



UNIVERSIDAD DE EXTREMADURA

**Escuela Politécnica
Ingeniero en Informática**

Proyecto Fin de Carrera

**Real-A: Una aproximación RIA a la Realidad
Aumentada en móviles usando MDD**

**Juan María Domínguez González
Septiembre, 2011**



UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Ingeniero en Informática

Proyecto Fin de Carrera

Real-A: Una aproximación RIA a la Realidad

Aumentada en móviles usando MDD

Autor: Juan María Domínguez González

Fdo.:

Director: Marino Linaje Trigueros

Fdo.:

Tribunal Calificador

Presidente:

Fdo.:

Secretario:

Fdo.:

Vocal:

Fdo.:

CALIFICACIÓN:

FECHA:

A mi mujer Noelia, por recordarme continuamente que lo tenía que hacer.

A mis hijas Ana y Lucia, porque todo lo hago por ellas.

A mis Padres, porque se lo debía.

Contenido

1. Resumen y Objetivos	17
2. Introducción.	19
3. Bases del proyecto.	23
3.1. Realidad Aumentada.	23
3.1.1. Actualidad de la Realidad Aumentada.	23
3.2. Introducción a Dispositivos Móviles.	25
3.2.1. Los teléfonos inteligentes y sus sistemas operativos.....	25
3.3. Desarrollo de Aplicaciones Móviles.	29
3.3.1. Aplicaciones Nativas.	29
3.3.2. Aplicaciones Web.....	31
3.3.3. Aplicaciones Híbridas.	32
3.3.4. Comparativa Aplicaciones Nativas, Web e Híbridas.	35
3.3.5. Limitaciones de RIAs para aplicaciones de Realidad Aumentada. .	39
3.4. Desarrollo dirigido por modelos.	43
4. Real-A: Aproximación RIA a la Realidad Aumentada móvil usando MDD. .	49
4.1. Introducción.	49
4.2. Reconocimiento de imágenes.....	53
4.2.1. Técnicas para la caracterización de objetos.	54
4.2.2. Técnicas para el reconocimiento de objetos.	56
4.2.3. El reconocimiento de objetos en Real-A.	57
4.3. Manual de desarrollador.	63
4.3.1. Configuración Log.	64
4.3.2. Control de Versiones mediante Subversion.	66
4.3.3. Dominio (Data Model).	67
4.3.4. Acceso a la aplicación.	71
4.3.5. Module View Mensajes.....	72
4.3.6. Publico y PublicoMobile.....	76
4.3.7. AccesoUsuario y AccesoUsuarioMobile.	80
4.3.8. Servlet UploadFile.	92
4.3.9. AccesoAdministrador y AccesoAdministrador Mobile.	94
4.3.10. Nivel de presentación.	103
4.4. Manual de usuario.	104
4.4.1. Estructura general de una pantalla.	105
4.4.2. Zona Pública.	107
4.4.3. Zona Usuario.....	111

Contenido

4.4.4. Zona Administrador.....	119
5. Resumen y conclusiones.....	135
6. Trabajo futuro.....	141
7. Bibliografía.....	143

Índice de Tablas:

Tabla 1.- Comparativa Sistemas Operativos Móviles.....	26
Tabla 2.- Comparativa de aplicaciones nativas, Web e hibridas.....	38
Tabla 3.- Site Views de Real-A.....	71
Tabla 4.- Estadísticas líneas de código generadas.....	136
Tabla 5.- Estadísticas líneas de código manuales.....	136
Tabla 6.- Alternativas de desarrollo de aplicaciones de Realidad Aumentada en móviles.....	138

Índice de Figuras:

Figura 1.- Diagrama de la Aplicación.....	20
Figura 2.- secuencia de realidad-virtualidad de Milgram.	23
Figura 3.- Sistemas Operativos Móviles.	31
Figura 4.- Desarrollo de aplicaciones Web.	32
Figura 5.- Desarrollo de aplicaciones Híbridas.....	34
Figura 6.- Inserción de imágenes en BBDD.....	58
Figura 7.- imágenes en BBDD.....	59
Figura 8.- Obtención nº de coincidencias entre una imagen y las imágenes de BBDD.	60
Figura 9.- Imagen comparativa de dos imágenes.	62
Figura 10.- Estructura de Real-A.	64
Figura 11.- Dominio Real-A.....	68
Figura 12.- Module View Mensajes.....	73
Figura 13.- Mensajes de Error.....	73
Figura 14.- Mensajes de Información.....	73
Figura 15.- Cambiar Idioma.....	75
Figura 16.- Site View Publico y PublicoMobile.	76
Figura 17.- Site views Usuario y UsuarioMobile (Mantenimiento de usuario).	81
Figura 18.- Site views Usuario y UsuarioMobile (Real-A y Álbum).....	82
Figura 19.- Site views Administrador y AdministradorMobile (Mantenimiento de Objetos, Mantenimiento de Descripciones y Mantenimiento de Imágenes).....	95
Figura 20.- Site views Administrador y AdministradorMobile (Querys).....	95
Figura 21.- Site views Administrador y AdministradorMobile (Mantenimiento de idiomas).	96
Figura 22.- Site views Administrador y AdministradorMobile (Mantenimiento de Usuarios).....	96
Figura 23.- Estructura general de pantalla para navegadores de sobremesa.	105
Figura 24.- Estructura general de pantalla para navegadores móviles.	105
Figura 25.- Popup de error, informativo y de confirmación.	106
Figura 26.- Login.....	108
Figura 27.- Login Móvil.....	108
Figura 28.- Crear usuario.	109
Figura 29.- Crear usuario Móvil.	109
Figura 30.- Recuperar contraseña.	110
Figura 31.- Recuperar contraseña Móvil.	110
Figura 32.- Real-A. Posición Horizontal.	111
Figura 33. Descarga e instalación APIBridge_v1_1.sis.	112

Índice de Figuras

Figura 34.- Real-A. Cámara nativa.....	112
Figura 35.- Real-A. Selección de imagen.....	113
Figura 36.- Buscando coincidencias.....	113
Figura 37.- Real-A. Información de la imagen obtenida.....	113
Figura 38.- Real-A. Grabadora Nativa.....	114
Figura 39.- Real-A. Información de la imagen obtenida.....	115
Figura 40.- Mantenimiento de Usuario.....	116
Figura 41.- Mantenimiento de Usuario Móvil.....	116
Figura 42.- Álbum.....	117
Figura 43.- Álbum Móvil.....	117
Figura 44.- Mantenimiento de fotos del álbum.....	118
Figura 45.- Mantenimiento de fotos del álbum Móvil.....	118
Figura 46.- Mantenimiento de Objetos.....	120
Figura 47.- Mantenimiento de Objetos Móvil.....	120
Figura 48.- Mantenimiento de objeto.....	121
Figura 49.- Mantenimiento de objeto Móvil.....	122
Figura 50.- Mantenimiento de objetos. Nueva imagen.....	122
Figura 51.- Mantenimiento de objetos. Nueva imagen. Móvil.....	123
Figura 52.- Mantenimiento de descripciones.....	124
Figura 53.- Mantenimiento de descripciones Móvil.....	124
Figura 54.- Mantenimiento de Usuarios.....	125
Figura 55.- Mantenimiento de Usuarios Móvil.....	125
Figura 56.- Mantenimiento de Usuario.....	126
Figura 57.- Mantenimiento de Usuario Móvil.....	126
Figura 58.- Mantenimiento de Idiomas.....	127
Figura 59.- Mantenimiento de Idiomas Móvil.....	127
Figura 60.- Mantenimiento de Idioma.....	128
Figura 61.- Mantenimiento de Idioma Móvil.....	128
Figura 62.- Búsqueda de imágenes.....	129
Figura 63.- Búsqueda de imágenes Móvil.....	130
Figura 64.- Resultados.....	131
Figura 65.- Resultados Móvil.....	131
Figura 66.- Comparación de imágenes.....	132
Figura 67.- Comparación de imágenes Móvil.....	132

Figura 68.- Información	133
Figura 69.- Información Móvil	133

Índice de Figuras

Índice de Código:

Código 1.- Consulta de coincidencias entre imágenes.....	61
Código 2.- Estructura punto clave.....	61
Código 3.- RTXLogConfig.properties.....	65
Código 4.- Ejemplo de mensajes de log en la aplicación.....	66
Código 5.- JSP correspondiente a las páginas “redirección”	71
Código 6.- WEB-INF/web.xml.....	72
Código 7.- ControlMensajes.groovy.....	75
Código 8.- CambiarIdioma.groovy.	78
Código 9.- MostrarErrorLogin.groovy.....	79
Código 10.- DescargarApiBridge.unit.template	83
Código 11.- CameraLayout.unit.templete, objeto Flash embebido en html.	84
Código 12.- CameraLayout.unit.templete, comunicación JavaScript - flash.....	84
Código 13.- TomarFoto.fla, llamada a takePhoto().....	85
Código 14.- TomarFoto.fla, función takePhoto().....	85
Código 15.- TomarFoto.fla, función onPhoto().....	85
Código 16.- TomarFoto.fla, función fileUpload().....	86
Código 17.- TomarFoto.fla, función onFileUpload() y pasarEstadoAJavascript().....	86
Código 18.- CameraLayout.unit.templete, función pasarEstadoAJavascript.....	87
Código 19.- BuscarResultadolImagen.groovy.....	88
Código 20.- ObtenerIdiomaActual.groovy.	89
Código 21.- Parámetro para grabar sonido mediante la clase NewFileService.....	90
Código 22.- método doPost de la clase UploadFileServlet.....	93
Código 23.- Definición de UploadFileServlet en WEB-INF/web.xml.....	94
Código 24.- Query HQL que obtiene las descripciones e idioma asociadas a un objeto.	98
Código 25.- ObtenerQueryImágenes.groovy.....	101
Código 26.- ObtenerComparacionImágenes.groovy.	102

1. Resumen y Objetivos.

La creciente demanda de teléfonos inteligentes (i.e., smartphones) y de conexiones a Internet móvil, así como la gran aceptación y proliferación que están teniendo las aplicaciones basadas en Realidad Aumentada, hacen que sea un momento atractivo para estudiar las posibilidades de desarrollo combinando este tipo de tecnologías.

La Realidad Aumentada está adquiriendo gran popularidad y algunas aplicaciones de Realidad Aumentada aprovechan ya los sensores disponibles en los móviles inteligentes que permiten capturar el contexto del mundo real. Este tipo de aplicaciones se vienen desarrollando como aplicaciones nativas y a medida, teniendo que desarrollar una solución en el mejor de los casos por plataforma.

Esta dispersión de plataformas y dispositivos se tiende a solventar mediante el uso de aplicaciones Web en numerosos dominios. No obstante, las aplicaciones Web corren sobre un sandbox por lo que se encuentran más limitadas en funcionalidad (e.g., acceso a sensores) que las aplicaciones nativas, lo cual limita el conjunto de aplicaciones móviles que se pueden desarrollar como aplicaciones Web pese a los beneficios multiplataforma de desarrollo que esto conllevaría.

Con esta motivación en este proyecto se estudian en este proyecto las posibilidades de desarrollo para una aplicación móvil de Realidad Aumentada (nativa, Web e híbrida). El principal aporte, más allá de la aplicación en sí, es precisamente ese estudio y la detección de limitaciones actuales de acceso a los sensores de los teléfonos inteligentes desde una aplicación Web. Adicionalmente se estudian distintos algoritmos para el reconocimiento de imágenes y la manera eficiente de procesarla como requisito de la aplicación a implementar.

2. Introducción.

La creciente demanda de teléfonos inteligentes (i.e., smartphones) y de conexiones a Internet móvil, así como la gran aceptación y proliferación que están teniendo las aplicaciones basadas en Realidad Aumentada, hacen que sea un momento atractivo para estudiar las posibilidades de desarrollo combinando este tipo de tecnologías.

La existencia de más de 10 millones de teléfonos inteligentes en España y más de 52 millones en Europa [1], los más de 9 millones de usuarios de Internet móvil [2], junto con la aparición de nuevos tipos de dispositivos móviles como son las tablet PC hacen de este sector un sector de negocio en pleno desarrollo y con gran futuro.

En cuanto al término de Realidad Aumentada, aunque no es nuevo, si está adquiriendo gran popularidad, sobre todo con la aparición de videojuegos. Algunas aplicaciones de Realidad Aumentada aprovechan ya los móviles inteligentes, este auge es en parte debido al amplio conjunto de sensores que incorporan (e.g., cámara, brújula, GPS, etc.) y que permiten capturar el contexto del mundo real.

La dispersión de desarrollo para varias plataformas que se produce actualmente con los dispositivos móviles ya ocurrió en el pasado en entornos de escritorio.

Las aplicaciones Web, permiten que la aplicación no se desarrolle para una única plataforma específica. Además, existe un creciente aumento en el número de herramientas de desarrollo que permiten disminuir los costes de desarrollo. Su limitación tecnológica es que no son capaces de acceder más que a un conjunto mínimo de sensores de entre todos los disponibles en el móvil inteligente, lo cual limita el conjunto de aplicaciones móviles que se pueden desarrollar como aplicaciones Web, con los beneficios que esto conllevaría.

De este modo, las motivaciones principales de este proyecto en relación con el desarrollo de aplicaciones móviles Web de Realidad Aumentada son el estudio de las posibilidades de desarrollo y las limitaciones actuales de acceso a los sensores de los teléfonos inteligentes desde una aplicación Web. Adicionalmente, se estudian distintos algoritmos para el reconocimiento de

imágenes y la manera eficiente de procesarla. Para ello se especifica y desarrolla una aplicación de Realidad Aumentada que permite al usuario mantener un álbum digital donde la información de un edificio de interés se obtiene automáticamente a través de una fotografía mediante técnicas de reconocimiento de imágenes.

El desarrollo de la aplicación se basará en herramientas de desarrollo dirigido por modelos (MDD – Model Driven Development) debido a que este tipo de tecnologías permiten el desarrollo rápido de aplicaciones, un corto periodo de aprendizaje, además de mejorar la abstracción y comprensión del sistema a desarrollar. Todas estas premisas son deseables en el desarrollo de aplicaciones orientadas a Internet y a tecnología móvil, ya que a la velocidad que se mueven ambos mundos es de suma importancia poder convertir las ideas en aplicaciones lo más rápidamente posible, ya que el hecho de que el desarrollo de una nueva idea se pueda alargar varios años, puede suponer un fracaso. En concreto se utilizará WebRatio, herramienta de generación automática de código basada en modelos WebML (Web Modelling Language).

Con estas expectativas se plantea una aplicación que a grandes rasgos responde al diagrama de la figura 1.

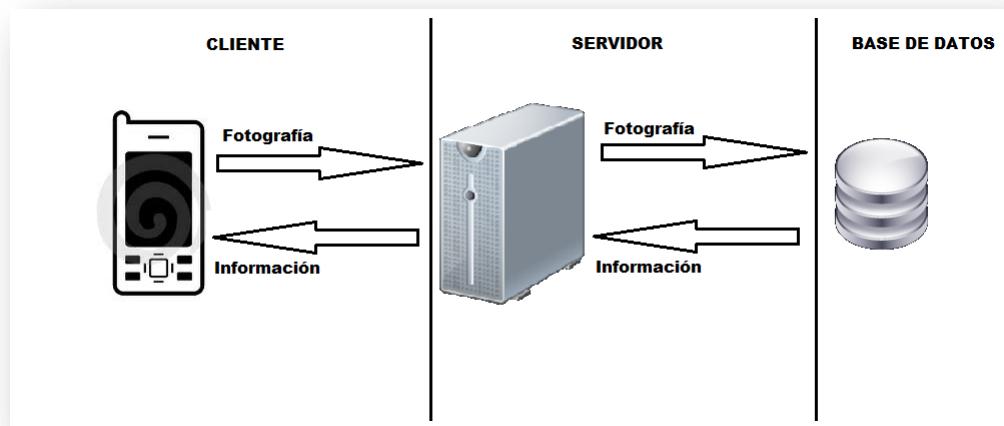


Figura 1.- Diagrama de la Aplicación

De manera resumida podemos presentar la funcionalidad de la aplicación como sigue: desde el móvil, y a través del navegador se obtiene una fotografía de un edificio característico, dicha foto se envía al servidor. Éste realiza la identificación de dicha imagen, en base a algoritmos de reconocimiento de imágenes, sobre imágenes de muestra previamente

almacenadas en una base de datos. La información relativa al edificio identificado llegará de nuevo al navegador del teléfono móvil como información que aumentará la realidad del usuario (un edificio del que no conoce su nombre, historia, etc.).

Podemos distinguir dos partes diferenciadas en base a la figura 1:

1. Cliente: En el cliente veremos la forma de acceder a la cámara de fotos desde el navegador del móvil. La aproximación en este caso, por las limitaciones tecnológicas, es ad-hoc y se ha realizado como ejemplo para móviles con sistema operativo Symbian S60 que soporten flash lite, utilizando una API (Application Programming Interface) de Nokia, llamada APIBrige, que ofrece las herramientas necesarias para la captura de imágenes y audio a través de aplicación nativa (no Web) del móvil, y ofrece un mecanismo para subir al servidor los archivos obtenidos. Esta API ofrece las interfaces necesarias para realizar las operaciones que se mencionan a continuación, aunque no todas ellas son accesibles a través del navegador:

- La captura de vídeo, imagen y audio.
- La lectura de archivos.
- Upload de archivos.
- La creación de la imagen en miniatura.
- Utilizar el servicio de registro.
- Utilizar el servicio de geolocalización.
- Utilizar el servicio de gestión de los medios de comunicación.
- El envío de tonos DTMF durante una llamada activa.

2. Servidor: Donde a su vez distinguimos dos partes:

- a. **Servidor Web:** Para la parte de servidor se utiliza un servidor Web Tomcat, ya que, además de ser de libre distribución y código abierto, soporta servlets, JSPs, y todo lo necesario para contener la aplicación generada a partir del modelo WebML. En el servidor se han creado dos servlets de manera manual, uno que proporciona el servicio para la subida de ficheros desde el cliente y el segundo detecta si el cliente que se conecta es un dispositivo móvil o un ordenador de sobremesa, proporcionando funcionalidades y apariencias de la aplicación distintas en cada caso.

b. Base de Datos: Se utiliza una base de datos PostgreSQL, por ser de libre distribución, de código abierto y por permitir la utilización de procedimientos almacenados. Es mediante este mecanismo, los procedimientos almacenados, como se soporta la búsqueda de imágenes a través de algoritmos de reconocimiento para mejorar su eficiencia. En concreto se ha elegido usar el algoritmo SURF (Speeded Up Robust Features), un detector y descriptor de características locales de la imagen que son invariantes a cambios de escala, rotación y cambios de intensidad por sus buenos resultados, no obstante esta implementación podría ser cambiada por otra sin cambios adicionales en el resto del proyecto.

Para la implementación de los procedimientos almacenados relacionados con SURF se ha utilizando el lenguaje C junto con la librería OpenCV (Open Source Computer Vision), que proporciona funciones para visión por computación en tiempo real.

3. Bases del proyecto.

3.1. Realidad Aumentada.

En la secuencia de realidad-virtualidad de Milgram [3] que define la secuencia que se extiende entre el mundo real y el mundo virtual, la Realidad Aumentada (AR) y la Realidad Virtual (AV) se encuentran en el medio, donde AR está más cerca del mundo real y AV está más cerca de un entorno virtual puro, como se ve en la figura. 2.

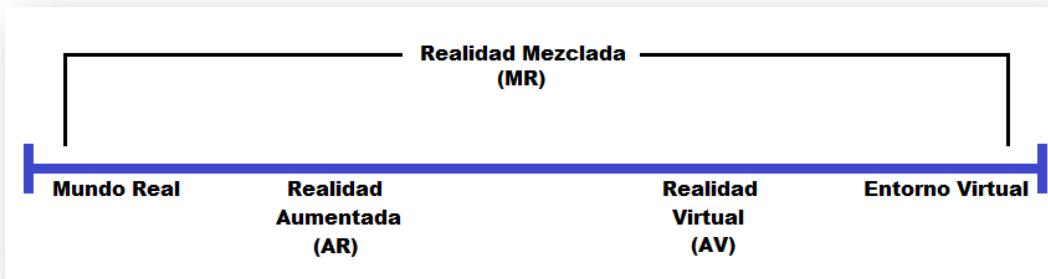


Figura 2.- secuencia de realidad-virtualidad de Milgram.

Se define AR como una visión en tiempo real, directa o indirectamente de un entorno real que se ha mejorado / aumentado añadiéndole información virtual computerizada [4]. AR mejora la percepción del usuario y la interacción con el mundo real, mientras que la realidad virtual envuelve al usuario en una realidad sintética sin contacto con el mundo.

El propósito de la AR es simplificar la vida del usuario ofreciéndole información virtual del entorno que le rodea, no solamente de forma visual, si no que puede ofrecer información a través de cualquiera de los sentido. Hay muchas clases de aplicaciones de AR, tales como la visualización médica, entretenimiento, publicidad, mantenimiento y reparación, planificación de trayectorias de robots, etc.

3.1.1. Actualidad de la Realidad Aumentada.

La principal barrera para el desarrollo de las soluciones de AR, se encuentra en las limitaciones técnicas; mezclar en tiempo real realidad y objetos virtuales tiene unos requisitos iniciales elevados, lo que ha llevado a

que primero se haya desarrollado en entornos profesionales para posteriormente pasar al entorno doméstico.

La AR en entornos profesionales no es algo reciente, y el mejor ejemplo lo podemos tener en los informativos meteorológicos, en la que se mezcla imagen real con virtual, empleando técnicas de AR. La AR en la televisión ha evolucionado y puede considerarse que presenta un nivel de madurez alto. Su uso cada día es más habitual, desde informativos hasta retransmisiones deportivas en las que la imagen real se complementa con datos virtuales (publicidad, distancias, estadísticas, etc.).

Como contrapartida, las aplicaciones de AR en entornos domésticos y personales son mucho más recientes, y debido a eso presentan una elevada capacidad de evolución. En este campo, el uso de la AR no ha hecho más que empezar.

El uso de AR en un escenario estático como puede ser un ordenador personal tiene demasiadas restricciones que minimizan su utilidad siendo uno de los principales campos de aplicación el marketing.

Hasta hace poco las limitaciones técnicas impedían su desarrollo en dispositivos móviles, pero la llegada de terminales personales cada día más potentes, ha supuesto un gran avance en las aplicaciones de AR, y gracias a ellos se ha roto la barrera de la movilidad, impulsando el desarrollo de soluciones y mostrando parte de su potencial.

Aunque existen aplicaciones de AR para móviles que utilizan distintos tipos de sensores, en general las aplicaciones existentes se basan en tres sensores o conjuntos de sensores: la cámara, el GPS y la brújula. Estos sensores permiten, por un lado capturar parte del mundo exterior que percibimos mediante la vista, y por otro posicionar al usuario en su localización exacta y la dirección en la que mira.

De esta forma, tenemos aplicaciones que utilizan el reconocimiento de imágenes para identificar la imagen, como la aplicación Word Lens [5], para traducción simultánea de textos en tiempo real con la cámara del iPhone.

Otras aplicaciones se basan en el uso de la geolocalización (GPS y brújula) como la aplicación Layar [6] que dispone de una librería con más de 1.500 capas virtuales que se pueden superponer sobre el mundo real, y se basan en el cálculo de la posición del usuario respecto a una serie de puntos

de información que se muestran sobre la imagen capturada por la cámara. De la misma forma trabaja Star Walk [7] que funciona como una guía interactiva de astronomía que permite al usuario localizar constelaciones, estrellas, planetas, satélites artificiales, etc.

Todas las aplicaciones para móviles identificadas son aplicaciones nativas que se ejecutan en el dispositivo, no existen aplicaciones Web que exploren el campo de la AR, por lo que estudiaremos la posibilidad de desarrollar una aplicación Web de AR para móviles con la tecnología disponible actual, a través del reconocimiento de imágenes.

3.2. Introducción a Dispositivos Móviles.

3.2.1. Los teléfonos inteligentes y sus sistemas operativos.

Los smartphones, o teléfonos inteligentes, son pequeños dispositivos que integran funcionalidades de teléfono móvil con las funcionalidades tradicionales de una PDA. Una de las características de los teléfonos inteligentes que los hace especialmente atractivos para la Realidad Aumentada, a parte de su movilidad, son la incorporación de sensores que hacen posible la iteración del dispositivo con el mundo que lo rodea, tal como la incorporación de cámaras, GPS, brújula, sensores de movimiento, etc.

A continuación se muestra una comparativa [8] de los principales sistemas operativos para móviles inteligentes en la actualidad:

	iOS	Android	webOS	Windows Mobile	Windows Phone	BlackBerry OS	Symbian	Maemo	bada
Compañía	Apple	Open Handset Alliance (Google)	HP/Palm, Inc	Microsoft	Microsoft	RIM	Symbian Foundation	Nokia	Samsung
Familia de Sistema Operativo	Mac OS X/Unix-like	Linux	Linux	Windows CE 5.2	Windows CE 7	Mobile OS	Mobile OS	Linux	RTOS propietario o Linux
Arquitectura de CPU soportadas	ARM	ARM, MIPS, Power Architecture, x86	ARM	ARM	ARM	ARM	ARM	ARM	ARM
Lenguajes de programación	C, C++, Objective-C	C, C++, Java	C	C++	Varias, .NET (Silverlight/XNA)	Java	C++	C/C++	C++
Licencia	EULA comercial, excepto para los componentes de código abierto	Libre y open source, excepto para la versión 3 Honeycomb	Libre y open source, excepto módulos de código cerrado	Propietario	Propietario	Propietario	Licencia Pública Eclipse	Libre y open source, excepto componentes de código cerrado	Propietario
Gestor de paquetes	iTunes	APK	App Catalog (Official) Preware (3rd party homebrew)	Windows Mobile Device Center/ActiveSync	Zune Software	Blackberry Desktop Manager	Nokia Ovi Suite	dpkg+apt-get	?

Motor de Navegador Web por defecto	Webkit	Webkit	Webkit	Trident	Trident	Webkit	Webkit	Gecko	Webkit
Tienda de Aplicaciones oficial	App Store	Android Market	App Catalog	Windows Marketplace for Mobile	Windows Phone Marketplace	App World	Symbian Horizon, Ovi store	maemo.org, Ovi store	Samsung Apps
Plataformas de SDK Oficial	Mac OS X usando iOS SDK	Linux, Mac OS X y Windows	Linux, Mac OS X y Windows	Windows	Windows	Windows	Windows usando Symbian SDK o Linux, Mac OS X y Windows usando Nokia Qt SDK	GNU/Linux	Windows
Coste extra de desarrollo	Libre (9\$/año por distribuir en App Store)	Libre (25\$ una vez para ofertarla en Android Market)	Libre	Libre	Libre (99\$/año para ofertarla en Windows Phone Marketplace)	?	Libre (1€ una vez para ofrecerla en Ovi Store)	Libre	?

Tabla 1.- Comparativa Sistemas Operativos Móviles. Traducción de [8].

Sistema Operativo iOS.

iOS, también denominado iPhone OS, es un sistema operativo móvil de Apple desarrollado originalmente para el iPhone, siendo después usado en el iPod Touch e iPad. Es un derivado de Mac OS X, que a su vez está basado en Darwin BSD. El iOS tiene 4 capas de abstracción: la capa del núcleo del sistema operativo, la capa de "Servicios Principales", la capa de "Medios de Comunicación" y la capa de "Cocoa Touch". La interfaz de usuario de iOS se basa en el concepto de manipulación mediante gestos multitáctiles. Provee de una interfaz fluida.

