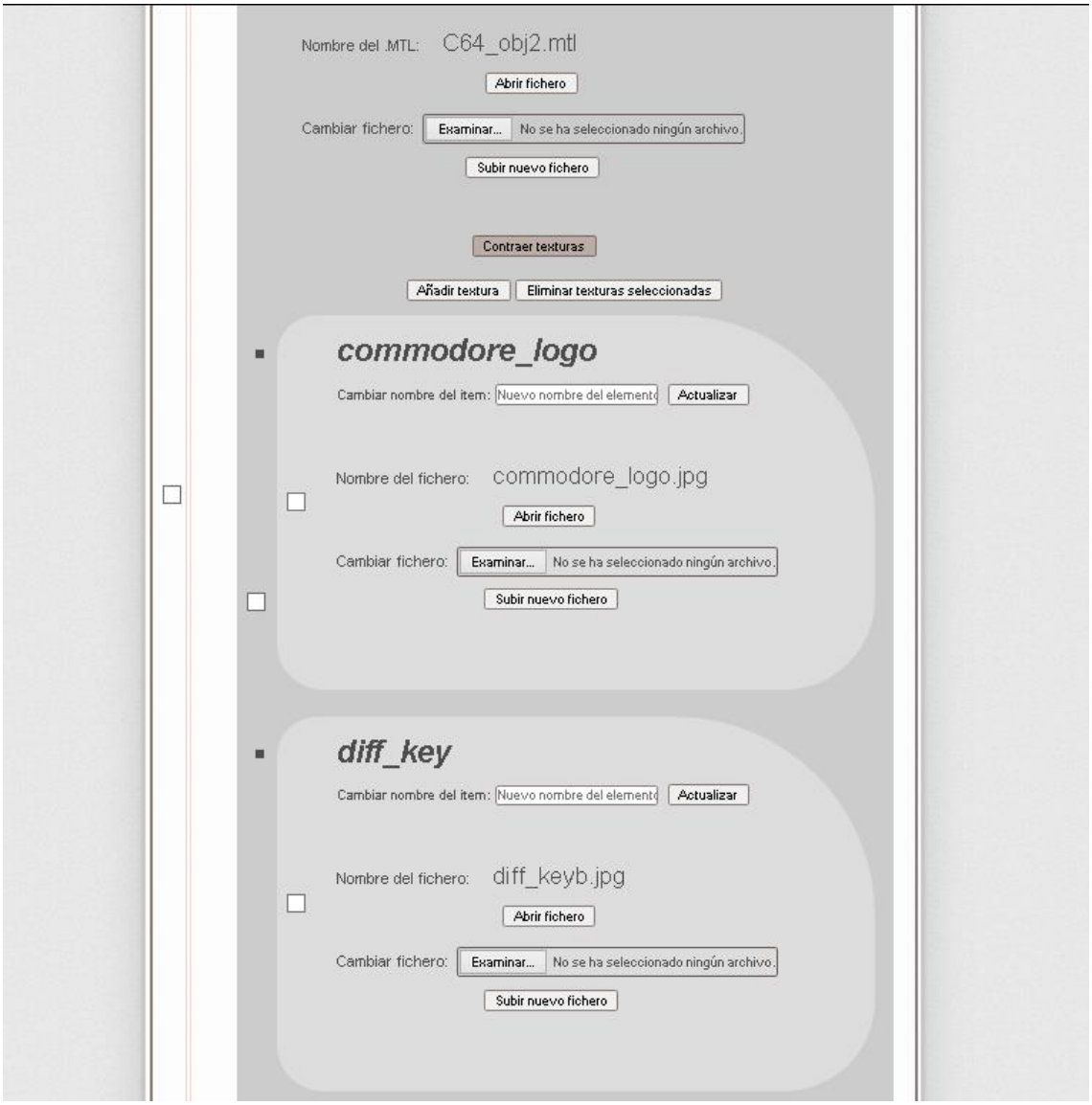

Figura D.25 Apartado de modelos 3D expandido con lista de texturas expandida

D.3. Marcadores

En este apartado se muestran los marcadores utilizados para cada una de las secciones. Se puede seleccionar cualquier imagen para que actúe de marcador. En este caso se han seleccionado unas imágenes que contienen una palabra clave en relación con la sección que referencia:

- Sección "Ordenadores Domésticos"

Figura D.26 Marcador de la sección "Ordenadores Domésticos"

- Sección "Ordenadores Apple"

Figura D.27 Marcador de la sección "Ordenadores Apple"

- Sección “ Estaciones de Trabajo”

Figura D.28 Marcador de la sección “Estaciones de Trabajo”

- Sección “Servidores y terminales”

Figura D.29 Marcador de la sección “Servidores y terminales”

- Sección “Compatibles PC”

Figura D.30 Marcador de la sección “Compatibles PC”