Sistema Operativo Android.

Android es un sistema operativo basado en Linux, diseñado originalmente para dispositivos móviles, tales como teléfonos inteligentes y tablets, que actualmente se encuentra en desarrollo para usarse en netbooks y PCs. Fue desarrollado inicialmente por Android Inc., firma comprada por Google en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio.

Android tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos.

Google liberó la mayoría del código de Android bajo la licencia Apache, una licencia libre y de código abierto. Android está escrito en C, C++, Java y XML, y se estructura en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución.

Sistema Operativo WebOS.

WebOS es un sistema operativo multitarea para sistemas embebidos basado en Linux, desarrollado por Palm Inc.

La interfaz gráfica de usuario de WebOS fue diseñada para dispositivos con pantalla táctil. Incluye un conjunto de aplicaciones para el manejo de la información personal y usa tecnologías Web como HTML5, JavaScript y CSS. Palm asegura que el diseño alrededor de estas tecnologías fue intencionado, para evitar así el aprendizaje de un nuevo lenguaje de programación por parte de los desarrolladores.

Sistema Operativo Windows Mobile y Windows Phone.

La plataforma Windows Mobile es una adaptación de Windows CE, adaptada a teléfonos móviles. A partir de la versión 6.5, el sistema operativo deja de llamarse Windows Mobile y pasa a denominarse Windows Phone.

Uno de los atractivos que tienen estas plataformas para el usuario es que, junto al sistema operativo, se incluyen las aplicaciones de ofimática más populares de Microsoft, como son Office Mobile, Internet Explorer y MSN Messenger.

Sistema Operativo BlackBerry OS.

BlackBerry OS es un sistema operativo móvil desarrollado por Research In Motion (RIM) para sus dispositivos BlackBerry.

El SO BlackBerry está claramente orientado a su uso profesional como gestor de correo electrónico y agenda.

Al igual que en el SO Symbian desarrolladores independientes también pueden crear programas para BlackBerry pero en el caso de querer tener acceso a ciertas funcionalidades restringidas necesitan ser firmados digitalmente para poder ser asociados a una cuenta de desarrollador de RIM.

Sistema Operativo Symbian.

Symbian es propiedad de un consorcio de los más importantes fabricantes de teléfonos celulares entre los que se destaca Nokia, Samsung, Siemens, Panasonic, Ericsson entre otros. Es un sistema operativo de 32 bits, está diseñado para los requerimientos específicos de los teléfonos móviles de 2G, 2.5 y 3G.

Symbian OS fue diseñado pensando en las características limitadas de los dispositivos móviles, lo que hace que sea muy robusto y estable. Realiza una eficiente administración de los recursos. Proporciona mecanismos para ahorrar espacio en memoria, contiene rutinas para la planificación eficiente del procesador, dando un uso más racional al hardware y por consiguiente menor consumo de energía. Con Symbian se han desarrollado interfaces como la serie 60 de Nokia. Symbian también ha dado lugar a plataformas propietarias.

Sistema Operativo Maemo.

Maemo es una plataforma de software desarrollada por la comunidad Maemo para teléfonos inteligentes y tablets de Internet. Se basa en la distribución Linux Debian, aunque no es totalmente de código abierto.

Maemo ha sido desarrollado por Nokia con la colaboración de proyectos de código abierto, utiliza el kernel de Linux Debian y GNOME, utilizando muchas de sus librerías y frameworks. Utiliza el gestor de ventanas Matchbox, y Hildon basado en GTK como GUI y marco de aplicaciones.

Sistema Operativo Bada.

Bada, océano o mar en coreano, es un sistema operativo para teléfonos móviles desarrollado por Samsung Electronics. Está diseñado para cubrir tanto los teléfonos smartphones de gama alta como los de gama baja.

Es una plataforma con un núcleo de arquitectura configurable, que permite el uso de cualquiera de los kernel de Linux, o cualquier otro kernel de sistema operativo en tiempo real (RTOS, real-time operating system). El kernel de Linux se utiliza en Smartphones de gama alta, mientras que RTOS se utiliza para los Smartphones más asequibles, debido a un consumo de memoria más pequeño.

Otros Sistemas Operativos.

Además de los Sistemas Operativos citados podemos destacar otros como:

- GridOS: de Fusion Garage, creado a partir del Kernel de Android.
- QNX: de RIM, es un RTOS basado en Unix.
- SHR: proyecto dirigido por la comunidad, basado en GNU/Linux.
- MeeGo: de The Linux Foundation, basado en Linux.
- Palm OS/Garnet OS: de Access Co. Es el antecesor de WebOS.

- Sistemas Operativos basados Linux: Existen muchos proyectos basados en Linux como pueden ser GridOS, B2G, LiMo, Openmoko, etc.

3.3. Desarrollo de Aplicaciones Móviles.

A la hora de afrontar el desarrollo de una aplicación para dispositivos móviles, existen tres alternativas a considerar en cuanto al tipo de aplicación: aplicaciones nativas, aplicaciones Web y aplicaciones híbridas. A continuación se detalla cada una de ellas.

3.3.1. Aplicaciones Nativas.

Las aplicaciones nativas están desarrolladas para la arquitectura propia de cada dispositivo móvil, son imágenes binarias ejecutables que se almacenan directamente en el sistema de ficheros del dispositivo y se ejecutan directamente en el sistema operativo. El desarrollo de este tipo de aplicaciones se realiza a través de la SDK (Software Development Kit - Kit de desarrollo de software) propia de cada plataforma, que en general contiene las herramientas, utilidades, APIs y controles, que permiten sacar el máximo partido a los dispositivos propios de cada plataforma.

Las aplicaciones nativas permiten un acceso completo a todas las características y sensores del dispositivo. Además permiten un alto rendimiento ya que se ejecutan directamente sobre el dispositivo optimizando la utilización del hardware. También se adaptan perfectamente al look & feel del dispositivo, haciendo que la aplicación tenga una apariencia uniforme respecto al propio dispositivo, mediante controles que permiten generar interfaces de usuario comunes a la plataforma.

Como mecanismo de distribución, las aplicaciones nativas permiten su incorporación a las app stores, que son portales de venta de aplicaciones propias de cada plataforma, muy populares en los últimos años, y que son la puerta de entrada al mercado de las aplicaciones móviles. Además esta característica permite la monetización de la propia aplicación, ya que desde estas tiendas se permite la venta de las aplicaciones mediante pago único, o incluso mediante suscripción al servicio que ofrezca dichas aplicaciones.

Sin embargo, las aplicaciones nativas son dependientes de la plataforma para la que está desarrollada, de forma que si se quiere tener una aplicación

funcionando en varias plataformas la única manera es realizar el desarrollo en cada una de ellas, en distintos lenguajes de programación, con distintas SDKs y distintas APIs, normalmente propietarias y no de libre distribución. Este tipo de aplicaciones obliga al desarrollador a conocer a fondo el lenguaje nativo que utiliza la plataforma para la que desarrolla, incluso deberá conocer a fondo la propia plataforma y la forma de acceder a las distintas características del dispositivo para el que se está desarrollando. Además las plataformas móviles están en constante evolución, lo que obliga al desarrollador a estar en continuo aprendizaje con el consecuente coste.

En la figura 3 se muestra la estructura de desarrollo de una aplicación nativa. El desarrollador debe conocer el lenguaje nativo de la plataforma sobre la que va a desarrollar (Objective-C, Java, C, C++, etc.), desarrolla los fuentes de la aplicación, formado por código fuente y los distintos recursos necesarios para la aplicación (Imágenes, sonido, etc.). A continuación se compila y enlaza generando los binarios de la aplicación, que se empaquetan en un fichero instalable en el formato requerido por la plataforma a la que va destinada la aplicación (.sys para Symbian, .apk para Android, .app para iOS, etc.), obteniéndose un paquete distribuible que puede ser subido a la tienda de aplicaciones de la plataforma para su distribución. Normalmente, cada plataforma posee su propia SDK, que facilita todas estas tareas. Esta aproximación es similar para todas las plataformas, produciendo grandes costes de desarrollo y mantenimiento.

Para el empaquetado, cada plataforma dispone de su Sistema de Gestión de Paquetes, que consiste en una colección de herramientas de software para automatizar el proceso de instalación, actualización, configuración y eliminación de paquetes de software para el sistema operativo de una manera consistente.

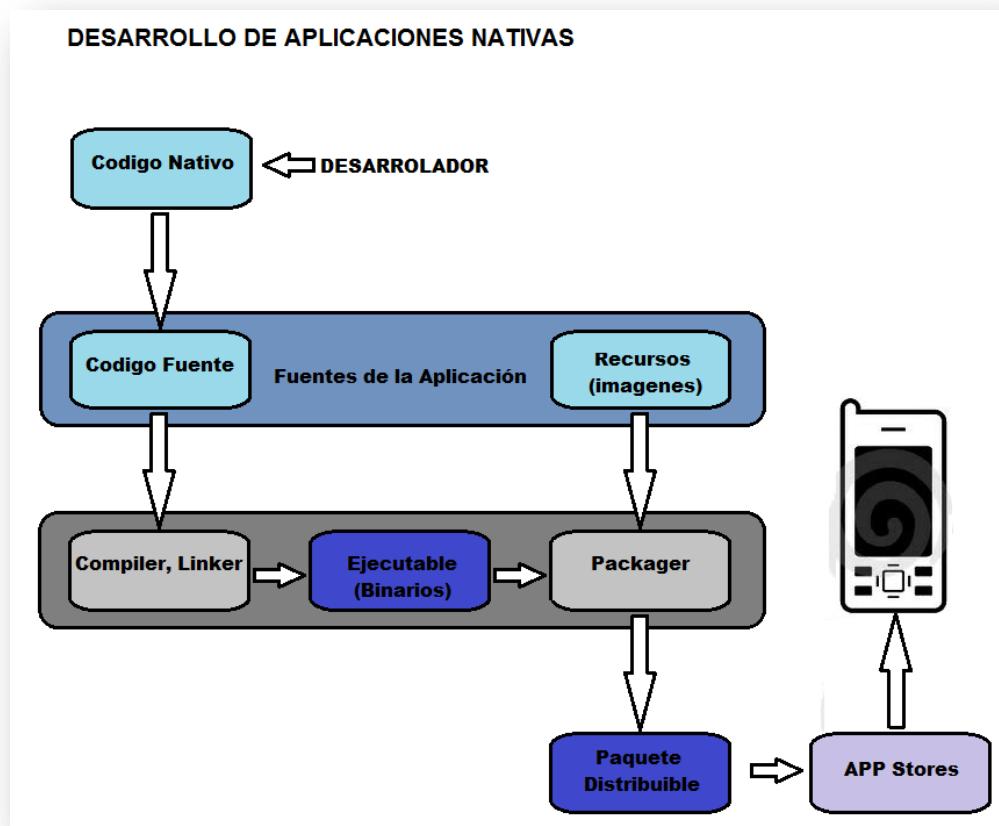


Figura 3.- Sistemas Operativos Móviles. Traducción de [9].

3.3.2. Aplicaciones Web.

Las aplicaciones Web son aquellas realizadas mediante tecnologías Web, como HTML, JavaScript, CSS, Java, etc. atendiendo a una arquitectura Web Cliente/Servidor. Son independientes del hardware y del Sistema Operativo del dispositivo, ya que se ejecutan a través del navegador y no en el Sistema Operativo, lo que elimina el problema de la fragmentación de plataformas. Además, el hecho de utilizar tecnologías ampliamente extendidas (la mayoría de ellas estándar), reduce los costes de aprendizaje e implementación respecto a las aplicaciones nativas que requieren distintos lenguajes, APIs, etc.

Sin embargo, las aplicaciones Web, al ejecutarse sobre un navegador, están sujetas a las limitaciones de recursos del propio navegador, de forma que no se tendrá un acceso ilimitado a los recursos de la plataforma sobre la que se ejecuta, por ejemplo no se podrá tener acceso al GPS del dispositivo móvil, si el navegador no soporta esta opción. De hecho, las aplicaciones Web para móviles están restringidas en su acceso a características tales como la cámara,

el micrófono, libreta de direcciones, y así sucesivamente. Si bien hay trabajos en curso con la incorporación de HTML5 para permitir el acceso a tales sensores, actualmente esta parte de la especificación se ha eliminado del borrador. No obstante, podemos asegurar que en un futuro a medio-largo plazo se permitirá la especificación de aplicaciones Web innovadoras en campos como el de la realidad aumentada que hagan uso de sensores mediante estándares de desarrollo Web.

Otro hándicap es que las aplicaciones Web no tienen cabida en las app stores de las distintas plataformas, aunque sí disponen de los medios de difusión y distribución a través de buscadores e incluso a través de Web stores como Chrome Web store.

La figura 4 muestra la estructura de desarrollo de una aplicación Web. El desarrollador implementa la aplicación utilizando lenguajes y tecnologías Web, despliega dicha aplicación en un servidor Web, de forma que queda disponible para que distintos dispositivos móviles accedan a través de sus navegadores, siendo la aplicación independiente del dispositivo que accede a ella.



Figura 4.- Desarrollo de aplicaciones Web.

3.3.3. Aplicaciones Híbridas.

Aplicaciones híbridas son aquellas aplicaciones nativas con una aplicación Web embebida, de forma que porciones específicas de la aplicación están desarrolladas usando tecnologías Web. Estas partes Web pueden ser descargadas desde Internet o empaquetadas directamente dentro de la aplicación y usadas offline. Para el usuario, una aplicación híbrida es casi indistinguible de una nativa en cierto sentido, ya que se descarga desde las app stores, se almacena en el dispositivo, y se ejecuta igual que cualquier otra aplicación. A nivel gráfico puede notarse que el look-and-feel es diferente,

aunque con esfuerzo se puede eliminar esta sensación. Sin embargo, para los desarrolladores hay una gran diferencia, ya que en lugar de reescribir la aplicación a partir de cero para cada sistema operativo móvil, parte del código de aplicación se escribe en HTML, CSS y JavaScript y es reutilizable para todos los dispositivos.

El desarrollo de este tipo de aplicaciones se pueden realizar a medida, pero lo habitual es usar frameworks que dan soporte a varias plataformas, de forma que permiten el desarrollo utilizando tecnologías Web (HTML, CSS, JavaScript, etc.), que mediante una o varias APIs, recubren el acceso a las funcionalidades nativas y proporcionan una interfaz de usuario uniforme.

Phonegap [10] es un ejemplo de framework que permite el desarrollo de aplicaciones hibridas. Se trata de un framework basado en estándares de código abierto para la construcción de aplicaciones en plataformas móviles con HTML, CSS y JavaScript para iPhone / iPad, Android, Palm, Symbian, BlackBerry, Windows Mobile y Bada. PhoneGap pone a disposición del desarrollador una serie de interfaces de funciones JavaScript que permiten acceder a las características nativas de cada plataforma de forma transparente y uniforme.

Otros frameworks de este tipo son:

- **Rhomobile Rhodes** [11]: Rhodes es un framework modelo-vista-controlador escrito en Ruby. Rhodes toma el código de Ruby y lo compila en código nativo para cualquier dispositivo. Esto elimina la necesidad de escribir código en el lenguaje nativo destino. Disponible para iPhone, Windows Mobile, RIM, Symbian y Android.
- **Appcelerator Titanium** [12]: al igual que PhoneGap, posee una librería de funciones JavaScript y JSON que eliminan la necesidad de conocer el lenguaje nativo de la plataforma en la que se va a desarrollar la aplicación, actualmente Appcelerator soporta Android y iOS.
- **Worklight** [13]: proporciona una plataforma de desarrollo de aplicaciones móviles abierta, completa y avanzada así como herramientas de software para smartphones y tablets. Permite el

desarrollo tanto de aplicaciones nativas como aplicaciones híbridas. Soporta iPhone, Windows Mobile, Android y BlackBerry.

En la figura 5 se muestra la estructura de desarrollo de una aplicación Híbrida. El desarrollador implementa parte de la aplicación de forma similar a las aplicaciones nativas, creando código fuente en el lenguaje nativo de la plataforma en la que se está desarrollando, compilándolo, enlazándolo y empaquetándolo para generar un paquete distribuible a través de la tienda de aplicaciones de la plataforma. A su vez desarrolla parte de la aplicación utilizando tecnologías Web, que será empaquetada junto a la parte nativa, o desplegada en un servidor Web y descargada al ejecutarse la aplicación. La parte Web de la aplicación es reutilizable para distintas plataformas móviles. La parte nativa no será reutilizable de unas plataformas a otras, sin embargo, se pueden utilizar frameworks que ofrecen soporte a distintas funcionalidades nativas a través de interfaces accesibles desde la parte Web, haciendo que la adaptación de la aplicación de unas plataformas a otras sea más sencilla que en las aplicaciones puramente nativas.

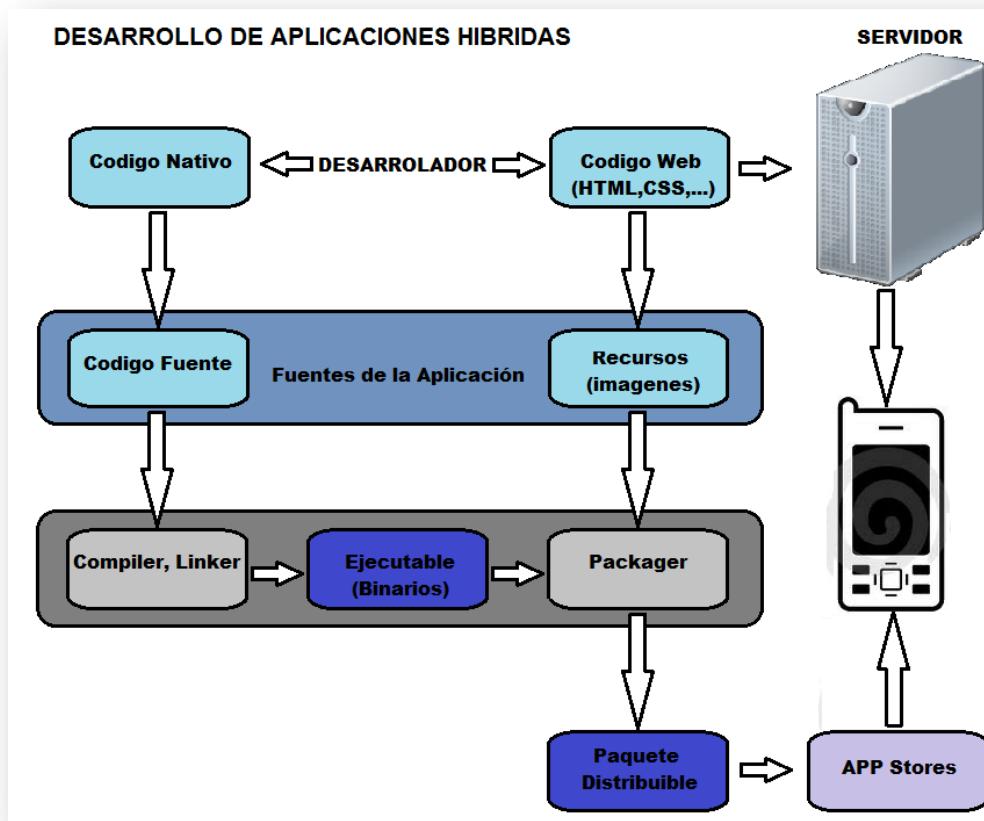


Figura 5.- Desarrollo de aplicaciones Híbridas. Traducción de [9].

3.3.4. Comparativa Aplicaciones Nativas, Web e Híbridas.

La siguiente tabla muestra una comparativa de los tres tipos de aplicaciones respecto al desarrollo:

	Aplicaciones Nativas	Aplicaciones Web	Aplicaciones Híbridas
Plataformas			
Plataformas	<p>Diferentes plataformas móviles:</p> <ul style="list-style-type: none"> • iPhone OS (iOS). • LiMo Platform. • Windows Mobile. • Android (fragmentada por versiones: teléfonos, tablets,...). • Palm OS. • WebOS. • BlackBerry OS. • Symbian. • Bada. 	<p>Navegadores Web Móviles. Aunque la fragmentación de plataformas se minimiza, siguen existiendo diferencias debido a las diferentes versiones de navegadores con diferentes soportes.</p>	<p>Diferentes plataformas de despliegue (como en las nativas) pero sin problemas con diferencias entre navegadores.</p>
Lenguajes			
Lenguajes	<p>Diferentes lenguajes para cada plataforma:</p> <ul style="list-style-type: none"> • Objective-C. • Java. • C/C++. • Visual C#/C++. 	<p>HTML, CSS, JavaScript, Java, Flash, Flex, etc.</p>	<p>Tanto lenguajes nativos como lenguajes web.</p>
Distribución, monetización y soporte			
Distribución	<p>A través de las tiendas de aplicaciones de las plataformas.</p>	<ul style="list-style-type: none"> • Al igual que aplicaciones de escritorio a través de buscadores, URL's, etc. • A través de tiendas Web, como Chrome Web Store. 	<p>A través de las tiendas de aplicaciones de las plataformas.</p>
Aprobación	<p>La aplicación se publica a través de una tienda de aplicaciones y pasa a través de proceso de revisión antes de ser aprobados.</p>	<p>No necesita proceso de aprobación.</p>	<p>La aplicación se publica a través de una tienda de aplicaciones y pasa a través de proceso de revisión antes de ser aprobados.</p>
Instalación	<p>Descarga a través de las tiendas de aplicaciones e</p>	<p>Abriendo la URL en el navegador, o a través de</p>	<p>Descarga a través de las tiendas de aplicaciones e</p>

	instalado en el dispositivo.	un acceso directo en el dispositivo.	instalado en el dispositivo.
Actualización	La aplicación actualizada pasa por un proceso de revisión, a continuación, una nueva descarga e instalación.	Actualizando la aplicación Web todos los usuarios tendrán acceso inmediato a la nueva versión.	La aplicación actualizada pasa por un proceso de revisión, a continuación, una nueva descargar e instalación.
Soporte y mantenimiento	Contra más plataformas soporten la aplicación más complicado será el soporte y el mantenimiento.	Más sencillo, ya que sólo se realiza una vez y está disponible para todas las plataformas.	Más sencillo, ya que sólo se realiza una vez y está disponible para todas las plataformas.
Monetización	Las tiendas de aplicaciones son una probada estrategia de monetización.	Principalmente: <ul style="list-style-type: none"> • Estrategia de monetización propia (e.g., publicidad, suscripción,...). • A través de las tiendas como Chrome Web Store. 	Las tiendas de aplicaciones son una probada estrategia de monetización.
Portabilidad	Obliga el aprendizaje y desarrollo para cada plataforma donde se quiere que funcione la aplicación.	Se construye una vez y se ejecuta en cualquier lugar. Aunque debido a diferencias entre navegadores es posible que se tenga que realizar pequeñas modificaciones sobre la interfaz de usuario.	Al igual que en las aplicaciones Web sólo se construye una vez, aunque el entorno de desarrollo utilizado deberá soportar la plataforma sobre la que se quiere utilizar la aplicación.
Experiencia			
Rendimiento	Más rápido, sobre todo si son necesarios procesos gráficos pesados.	HTML5 mejora la infraestructura de la Web y permite aplicaciones más rápidas y más funcionales. Los motores de JavaScript son cada vez más rápidos y suficientemente buenos para la mayoría de las aplicaciones Web.	Las partes de la aplicación en las que el rendimiento sea crítico se puede desarrollar como nativo.
Experiencia de usuario	Las aplicaciones nativas permiten gran cantidad de efectos en la interfaz de usuario, lo que las hace muy atractivas.	Puede llegar a ser muy bueno, además se puede ir mejorando una vez implantada la aplicación.	Se puede utilizar lo mejor de ambos tipos.
Acceso a las funciones del dispositivo			

Audio/Video	Incorporado.	Disponible dependiendo del navegador.	Incorporado.
Acceso offline	Las aplicaciones nativas pueden funcionar sin conexión.	El modo offline se puede lograr con HTML5.	Las aplicaciones Web pueden funcionar offline.
Modo de pantalla completa	Incorporado.	Disponible si lo soporta el navegador.	Incorporado.
Integración con servicios del dispositivo	Una buena integración con los servicios del dispositivo e.g.,: <ul style="list-style-type: none">• Contactos.• Calendario.• Otras aplicaciones.	Limitado.	Una buena integración con los servicios del dispositivo.
Integración con el hardware del dispositivo	Completa integración con el hardware del teléfono: <ul style="list-style-type: none">• Cámara / Video.• GPS.• Brújula.• Acelerómetro.• Otros sensores y dispositivos.	Algún soporte, como la cámara y el GPS a través de HTML5 o W3C, sólo en navegadores que lo soporten.	Buena integración con el hardware del teléfono.
Desarrollo.			
Habilidades	Es necesario el aprendizaje de una o más plataformas móviles y sus lenguajes de programación subyacentes y SDKs. Cada plataforma también tiene su propio enfoque y el estilo de desarrollo y diseño de interfaces de usuario móviles.	Los desarrolladores pueden utilizar HTML, JavaScript y CSS para crear aplicaciones Web móviles sin aprendizaje de nuevos idiomas al código de las aplicaciones nativas. Pero, eso no significa que sea necesario algún tipo de formación, debido a la necesidad de adaptar el interfaz de usuario a los distintos dispositivos.	Además de la tecnología Web es necesario el aprendizaje de algunos de los frameworks que permiten la generación de aplicaciones híbridas, como Phonegap.
Coste de desarrollo	Caro, siendo relativamente pequeño el número de desarrolladores que dominan estas habilidades.	Puede ser significativamente más barato. Gran número de desarrolladores poseen habilidades de desarrollo Web. Sin embargo, es necesario algún tipo de formación para aprender a	Se reduce considerablemente el coste respecto a las aplicaciones nativas.

		desarrollar la interfaz de usuario para aplicaciones móviles.	
Frameworks de desarrollo	Cada plataforma móvil tiene su propio SDK.	Gran número de opciones permiten el desarrollo de aplicaciones Web.	Distintos frameworks permite el desarrollo de aplicaciones híbridas.

Tabla 2.- Comparativa de aplicaciones nativas, Web e hibridas.

Se puede ver en la tabla que cada tipo de aplicación tiene sus ventajas e inconvenientes. Esta tabla se basa en la original de [14] y ha sido ampliada para incluir lenguajes y la columna de híbridas en base a las lecturas citadas en este documento.

El principal hándicap de las aplicaciones nativas es el alto coste de desarrollo, el desarrollo de una aplicación para varias plataformas implica varios desarrollos para cada una de ellas, además en distintos lenguajes de programación y utilizando distintas APIs, incluso muchas veces el desarrollo de una aplicación para una plataforma no implica su funcionamiento en otras versiones de la misma plataforma.

El coste de desarrollo, aunque menor, es mayor para las aplicaciones híbridas respecto a las aplicaciones web.

Respecto a las aplicaciones Web, ofrecen menor coste de desarrollo, ya que se utilizan tecnologías ampliamente conocidas, además de no ser dependiente de ninguna plataforma. Otro punto a favor de las aplicaciones web, es que en general necesitan menos requerimientos hardware que otro tipo de aplicaciones, ya que la mayoría de procesos pesados se realiza en el servidor. Proporcionan la ubicuidad que ofrece internet, estando la aplicación disponible en cualquier lugar, en cualquier momento. Sin embargo, este tipo de aplicaciones no puede sacar todo el partido de los dispositivos móviles ya que se dispone de acceso restringido a las funcionalidades de los mismos (cámara, acelerómetro, GPS, etc.), aunque en un futuro próximo si será posible la disponibilidad de acceso a estas funcionalidades.

3.3.5. Limitaciones de RIAs para aplicaciones de Realidad Aumentada.

El objetivo de las RIAs (Rich Internet Applications) es el de proveer IUs y funcionalidad en aplicaciones Web que permitan aproximar las características de éstas a las de las aplicaciones tradicionales de escritorio.

Actualmente se pueden desarrollar aplicaciones Web altamente interactivas, pudiéndose generar interfaces de usuario muy atractivos. En general esto es así también para aplicaciones Web móviles, ya que los navegadores Web soportan tecnologías tales como AJAX, Adobe Flash lite, Flex, OpenLaszlo, etc.

El principal problema que existe es que en la actualidad es que no todas las plataformas ni navegadores móviles soportan todas las tecnologías citadas, por ejemplo, si se incluye cierto componente en una página Web utilizando flash, funcionará correctamente en el navegador Web para S60 pero no en Opera Mobile, sin embargo Opera Mobile tiene soporte para geolocalización que no está disponible en el navegador para S60. Esto implica que aplicaciones que necesiten de este tipo de componentes, pueda funcionar en unos navegadores y en otros no, manteniéndose la fragmentación de plataformas.

En cuanto a aplicaciones de Realidad Aumentada, es una necesidad el acceso al contexto exterior en el que se sitúa el usuario, de forma que la aplicación pueda ofrecer información sobre esa realidad exterior. La única forma de conseguir esto es a través de los distintos sensores y los distintos dispositivos hardware que incluyen los teléfonos inteligentes. Esto también se convierte en un problema dada la dificultad de acceso a estos dispositivos mediante tecnologías Web.

A continuación se numeran los dispositivos que normalmente incluyen los actuales teléfonos inteligentes y que pueden ofrecer información contextual del mundo real:

- **Cámara fotográfica:** fundamental en Realidad Aumentada, ya que permite mostrar en la pantalla del dispositivo móvil la realidad exterior, generando la sensación de que la información adicional se superpone con el mundo real. En cuanto al acceso a este dispositivo

desde una aplicación Web está realmente limitado, ya que actualmente se puede acceder a la aplicación nativa de cada dispositivo a través de API's como ApiBridge de Nokia, PhoneGap y similares, obtener una fotografía o video y retornar el fichero a la aplicación para su tratamiento. Lo que aunque efectivo en cuanto a poder obtener información sobre una imagen determinada, no proporciona la misma sensación de Realidad Aumentada que pueden proporcionar las aplicaciones nativas, que muestra la información directamente sobre la imagen de la cámara en tiempo real.

En el caso del actual proyecto la imagen obtenida a través de la cámara es la que tiene todo el peso para la obtención de la información adicional, ya que se utiliza el reconocimiento de imágenes para obtener dicha información.

- **GPS, brújula y acelerómetro:** El acelerómetro detecta el movimiento 3D a lo largo de los ejes x, y y z, midiendo el movimiento, vibraciones e inclinación a las que está sometido el dispositivo. La brújula detecta la dirección o rumbo hacia el que señala el dispositivo, midiendo el rumbo en grados de 0 a 359,99, siendo 0 el norte. El GPS proporciona información sobre la ubicación del dispositivo, tales como latitud y longitud. Mediante estos tres dispositivos, podemos conocer la posición del usuario mediante el GPS, la dirección hacia la que está mirando mediante la brújula y la inclinación del teléfono gracias al acelerómetro, además el GPS nos proporciona información de la velocidad a la que se mueve el usuario y información sobre la altitud a la que se encuentra, el acelerómetro también nos informa de la velocidad con la que se mueve el dispositivo móvil. Con estos tres dispositivos se puede conseguir un motor de Realidad Aumentada completo, ya que posiciona completamente al usuario y indica hacia donde está mirando con la cámara, con lo que podemos obtener información de todos los puntos de interés que lo circundan y la distancia que lo separa de ellos de forma que es relativamente fácil representarlos gráficamente sobre la imagen que se está obteniendo a través de la cámara. Este mecanismo es el más utilizado en la actualidad, ya que es

relativamente sencillo realizar el tratamiento de la información obtenida a través de estos tres elementos y mostrar la información sobre la imagen de la cámara en tiempo real consigiéndose una gran sensación de Realidad Aumentada. Sin embargo el acceso a estos tres elementos desde una aplicación Web es complicado en la actualidad, y aunque existen navegadores que soportan el acceso a los datos de localización mediante el GPS, no soportan el acceso al acelerómetro ni a la brújula. Por tanto la opción de utilizar estos tres componentes no es factible de momento, aunque será cuestión de tiempo su disponibilidad para aplicaciones Web, ya que serán soportados por tecnologías HTML5.

- **Micrófonos:** permite recoger el ruido que rodea al dispositivo móvil, lo que permitiría crear aplicaciones de realidad Web para el reconocimiento de sonidos. El acceso al micrófono es factible en aplicaciones Web, aunque limitado y similar a la cámara fotográfica, ya que lo que se ha conseguido es llamar a la grabadora de sonidos nativa que retorna un fichero que se puede tratar desde la aplicación.
- **Altavoces:** como la Realidad Aumentada no tiene por qué sólo afectar a la vista, sino que la información puede ser transmitida a cualquier sentido, el audio también puede ser un medio importante para transmitir la información aumentada. La reproducción de sonidos desde una aplicación Web es totalmente posible.
- **Vibración:** los móviles incorporan un vibrador que permiten emitir señales que son perceptibles por el tacto del usuario.

La conclusión es que actualmente es difícil poder crear una interfaz Web que permita acceder a las funcionalidades mencionadas del dispositivo móvil, por lo que es complicado realizar una aplicación Web de Realidad Aumentada que permita una interacción con el mundo en tiempo real.

Como se ha visto anteriormente, existen frameworks como Phonegap que ofrecen una interfaz JavaScript que da soporte a distintas funcionalidades de los dispositivos móviles, pero este soporte funciona para aplicaciones híbridas instaladas directamente en los móviles, pero no funcionan correctamente desde aplicaciones Web ejecutadas en un navegador.

Con el objetivo de desarrollar la aplicación como aplicación Web por las ventajas que conllevaría, se nos ha hecho necesario simular que un navegador Web o servicio del móvil proporcionaría acceso a los dispositivos y sensores de este desde el propio navegador. Para ello, en este proyecto se ha obtenido acceso a la aplicación nativa de la cámara y obtener un fichero de imagen para su tratamiento a través de dos componentes, la API de Nokia ApiBridge [15] y Flash Lite [16] soportado por el navegador Web S60, por lo que aunque el resto de la aplicación es soportado sobre cualquier navegador que soporte AJAX, la parte de captura de datos desde la cámara sólo es soportado por dispositivos con Symbian S60. De la misma forma que se realiza el acceso a la cámara, en la aplicación, también se accede a la grabadora nativa de sonidos, a través de Nokia ApiBridge y Flash Lite.

ApiBridge ofrece una interfaz JavaScript para la creación de aplicaciones hibridas ofreciendo los siguientes servicios:

- Carga de archivos a un servidor.
- Captura de imágenes, video y audio.
- Lectura de ficheros locales.
- Redimensión de imágenes.
- Creación de imágenes en miniatura.
- Uso del servicio de log de llamadas del dispositivo.
- Uso del servicio de localización.
- Utilizar el servicio de gestión de ficheros.
- Envió de tonos DTMF en una llamada activa.

Aunque esta API es JavaScript, no es utilizable desde el navegador y solamente es accesible desde aplicaciones hibridas instaladas en el propio dispositivo, lo que Nokia llama WRT Widgets, que son pequeñas aplicaciones que se ejecutan sobre Web RunTime (WRT) sin necesidad de un navegador.

Sin embargo, ApiBridge tiene una interfaz para ActionScript, utilizable desde Flash Lite, dando soporte a los mismos servicios que la interfaz de JavaScript, pero que es utilizable desde el navegador, por lo que es factible su utilización en aplicaciones Web, siendo ésta la opción utilizada en este proyecto. A pesar de todo, no todos los servicios son accesibles desde la Web, por ejemplo el servicio de localización no funciona correctamente.

En definitiva, aunque en un futuro próximo, con implantación total de HTML5, se podrá acceder a la mayoría de las funcionalidades de los teléfonos inteligentes, el hecho es que en la actualidad este acceso es complicado, algunos navegadores soportan la localización mediante el estándar del W3C, como es el caso de Opera Mobile, navegador Web disponible para Android y Symbian, siendo este soporte dependiente de la plataforma también, ya que lo soporta para Symbian pero no para Android, mientras soporta flash para Android y no para Symbian. Por lo que en general, aunque para gran número de aplicaciones, la tecnología Web permite evitar el problema de la fragmentación de plataformas, en el caso de aplicaciones que necesiten hacer uso las funcionalidades mencionadas, la fragmentación de plataformas sigue siendo un problema, acentuado por el amplio número de navegadores y versiones de estos que existen.

3.4. Desarrollo dirigido por modelos.

El Desarrollo Dirigido por Modelos (MDD) es un enfoque que afronta el desarrollo de software con un nivel más alto de abstracción, mediante el uso de modelos como principal método de desarrollo. Utiliza los modelos como especificación de software y transformaciones de los modelos para obtener el código fuente. Los modelos permiten un alto nivel de abstracción separando responsabilidades entre distintos modelos (e.g., modelo de datos, de presentación,...). El modelado permite a los desarrolladores concentrarse en aspectos diferentes a la implementación como pueden ser la funcionalidad, la usabilidad o la optimización. Además dado que gran parte del código se genera a partir de dichos modelos aumenta la eficiencia de los procesos generados y se reducen los defectos introducidos en las etapas manuales del desarrollo.

El actual proyecto utiliza WebRatio [17] como entorno de desarrollo MDD, debido a que diversos estudios (e.g., [18]) apuntan a él como un entorno apropiado para el desarrollo de aplicaciones Web avanzadas. WebRatio permite la especificación de modelos WebML [19] y la generación de código fuente libre desde estos.

WebML proporciona un modelo conceptual independiente de plataforma para sitios Web. Es un tema de investigación activo, con ampliaciones propuestas para una serie de conceptos tales como servicios Web, modelado

de procesos, excepciones y conocimiento de contexto. WebML se compone de cuatro modelos principales que se resumen a continuación:

1. **Modelo estructural:** expresa la estructura de los datos del sitio Web, en términos de entidades y las relaciones para el dominio de la aplicación. WebML propone un lenguaje compatible con notaciones clásicas como el modelo E/R, el modelo ODMG orientado a objetos, y diagramas de clases UML.
2. **Modelo Hipertexto:** describe uno o más hipertextos que pueden ser publicados en el sitio. Cada hipertexto se denomina vista del sitio (site view). La descripción de cada vista, a su vez consta de dos submodelos:
 - a. **Modelo de composición:** especifica las páginas que componen la vista, y las unidades contenidas en una página.
 - b. **Navegación Modelo:** expresa la forma en se enlazan las páginas del hipertexto y las unidades contenidas en las páginas. Los enlaces pueden ser no-contextuales, cuando conectan páginas semánticamente independientes, o contextuales, cuando el contenido de la unidad destino depende del contenido de la entidad origen.
3. **Modelo de Presentación:** permite expresar el aspecto gráfico de las páginas, independientemente del dispositivo y el lenguaje de salida, expresado mediante sintaxis XML. Las especificaciones de presentación permiten representar páginas específicas o genéricas.
4. **Modelo de Personalización:** permite modelar explícitamente usuarios y grupos de usuarios en forma de entidades predefinidas llamadas usuarios y grupos. Las características de estas entidades se pueden utilizar para representar contenidos individuales o de grupos.

Mediante los diagramas WebML permite expresar de manera visual todos los requisitos de la aplicación Web. Además WebRatio ofrece las siguientes características:

- **Generación de código automático:** WebRatio genera automáticamente aplicaciones Web que cumplen con el estándar

Java/JSP 2.0 y se pueden instalar en cualquier servidor de aplicaciones, incluyendo:

- Apache Tomcat
- JBoss
- Caucho Resin
- Oracle WebLogic Application Server
- IBM WebSphere

- **Independencia del sistema de gestión de bases de datos:** las aplicaciones Web construidas con WebRatio utilizan las librerías de Hibernate , por lo que pueden conectarse a cualquier sistema de gestión de base de datos, incluyendo:
 - PostgreSQL
 - MySQL
 - Oracle 8i/9i/10g/11g
 - IBM DB2
 - Microsoft SQL Server 2000/2005/2008
 - Apache Derby
 - Cualquier otro producto que soporte el estándar JDBC, para el cual esté disponible un conector de Hibernate.
- **Soporte completo de SOA y Servicios Web:** WebRatio permite publicar, consumir, y organizar servicios Web.
- **Interoperabilidad con sistemas heredados:** mediante el desarrollo e integración de componentes personalizados, las aplicaciones creadas con WebRatio pueden conectarse con cualquier sistema heredado.
- **Seguridad, autenticación, perfiles:** mediante la utilizan el protocolo SSL permite la protección de datos, además permite administrar la autenticación de usuarios mediante la integración de los sistemas de gestión de identidad más populares, como LDAP y Active Directory de MS. El modelo de aplicación puede utilizarse para representar el perfil del usuario, haciendo que la aplicación sea completamente personalizable.

- **Aplicaciones multilenguaje:** las aplicaciones multilenguaje están totalmente soportadas. El editor del modelo de aplicación permite definir una traducción para cada etiqueta y en un formato adoptado a cada tipo de datos, en todos los idiomas deseados.
- **Interfaz e interacción enriquecidas:** las reglas de generación de código pueden incorporar cualquier plantilla gráfica: HTML, CCS, AJAX, JavaScript y también las interfaces de aplicaciones de Internet enriquecidas.
- **Alto rendimiento:** las reglas de generación producen código altamente optimizado, que cumple con los estándares requeridos por aplicaciones de misión crítica. Si es necesario, optimizaciones de postproducción pueden aplicarse al código generado y también ser incorporadas en las reglas de generación para que se apliquen a todas las versiones posteriores.
- **Trabajo colaborativo y control de versiones:** todos los archivos fuente de un proyecto se pueden compartir a través de un servidor de trabajo colaborativo y de control de versiones (CVS o Subversion).
- **Código abierto y librerías estándar:** las aplicaciones Web creadas con WebRatio emplean las librerías Java más populares, incluyendo Hibernate, Struts, JSTL, JSP y Java servlet. El código generado es completamente abierto y no depende de componentes propietarios. Las aplicaciones pueden mantenerse manualmente, como cualquier aplicación de Java.

WebRatio, al ser un entorno de desarrollo dirigido por modelos, permite optimizar el tiempo de desarrollo, así como minimizar errores de codificación gracias a su generación de código automática. Además permite la generación rápida de prototipos de aplicación totalmente funcionales, que permiten la implementación, mantenimiento y evolución de forma sencilla y rápida.

La generación de una aplicación mediante WebRatio se realiza en tres pasos:

1. **Construcción del modelo:** Los requisitos de la aplicación se generan, como se ha comentado utilizando el lenguaje de modelado WebML, describiendo visualmente:

- La lógica de la extracción de datos y cómo se utilizan los datos para componer la interfaz de usuario.
- La integración con componentes y servicios Web.
- La navegación a través de las páginas y los comandos de interacción

2. **Personalización de reglas:** WebRatio utiliza reglas de generación de código que son totalmente adaptables y extensibles. WebRatio se puede ampliar para:

- Definir estilos de presentación: La identidad visual puede ser controlada mediante reglas de generación de código recogidas en un estilo de presentación. El estilo de presentación y el modelo de aplicación son independientes y por tanto, es posible generar una aplicación Web con estilos completamente diferentes con el mismo modelo o aplicar el mismo estilo de presentación para generar muchas aplicaciones.
- Agregar nuevos componentes: gran cantidad de componentes están predefinidos en el entorno de desarrollo, pero se pueden definir e integrar componentes propios para obtener cualquier comportamiento deseado. Los componentes están codificados en Java y deben seguir la estructura de componentes especificada por WebRatio.

3. **Generación de la aplicación:** WebRatio genera aplicaciones estándar Java, sin entorno de ejecución ni componentes propietarios, de forma que el código producido se puede ejecutar en cualquier servidor de aplicaciones Java. Además, la aplicación puede generarse en cualquier momento del ciclo de vida de desarrollo. Se puede obtener un prototipo realista totalmente funcional sin ningún costo, para verificar el cumplimiento de los requisitos más rápidamente. Esta capacidad crea un círculo de análisis, modelización, generación y validación, que permite converger rápidamente hacia la solución final.

4. Real-A: Aproximación RIA a la Realidad Aumentada móvil usando MDD.

4.1. Introducción.

Real-A es para este proyecto una aplicación que permite explotar las posibilidades de la Realidad Aumentada en móviles utilizando tecnologías Web y utilizando MDD (en concreto WebRatio).

La aplicación nos ha permitido explorar las limitaciones existentes actualmente debidas sobre todo a la fragmentación de plataformas móviles, y la dificultad de acceso a los sensores y dispositivos de los propios móviles inteligentes, como puede ser la cámara de fotos, el GPS, etc. fundamentales en este tipo de aplicaciones, dada la necesidad de acceder al contexto que rodea al usuario en cada momento, y poder así proporcionarle información añadida del mundo que le rodea.

Real-A es una aplicación completa, que permite gestionar dos tipos de roles, *usuarios* y *administradores*, con accesos a dos partes bien diferenciadas de la aplicación, una para cada tipo de usuario, además de una parte de acceso público, constituyendo tres siteviews en WebRatio como sigue:

- **Zona de acceso Público:** La zona de acceso público gestiona la entrada a la aplicación, mediante una pantalla de login. También gestiona la creación de nuevos usuarios, y la recuperación de contraseñas para usuarios que la hayan olvidado, ya que cada usuario deberá acceder a la aplicación mediante un nombre de usuario, y una contraseña.

Para crear un usuario nuevo, se deberá introducir el nombre de usuario y una cuenta de correo, donde la aplicación envía una contraseña generada, que permitirá al menos el primer acceso a la aplicación, sirviendo además como manera de controlar que la cuenta de correo introducida es válida, ya que si se proporciona una cuenta de correo errónea o falsa, el usuario no podrá tener acceso a la aplicación, al no tener disponible la contraseña.

Si el usuario olvida la contraseña, podrá obtener una nueva proporcionando su correo electrónico.

- **Zona de acceso de Usuarios:** La zona de usuarios normales de la aplicación es la que contiene la parte de Realidad Aumentada, de forma que desde la página principal, permite al usuario obtener una fotografía, y a partir de ella obtener información relevante sobre dicha fotografía. Además permite ir almacenando las fotografías, junto con la información obtenida en un álbum accesible por el usuario. Junto con la información y la fotografía, se le permite al usuario añadir notas escritas y grabaciones de sonido accediendo desde la aplicación a la grabadora del dispositivo móvil.

El usuario también tiene acceso al mantenimiento de su cuenta de usuario, pudiendo modificar cualquiera de sus datos e incluso eliminar su cuenta de usuario.

Finalmente el usuario tiene también disponible el álbum de fotos, que le permite mantener las fotografías, informaciones, notas y grabaciones de audio que ha ido obteniendo, así como su eliminación.

- **Zona de acceso de Administradores:** La zona de administración contiene principalmente la gestión del repositorio de objetos e imágenes que servirán de base para la comparación y reconocimiento de imágenes. En general, esto se lleva a cabo mediante el mantenimiento y gestión de una estructura llamada genéricamente “Objetos” cada objeto, contendrá asociada varias imágenes y una descripción, de forma que cada imagen contiene la información necesaria para su comparación con las imágenes obtenidas por el usuario, y la descripción permite añadir la información relevante sobre el objeto, información que se mostrará al usuario cada vez que se detecte el objeto en alguna de las imágenes obtenidas por él. El mantenimiento de estas descripciones contiene un editor HTML, de forma que se pueden crear informaciones relativamente elaboradas.

La zona de administración contiene una gestión completa de los usuarios de la aplicación, que permite crear, consultar y modificar los usuarios de la aplicación.

También existe un mantenimiento de idiomas, que permite dar de alta nuevos idiomas para explotar las posibilidades de multilenguaje por parte de la aplicación.

Finalmente la zona de administración contiene una pantalla de búsquedas que permite realizar búsquedas de imágenes sobre el repositorio de imágenes para la comprobación del reconocimiento.

Como se ha comentado, la aplicación explota las posibilidades de multilenguaje que ofrece WebRatio, configurado inicialmente para soportar español e inglés, es fácilmente configurable para añadir nuevos idiomas, simplemente traduciendo el fichero ApplicationResources_es.properties en español, o el ApplicationResources_en_US.properties en inglés, al idioma deseado, generando un nuevo ApplicationResources_idioma_pais.properties incluyéndolo en los ejecutables de la aplicación, o directamente creándolo y traduciendo en la opción de localización que proporciona WebRatio y generando de nuevo la aplicación, y añadiendo una nueva entrada en la gestión de idiomas en la zona de administración de la aplicación

En cuanto al soporte para los datos de la aplicación se utiliza PostgreSQL como SGDB (Sistema de Gestión de Base de Datos) relacional. Principalmente, se ha optado por este SGDB por ser de libre distribución, publicado bajo la licencia BSD, pero también por permitir la utilización de procedimientos almacenados utilizando PL/PGSQL, C y C++, que permiten el desarrollo del reconocimiento de imágenes y la búsqueda de imágenes directamente sobre la base de datos. Esto último mejora considerablemente el rendimiento del proceso de reconocimiento de imágenes, ya que es un proceso costoso y su implementación directa sobre la base de datos con la potencia del lenguaje compilado C, mejora considerablemente cualquier implementación Java sobre la aplicación tal y como comprobamos en la práctica.

Otro punto a tener en cuenta en la aplicación es su apariencia, debido a las reducidas dimensiones de las pantallas de los móviles, en comparación con cualquier monitor de sobremesa, obliga a la utilización de dos estilos de presentación distintos para sobremesa y para móviles, la detección del tipo de cliente que accede a la aplicación se detecta en el servidor mediante la utilización de WURFL (Wireless Universal Resource File) [20], repositorio de descripción de dispositivos, basado en un fichero XML con la configuración de cientos de dispositivos y sus características junto a APIs en varios lenguajes de programación para el acceso a dichos datos, en concreto en la aplicación, se utiliza su API Java, así, mediante un servlet que gestiona la entrada a la

aplicación se detecta si el dispositivo que intenta acceder es un móvil o no y redirige a las páginas con la apariencia apropiada. En la actualidad son muchas las aplicaciones y sitios Web que tienen distintas apariencias que se adaptan a los dispositivos más comerciales, de forma que un único sitio Web puede tener decenas de maquetaciones distintas adaptadas a las marcas y dispositivos más comunes, consiguiéndose interfaces que se adaptan perfectamente a cada dispositivo y pantalla. En Real-A se considera únicamente dos estilos distintos líquidos que se ajustan a distintos tamaños de pantalla, además el estilo para móviles evita en la medida de lo posible la utilización de iconos e imágenes intentando utilizar al máximo las posibilidades que ofrece CSS, mejorando así el tráfico de datos para la parte de presentación y la rapidez en el refresco de las pantallas, críticos sobre todo para dispositivos móviles.

El punto más crítico de la aplicación a nivel de procesamiento es el referente a la Realidad Aumentada, debido a la dificultad de acceso a las funcionalidades de los dispositivos móviles. La pretensión de realizar una aplicación multiplataforma es sumamente complicada actualmente, dada la fragmentación de plataformas, junto a la fragmentación de navegadores móviles, y la falta de un estándar que soporte el acceso a funcionalidades como GPS, cámara, acelerómetro, brújula y demás elementos. Para la aplicación se ha conseguido el acceso a la cámara únicamente para dispositivos Symbian, utilizando APIBridge de Nokia con Flash Lite, sin embargo no se ha conseguido un resultado positivo usando el mismo procedimiento para acceder al GPS, cosa que reduciría el coste de tiempo del reconocimiento de imágenes al limitar la búsqueda a las imágenes localizadas cerca de la posición del dispositivo móvil. De esta forma, al menos se llega a una solución funcional con la obtención de imágenes a través de la cámara y el reconocimiento de imágenes.

Adicionalmente se han probado otras alternativas como la utilización de la API JavaScript de PhoneGap desde navegadores en distintas plataformas como Android con el objetivo de obtener datos reales para el estudio de desarrollo realizado en este proyecto..

La utilización de ApiBridge implica la necesidad de tener instalado en el dispositivo móvil la aplicación APIBridge_v1_1.sis. Esta aplicación funciona

como servidor, exponiendo los servicios necesarios mediante una interfaz HTTP.

Por tanto se puede concluir que la tecnología actual imposibilita la creación de una aplicación Web multiplataforma de Realidad Aumentada que permita acceder a las funcionalidades de los móviles para obtener el contexto externo al móvil, además, para este tipo de aplicaciones las aplicaciones nativas con un acceso total a las funcionalidades mencionadas siguen siendo la opción más factible, ya que además del acceso en tiempo real a la cámara permite obtener la información a partir de los datos del GPS y la brújula, opción más fácil de implementar y con mejor rendimiento que la utilización de técnicas de reconocimiento de imágenes.

A pesar de lo mencionado con anterioridad, la aplicación conseguida es totalmente multiplataforma, salvo la página que gestiona la Realidad Aumentada, necesitándose únicamente un navegador con soporte JavaScript y AJAX. Por tanto, aunque las aplicaciones Web no son el medio más indicado para soportar aplicaciones de Realidad Aumentada en la actualidad, si son un tipo de aplicación muy a tener en cuenta en otros ámbitos de negocio y las herramientas MDD son un gran apoyo a la hora de afrontar el desarrollo con eficacia y rapidez.

4.2. Reconocimiento de imágenes.

Dada las limitaciones de acceso a las funcionalidades de los dispositivos móviles desde aplicaciones Web y a la carga de almacenamiento y proceso que puede llegar a requerir el sistema, en el actual proyecto se opta por el reconocimiento de imágenes para poder obtener la información referente a la imagen obtenida por el usuario. En base a esto es necesario conocer las distintas técnicas y algoritmos existentes para el reconocimiento de imágenes.

En general dichos algoritmos se basan en dos pasos:

1. **Caracterización de la imagen:** se detecta una serie de puntos característicos de la imagen que se está procesando.
2. **Reconocimiento de la imagen:** se comparan los puntos característicos obtenidos y se comparan con los de las imágenes.

4.2.1. Técnicas para la caracterización de objetos.

Existen multitud de algoritmos, que básicamente lo que hacen es recopilar información invariante de pequeñas regiones de la imagen que consideran de interés y que permiten la comparación de imágenes en las que aparecen los mismos objetos o escenas y saber así su grado de similitud.

Sin pretender entrar en consideraciones matemáticas, dado que no es el objetivo de este proyecto, a continuación se exponen algunos de los más utilizados:

Algoritmo SIFT.

Scale-Invariant Feature Transform (SIFT), algoritmo creado por David G. Lowe en 1999 [21] y actualizado en 2004 [22], es muy utilizado en la actualidad y su función principal es la de extraer las características de un objeto. Proporciona un reconocimiento de objeto de forma general mediante la comparación de sus descriptores.

El funcionamiento principal de este algoritmo es la transformación de la imagen por medio de derivadas y máscaras de bits para un conjunto de puntos invariantes frente a cambios en la rotación, iluminación y tamaño.

El algoritmo consta de cuatro fases:

1. **Detección de los extremos de la escala.:** la primera fase de la detección consiste en buscar puntos característicos que se mantengan ante cambios de escala. SIFT busca extremos en el espacio de escalas en la función de diferencia Gaussiana (DoG) convolucionada con la imagen de entrada.
2. **Localización de puntos clave:** no todos los puntos de interés localizados en la fase anterior son realmente invariantes. Los puntos inestables en los bordes y con bajo contraste son candidatos de sufrir inestabilidad frente a cambios de iluminación, puntos de vista y ruido. Por tanto, en esta fase se evalúa los puntos de interés detectados anteriormente y eliminan los que no sean estables.
3. **Asignación de la orientación:** el objetivo de esta fase es asignar una orientación a cada punto invariante. Con esto se consigue que la caracterización del objeto sea independiente a la rotación. Pueden aparecer varias direcciones dominantes para un mismo punto y

cuantas más direcciones tengan, más invariante será. En esta fase se obtiene la dirección o direcciones del punto de interés.

4. **Descriptor de los puntos clave:** en la fase final se crea el vector de las características de cada punto de interés, llamado descriptor. El vector normalizado que se obtiene en esta fase se encarga de reducir los problemas por cambio de iluminación.

Ventajas SIFT:

- Invariante frente a cambios en la escala y rotación.
- Buen comportamiento frente al ruido.
- Bueno respecto a cambios lineales en la iluminación.

Desventajas SIFT:

- Proporciona malos resultados frente a cambios no lineales de iluminación.

Algoritmo SURF.

Speeded Up Robust Features (SURF) es un algoritmo definido en 2006 por Herbert Bay, Tinne Tuytelaars y Luc Van Gool. La última versión oficial es de 2008 [23] y está creada por los autores anteriores y Andreas Ess. Al igual que SIFT este algoritmo se basa en la extracción de puntos de interés que identifican a una imagen.

El algoritmo consta de los siguientes pasos:

1. **Detección de los puntos de interés:** para el cálculo de los puntos de interés se utiliza el cálculo de imagen-integral diseñada por Paul Viola y Michael Jones y matrices Hessianas y aproximaciones Gaussianas que reducen el tiempo de procesamiento sin perder precisión.
2. **Detección de la escala y localización de puntos de interés:** similar al cálculo de diferencia de Gaussianas de SIFT para la localización de puntos invariantes a diferentes escalas en la que varían los filtros utilizados y no la imagen, de forma que junto a las imágenes integrales consigue una disminución significativa del cálculo computacional.
3. **Descriptor de los puntos de interés:** para cada punto de interés obtenido, se calcula un vector con información de la orientación,

basándose en el cálculo de subregiones y respuestas de Haar-wavelet. El descriptor obtenido para cada punto hace que éste sea fuerte en cuanto a ruido, variaciones de iluminación y cambios en el punto de vista.

Ventajas SURF:

- Es mucho más rápido que SIFT.
- Bastante robusto a cambios de iluminación.

Desventajas SURF:

- En los cambios de escala y rotación se obtienen peores resultados que con el algoritmo SIFT.
- Se detectan menos puntos invariantes y menor número de emparejamientos correctos junto con más positivos falsos que con el algoritmo SIFT.

4.2.2. Técnicas para el reconocimiento de objetos.

Una vez obtenidos los puntos característicos de la imagen hay que compararla con las imágenes de muestra para encontrar coincidencias. Las técnicas de reconocimiento de objeto se basan en emparejar los puntos invariantes de cada imagen y a partir de una cierta probabilidad se deberá decidir si el objeto es reconocido o no. Estas técnicas son independientes del algoritmo utilizado para la extracción de características.

Entre las técnicas más sencillas y utilizadas cabe destacar:

Vecino más cercano.

Esta técnica se utiliza para detectar emparejamientos (matching) y se utiliza tanto con SIFT como con SURF. Consiste en comparar los vectores de los descriptores de las dos imágenes y encontrar la distancia Euclídea más cercana.

La distancia Euclídea se puede interpretar como una medición de la distancia entre dos vectores.

Best-Bin-First (BBF).

Algoritmo de aproximación creado por Beis y Lowe en 1997 [24].

Es un algoritmo rápido que devuelve el vecino más cercano con gran fiabilidad utilizando distancias Euclídeas y árboles kd-tree.

Además estas técnicas se pueden complementar con otras, para la eliminación de falsos positivos, mejorando así el reconocimiento, un ejemplo de este tipo de algoritmos es el algoritmo Graph Transformation Matching, que a partir de los resultados obtenidos en el matching construye un grafo bidireccional por cada imagen que vincula cada punto característico emparejado por sus K-vecinos más cercanos y a partir de los dos grafos se van eliminando los emparejamientos que distorsionan las relaciones de vecindad.

4.2.3. El reconocimiento de objetos en Real-A.

Sin pretender que el proyecto actual sea un estudio exhaustivo de las técnicas de reconocimiento de imágenes, si es un punto crítico dentro de la aplicación, por tener un gran peso dentro de la misma.

En la aplicación se utiliza el algoritmo SURF para la descripción de imágenes, junto con el algoritmo Vecino Más Cercano para la comparación de imágenes.

Se ha optado por SURF, debido a que proporciona mejores tiempos de ejecución que SIFT, además se han realizado distintas pruebas con distintas implementaciones en Java que proporcionaban tiempos de ejecución muy altos, por lo que se desechó la opción de utilizar una implementación Java dentro de la propia aplicación, y se ha optado por una implementación en C, que proporciona un rendimiento adecuado para el propósito de la aplicación. En cuanto a la elección del algoritmo de vecino más cercano se ha tenido en cuenta su sencillez de implementación sobre otros criterios.

Se ha utilizado OpenCV [25][26] que es una biblioteca de funciones de programación para la visión de computadora en tiempo real. OpenCV es liberado bajo una licencia BSD, y es gratuito tanto para uso académico como comercial.

OpenCV contiene implementado tanto el algoritmo SURF como SIFT, y proporciona las funciones en C necesarias para la obtención de los puntos de interés de cualquier imagen.

La implementación del reconocimiento de imágenes se ha realizado directamente sobre la base de datos de la aplicación que se encuentra en PostgreSQL [27].

Se han implementado varios procedimientos almacenados utilizando PL/PGSQL [28], el lenguaje PL de PosgreSQL, y varios procedimientos implementados en C. A continuación se detalla el proceso de reconocimiento de imagen en la aplicación:

Inserción de Imágenes en la BBDD:

Para el reconocimiento de las imágenes en la aplicación se ha creado un repositorio de imágenes, estas imágenes que sirven como muestra sobre las que se realizarán las búsquedas con las imágenes externas que proporciona el usuario. La inserción de imágenes en la base de datos se realiza atendiendo al diagrama de la figura 6.

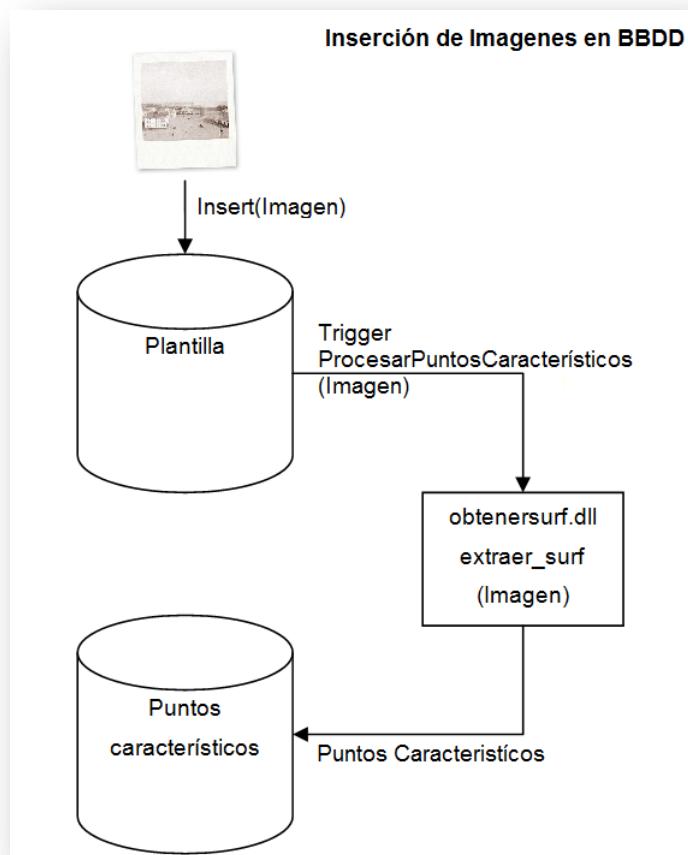


Figura 6.- Inserción de imágenes en BBDD.

La estructura de las tablas en las que se almacenan las imágenes queda detallada en la figura 7.

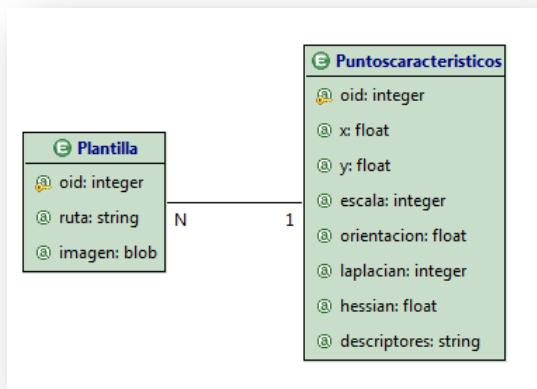


Figura 7.- imágenes en BBDD.

Al realizar una inserción (insert) de una imagen en la tabla Plantilla, se lanza el trigger ProcesarPuntosCaracteristicos automáticamente, de forma que este trigger calcula los puntos característicos de la imagen y los almacena en la tabla Puntoscaracteristicos. El cálculo de puntos se realiza mediante una llamada al procedimiento extraer_surf, implementado en C e incluido en la librería creada para tal efecto llamada obtenersurf.dll.

Obtención de coincidencias entre una imagen y una imagen del repositorio:

Para obtener el número de coincidencias entre cualquier imagen y las imágenes de la base de datos se ha creado el procedimiento almacenado numero_de_coincidencias en base al diagrama detallado en la figura 8.

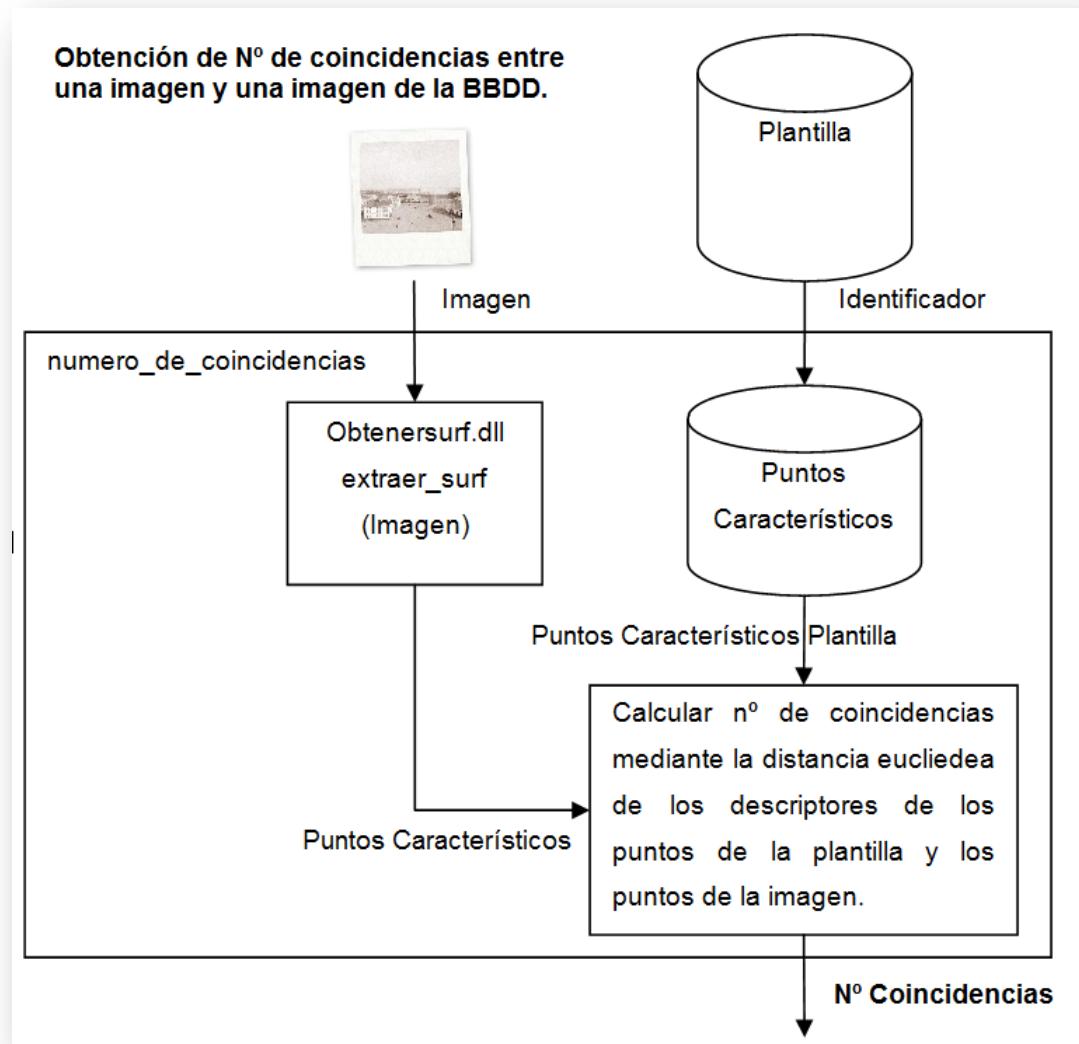


Figura 8.- Obtención nº de coincidencias entre una imagen y las imágenes de BBDD.

Al procedimiento `numero_de_coincidencias`, se le pasa la imagen que se quiere comparar y el identificador de la imagen de la tabla **Plantilla** con la que se quiere comparar, dicho procedimiento obtiene los puntos característicos mediante la función `extraer_surf`, a continuación cada punto obtenido, lo compara con todos los puntos de la imagen del repositorio mediante la función `distancia_cuadrada` que también ha sido implementada dentro de la librería `obtenersurf.dll`, que obtiene la distancia euclídea entre los descriptores de dos puntos y decide si se trata de una coincidencia o no. Finalmente el procedimiento retorna el número de coincidencias existentes.

Gracias a este procedimiento, desde la aplicación se pueden realizar consultas sobre la base de datos como la que sigue:

```
SELECT plantilla.oid, numero_de_coincidencias(plantilla.oid, 'imagen') AS
coincidencias FROM plantilla ORDER BY coincidencias DESC LIMIT 1
```

Código 1.- Consulta de coincidencias entre imágenes.

Si 'imagen' es la ruta de una imagen, esta consulta obtiene el identificador de la plantilla que contiene más coincidencias, es decir, sería la imagen en base de datos que se corresponde con la imagen pasada.

El hecho de obtener coincidencias para una imagen, no significa que la imagen buscada coincida con la obtenida debido a los falsos positivos, de esta forma, para la aplicación se considera que se necesitan al menos 10 coincidencias para poder decir que las imágenes coinciden con probabilidad alta. A este número mínimo de coincidencias se ha llegado empíricamente, realizando pruebas con la aplicación sobre distintas imágenes.

A continuación se detallan los procedimientos creados para el manejo de imágenes:

- **extraer_surf(cstring):** función que se encarga de extraer los puntos característicos de una imagen a través del algoritmo SURF, implementada en C en el fichero obtenersurf.c. Recibe como parámetro de entrada una cadena con el path completo del fichero con la imagen, convierte la imagen del formato que tenga a escala de grises, y obtiene y retorna un conjunto de PuntoClave, calculados mediante el algoritmo SURF que proporciona la librería OpenCV. PuntoClave se corresponde con la estructura de punto característico definida en SURF:

```
/*Descripción PuntoClave */
typedef struct{
    float x, y;           //Pixel que representa el punto.
    int scale;             //Escala.
    float orientation;    //Orientación del punto.
    int laplacian;         //Laplacian.
    float hessian;         //hessian.
    float descriptor[128]; //Descriptores del punto.
} PuntoClave;
```

Código 2.- Estructura punto clave.

- **obtener_imagen_coin(cstring,cstring,cstring):** obtiene una imagen comparativa de dos imágenes enlazando los puntos característicos coincidentes mediante rectas, tal y como se muestra en la figura 9. Como parámetro de entrada recibe tres cadenas de caracteres, la primera y la

segunda contendrán el path completo de los ficheros con las imágenes, la tercera cadena es el path completo del fichero de salida. La función calcula los puntos característicos coincidentes entre las dos imágenes y genera una tercera imagen, uniendo las anteriores y representando las coincidencias. Esta función también está implementada en C en el fichero obtenersurf.c, aprovechando las facilidades de OpenCV para el manejo de imágenes.

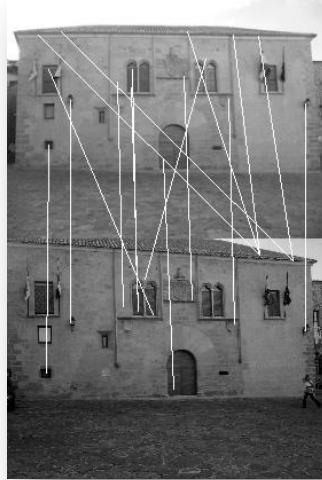


Figura 9.- Imagen comparativa de dos imágenes.

- **distanzia_cuadrada(double precision[],double precision[]):** esta función recibe dos vectores y obtiene la distancia cuadrada entre ellas, retornando el valor de dicha distancia. Esta función también está implementada en C en el fichero obtenersurf.c.
- **procesar_puntos_caracteristicos():** trigger que se lanza al insertar o modificar cualquier imagen en la tabla “Plantilla” de forma que genera los puntos característicos de la imagen, mediante una llamada a extraer_surf(cstring), y los almacena en la tabla “Puntoscaracteristicos”. Esta función se ejecuta al borrar una imagen en la tabla “Plantilla”, eliminando los puntos correspondientes de la tabla “Puntoscaracteristicos”. Esta función está implementada en PL/PGSQL.
- **numero_de_coincidencias(oid_imagen integer, ruta_imagen character varying(255)):** esta función recibe el identificador de la imagen de la tabla “Plantilla” y el path completo de la imagen con la que se quiere comparar, la función se encarga de comparar los descriptores de los puntos de ambas

imágenes mediante el algoritmo del vecino más cercano. Para ello calcula las distancias cuadradas, de cada punto de la imagen a comparar con todos los puntos de la imagen que sirve de plantilla, llamando a distancia_cuadrada(double precision[],double precision[]). Obtiene la distancia menor y la segunda menor, si la distancia menor es menor al 60% de la segunda menor, se considera que hay coincidencia entre el punto comparado y el punto con la menor distancia. Esta función está implementada en PL/PGSQL.

- **obtener_imagen_coincidencias(ruta_imagen1 character varying(255), ruta_imagen2 character varying(255), ruta_salida character varying(255))**: función implementada en PL/PGSQL, que se encarga de llamar a la ya mencionada función obtener_imagen_coin(cstring, cstring, cstring) para obtener una imagen comparativa ruta_salida donde se muestran las coincidencias de las dos imágenes de entrada ruta_imagen1 y ruta_imagen2.

Como se ha comentado, es necesaria la librería obtenersurf.dll, generada a partir del código fuente incluido en el fichero obtenersurf.c, la generación de la dll se puede realizar utilizando los ficheros bat creados para tal propósito, compila.bat y enlaza.bat. El fichero obtenersurf.dll debe ir en la carpeta “lib” de la instalación de PostgreSQL, en este caso en PostgreSQL\8.4\lib.

El script SQL REAL-A.sql contiene el script SQL necesario para crear todas las funciones en la base de datos de la aplicación.

Para más detalle en sobre la codificación de estas funciones se remite al lector a los ficheros obtenersurf.c y REAL-A.sql, así como compila.bat y enlaza.bat, entregados junto al proyecto, o en google code en la url:

<http://code.google.com/p/pfc-real-a/source/browse/#svn/trunk/Real-A/Reconocimiento>

4.3. Manual de desarrollador.

Como se ha comentado con anterioridad, la aplicación Web Real-A está desarrollada utilizando WebRatio, de forma que la aplicación se divide en varios modelos de hipertexto que describen las distintas vistas de las que se compone la aplicación (site views) y un modelo estructural con la definición del dominio de la aplicación.

Estas tres vistas (público, Usuario y Administrador) a su vez están duplicadas, de forma que hay tres vistas para dispositivos móviles y tres vistas para dispositivos de escritorio, con apariencia y funcionalidad adaptada a cada tipo. La aplicación detecta el tipo de dispositivo con que el usuario entra a la aplicación y redirecciona a la vista adecuada. La única diferencia entre las vistas para dispositivos de escritorio y las vistas para dispositivos móviles, además de la apariencia, es que la vista de usuario de dispositivos de escritorio no tiene acceso a la pantalla de Realidad Aumentada debido que no tendría razón de ser para un ordenador de sobremesa.

En la figura 10 se muestra la estructura de módulos más abstracta de Real-A en WebRatio.

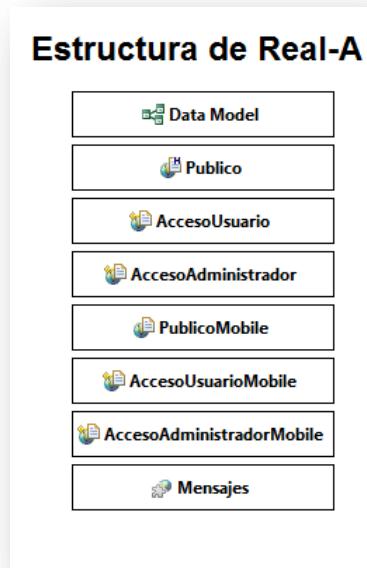


Figura 10.- Estructura de Real-A.

En los siguientes capítulos se entra en el detalle de cada módulo que componen la aplicación y otros aspectos de configuración como el control de versionados y la generación de logs.

4.3.1. Configuración Log.

Para poder depurar la aplicación y llevar el control de errores, WebRatio, haciendo uso del servicio de login log4j, permite configurar la aplicación para poder mostrar tanto en consola como por ficheros distintos mensajes de log.

En la aplicación, además de los ficheros de log que configura WebRatio por defecto, se ha configurado para poder ver en tiempo real y por consola

mensajes de log que se han ido insertando por el código escrito manualmente, de forma que es fácil seguir la ejecución y controlar cualquier error, para ello se ha configurado el fichero RTXLogConfig.properties introduciendo las siguientes líneas:

```
log4j.appenders.RTXConsoleAppender=org.apache.log4j.ConsoleAppender
log4j.appenders.RTXConsoleAppender.layout=org.apache.log4j.PatternLayout
log4j.appenders.RTXConsoleAppender.layout.ConversionPattern= %d{HH:mm:ss,SSS}
%p (%c:%L) - %m%n
log4j.appenders.RTXConsoleAppender.Threshold=INFO
log4j.appenders.RTXConsoleAppender.Target=System.out
```

Código 3.- RTXLogConfig.properties.

De esta forma se muestran por consola los distintos mensajes a nivel INFO insertados en el código.

Log4j permite configurar los log en 6 niveles de menor a mayor detalle:

- OFF: es el nivel de mínimo detalle, deshabilita todos los logs.
- FATAL: se utiliza para mensajes críticos del sistema, generalmente después de utilizar este tipo de mensaje la aplicación abortará.
- ERROR: se utiliza en mensajes de error de la aplicación que se desea guardar, estos eventos afectan a la aplicación pero lo dejan seguir funcionando.
- WARN: se utiliza para mensajes de alerta sobre eventos que se desea mantener constancia, pero que no afectan al correcto funcionamiento del programa.
- INFO: se utiliza para mensajes de información.
- DEBUG: se utiliza para escribir mensajes de depuración..
- TRACE: se utiliza para mostrar mensajes con un mayor nivel de detalle que DEBUG.
- ALL: es el nivel de máximo detalle, habilita todos los logs.

En el caso de WebRatio, mucho del código generado automáticamente contiene logs a nivel de DEBUG, que muestran mucha información sobre la ejecución, por esta razón, para evitar un log muy extenso y poder tener sólo información referente al código que se ha implementado manualmente, se ha optado por configurar la aplicación a nivel INFO, e insertar todos los mensajes de log en el código con este nivel.

El nivel de log incide en los tiempos de ejecución, por lo que para una aplicación en producción es aconsejable dejar el nivel de log a ERROR,

mostrando sólo errores y excepciones en la ejecución, minimizando el tiempo consumido por log4j, incluso se podría desactivar con el nivel OFF y volver a activar ante cualquier anomalía o error.

A continuación se muestra un ejemplo de log introducido en el código:

```
if (log.isInfoEnabled()) log.info("*****");
if (log.isInfoEnabled()) log.info("* ValidarCamposUsuario.groovy *****");
if (log.isInfoEnabled()) log.info("ENTRADA:");
if (log.isInfoEnabled()) log.info("usuario:" + usuario);
if (log.isInfoEnabled()) log.info("email:" + email);
```

Código 4.- Ejemplo de mensajes de log en la aplicación.

Además WebRatio incluye por defecto cuatro ficheros de log, que se encuentran en \Real-A\WEB-INF\log en donde se encuentre desplegada la aplicación:

- **RTX.log:** histórico de las unidades ejecutadas, las páginas visitadas, consultas ejecutadas por cada unidad y, en su caso, los registros de errores de ejecución.
- **RTXError.log:** realiza un seguimiento de todos los errores de ejecución.
- **WRStruts.log:** realiza un seguimiento de la ejecución de cada unidad y en cada página, los enlaces y los datos transportados, junto con los resultados de cada ejecución.
- **WRStrutsError.log:** registra los errores del framework Struts.

4.3.2. Control de Versiones mediante Subversion.

Para el control de versiones y soporte de seguridad, en el proyecto se ha optado por utilizar Subversion (SVN), que es un sistema de versionado de libre distribución y código abierto, permitiendo mantener un repositorio con los fuentes de la aplicación, manteniendo un histórico con los cambios que se van produciendo en cada fichero.

Como cliente de cliente SVN en WebRatio, se ha utilizado el plug-in Subversive 0.7 para eclipse y como servidor se ha creado un proyecto en google code (<http://code.google.com/p/pfc-real-a/>), ya que google code soporta este sistema de versionado .

El uso del control de versiones asegura al proyecto dos objetivos:

1. Permite revertir archivos a un estado anterior, revertir el proyecto entero a un estado anterior, comparar cambios a lo largo del tiempo,

comprobar que puede estar causando un problema comparando con versiones anteriores. Además de poder recuperar ficheros eliminados por error.

2. Tener una copia de seguridad en un servidor externo y on-line, de forma que además de evitar cualquier problema de pérdida por errores hardware, incendios y catástrofes en general, permite tener acceso al proyecto desde cualquier lugar. Además, es una forma rápida de mantener una copia de seguridad, ya que sólo se actualizan los ficheros sobre los que se han producido cambios.

Este tipo de herramientas se hacen imprescindibles en proyectos con varios integrantes, ya que, además de lo ya mencionado, ayudan a gestionar la colaboración y la compartición del proyecto, permitiendo bloquear ficheros, comparar y mezclar ficheros que han tocado varias personas, entre otros beneficios.

Dado que en el actual documento sólo se entra en detalle del código más relevante, si el lector desea entrar en más detalle sobre algún aspecto de codificación, tiene todo el código completamente disponible online en:

<http://code.google.com/p/pfc-real-a/>

4.3.3. Dominio (Data Model).

WebRatio permite sincronizar directamente el modelo con la base de datos, generando dicho modelo en la base de datos a la que está conectada.

En el caso de la aplicación se ha utilizado PostgreSQL 8.4, sin embargo, para el correcto funcionamiento de la aplicación, además de la sincronización que proporciona WebRatio, hay que cargar en la base de datos un script, creado con las funciones necesarias para el reconocimiento de imágenes comentadas en el punto 3.2.3, el script se encuentra en el fichero “Real-A.SQL”.

También se adjunta con el proyecto un script llamado “Carga_Inicial.sql” que contiene la carga de los datos básicos de la aplicación, como son los grupos, módulos y usuarios e idiomas por defecto, así como una carga de objetos para el reconocimiento de imágenes, basada en 20 edificios característicos de la parte antigua de la ciudad de Cáceres, con informaciones obtenidas de la página Web de Cáceres Joven [29] y del portal de turismo del

Ayuntamiento de Cáceres [30], con lo que se consigue una carga inicial operativa de la base de datos.

Los scripts, además de encontrarse adjuntos al proyecto, están accesibles en:

<http://code.google.com/p/pfc-real-a/source/browse/#svn/trunk/Real-A/Reconocimiento>

Las imágenes necesarias para el reconocimiento de los edificios se encuentran en:

<http://code.google.com/p/pfc-real-a/source/browse/#svn/trunk/Real-A/WebContent/upload/caceres>

En la figura 11 se muestra el modelo estructural de la aplicación, que define el dominio de datos.

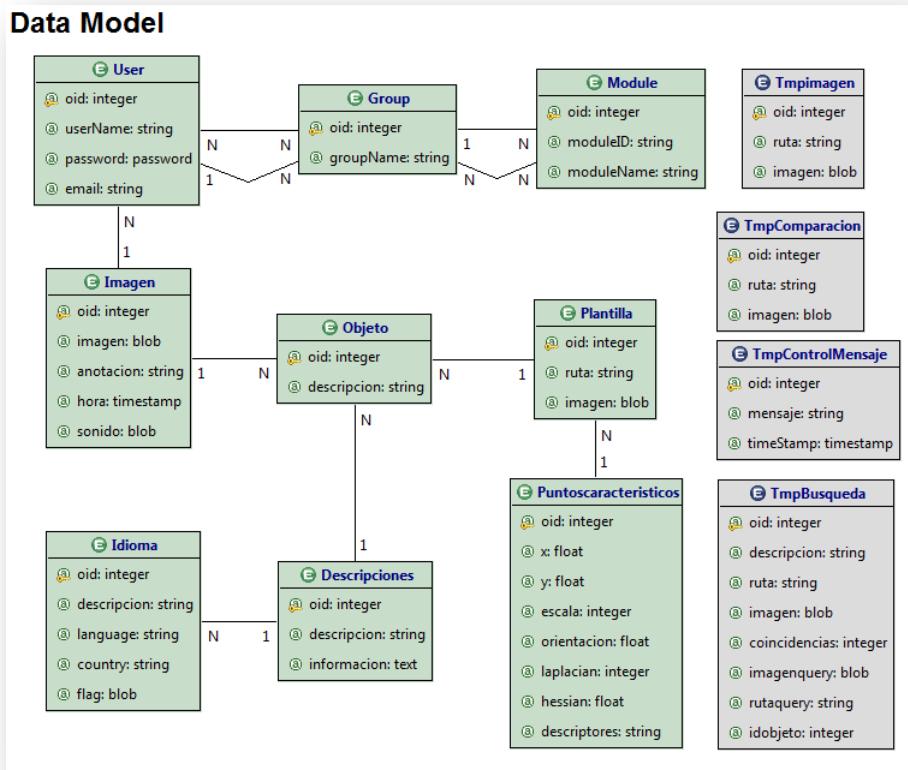


Figura 11.- Dominio Real-A.

Las tablas User, Group y Module son tablas que WebRatio crea por defecto y que sirven para controlar las restricciones de acceso a la aplicación:

- **User**: Representa un usuario de la aplicación, por defecto contiene los campos username y password incluidos para la identificación, además se le ha añadido el campo email, que se utiliza en la aplicación para el control de password.
- **Group**: Representa un conjunto de usuarios con las mismas propiedades.

- **Module:** Representa la parte lógica de la aplicación a la que queremos restringir el acceso (site views, áreas y páginas).

De esta forma WebRatio permite definir usuarios, cada usuario puede estar conectado a un grupo por defecto y a otros grupos. Todo grupo está conectado a un módulo por defecto y a otros módulos. Un módulo es la representación de una parte de la aplicación con acceso restringido. De este modo WebRatio conecta a cada usuario con las partes protegidas a las que puede tener acceso, a través de las relaciones con grupos.

En Real-A, existen dos grupos creados por defecto, "usuarios" y "admin", y cuatro módulos, "Acceso Usuarios", "Repositorio Imagenes", "Acceso Usuario Movil", "Repositorio Imagenes Movil", que se corresponden con los site views restringidos, de forma que "usuarios" se relaciona con "Acceso Usuarios" y "Acceso Usuario Movil" y "admin" se relaciona con "Repositorio Imagenes Movil" y "Repositorio Imagenes Movil", con lo que queda totalmente definido el acceso a la aplicación para cada tipo de usuario de forma fácil gracias a las facilidades ofrecidas por WerRatio.

Las entidades propias de la aplicación son:

- **Objeto:** Objeto es la entidad principal del repositorio, representa los objetos que servirán de muestra para las búsquedas, cada objeto estará relacionado sobre un conjunto de imágenes para su reconocimiento en la entidad "Plantilla" e informaciones relativas a dicho objeto, una por cada idioma definido en la aplicación, en la entidad "Descripciones". De esta forma si pretendemos que la aplicación ofrezca información sobre monumentos, cada "Objeto" representará un monumento dado.
- **Plantilla:** esta entidad contendrá la información de las distintas imágenes asociadas a cada objeto, esta entidad junto con la de "Puntoscaracteristicos" es la utilizada para el reconocimiento de imágenes. Contiene la ruta y el nombre del fichero físico con la imagen.
- **Puntoscaracteristicos:** entidad que contiene los datos de los puntos característicos de cada imagen, estos datos son generados automáticamente como se indica en el apartado 3.2.3 utilizando el algoritmo SURF.

- **Descripciones:** contiene la información relativa al objeto con el que está relacionado, y es la información que se muestra al usuario tras reconocer la imagen que proporciona. La entidad contiene un campo “descripción” que es una cadena de texto, y un campo “información” que puede albergar un texto escrito en html, de forma que se puede almacenar información elaborada sobre el objeto relacionado. Además cada descripción se relaciona con la entidad “Idioma” de forma que se pueda mantener una descripción por cada idioma asociada a cada objeto y poder soportar así el multilenguaje en la aplicación.
- **Idioma:** la entidad “Idioma” proporciona información sobre los lenguajes soportados por la aplicación, contiene cuatro campos, “descripción” que describe el idioma, “languaje” y “country” que permitirá a la aplicación cargar las variables de contexto CountryISOCtxParam y LanguageISOCtxParam, que son las que utiliza WebRatio para el control del multilenguaje en la aplicación, y el campo “flag” que identifica el fichero
- **Imagen:** esta entidad es la que contiene la información relativa al álbum de fotos que puede ir creando el usuario con las fotos que va generando, se relaciona con la entidad “Objeto” si la imagen ha sido reconocida, y por tanto con la información relativa a dicho objeto, y también se relaciona con el propietario de la imagen, mediante la entidad “User”.

Las entidades indicadas son todas entidades persistentes (entidades con fondo verde en la figura 11), que tienen reflejo en la base de datos, además existen otras cuatro entidades volátiles (entidades con fondo gris en la figura 11, situadas a la derecha), que son entidades que sólo perduran en el sistema a nivel de sesión. Estas entidades son utilizadas en la aplicación como entidades temporales para el reconocimiento de imágenes y para almacenar mensajes de error y de información. Se irán comentando a lo largo de los siguientes subcapítulos.

4.3.4. Acceso a la aplicación.

De la detección del dispositivo se encarga un servlet creado para este fin, para ello se ha creado la clase java “Redireccion” que extiende de HttpServlet.

La aplicación se compone de seis vistas:

Dispositivo	Tipo de Usuario	SiteView
Escritorio		Publico
	“usuario”	AccesoUsuario
	“admin”	AccesoAdministrador
Móviles		PublicoMobile
	“usuario”	AccesoUsuarioMobile
	“admin”	AccesoAdministradorMobile

Tabla 3.- Site Views de Real-A.

Las tres vistas de escritorio son las de entrada por defecto a cada uno de los módulos, en cada una de ellas se ha creado como página principal (home) una página denominada “redirección”, estas páginas se generan en una jsp del estilo:

```
<jsp:forward page="/[%=redireccion%]"></jsp:forward>
```

Código 5.- JSP correspondiente a las páginas “redirección”.

Donde [%=redireccion%] se corresponde con publico.htm, usuario.htm o admin.htm, según corresponda. Así, se configura el fichero WEB-INF/web.xml, definiendo un servlet para cada tipo de acceso, y se mapea con el patrón de url que hará que se ejecute, de esta forma se añaden a web.xml las entradas que se muestran a continuación:

```
<servlet>
    <servlet-name>RedireccionPublivоСervlet</servlet-name>
    <servlet-class>es.real.a.redireccion.Redireccion</servlet-class>
    <init-param>
        <param-name>entrada</param-name>
        <param-value>PUBLICO</param-value>
    </init-param>
</servlet>

<servlet>
    <servlet-name>RedireccionUsuarioServlet</servlet-name>
    <servlet-class>es.real.a.redireccion.Redireccion</servlet-class>
    <init-param>
        <param-name>entrada</param-name>
        <param-value>USUARIO</param-value>
    </init-param>
</servlet>

<servlet>
    <servlet-name>RedireccionAdminServlet</servlet-name>
    <servlet-class>es.real.a.redireccion.Redireccion</servlet-class>
```

```
<init-param>
    <param-name>entrada</param-name>
    <param-value>ADMIN</param-value>
</init-param>
</servlet>

<servlet-mapping>
    <servlet-name>RedireccionPublivoSrvlet</servlet-name>
    <url-pattern>/publico.htm</url-pattern>
</servlet-mapping>

<servlet-mapping>
    <servlet-name>RedireccionUsuarioSrvlet</servlet-name>
    <url-pattern>/usuario.htm</url-pattern>
</servlet-mapping>

<servlet-mapping>
    <servlet-name>RedireccionAdminSrvlet</servlet-name>
    <url-pattern>/admin.htm</url-pattern>
</servlet-mapping>
```

Código 6.- WEB-INF/web.xml

De esta forma, el método “processRequest” de la clase java “redirección” se ejecutará cada vez que se entre en las distintas páginas “redirección” pasándole como parámetro el tipo de acceso, PUBLICO, USUARIO o ADMIN que corresponda, “processRequest” se encarga de decidir el tipo de dispositivo que está accediendo a la aplicación utilizando la librería wurfl-1.2.2.jar y el fichero xml con la definición de dispositivos que proporciona Wurfl, que se encuentra comprimido WEB-INF/wurfl-latest.zip junto con el parche para navegadores WEB-INF/Web_browsers_patch.xml. Una vez que el servlet decide el tipo de dispositivo con el que se accede redirecciona la aplicación a la página real correspondiente.

El código completo de la clase “redireccion.java” se remite al lector al fichero entregado junto al proyecto o en google code en la url:

<http://code.google.com/p/pfc-real-a/source/browse/#svn/trunk/RedireccionDispositivo/src/es/real/a/redireccion>

4.3.5. Module View Mensajes.

WebRatio permite crear colecciones de módulos reutilizables que pueden contener unidades de operación, unidades de contenido, grupos de operación y páginas, que pueden ser utilizados en las demás vistas que componen la aplicación.

Este tipo de vistas se ha utilizado para el diseño de varios módulos de contenido que son utilizados en el resto de vistas de la aplicación.

La figura 12 muestra el mayor nivel de abstracción del Module View Mensajes.

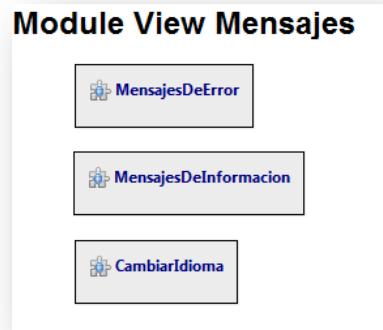


Figura 12.- Module View Mensajes.

Mensajes de Error y Mensajes de Información:

Ambos módulos se han creado para mostrar los mensajes de error o información que se genera la aplicación y que se muestran al usuario. Dichos mensajes se muestran en un popup AJAX sobre la aplicación y soportan multilenguaje de forma que el mensaje se muestra en el idioma que tiene seleccionado el usuario. La diferencia entre los mensajes de error y los mensajes de información son únicamente a nivel de presentación.

Los modelos de ambos módulos de mensajes quedan detallados en las figuras 13 y 14.

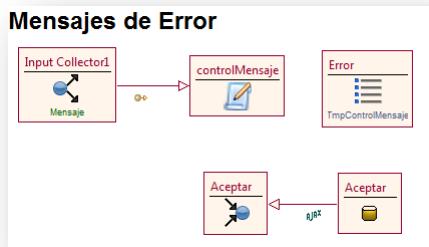


Figura 13.- Mensajes de Error.

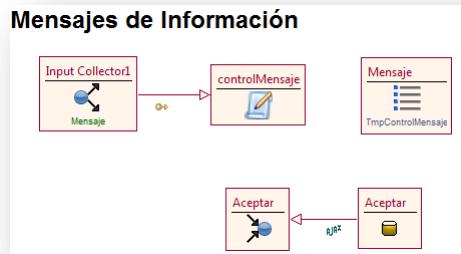


Figura 14.- Mensajes de Información.

Los mensajes de error, y los de información, tienen un colector de parámetros de entrada (Input Collector Unit) donde recibe el mensaje en forma de localización, esta clave la pasa al script “ControlMensajes.groovy”, que, en caso de recibir un mensaje como parámetro de entrada, se encarga de acceder al contexto para obtener el país y el lenguaje que tiene seleccionado el usuario, y así acceder al texto en dicho lenguaje. Este mensaje lo almacena en una tabla temporal, correspondiente a la entidad virtual “TmpControlMensaje” del dominio. Siendo las entradas de esta tabla temporal, las que muestra en la unidad índice (Index Unit) “Mensaje” en forma de tabla.

Este sistema de la tabla temporal y la unidad índice, permite además mostrar una colección de mensajes, almacenados en la tabla desde otros scripts groovy, en los que se realizan varias validaciones a la vez.

Finalmente se ha añadido una unidad de contenido sin operación (No Op Content Unit) “Aceptar” y la señal “Aceptar”, que se encarga de cerrar el popup AJAX y retornar el control a la aplicación.

Como se ha comentado, y se puede ver en el modelo, ambos módulos son similares, cambiando sólo a nivel de presentación.

El código de “ControlMensajes.groovy” es el siguiente:

```
//inputs=mensaje
import java.sql.Timestamp
import com.webratio.rtx.RTXConstants
import com.webratio.rtx.vdb.VirtualEntityTable
import com.webratio.webapp.TmpControlMensaje
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

Log log = LogFactory.getLog(this.getClass());
if (log.isInfoEnabled())
    log.info("*****");
if (log.isInfoEnabled())
    log.info("* ControlMensajes.groovy *****");
if (log.isInfoEnabled()) log.info("ENTRADA:");
if (log.isInfoEnabled()) log.info("mensaje:" + mensaje);

if(null != mensaje && mensaje != ""){
    /* OBTIENE EL MENSAJE INTERNACIONALIZADO
    *****/
    try{
        String lenguaje = sessionContext.get(RTXConstants.LANGUAGE_CTX_PARAM_KEY);
        String pais = sessionContext.get(RTXConstants.COUNTRY_CTX_PARAM_KEY);
        mensaje = ResourceBundle.getBundle("ApplicationResources",
                new Locale(lenguaje,pais)).getString(mensaje);
    }catch(e){
        mensaje = "????" + mensaje + "??";
    }
    /*
    ****
}

if (log.isInfoEnabled()) log.info("mensaje:" + mensaje);

def tmpEntityId = "ent8";
VirtualEntityTable tmpTable = new VirtualEntityTable(tmpEntityId, sessionContext,
        service)
```

```

//Borrado de la tabla temporal
tmpTable.delete(tmpTable.getRows());

//Genera el registro con el mensaje de error
TmpControlMensaje tmpControlMensaje = new TmpControlMensaje();
tmpControlMensaje.mensaje = mensaje;
tmpControlMensaje.timeStamp = new Timestamp(System.currentTimeMillis());
tmpTable.save(tmpControlMensaje);
}

if (log.isInfoEnabled())
    log.info("* FIN ControlMensajes.groovy *****");
if (log.isInfoEnabled())
    log.info("*****");

return ["resultCode":"success"]

```

Código 7.- ControlMensajes.groovy

Cambiar Idioma:

Este módulo se ha diseñado para manejar la selección de idioma en todas las páginas de la aplicación, que permita al usuario seleccionar el idioma en el que quiere visualizar la aplicación. El diseño de este módulo queda recogido en la figura 15.

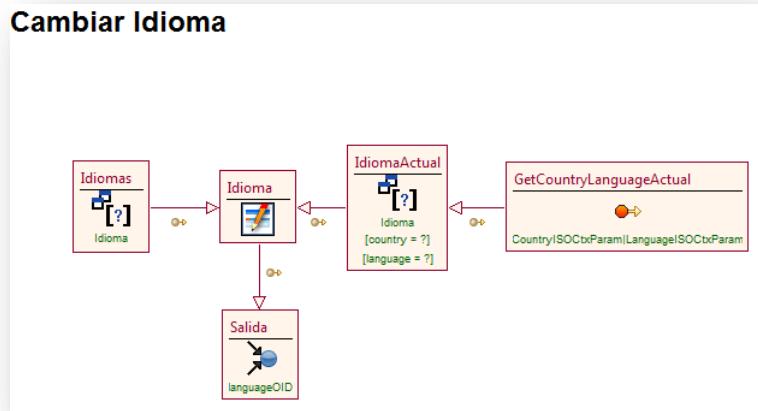


Figura 15.- Cambiar Idioma.

El modelo se compone de la unidad de entrada (Entry Unit) “Idioma”, que contiene un campo de selección múltiple, donde se cargan los idiomas de la entidad “Idiomas” del modelo de datos, a través de la unidad de selección (Selector Unit) “Idiomas”, también se carga el valor por defecto que se corresponderá con los valores de contexto LanguageISOCtxParam y CountryISOCtxParam, tras seleccionar el valor actual a través de la unidad de selección “Idioma Actual”. Finalmente, cada vez que se selecciona un nuevo idioma, retorna el valor del identificador del registro seleccionado a través del colector de parámetros de salida (Output Collector Unit) “Salida”.

4.3.6. Publico y PublicoMobile.

Los sitios “Publico” y “PublicoMobile” son los modelos que se corresponden con las páginas de acceso público, es decir, accesibles sin necesidad de tener un usuario registrado en la aplicación, siendo además el punto de entrada a la aplicación.

Ambas vistas, como en el resto de los casos, son idénticas, sólo diferenciándose a nivel de presentación, de forma que “Publico” tiene una apariencia diseñada para el acceso desde dispositivos de escritorio, y “PublicoMobile” para el acceso desde dispositivos móviles.

El modelo de la aplicación para estos siteviews queda reflejado en la figura 16.

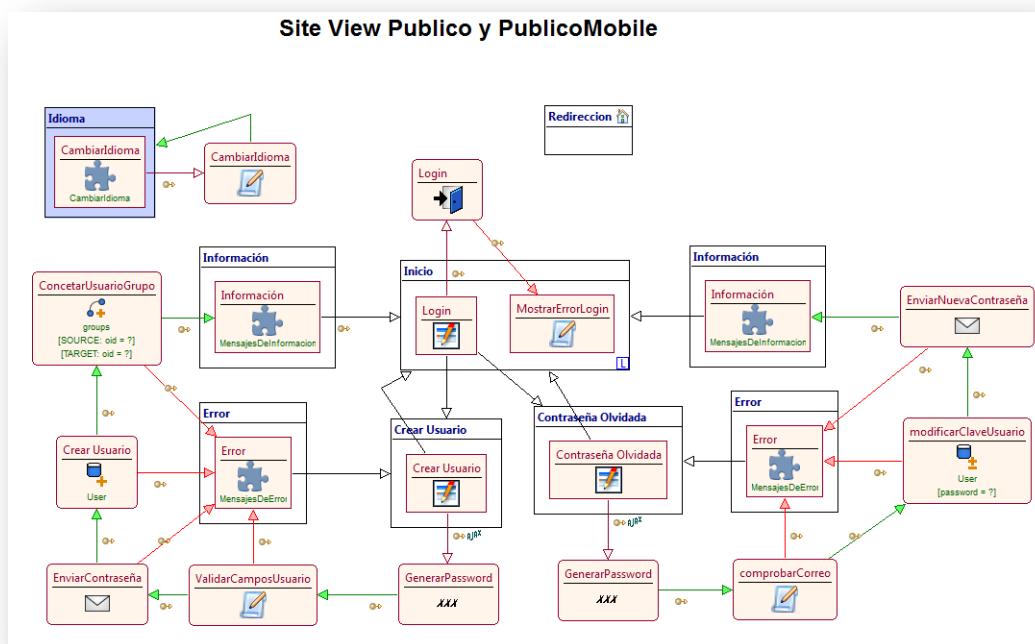


Figura 16.- Site View Publico y PublicoMobile.

En este modelo podemos destacar los siguientes componentes:

Página Redirección.

La página Redirección es una página sin contenido, marcada como “Home” del sitio, que interviene en el mecanismo de redirección comentado en el punto 3.3.4. Acceso a la aplicación. Esta página sólo se incluye en la vista “Publico” que es la vista Home, es decir la vista de entrada a la aplicación. En “PublicoMobile” no es necesaria, ya que a esta vista sólo se accederá mediante la redirección, una vez detectado que el dispositivo es un móvil.

Página Maestra Idioma.

Una página maestra (Master Page) es una página especial cuyo contenido se inserta en las demás páginas de la vista, a la hora de generarlas, a no ser que se indique explícitamente en las propiedades de la página que se quiere omitir la página maestra.

Como la opción de seleccionar idioma se desea que aparezca en todas las páginas de la vista, se ha optado por diseñarla dentro de una página maestra, en vez de tenerla que repetir por cada una de las páginas de la vista.

Así, esta página contiene una unidad de módulo (Module Unit) que hace referencia al módulo “Cambiar Idioma”, que como hemos comentado, cada vez que se selecciona un idioma en dicho módulo, éste retorna el identificador del registro “Idioma” seleccionado, este identificador se pasa por parámetro al script “CambiarIdioma.groovy” que se encarga de buscar el registro en base de datos utilizando el identificador, y recarga las variables de contexto con los valores obtenidos de base de datos, de forma que al refrescarse la página, se carga con los textos y etiquetas en el idioma seleccionado.

```
//inputs=id
//outputs=id|Language|Country
import com.webratio.rtx.db.DBTransaction
import com.webratio.rtx.db.HibernateService
import com.webratio.rtx.core.BeanHelper
import org.apache.commons.lang.math.NumberUtils
import com.webratio.rtx.RTXConstants
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

Log log = LogFactory.getLog(this.getClass());

if (log.isInfoEnabled())
    log.info("*****");
if (log.isInfoEnabled())
    log.info("* CambiarIdioma.groovy*****");
if (log.isInfoEnabled()) log.info("ENTRADA:");
if (log.isInfoEnabled()) log.info("id:" + id);

def dbId = "db1";
def session = getDBSession(dbId);
language = null;
country = null;

def idio = null;
def idioOID = null;
if(null != id) {
    idio = BeanHelper.asString(id);
    idioOID = NumberUtils.toInt(idio);
}

//Realiza la búsqueda de las imágenes
def select = "SELECT language ,country FROM \"idioma\" WHERE oid = :idiomaOid

def querySeleccion = session.createSQLQuery(select);
querySeleccion.setParameter("idiomaOid", idioOID);
def result = querySeleccion.list();

//Genera la tabla temporal con los registros de la query
for (Iterator iter = result.iterator(); iter.hasNext();) {
```

```

Object[] row = (Object[]) iter.next();
language = (String) row[0];
country = (String) row[1];
sessionContext.put(RTXConstants.LANGUAGE_CTX_PARAM_KEY, language);
sessionContext.put(RTXConstants.COUNTRY_CTX_PARAM_KEY, country);
}

if (log.isInfoEnabled()) log.info("SALIDA:");
if (log.isInfoEnabled()) log.info("id: " + id);
if (log.isInfoEnabled()) log.info("Language: " + language);
if (log.isInfoEnabled()) log.info("Country: " + country);
if (log.isInfoEnabled())
    log.info("* FIN CambiarIdioma.groovy *****");
if (log.isInfoEnabled())
    log.info("*****");

return ["resultCode":"success", "id":id, "Language":language, "Country":country]

```

Código 8.- CambiarIdioma.groovy.

Página Inicio.

Página de entrada a la aplicación, contiene la unidad de entrada (Entry Unit) “Login” con dos campos, usuario y contraseña, junto con una señal “Aceptar”, que envía ambos campos a una unidad de login (Login Unit), que verifica las credenciales del usuario para permitir el acceso a las vistas no públicas. En caso de producirse un error en el acceso, retorna a la pantalla mediante una señal de error a la unidad de script (Script Unit) “MostrarErrorLogin”, con el script “MostrarErrorLogin.groovy”, que se encarga de traducir la clave pasada en la señal de error, obtener la traducción al idioma seleccionado y mostrar el error en pantalla.

```

//inputs=mensaje
//outputs=mensaje
import com.webWebratio.rtx.RTXConstants
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

Log log = LogFactory.getLog(this.getClass());

if (log.isInfoEnabled())
    log.info("*****");
if (log.isInfoEnabled())
    log.info("* Localize.groovy *****");
if (log.isInfoEnabled()) log.info("ENTRADA:");
if (log.isInfoEnabled()) log.info("mensaje:" + mensaje);

/* OBTIENE EL MENSAJE INTERNACIONALIZADO *****/
if(null != mensaje){
    try{
        String lenguaje = sessionContext.get(RTXConstants.LANGUAGE_CTX_PARAM_KEY);
        String pais = sessionContext.get(RTXConstants.COUNTRY_CTX_PARAM_KEY);
        mensaje = ResourceBundle.getBundle("ApplicationResources", new
            Locale(lenguaje,pais)).getString(mensaje);
    }catch(e){
        mensaje = "???" + mensaje + "??";
    }

    if (log.isInfoEnabled()) log.info("SALIDA:");
    if (log.isInfoEnabled()) log.info("mensaje:" + mensaje);
    if (log.isInfoEnabled())
        log.info("* FIN Localize.groovy *****");
    if (log.isInfoEnabled())
        log.info("*****");
}

return mensaje;

```

```

} else {
    if (log.isInfoEnabled())
        log.info("** FIN Localize.groovy *****");
    if (log.isInfoEnabled())
        log.info("*****");
    return "";
}

```

Código 9.- MostrarErrorLogin.groovy.

La unidad “Login” tiene dos señales más, una que va a la página “Crear Usuario” y otra que va a la página “Contraseña Olvidada”.

Página Crear Usuario.

Contiene la unidad de entrada (Entry Unit) “Crear Usuario”, que contiene los campos `userName` y `email`, y dos señales, “Aceptar” y “Cancelar”, esta última retorna a la página “Inicio”, mientras que la señal “Aceptar” a grandes rasgos sigue el siguiente flujo:

1. Genera el valor para la `password` mediante una unidad de `password` (Password Unit).
2. Mediante la unidad de script “ValidarCampos”, se ejecuta “ValidarCampos.groovy” pasándole los campos `userName`, `email` y `password`. El script se encarga de comprobar que dichos campos tienen valor y no existen ya en la base de datos, si se producen errores en la validación, almacena los mensajes en la entidad virtual “TmpControlMensaje” y los muestra mediante una ventana de error a través del módulo “Mensaje de Error” y retorna a la página “Inicio”. Si no hay errores continúa.
3. Envía la contraseña generada a la dirección de correo proporcionada, mediante una unidad de mail (Mail Unit). Si hay error en el envío, lo muestra con el módulo “Mensaje de Error” y retorna a la página “Inicio”. Si no hay errores continúa.
4. Crea el registro con los datos del nuevo usuario en la entidad “User” mediante una unidad de creación (Create Unit). Si hay error en la creación del registro, lo muestra con el módulo “Mensaje de Error” y retorna a la página “Inicio”. Si no hay errores continúa.
5. Conecta el usuario creado con la entidad “Group” asignando el nuevo usuario al grupo “Usuarios”, mediante una unidad de conexión (Connect Unit), en caso de error se comporta como los puntos anteriores, mostrando el error y retornando a la pantalla “Inicio”. En

caso de ir bien, muestra un mensaje de información con el módulo “Mensaje de Información” y retorna a la pantalla “Inicio”.

Página Contraseña Olvidada.

Contiene la unidad de entrada (Entry Unit) “Contraseña Olvidada”, que contiene un campo email, y dos señales, “Aceptar” y “Cancelar”, esta última retorna a la página “Inicio”, mientras que la señal “Aceptar” sigue el siguiente flujo:

1. Genera el valor para la password mediante una unidad de password (Password Unit).
2. Valida que el correo electrónico introducido pertenece a alguno de los usuarios almacenados en la entidad “User”, mediante el script “ComprobarCorreo.groovy”. Si el script no encuentra el correo electrónico se muestra un error con el módulo “Mensaje de Error” y retorna a la página “Inicio”, en caso contrario continua.
3. Modifica el registro del usuario al que pertenece el correo introducido con la password generada, mediante una unidad de modificación (Modify Unit). En caso de error en la modificación lo muestra con el módulo “Mensaje de Error” y retorna a la página “Inicio”, en caso contrario continua.
4. Envía la contraseña generada a la dirección de correo proporcionada, mediante una unidad de mail (Mail Unit). Si hay error en el envío, lo muestra con el módulo “Mensaje de Error” y retorna a la página “Inicio”. Si no hay errores indica que todo ha ido bien con el módulo “Mensaje de Información” y retorna a la pantalla “Inicio”.

4.3.7. AccesoUsuario y AccesoUsuarioMobile.

Las vistas “AccesoUsuario” y “AccesoUsuarioMobile” son los modelos que definen las páginas accesibles por los usuarios que pertenecen al grupo “Usuario”, es decir, los usuarios normales. Entre la vista “AccesoUsuario” y “AccesoUsuarioMobile” existen dos diferencias, además de la ya mencionada diferencia a nivel de presentación, estas diferencias son:

- “AccesoUsuarioMobile” contiene la página Real-A con el modelado de la parte de Realidad Aumentada, que permite capturar una imagen y

mostrar información al usuario respecto a esta imagen. Esta parte sólo es accesible en esta vista ya que en la parte definida para dispositivos de escritorio queda fuera del alcance del proyecto, aunque es relativamente fácil de implementar el acceso a una Webcam utilizando Flash, pudiéndose tener acceso en tiempo real a las imágenes obtenidas. Ejemplo:

<http://mroth.github.com/cameraform/htdocs/test.html>.

- “AccesoUsuario” contiene una página vacía llamada “Redirección” utilizada para dirigir a la vista correcta, según el tipo de dispositivo que accede a la aplicación.

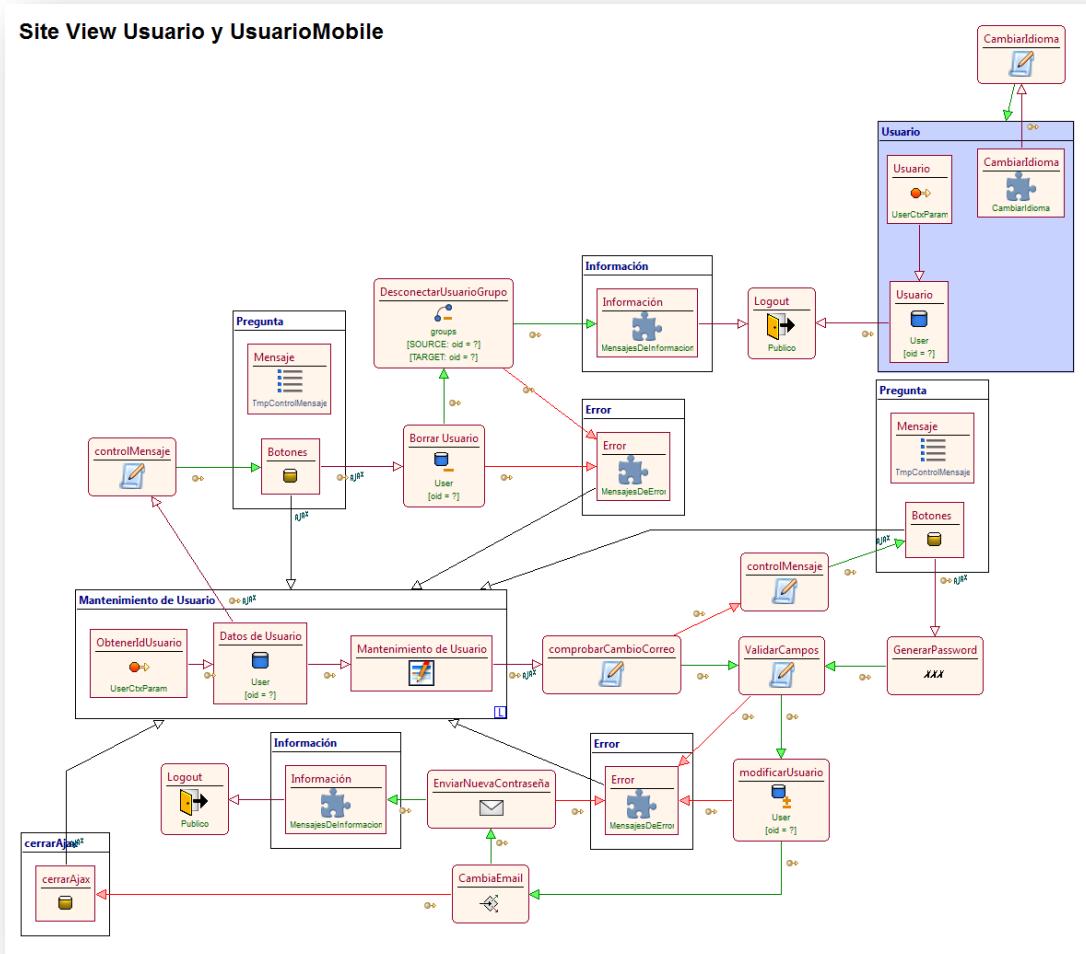


Figura 17.- Site views Usuario y UsuarioMobile (Mantenimiento de usuario).

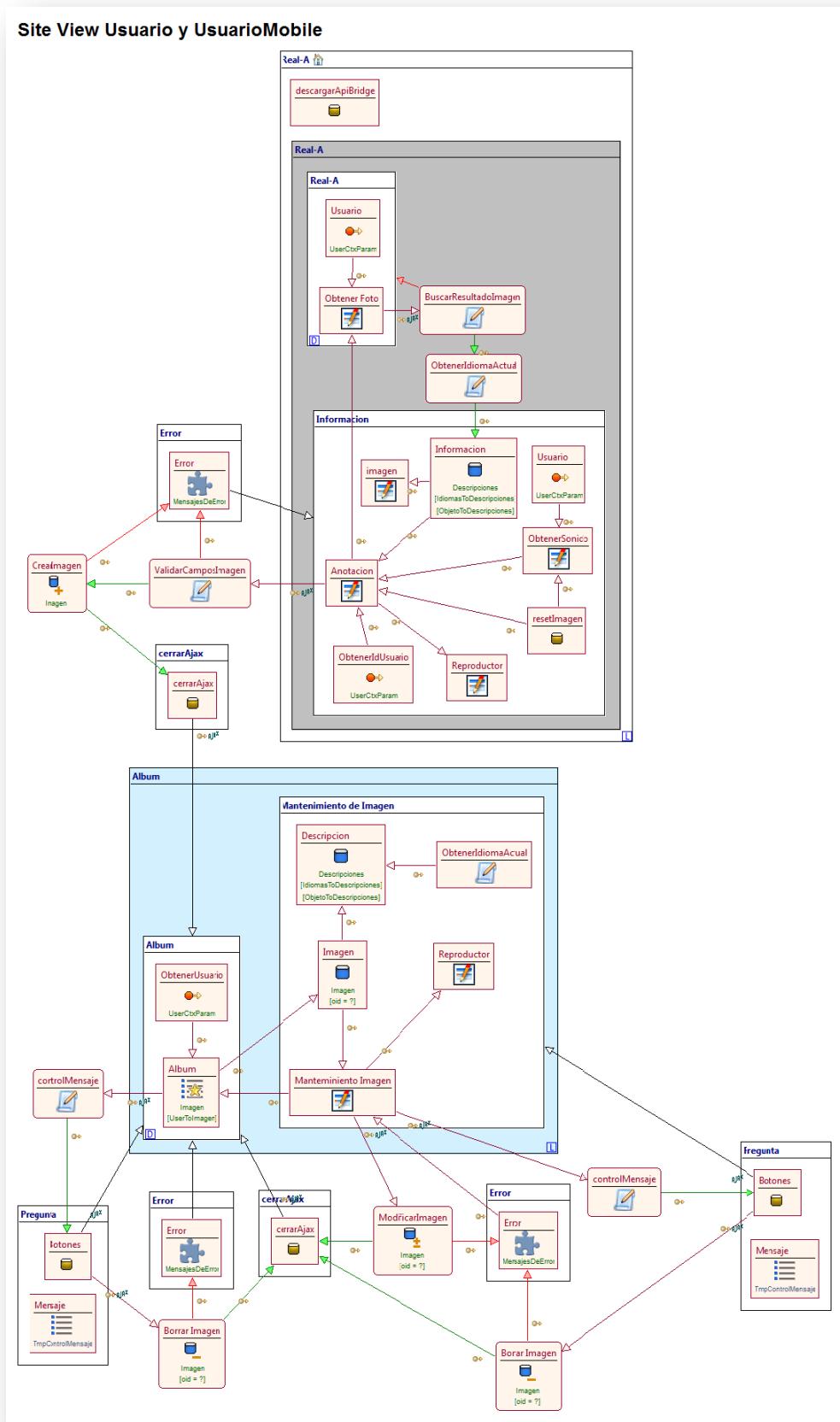


Figura 18.- Site views Usuario y UsuarioMobile (Real-A y Álbum).

En las figuras 17 y 18 se muestra el modelo creado para las vistas de “Usuario”, de las que podemos destacar los siguientes componentes:

Página Real-A.

Es la página principal para la vista “AccesoUsuarioMobile”, y es la que lleva el peso de la Realidad Aumentada. Dicha página contiene una Alternativa (Alternate). Una Alternativa es un contenedor de subpáginas que se muestran una a una, alternativamente. La alternativa contiene dos subpáginas, “Real-A” e “Información”.

En dicha página se ha incluido una unidad sin operación llamada “descargarApiBridge”, para la que se ha implementado una plantilla de diseño especial, la cual solo contiene un botón, desde el cual se permitirá al usuario descargar e instalar el fichero 'APIBridge_v1_1.sis', que instala ApiBrige en el móvil, siendo este imprescindible para el funcionamiento del acceso a la cámara y a la grabadora nativas.

```
#?delimiters [% , %], [=, =]
<input title="onclick="location.href='APIBridge_v1_1.sis'" class="bthAltaPersona"  
id="descargaApiBridge" name="descargaApiBridge" type="button" value="key="descarga.api.bridge"/>">
```

Código 10.- DescargarApiBridge.unit.template

Subpágina Real-A.

Está dentro un contenedor Alternativa (Alternative) en la página Real-A. Esta subpágina es la que accede a la cámara fotográfica del dispositivo móvil para la captura de imágenes. Contiene una unidad Get (Get Unit) para acceder a la variable de contexto UserCtxParam, cuyo valor se le pasa a una unidad de entrada (Entry Unit), llamada “Obtener Foto”, para esta unidad se ha creado un template layout en el que se ha introducido un objeto embebido implementado en Flash Lite 2.0 ActionScript, desde el que se accede a la cámara fotográfica:

```
<tr class="row">
    <td valign="middle" nowrap="nowrap" class=" value">
        <object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" width="1"
            height="1" id="TomarFoto" align="middle">
            <param name="allowScriptAccess" value="sameDomain" />
            <param name="movie" value="[%= nombrePeliculaFlash %]" />
            <param name="quality" value="high" />
            <param name="bgcolor" value="#F6F6F6" />
            <embed id="PeliculaTomarFoto" src="[%= nombrePeliculaFlash %]" quality="high"
                bgcolor="#F6F6F6" width="1" height="1" name="TomarFoto" align="middle"
                allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
                pluginspage="http://www.macromedia.com/go/getflashplayer" />
        </object>
    </td>
</tr>
<tr class="row">
    <td valign="middle" nowrap="nowrap" class=" value">
        <input title="
```

```

        class="btnFoto" type="button" value="
    </td>
</tr>

```

Código 11.- CameraLayout.unit.templete, objeto Flash embebido en html.

El mecanismo de funcionamiento para la captura de imágenes es el siguiente:

1. Cuando se pulsa sobre el botón “btnFoto”, se lanza el evento “onclick” que llama a la función JavaScript “obtenerFoto()”, que es la que se encarga de comunicar a la película flash que debe obtener una fotografía. Además le pasa a la película la ruta del servidor donde se va a realizar la subida del fichero con la imagen y el usuario que está conectado.

El paso de parámetros desde JavaScript a Flash Lite se realiza media la función SetVariable, indicándole que variable Flash se quiere modificar y el valor.

La función comienza llamando a la función MostrarPelículasFlash(), que se encarga de hacer visible la película en la página, ya que en el momento de la ejecución de la función se puede encontrar oculta. La película Flash se muestra y oculta en determinados casos, ya que se produce un pequeño defecto visual al moverse entre pantallas o al realizar el scroll en la pantalla donde se encuentra, defecto que se evita ocultándola y mostrándola dependiendo de la ocasión.

```

/*
 * Función que llama a la pelicula flash para que obtenga la foto.
 */
function obtenerFoto() {
try {
    mostrarPelículasFlash();

    //Pasa la ruta del upload a flash
    window.document.TomarFoto.SetVariable("rutaDelServidor", "[%=urlUpload%]");

    //identificador de usuario
    window.document.TomarFoto.SetVariable("usuario", $('[%=idUsuario%]').value);

    //Se modifica el valor de testValue para que la pelicula lance la cámara
    window.document.TomarFoto.SetVariable("testValue", "ObtenerFoto");

} catch (e) {
    var error = e.toString();
    alert(error);
}
}

```

Código 12.- CameraLayout.unit.templete, comunicación JavaScript - flash.

2. La película Flash, comienza con la obtención de la fotografía cuando se confirma que se ha modificado el valor de la variable “testValue” desde JavaScript y llama a la función takePhoto().

```
//Llamada desde JavaScript para obtener la foto.
var testValue:String = "";
llamadaDesdeJavaScript = function (id, oldval, newval):String {
    takePhoto();
    return newval;
};
//Vigila la variable "testValue" y si cambia, llama a llamadaDesdeJavaScript.
root.watch("testValue", llamadaDesdeJavaScript);
Código 13.- TomarFoto.fla, llamada a takePhoto().
```

3. La función takePhoto, a través de la clase NewFileService de ApiBridge de Nokia, se encarga de ejecutar la aplicación nativa de la cámara del dispositivo, y queda en espera de que dicha aplicación retorne, para ejecutar la función de callback onPhoto.

```
//Llama a la aplicación nativa de la cámara para obtener la foto.
function takePhoto() {
    textoEstado.text = "Obtener fotografía...";
    var inParams = {NewFileType:"Image"};
    //Se crea una instancia de NewFileService que llama a la aplicación de la
    //cámara.
    var fileService = bridge.Service("Service.NewFileService", "IDataSource")
    fileService.TakePhoto(inParams, onPhoto);
}
Código 14.- TomarFoto.fla, función takePhoto().
```

4. Cuando el usuario obtiene la fotografía y finaliza la aplicación nativa, se ejecuta la función onPhoto, si la aplicación nativa retorna un error lo muestra en pantalla. En caso contrario llama a la función fileUpload pasándole como parámetro el nombre del fichero donde se almacenó la foto obtenida.

```
//Callback de obtener foto.
function onPhoto (transactionID:Number, eventID:String, outParam:Object) {
    if (outParam.ErrorCode != 0){
        //Si se produce un error se muestra en pantalla.
        textoEstado.text = "Ocurrió un error mientras se obtenía la foto.";
        pasarEstadoAJavascript(CONSTANTE_ERROR);
        return;
    } else {
        fileUpload(outParam.ReturnValue.Src.toString())

        //Se modifica la ruta del fichero cambiando "\" por "/" para JavaScript
        pathFotografia = cambiarSlash(outParam.ReturnValue.Src.toString());
        return;
    }
}
Código 15.- TomarFoto.fla, función onPhoto().
```

5. La función fileUpload recibe el nombre de la imagen, y se encarga de realizar la subida de dicho fichero al servidor usando el servicio “UploadFile” proporcionado por ApiBrige. Dicha función utiliza la URL que se pasó a la película flash en la variable “rutaDelServidor” y que es la URL del servidor que se encarga de la subida, y la el usuario en

la variable “usuario”. Para la gestión de las subidas de ficheros, se ha creado un servlet en la parte del servidor, como se explica en el apartado “3.3.8.- Servlet UploadFile”.

Tras llamar al servicio “UploadFile” queda en espera a la finalización de éste para ejecutar la función de callback onFileUpload.

```
//Función que sube un fichero al servidor mediante APIBridge
function fileUpload(filePath:String){
    textoEstado.text = "Subiendo fichero con imagen: " + filePath.toString();
    pathUpload = filePath.toString();
    var inParams:Object = new Object();
    inParams.VarName = "file";
    inParams.FileName = filePath.toString();
    inParams.Url = rutaDelServidor;

    var paramUser:Object = new Object();
    paramUser.Name= "usuario";
    paramUser.Value = usuario;

    inParams.Parameters = new Array();
    inParams.Parameters[0] = paramUser;

    bridge.UploadFile(inParams,onFileUpload);
}
```

Código 16.- TomarFoto.fla, función fileUpload().

6. La función onFileUpload se ejecuta cuando termina de la subida del fichero. Si se ha producido un error lo muestra en pantalla, si no, se encarga de indicar que se ha terminado de tomar y subir la foto mediante una llamada a pasarEstadoAJavascript, que a su vez se encarga de llamar a la función homónima JavaScript, dando por finalizada la obtención de la fotografía.

```
//Callback del fileUpload
function onFileUpload (transactionID:Number, eventID:String, outParam:Object)
{
    if (outParam.ErrorCode != 0){
        textoEstado.text = "Ocurrió un error " + outParam.ErrorCode + " en la
        subida del fichero (" + pathUpload + ")";
        return;
    } else {
        textoEstado.text = "Descarga correcta de fichero.";
        pasarEstadoAJavascript(pathFotografia);
        return;
    }
}

//Llamada a JavaScript pasándole el estado final del proceso.
//Si todo ha ido correcto se pasará la ruta del fichero.
function pasarEstadoAJavascript(estado:String) {
    var msg:String = '"' + estado + '"';
    var callJas:String = 'JavaScript:pasarEstadoAJavascript(' + msg + ')';
    getURL(callJas);
    textoEstado.text = "Proceso finalizado.";
}
```

Código 17.- TomarFoto.fla, función onFileUpload() y pasarEstadoAJavascript().

7. La función JavaScript pasarEstadoAJavascript se encarga de cargar el nombre del fichero y la ruta donde se encuentra en los campos “Nombre imagen” e “Imagen” de la unidad de entrada (Entry Unit)

“Obtener Foto”. De esta forma se habría obtenido la fotografía a través del dispositivo móvil y tendríamos los datos necesarios en la unidad indicada. A continuación lanza la señal “FotoObtenida” que pasa el valor de los campos de la unidad de entrada al script “BuscarResultadolImagen”.

```
/*
    Función que se llamada desde la película flash una vez obtenida la foto.
*/
function pasarEstadoAJavaScript(outPut) {
    try {
        if(outPut != CONSTANTE_ERROR) {
            pathFotoObtenida = CONSTANTE_INICIO_RUTA_IMAGEN + outPut;
            nombreFotoSubida = obtenerNombreFichero(pathFotoObtenida);
            pathFotoSubida = "[%= rutaUpload %]" + ${'[%= idUsuario %]'} .value +
                "/" + nombreFotoSubida;

            ${'[%= idFichero %]'} .value = pathFotoSubida;
            ${'[%= idNombreFichero %]'} .value = nombreFotoSubida;
            ${'button#wr:Id context="link"/>'} .click();
            ocultarPelículasFlash();
        }
    } catch (e) {
        var error = e.toString();
        alert(error);
    }
}
```

Código 18.- CameraLayout.unit.templete, función pasarEstadoAJavaScript.

8. BuscarResultadolImagen.groovy se encarga de realizar la consulta a la base de datos para buscar coincidencias con la imagen obtenida. Se forma y ejecuta una query utilizando la función numero_de_coincidencias() creada en PosgreSQL, para obtener la imagen con el mayor número de puntos coincidentes respecto a la imagen obtenida. Además se considera que es necesario un número mayor o igual a 10 para que se pueda considerar coincidencia. A este número mínimo de coincidencias se ha llegado de forma empírica a través de pruebas, de forma que coincidencias por debajo de este umbral, en general no se puede considerar que existe correspondencia.

En caso de no obtener coincidencia se retorna el identificador 999 de la entidad “objeto”, que es un objeto especial creado para indicar que no existe coincidencia utilizando multilenguaje, de forma que se obtiene el mismo comportamiento, exista o no coincidencias.

```
//inputs=imagen|nombreImagen
//outputs=fecha|imagen|nombreImagen|idObjeto
import java.sql.Timestamp
import com.webratio.rtx.core.BeanHelper
import org.apache.commons.lang.math.NumberUtils
import com.webratio.rtx.db.DBTransaction
import com.webratio.rtx.db.HibernateService
import com.webratio.rtx.vdb.VirtualEntityTable
```

```

import com.webratio.webapp.TmpBusqueda
import com.webratio.rtx.RTXBLOBData;
import com.webratio.rtx.blob.BLOBData;
import org.apache.commons.io.FilenameUtils
import com.webratio.units.content.rtx.blob.UploadedBLOBData
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

Log log = LogFactory.getLog(this.getClass());
if (log.isInfoEnabled()) log.info("*****");
if (log.isInfoEnabled()) log.info("* BuscarResultadoImagen.groovy *****");

String rutaEntrada = "/" +
FilenameUtils.separatorsToUnix(FilenameUtils.getPath(rtx.getUploadDirectory()));

String pathCompleto = rutaEntrada + imagen;
Timestamp fecha = new Timestamp(System.currentTimeMillis());

if (log.isInfoEnabled()) log.info("RutaEntrada:" + rutaEntrada);
if (log.isInfoEnabled()) log.info("PathCompleto:" + pathCompleto);
if (log.isInfoEnabled()) log.info("Imagen:" + imagen);
if (log.isInfoEnabled()) log.info("NombreImagen:" + nombreImagen);

pathCompleto = pathCompleto.replace("{", " ");
pathCompleto = pathCompleto.replace("}", " ");

if (log.isInfoEnabled()) log.info("PathCompleto:" + pathCompleto);
def dbId = "db1";
def session = getDBSession(dbId);
//Realiza la búsqueda de las imágenes
def select = "SELECT o.oid, " +
"\\"plantilla\".ruta, \\"plantilla\".imagen, " +
"numero_de_coincidencias(\"plantilla\".oid,:pathCompleto) AS coincidencias " +
"FROM \\"plantilla\" " +
"INNER JOIN \\"objeto\" AS o ON o.oid = \\"plantilla\".objeto_oid " +
"ORDER BY coincidencias DESC LIMIT 1";
if (log.isInfoEnabled()) log.info("Select:" + select);
try{
    def querySeleccion = session.createSQLQuery(select);
    querySeleccion.setParameter("pathCompleto", pathCompleto);
    def result = querySeleccion.list();
    //Genera la tabla temporal con los registros de la query
    for (Iterator iter = result.iterator(); iter.hasNext();) {
        Object[] row = (Object[]) iter.next();
        Integer oid = (Integer) row[0];
        String ruta = (String) row[1];
        String imagen = (String) row[2];
        int coincidencias = (int) row[3];
        if (log.isInfoEnabled()) log.info("coincidencias:" + coincidencias);
        if(coincidencias >= 10){
            idObjeto = oid;
        } else {
            idObjeto = 999; //no correspondencia
        }
    }
    if (log.isInfoEnabled()) log.info("Identificador Objeto:" + idObjeto);
    if (log.isInfoEnabled()) log.info("* FIN BuscarResultadoImagen.groovy ****");
    if (log.isInfoEnabled()) log.info("*****");
    return ["resultCode":"success", "idObjeto":idObjeto, "fecha":fecha,
"imagen":imagen, "nombreImagen":nombreImagen]
}catch(e){
    if (log.isInfoEnabled()) log.info("ERROR:" + e);
    return ["resultCode":"error"]
}
}

```

Código 19.- BuscarResultadoImagen.groovy.

9. A continuación se obtiene el idioma actual a través del script ObtenerIdiomaActual.groovy.

```

//outputs=idIdioma
import com.webratio.rtx.db.DBTransaction
import com.webratio.rtx.db.HibernateService
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import com.webratio.rtx.RTXConstants

```

```

def leng = sessionContext.get(RTXConstants.LANGUAGE_CTX_PARAM_KEY);
def pais = sessionContext.get(RTXConstants.COUNTRY_CTX_PARAM_KEY);
def idIdioma = null;

Log log = LogFactory.getLog(this.getClass());
if (log.isInfoEnabled()) log.info("*****");
if (log.isInfoEnabled()) log.info("* ObtenerIdiomaActual.groovy *****");
if (log.isInfoEnabled()) log.info("Variables de sesión:");
if (log.isInfoEnabled()) log.info("Lenguaje:" + leng);
if (log.isInfoEnabled()) log.info("Pais:" + pais);

def dbId = "db1";
def session = getDBSession(dbId);

//Realiza la busqueda de las imagenes
def select = "SELECT oid FROM idioma"
if((null != leng && "" != leng) || (null != pais && "" != pais)){
    select = select + " WHERE"
    if(null != leng && "" != leng){
        select = select + " language = :idioma ";
    }
    if(null != leng && "" != leng) && (null != pais && "" != pais){
        select = select + " AND"
    }
    if(null != pais && "" != pais){
        select = select + " country = :pais";
    }
}
if (log.isInfoEnabled()) log.info("Query:" + select);

def querySeleccion = session.createSQLQuery(select);

if(null != leng && "" != leng){
    querySeleccion.setParameter("idioma", leng);
}
if(null != pais && "" != pais){
    querySeleccion.setParameter("pais", pais);
}

idIdioma = querySeleccion.uniqueResult();

if (log.isInfoEnabled()) log.info("SALIDA:");
if (log.isInfoEnabled()) log.info("idIdioma:" + idIdioma);
if (log.isInfoEnabled()) log.info("* FIN ObtenerIdiomaActual.groovy *****");
if (log.isInfoEnabled()) log.info("*****");

return ["resultCode":"success", "idIdioma":idIdioma]

```

Código 20.- ObtenerIdiomaActual.groovy.

10. Finalmente se pasan todos los datos obtenidos (identificador de la imagen, idioma actual, nombre del fichero a la subpágina “Información”.

Subpágina Información.

Una vez obtenida la imagen y la información relativa a la imagen, se muestra en la subpágina “Información”. Esta página se compone una unidad de datos (Data Unit) que se encarga de obtener la descripción de la entidad “Descripciones” a partir del identificador del “objeto” y el identificador del “idioma” pasados en el proceso anterior.

Además la subpágina contiene las siguientes unidades:

- **Entry Unit “Imagen”:** Entidad que se encarga de mostrar la imagen obtenida por el usuario.
- **Entry Unit “ObtenerSonido”:** Esta unidad se encarga de obtener la grabación de un sonido a través de ApiBridge y la aplicación nativa de grabación del dispositivo móvil. El funcionamiento es casi idéntico a “Obtener Foto”, utilizando ActionScript y JavaScript, sólo modificando el parámetro que se le pasa al servicio de la clase NewFileService de ApiBridge en la función takePhoto() en la película Flash (en este caso GrabarSonido.fla):

```
var inParams = {NewFileType:"Audio"};
```

Código 21.- Parámetro para grabar sonido mediante la clase NewFileService.

- **Entry Unit “Reproductor”:** Para esta unidad también se ha creado un template layout especial, que mediante una película flash se encarga de reproducir el sonido grabado y subido al servidor mediante la unidad “ObtenerSonido”. La película flash creada para tal efecto es ReproducirSonido.fla.
- **Entry Unit “Anotación”:** Esta unidad contiene un campo “Anotación” donde se permite al usuario introducir un texto relativo a la imagen obtenida. Además canaliza la información del resto de unidades para poder grabar un registro en la entidad “Imagen”. Esta unidad tiene dos señales de salida, “Nueva Foto” que se encarga de volver a la subpágina “Real-A” para la obtención de una nueva fotografía, y “Guardar en Álbum” que se encarga de validar los campos pasados mediante el script “ValidarCamposImagen.groovy”, guardar el registro en la entidad “Imagen” mediante una Unidad Crear (Create Unit) y acceder a la página “Álbum” si todo ha ido bien, o mostrar el error correspondiente si se produjo algún error en el flujo.

Página Álbum.

La página contiene un una Unidad Índice (Index Unit), llamada “Álbum”, donde se cargan los registros de la entidad “Imagen” pertenecientes al usuario conectado, dicho usuario se obtiene del la variable de contexto “UserCtxParam”, de forma el usuario sólo tenga acceso a los registros creados por él.

La página contiene además dos enlaces. El primer enlace llamado “Borrar Imagen” que se encarga procesar el borrado de la entrada seleccionada de la entidad “Imagen”. El segundo enlace llamado “Mantenimiento” se encarga de ir a la página “Mantenimiento de Imagen” pasándole la entrada seleccionada en la unidad índice.

Página Mantenimiento de Imagen.

Esta página recibe el identificador de la imagen seleccionada en la Unidad Índice de la página anterior, carga la imagen mediante una unidad de datos (Data Unit) que hace referencia a la entidad “Imagen”, carga la información mediante una Unidad de Datos (Data Unit) referente a la entidad “Descripciones”, obteniendo el registro relacionado con la imagen, en el idioma seleccionado en ese momento, que es obtenido mediante el script “ObtenerIdiomaActual.groovy”.

La página contiene tres enlaces. El primero enlace, llamado “Cancelar”, que retorna el control a la página “Álbum” sin realizar ninguna operación. Un segundo enlace llamado “Modificar” que inicia el flujo que procesa la modificación del registro cargado. Y finalmente, el tercer enlace, llamado “Borrar Imagen”, que se de iniciar el flujo que procesa el borrado del registro.

Página Mantenimiento de Usuario.

Esta página contiene las unidades de contenido, señales y flujos que permiten gestionar el registro de la entidad “User” perteneciente al usuario conectado, es decir el usuario cuyo identificador se encuentra en la variable de contexto “UserCtxParam”.

Esta página contiene los flujos y operaciones necesarios para modificar todos los datos del usuario, y eliminar el propio usuario. Reseñar dos puntos importantes en estas operaciones:

- La modificación de la cuenta de correo implica la generación de una nueva contraseña, el envío de dicha contraseña y la desconexión del usuario mediante una Unidad Logout (Logout Unit).
- La eliminación de un usuario finaliza con la desconexión del usuario mediante una Unidad Logout (Logout Unit).

Página Maestra Usuario.

La página maestra (Master Page) “Usuario” aparece en el resto de páginas de la vista. Contiene una unidad de módulo (Module Unit) que hace referencia al módulo “Cambiar Idioma” que, junto con el script “CambiarIdioma.groovy”, permite cambiar el idioma seleccionado en la aplicación.

Además muestra la información del usuario conectado mediante la Unidad de Datos (Data Unit) “Usuario” y permite la desconexión de dicho usuario mediante el enlace “Logout” y una Unidad Logout (LogOut Unit).

4.3.8. Servlet UploadFile.

La necesidad de subir ficheros desde el cliente, ya sea una imagen obtenida a través de la cámara, o un sonido a través de la grabadora, hace necesario este servlet, que se encarga de este proceso.

Se crea la clase UploadFileServlet que extiende de HttpServlet, implementando el método doPost como sigue:

```
@SuppressWarnings("unchecked")
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    if (log.isInfoEnabled()) log.info("*****");
    if (log.isInfoEnabled()) log.info("* UploadFileServlet.doPost *****");

    ServletContext contexto = this.getServletContext();
    DefaultRTXManager defaultRTXManager = (DefaultRTXManager)
        contexto.getAttribute(WRGlobals.RTX_KEY);

    PrintWriter out = response.getWriter();

    response.setContentType("text/html");

    int estado = HttpServletResponse.SC_OK;
    String textoEstado = "OK";

    String realPathTemporal = getServletContext().getRealPath(TMP_DIR_PATH);
    File tmpDir = new File(realPathTemporal);
    if(!tmpDir.isDirectory()) {
        estado = HttpServletResponse.SC_INTERNAL_SERVER_ERROR;
        textoEstado = "INTERNAL_SERVER_ERROR";
        response.setStatus(estado);
        out.write(textoEstado);
        out.close();
        throw new ServletException(TMP_DIR_PATH + " no es un directorio");
    }

    String realPathDestino = defaultRTXManager.getUploadDirectory();
    File destinationDir = new File(realPathDestino);
    if(!destinationDir.isDirectory()) {
        estado = HttpServletResponse.SC_INTERNAL_SERVER_ERROR;
        textoEstado = "INTERNAL_SERVER_ERROR";
        response.setStatus(estado);
        out.write(textoEstado);
        out.close();
        throw new ServletException(realPathDestino + " no es un directorio");
    }

    DiskFileItemFactory fileItemFactory = new DiskFileItemFactory();

    //Umbral de tamaño, por encima del cual el contenido se almacena en el disco.
```

```

fileItemFactory.setSizeThreshold(10*1024*1024); //1 MB

//Directorio temporal para almacenar ficheros por encima del umbral
fileItemFactory.setRepository(tmpDir);

ServletFileUpload uploadHandler = new ServletFileUpload(fileItemFactory);
try {
    //Analiza la solicitud
    List items = uploadHandler.parseRequest(request);
    for (Iterator iterator = items.iterator(); iterator.hasNext();) {
        FileItem item = (FileItem) iterator.next();

        if(item.isFormField()) {
            //Se controlan los campos de formulario
            if (log.isInfoEnabled()) log.info("Nombre del campo....:" +
                item.getFieldName());
            if (log.isInfoEnabled()) log.info("Valor.....:" +
                item.getString());

            String campo = null ==
                item.getFieldName()?null:item.getFieldName().replaceAll("<data>", "");
            campo = null == campo?null:campo.replaceAll("</data>", "");
            String valor = null ==
                item.getString()?null:item.getString().replaceAll("<data>", "");
            valor = null == valor?null:valor.replaceAll("</data>", "");
            if(null != campo && campo.equals("usuario")){
                destinationDir = new File(realPathDestino,valor);
                if(!destinationDir.exists()){
                    if(!destinationDir.mkdir()){
                        estado = HttpServletResponse.SC_INTERNAL_SERVER_ERROR;
                        textoEstado = "INTERNAL SERVER ERROR";
                        response.setStatus(estado);
                        out.write(textoEstado);
                        out.close();
                        throw new ServletException("Error al crear directorio del
                            usuario");
                    }
                }
            } else {
                //Se manejan los ficheros a subir
                if (log.isInfoEnabled()) log.info("Nombre de campo....:" +
                    item.getFieldName());
                if (log.isInfoEnabled()) log.info("Nombre de fichero.." + item.getName());
                if (log.isInfoEnabled()) log.info("Tipo de contenido.." +
                    item.getContentType());
                if (log.isInfoEnabled()) log.info("Tamaño.....:" + item.getSize());

                //Se escribe el fichero al directorio de destino
                File fichero = new File(item.getName());
                String nombreFichero = fichero.getName();
                File file = new File(destinationDir, nombreFichero);
                item.write(file);
            }
        }
    }catch(FileUploadException ex) {
        estado = HttpServletResponse.SC_INTERNAL_SERVER_ERROR;
        textoEstado = "INTERNAL SERVER ERROR";
        if (log.isInfoEnabled()) log.error("Error encontrado durante el analisis de
            solitud. ",ex);
    } catch(Exception ex) {
        estado = HttpServletResponse.SC_INTERNAL_SERVER_ERROR;
        textoEstado = "INTERNAL SERVER ERROR";
        if (log.isInfoEnabled()) log.error("Error encontrado durante la descarga del
            fichero. ",ex);
    }

    response.setStatus(estado);
    out.write(textoEstado);
    out.close();

    if (log.isInfoEnabled()) log.info("* FIN UploadFileServlet.doPost *****");
    if (log.isInfoEnabled()) log.info("*****");
}

```

Código 22.- método doPost de la clase UploadFileServlet.

Se configura el fichero WEB-INF/web.xml, definiendo el servlet para la clase UploadFileServlet, y se mapea con el patrón de url que hará que se ejecute, de esta forma se añaden a web.xml las entradas que se muestran a continuación:

```
<servlet>
    <servlet-name>UploadFileServlet</servlet-name>
    <servlet-class>es.real.a.uploadfile.UploadFileServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>UploadFileServlet</servlet-name>
    <url-pattern>/UploadFileServlet</url-pattern>
</servlet-mapping>
```

Código 23.- Definición de UploadFileServlet en WEB-INF/web.xml.

De esta forma, se podrá utilizar el servicio de upload a través de la URL http://ruta_servidor/Real-A/UploadFileServlet, siendo esta URL la que se pasa por parámetro a la película flash “TomarFoto.fla” y en “GrabarSonido.fla” para utilizarla en el servicio “UploadFile” proporcionado por ApiBrige como dirección de servidor de upload.

Como directorio de descarga se utiliza la ruta .\upload\id_usuario, a partir de la ruta donde esté desplegada la aplicación, id_usuario es el identificador del usuario conectado, de forma, que cada usuario tendrá su propio directorio, evitando así, que dos usuarios puedan subir un fichero con el mismo nombre a la vez, provocando la pérdida de uno de ellos y propiciando que el usuario cuya imagen no ha sido salvada correctamente, pueda acceder a la imagen obtenida por el otro.

4.3.9. AccesoAdministrador y AccesoAdministrador Mobile.

Las vistas “AccesoAdministrador” y “AccesoAdministrador Mobile” contienen los modelos que definen las páginas accesibles por los usuarios que pertenecen al grupo “Administrador”.

En las siguientes figuras se muestra el modelo de los sitios “AccesoAdministrador” ya que ambas vistas son similares, existiendo sólo diferencias a nivel de presentación, junto con la página vacía llamada “Redirección” existente únicamente en la vista “AccesoAdministrador”, utilizada para dirigir a la vista correcta, según el tipo de dispositivo que accede a la aplicación.

El diseño de estas vistas del sitio se muestra de la figura 19 a la 22.

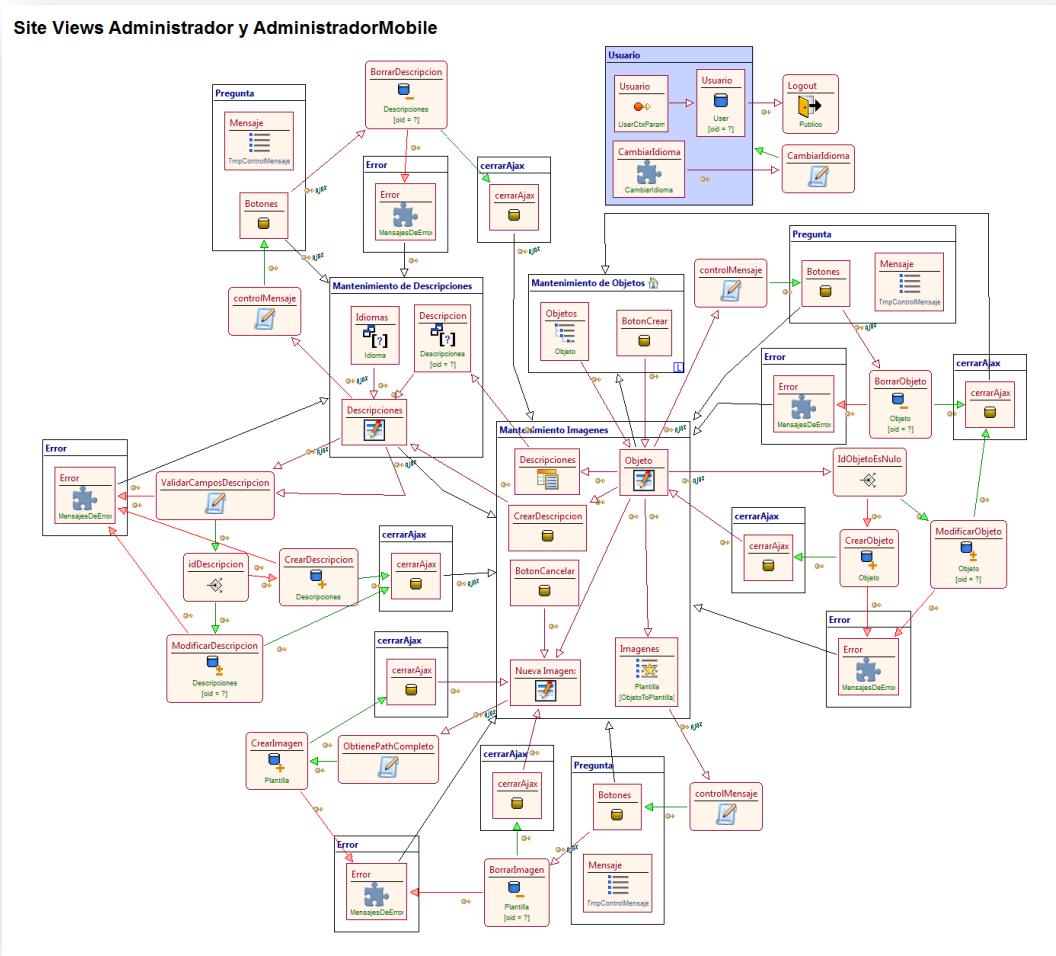


Figura 19.- Site views Administrador y AdministradorMobile (Mantenimiento de Objetos, Mantenimiento de Descripciones y Mantenimiento de Imágenes).

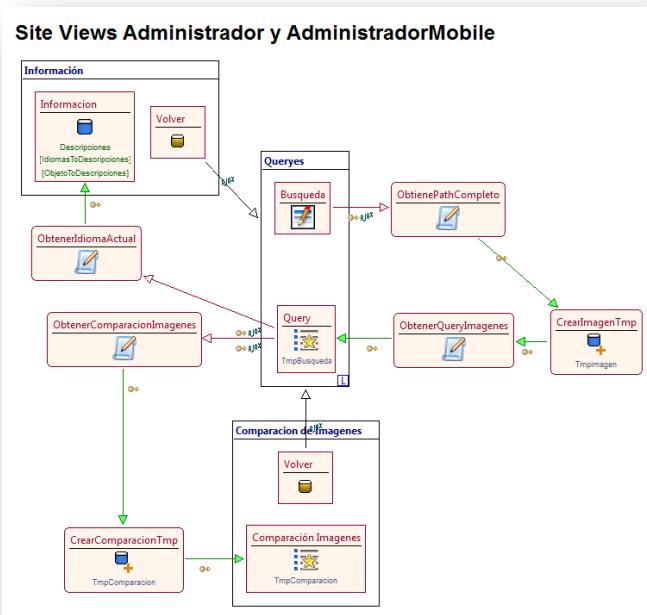


Figura 20.- Site views Administrador y AdministradorMobile (Querys).

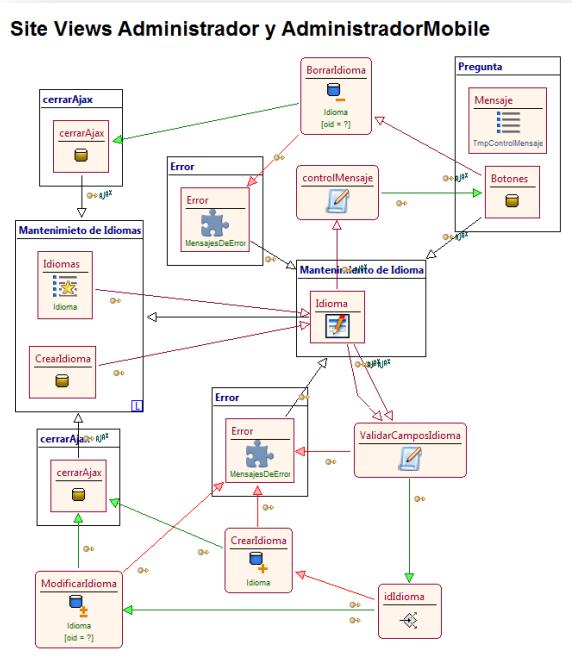


Figura 21.- Site views Administrador y AdministradorMobile (Mantenimiento de idiomas).

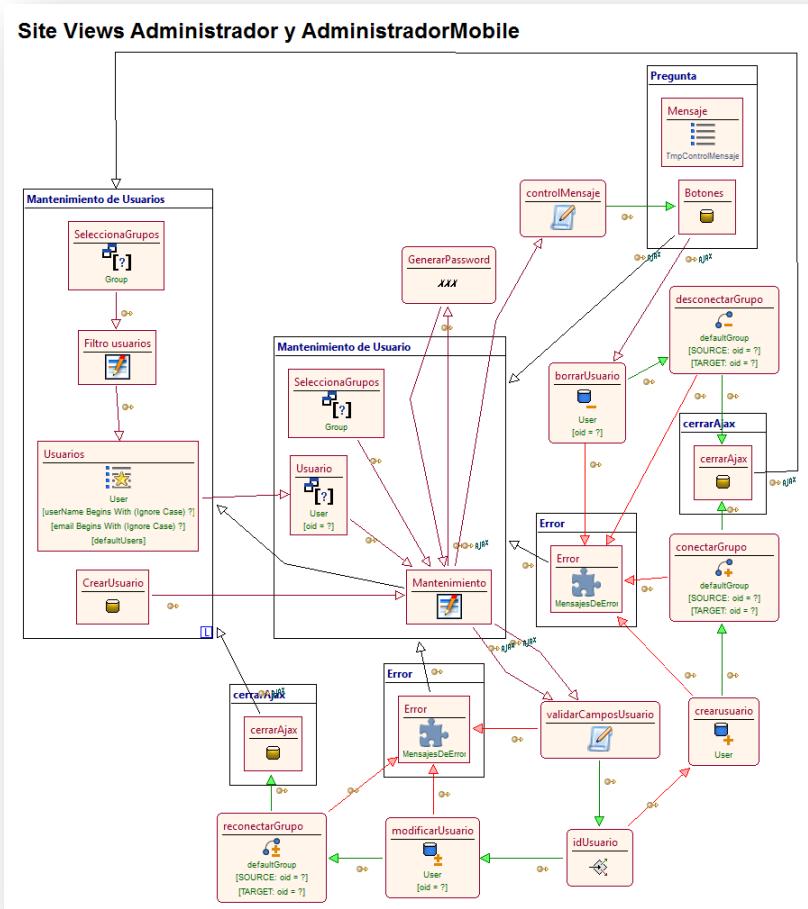


Figura 22.- Site views Administrador y AdministradorMobile (Mantenimiento de Usuarios).

Página Mantenimiento de Objetos.

Permite el mantenimiento de la entidad “Objeto”, con contiene los objetos para la comparación de imágenes.

Contiene una unidad de índice jerárquica (Hierarchical Index Unit), con dos niveles, el principal que hace referencia a la entidad “Objeto”, donde se cargan todos los registros de dicha entidad, y un segundo nivel, con referencia a la entidad “Plantilla”, donde se cargan las imágenes almacenadas en dicha entidad, relacionada con el registro “Objeto” padre.

Esta unidad tiene una señal llamada “Modificar” que envía el identificador y la descripción del registro “Objeto” seleccionado en un momento dado a la pantalla “Mantenimiento de Imágenes” para su modificación, es decir, entraría en dicha pantalla en modo modificación.

También contiene una Unidad Sin Operación (No Op Unit), con una señal llamada “Crear” hacia la pantalla “Mantenimiento de Imágenes”, sin pasar ningún tipo de información, con lo que entraría en dicha pantalla en modo creación.

Página Mantenimiento de Imágenes.

Como hemos visto, a esta página se llegará desde la anterior en dos modos distintos, creación o modificación. Esta página contiene una Unidad de Entrada (Entry Unit) llamada “Objeto”, con dos campos, “id” y “descripción”. Contiene una Unidad Índice Mejorada (Power Index Unit), llamada “Imagenes”, que está asociada a la entidad “Plantilla” donde se cargan las imágenes asociadas al objeto. Una segunda Unidad de Entrada (Entry Unit), llamada “Nueva Imagen”, con un campo “Imagen” de tipo Blod, que nos permitirá subir una nueva imagen.

Además contiene una Unidad de Consulta (Query Unit), llamada “Descripciones”, que carga información de las relaciones entre la entidad “Objeto” y las entidades “Idioma” y “Descripciones”, de forma que se obtiene las descripciones junto a su idioma asociados a un objeto, utilizando la siguiente query en HQL:

```
Select d.oid as descripcionOid,
       i.descripcion as Idioma,
       d.descripcion as Descripcion
  from Objeto o
 inner join o.objetoToDescripciones d
 inner join d.descripcionesToIdiomas i
```

```
where o.id = :objectId
```

Código 24.- Query HQL que obtiene las descripciones e idioma asociadas a un objeto.

La Unidad de Entrada (Entry Unit) “Objeto”, tiene tres enlaces de salida, uno llamado “Aceptar”, que se encarga de comenzar el flujo para crear o modificar un registro “objeto”, dependiendo si se está en modo de creación o modificación, esto lo detecta a través de una Unidad No Es Nulo (Is Not Null Unit), dependiendo de si el identificador pasado desde la página “Mantenimiento de Objetos” es nulo o no. El segundo flujo, llamado “Eliminar”, se encarga de iniciar el flujo para eliminar el registro “Objeto” cuyo identificador es igual al pasado desde la página anterior. Finalmente, el enlace “Cancelar” retorna el control a la página “Mantenimiento de Objetos” sin realizar ninguna acción.

La Unidad de Entrada (Entry Unit) “Nueva Imagen” tiene un enlace de salida, llamado “Guardar”, que se lanza el flujo encargado de guardar una nueva imagen asociada al registro “Objeto” que se está modificando.

La Unidad de Consulta (Query Unit) “Descripciones” tiene un enlace de salida, llamado “Modificar” que envía el identificador de la descripción a modificar y deja el control a la página “Mantenimiento de Descripciones”.

También existe una Unidad Sin Operación (No Op Unit) con un enlace de salida llamado “CrearDescripcion” que cede el control a la misma página en modo de creación, que permite crear nuevas descripciones asociadas al objeto que se está modificando.

La Unidad Índice Mejorada (Power Index Unit) “Imágenes” tiene una única señal, “Borrar”, que inicia el flujo que realiza las operaciones oportunas para borrar la imagen seleccionada de la entidad “Plantilla”.

Página Mantenimiento de Descripciones.

Esta página modela la gestión de las entidades “Descripciones” asociadas a un registro de la entidad “Objeto”. Tendrá dos modos de funcionamiento:

Uno, si recibe el identificador de un registro de la entidad “Descripciones”, enviado desde la página anterior estará en modo edición, permitiendo los flujos de operación encargados de modificar y borrar dicho registro.

Dos, si no se recibe dicho identificador, estará en modo creación, permitiendo únicamente el flujo para el alta de un nuevo registro.

Esta página permite también retornar el control a la anterior sin realizar ninguna operación a través de un enlace llamado “Cancelar”.

Página Mantenimiento de Idiomas.

Esta página contiene una Unidad Índice Mejorada (Power Index Unit) con los datos de la entidad “Idioma”, permite la modificación en dicha unidad, mediante un enlace llamado “Modificar” pasándole el identificador del registro seleccionado a la página “Mantenimiento de Idiomas”.

Y mediante Unidad Sin Operación (No Op Unit) con un enlace de salida llamado “Crear Idioma” que cede el control a la misma página en modo de creación, que permite crear nuevos registros.

Página Mantenimiento de Idioma.

Esta página modela la gestión de la entidad “Idioma”.

Si recibe el identificador de un registro de dicha entidad, enviado desde la página anterior, permite los flujos de operación encargados de modificar y borrar dicho registro. Si no se recibe dicho identificador, permite únicamente el flujo para el alta de un nuevo registro.

Esta página permite también retornar el control a la anterior sin realizar ninguna operación a través de un enlace llamado “Cancelar”.

Página Mantenimiento de Usuarios.

El modelado de esta página es muy similar al de las páginas “Mantenimiento de Objetos” y “Mantenimiento de Idiomas”, y permite crear un nuevo usuario o seleccionar un registro de la entidad “User” para su modificación.

Página Mantenimiento de Usuario.

Modelado muy similar al de las páginas “Mantenimiento de Objeto” y “Mantenimiento de Idioma”, permitiendo los flujos que realizan la operación de modificar y borrar una nueva entidad “User” si se ha recibido el identificador del usuario, o crear un nuevo registro en caso de no recibir ningún identificador.

Página Querys.

Esta página se ha diseñado con la idea de poder probar el reconocimiento de imágenes mediante búsqueda de una imagen en la base de datos.

Se compone de una Unidad de Entrada (Entry Unit) denominada “Búsqueda”, con un campo “Imagen” de tipo blob que permite subir un fichero con la imagen que se desea comparar.

El enlace “Buscar” inicia el flujo que realiza la query para buscar las coincidencias de la imagen con las imágenes almacenadas en base de datos.

En este proceso destaca el script encargado de realizar dicha query:

```
//inputs=imagenEntrada

import com.webratio.rtx.core.BeanHelper
import org.apache.commons.lang.math.NumberUtils
import com.webratio.rtx.db.DBTransaction
import com.webratio.rtx.db.HibernateService
import com.webratio.rtx.vdb.VirtualEntityTable
import com.webratio.webapp.TmpBusqueda
import com.webratio.webapp.TmpComparacion
import com.webratio.rtx.RTXBLOBData;
import com.webratio.rtx.blob.BLOBData;
import org.apache.commons.io.FilenameUtils
import com.webratio.units.content.rtx.blob.UploadedBLOBData
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

Log log = LogFactory.getLog(this.getClass());

if (log.isInfoEnabled()) log.info("*****");
if (log.isInfoEnabled()) log.info("* ObtenerQueryImagenes.groovy *****");
if (log.isInfoEnabled()) log.info("ENTRADA:");
if (log.isInfoEnabled()) log.info("imagenEntrada:" + imagenEntrada);

String rutaQuery = "/" +
FilenameUtils.separatorsToUnix(FilenameUtils.getPath(rtx.getUploadDirectory()));
String imagenQuery = "" + imagenEntrada;
String pathCompleto = rutaQuery + imagenEntrada;

pathCompleto = pathCompleto.replace("{", "");
pathCompleto = pathCompleto.replace("}", "");

imagenQuery = imagenQuery.replace("{", "");
imagenQuery = imagenQuery.replace("}", "");

if (log.isInfoEnabled()) log.info("rutaQuery:" + rutaQuery);
if (log.isInfoEnabled()) log.info("imagenQuery:" + imagenQuery);
if (log.isInfoEnabled()) log.info("pathCompleto:" + pathCompleto);

def dbId = "db1";
def session = getDBSession(dbId);

def tmpBusquedaEntityId = "ent5";
VirtualEntityTable tmpBusquedaTable = new VirtualEntityTable(tmpBusquedaEntityId,
sessionContext, service)

//Borrado de la tabla temporal
tmpBusquedaTable.delete(tmpBusquedaTable.getRows());

def tmpComparacionEntityId = "ent7";
VirtualEntityTable tmpComparacionTable = new
VirtualEntityTable(tmpComparacionEntityId, sessionContext, service)

//Borrado de la tabla temporal
tmpComparacionTable.delete(tmpComparacionTable.getRows());

//Realiza la busqueda de las imagenes
```

```

def select = "SELECT o.oid, o.descripcion, " +
    "p.ruta, p.imagen, " +
    "numero_de_coincidencias(p.oid,:pathCompleto) AS coincidencias " +
    "FROM \"plantilla\" AS p " +
    "INNER JOIN \"objeto\" AS o ON o.oid = p.objecto_oid " +
    "ORDER BY coincidencias DESC LIMIT 10";

def querySeleccion = session.createSQLQuery(select);
querySeleccion.setParameter("pathCompleto", pathCompleto);
def result = querySeleccion.list();

//Genera la tabla temporal con los registros de la query
for (Iterator iter = result.iterator(); iter.hasNext();) {
    Object[] row = (Object[]) iter.next();
    Integer idObjeto = (Integer) row[0];
    String descripcion = (String) row[1];
    String ruta = (String) row[2];
    String imagen = (String) row[3];
    Integer coincidencias = (Integer) row[4];

    RTXBLOBData blobFile = new BLOBData(imagen, rtx);
    RTXBLOBData blobImagenEntrada = new BLOBData(imagenQuery, rtx);

    TmpBusqueda tmpBusqueda = new TmpBusqueda();
    tmpBusqueda.setIdobjeto(idObjeto);
    tmpBusqueda.setDescripcion(descripcion);
    tmpBusqueda.setRuta(ruta);
    tmpBusqueda.setImagen(blobFile);
    tmpBusqueda.setCoincidencias(coincidencias);
    tmpBusqueda.setRutaQuery(rutaQuery);
    tmpBusqueda.setImagenQuery(blobImagenEntrada);

    tmpBusquedaTable.save(tmpBusqueda);
}

if (log.isInfoEnabled()) log.info("* FIN ObtenerQueryImagenes.groovy *****");
if (log.isInfoEnabled()) log.info("*****");

return ["resultCode":"success"]

```

Código 25.- ObtenerQueryImagenes.groovy.

Este script busca las 10 imágenes que tienen más puntos coincidentes con respecto a la imagen buscada. Por cada una de estas imágenes crea un registro en la entidad virtual “TmpBusqueda”, que es cargada a su vez en una Unidad Índice Mejorada (Power Index Unit) llamada “Query”, donde se mostrarán los datos.

Esta unidad tiene dos enlaces, “verInformacion” que permite mostrar la información relativa a la imagen seleccionada, mediante la página “Información”, y “Comparar Imágenes” que crea una imagen que contiene una comparativa de la imagen buscada y la imagen seleccionada de las obtenidas en la búsqueda y las muestra en la página “Comparación de imágenes”. Para la obtención de la imagen comparativa se utiliza el siguiente script:

```

//inputs=rutaImagen1|nombreImagen1|rutaImagen2|nombreImagen2
//outputs=rutaSalida|nombreSalida

import com.sun.org.apache.xml.internal.serializer.ToStream
import com.webratio.rtx.core.BeanHelper
import org.apache.commons.lang.math.NumberUtils
import com.webratio.rtx.db.DBTransaction
import com.webratio.rtx.db.HibernateService
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

```

```

Log log = LogFactory.getLog(this.getClass());

if (log.isInfoEnabled()) log.info("*****");
if (log.isInfoEnabled()) log.info("* ObtenerComparacionImagenes.groovy *****");
if (log.isInfoEnabled()) log.info("ENTRADA:");
if (log.isInfoEnabled()) log.info("rutaImagen1:" + rutaImagen1);
if (log.isInfoEnabled()) log.info("nombreImagen1:" + nombreImagen1);
if (log.isInfoEnabled()) log.info("rutaImagen2:" + rutaImagen2);
if (log.isInfoEnabled()) log.info("nombreImagen2:" + nombreImagen2);

String rutaSalida = rutaImagen2;
String nombreSalida = nombreImagen2;
int dotPos = nombreSalida.lastIndexOf(".");
String extension = nombreSalida.substring(dotPos);
Calendar fecha = new GregorianCalendar();
nombreSalida = nombreSalida.replace(extension, fecha.getTimeInMillis().toString() +
extension);

String pathCompleto1 = rutaImagen1 + nombreImagen1;
String pathCompleto2 = rutaImagen2 + nombreImagen2;
String pathSalida = rutaSalida + nombreSalida;

pathCompleto1 = pathCompleto1.replace("{", "");
pathCompleto1 = pathCompleto1.replace("}", "");
pathCompleto2 = pathCompleto2.replace("{", "");
pathCompleto2 = pathCompleto2.replace("}", "");

def dbId = "db1";
def session = getDBSession(dbId);

//Realiza la busqueda de las imagenes
def select = "SELECT obtener_imagen_coincidencias(:pathCompleto1, :pathCompleto2,
:pathSalida)";

def querySeleccion = session.createSQLQuery(select);
querySeleccion.setParameter("pathCompleto1", pathCompleto1);
querySeleccion.setParameter("pathCompleto2", pathCompleto2);
querySeleccion.setParameter("pathSalida", pathSalida);
def coincidencias = querySeleccion.uniqueResult();

if (log.isInfoEnabled()) log.info("SALIDA:");
if (log.isInfoEnabled()) log.info("rutaSalida:" + rutaSalida);
if (log.isInfoEnabled()) log.info("nombreSalida:" + nombreSalida);
if (log.isInfoEnabled()) log.info("* FIN ObtenerComparacionImagenes.groovy *****");
if (log.isInfoEnabled()) log.info("*****");

return ["resultCode":"success", "rutaSalida":rutaSalida, "nombreSalida":nombreSalida]

```

Código 26.- ObtenerComparacionImagenes.groovy.

Este script obtiene un fichero con la comparativa utilizando la función almacenada que se ha creado para tal efecto en la base de datos PostgreSQL, tal y como se explica en el punto 3.2.3.

Con la imagen creada se crea un registro en la entidad virtual “TmpComparación”.

Página Información.

El diseño de esta página se ha realizado para que se muestre en un popup AJAX.

Contiene una Unida de Datos (Data Unit) relacionada con la entidad “Descripciones”, donde se cargará el registro de la descripción de la imagen

seleccionada, en el idioma actual que previamente ha sido obtenido mediante el script “ObtenerIdiomaActual”.

Para cerrar el popup AJAX, existe una Unidad Sin Operación (No Op Unit) que retorna el control a la página “Querys” mediante una señal llamada “Cerrar”.

Página Comparación de Imágenes.

Esta página, diseñada como un popup AJAX como la anterior, muestra el registro de la entidad “TmpComparación” en una Unidad Índice Mejorada (Power Index Unit). El registro mostrado, que contiene la imagen comparativa, es creado en el flujo de datos que inicia el enlace “Comparar Imágenes” de la página “Query”, que da acceso a esta.

Permite volver a “Query” mediante una Unidad Sin Operación (No Op Unit) y un enlace llamado “Cerrar”.

Página Maestra Usuario.

Esta página está modelada igual que la página maestra “Usuario” en la vista “AccesoUsuario” y “AccesoUsuarioMobile”, teniendo las mismas funcionalidades.

4.3.10. Nivel de presentación.

En WebRatio, los elementos gráficos de la aplicación se manejan a través de proyectos de estilo (style projects), que contienen las plantillas de diseño (layout templates) para las unidades, páginas, enlaces, etc., y recursos como imágenes y hojas de estilo (CSS), necesarios para dar el aspecto deseado a la aplicación.

En Real-A se han creado dos proyectos de estilo, StyleRealA y StyleRealAMobile, uno para utilizar desde las vistas de escritorio y otro para las vistas para móviles. Ambos proyectos tienen la misma estructura y una implementación similar, ya que solamente varían en pequeños aspectos, sobre todo a nivel de hojas de estilo, para adaptar cada estilo al tamaño de las pantallas para el cual se ha diseñado.

De forma que el proyecto de estilo StyleRealA está especialmente para pantallas grandes, con utilización de iconos e imágenes. Mientras que

StyleRealAMobile está diseñado para pantallas pequeñas, evitando en lo posible los iconos e imágenes, maximizando en lo posible la utilización de CSS, para minimizar el tráfico de datos entre el servidor y el cliente, ya que el ancho de banda es un aspecto crítico en los móviles.

Se ha procurado que el aspecto de la aplicación sea líquido, de forma que la maquetación se adapta a distintos tamaños de pantalla, evitando así tener varias maquetaciones distintas para distintos tipos de tamaño de pantalla, y consiguiendo que el aspecto de la aplicación se adapte a la pantalla del móvil, independientemente de que éste, esté siendo observado en posición horizontal o vertical.

Debido a que los móviles se utilizan normalmente en condiciones de iluminación no ideales, y teniendo en cuenta que la aplicación está diseñada para su uso en el exterior, donde las condiciones de iluminación son extremas, para la aplicación se ha optado por colores que se adaptan a condiciones de baja visibilidad.

4.4. Manual de usuario.

La aplicación Real-A se divide en tres zonas, Zona Pública, Zona Usuario y Zona Administrador, que se corresponden con los tres niveles de acceso posibles.

La Zona Pública es la zona de acceso a la aplicación, accesible por cualquier persona, sin necesidad de que ésta, esté registrada en la aplicación, desde aquí se podrá crear un usuario, recuperar una contraseña olvidada e ingresar un usuario ya existente para acceder a una de las zonas restringidas.

La Zona Usuario, que requiere un usuario registrado en la aplicación, es la zona donde se desarrolla la Realidad Aumentada, que permite al usuario realizar una fotografía desde el móvil y obtener información relevante a dicha fotografía. Además le permite mantener un álbum con las fotografías obtenidas y mantener sus datos de usuario.

Finalmente, la Zona Administrador, que requiere un usuario registrado con permisos de Administrador, es la zona que permite el mantenimiento propio de la aplicación. Desde donde se pueden crear y mantener los objetos reconocibles, junto con sus imágenes y descripciones, además de permitir la gestión de usuarios e idiomas permitidos en la aplicación.

En los siguientes puntos se describe cómo utilizar cada zona así como la estructura general de pantalla dentro de la aplicación.

4.4.1. Estructura general de una pantalla.

En las figuras 23, 24 y 25 se muestra la estructura básica de las pantallas pertenecientes a la aplicación.

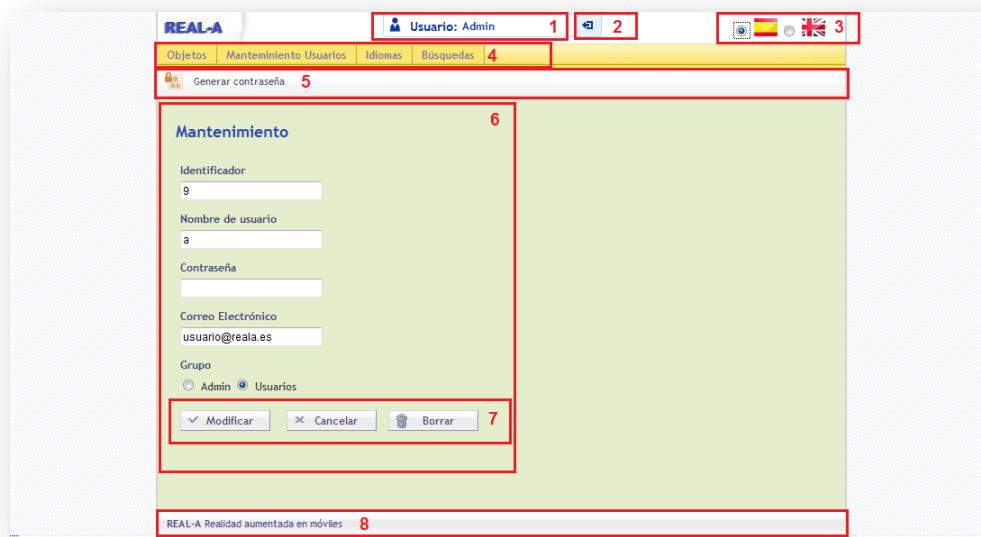


Figura 23.- Estructura general de pantalla para navegadores de sobremesa.

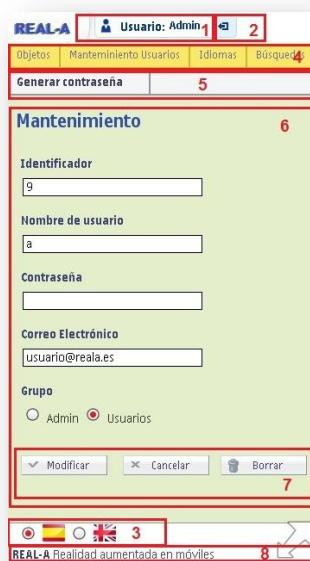


Figura 24.- Estructura general de pantalla para navegadores móviles.

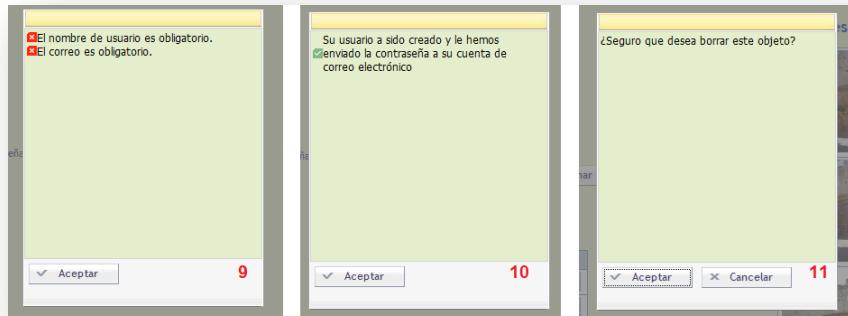


Figura 25.- Popup de error, informativo y de confirmación.

Los elementos que conforman la pantalla se han numerado en rojo y corresponden a la siguiente descripción:

1. Usuario: panel que muestra el nombre de usuario conectado. Este panel no está disponible en la Zona Pública, ya que en dicha zona el usuario aún no se ha conectado.
2. Botón Desconexión: permite desconectar al usuario cerrando la sesión activa y volviendo a la pantalla inicial de conexión. Este botón tampoco está en la Zona Pública.
3. Selección de Idioma: muestra el idioma seleccionado en un momento dado, y permite al usuario seleccionar otro idioma de los definidos en la aplicación, seleccionando el check box correspondiente.
4. Menú de pestañas: muestra las opciones de menú accesibles en cada Zona.
5. Panel de botones generales: contiene botones que realizan operaciones generales sobre los datos de la pantalla correspondiente.
6. Formulario: zona específica de cada pantalla, en la que se muestra y solicita al usuario la información relativa a dicha pantalla. Todas las partes de la pantalla son fijas. En caso de que el formulario crezca fuera de los límites que le corresponden, el usuario podrá moverse en él gracias al scroll horizontal y vertical que aparece automáticamente en este caso.
7. Botones del formulario: realizan operaciones sobre los datos del formulario.

8. Pie de página: limita la aplicación en su parte inferior, mostrando información de la aplicación.
9. Popup de error: se muestran en caso de producirse algún tipo de error en las validaciones de los campos de formulario introducidos por el usuario o producirse algún tipo de error en alguno de los procesos internos de la aplicación.
10. Popup informativo: muestran información sobre algún proceso al usuario.
11. Popup confirmación de acción: ante operaciones de riesgo, como el borrado de registros, la aplicación solicita la confirmación de dichas operaciones mediante este tipo de popup's.

4.4.2. Zona Pública.

Accesible por cualquier usuario, sin necesidad de que esté registrado en la aplicación. Esta zona sólo proporciona una página de conexión junto con los mecanismos necesarios para que los nuevos usuarios se puedan dar de alta en la aplicación, o puedan recuperar sus datos en caso de olvido.

Login.

La página principal es la página de conexión a la aplicación. El usuario deberá introducir su nombre de usuario y contraseña y pulsar el botón “Aceptar”. Si el nombre y contraseña son correctos, entrará en la Zona de Usuario o en la Zona de Administrador, dependiendo del tipo de usuario.

En caso de que el usuario no esté registrado aún en la aplicación podrá crear un nuevo usuario mediante el botón “Crear nuevo usuario” en el panel de botones generales.

En caso de que el usuario esté registrado en la aplicación pero no recuerde su contraseña, podrá volver a obtener una contraseña válida a través del botón “Recuperar contraseña” en el panel de botones generales.

Las figuras 26 y 27 muestran las pantallas de Login tanto para escritorio como para móviles.

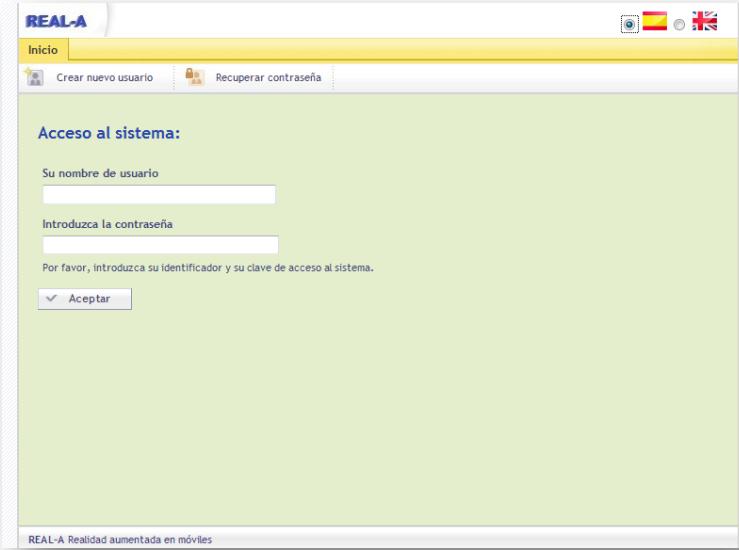


Figura 26.- Login.

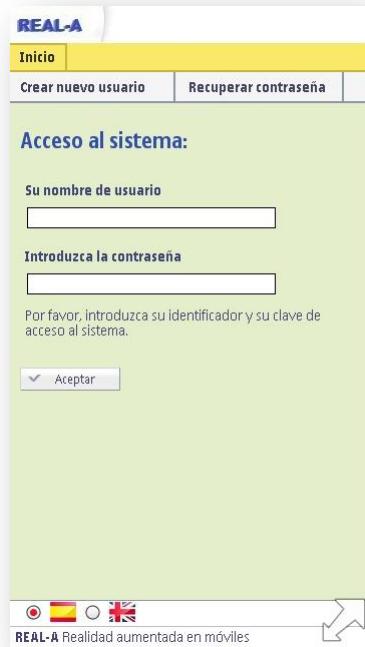


Figura 27.- Login Móvil.

Crear Usuario.

Para registrarse en la Real-A, el nuevo usuario deberá ingresar un nombre de usuario, y una cuenta de correo valida. La aplicación validará que el nuevo usuario no existe aún en el sistema y que el correo electrónico proporcionado es válido.

La aplicación envía un correo de bienvenida con la primera contraseña, de forma que el usuario deberá recibir este correo antes de poder conectar a la aplicación. Los usuarios creados desde aquí son usuarios registrados.

Las figuras 28 y 29 muestran las pantallas para la creación de nuevos usuarios.

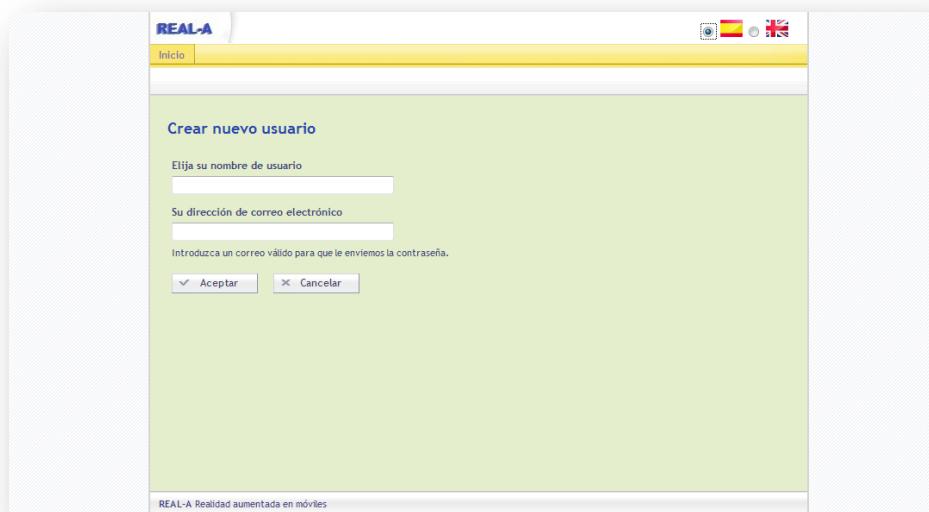


Figura 28.- Crear usuario.

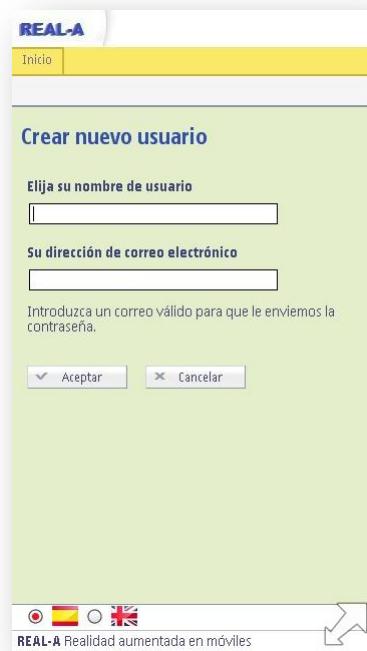


Figura 29.- Crear usuario Móvil.

Recuperar Contraseña.

En caso de que el usuario olvide su contraseña, desde esta pantalla se le permite recuperarla.

El usuario deberá ingresar en el formulario su correo electrónico, el sistema comprueba que el correo indicado pertenece a un usuario dado de alta y envía un correo indicando el nombre de usuario y la nueva contraseña.

Las figuras 30 y 31 muestran las pantallas para la recuperación de contraseñas.



Figura 30.- Recuperar contraseña.

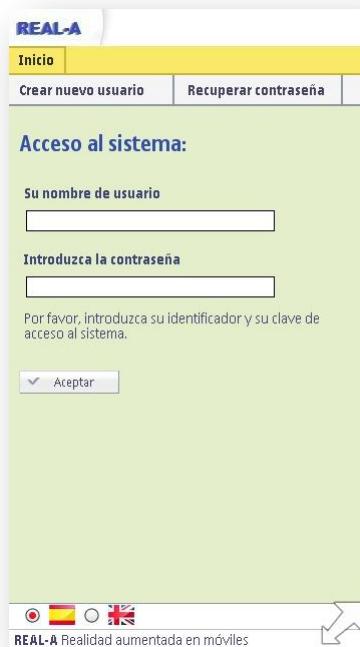


Figura 31.- Recuperar contraseña Móvil.

4.4.3.Zona Usuario.

Una vez que un usuario normal se conecta en la aplicación accede a la Zona de Usuario.

Esta zona es distinta dependiendo del tipo de dispositivo desde el que el usuario accede a la aplicación. En caso de que el usuario acceda desde un móvil, tendrá acceso a un menú con tres opciones:

- Real-A.
- Usuario.
- Álbum.

En caso de que acceda desde un ordenador de sobremesa, el menú se compone únicamente de dos opciones:

- Usuario.
- Álbum.

A continuación se detalla el funcionamiento de cada una de las opciones de menú:

Real-A.

Esta opción de menú sólo es accesible para dispositivos móviles.

Esta opción contiene las operaciones de Realidad Aumentada para las que está diseñada la aplicación. La figura 32 muestra la pantalla principal:

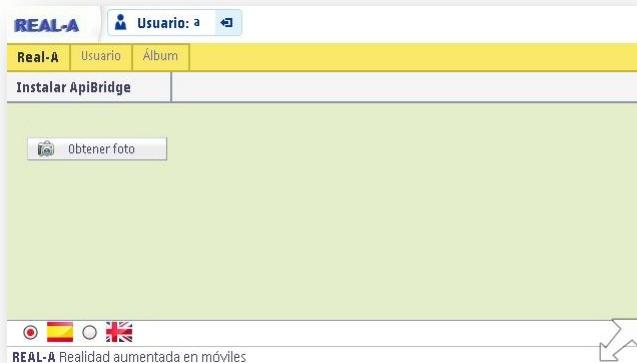


Figura 32.- Real-A. Posición Horizontal.

La opción de acceso a la cámara nativa del dispositivo móvil solo está disponible para móviles Symbian S60 con ApiBridge Instalado, para lo que es necesario instalar APIBridge_v1_1.sis en el dispositivo móvil desde el que se va a utilizar la aplicación.

Desde el botón “Instalar ApiBridge” (figura 33, Izquierda), se puede descargar directamente APIBridge_v1_1.sis. Una vez descargada, la aplicación siempre solicitará confirmación para comenzar la instalación (figura 33, Derecha).

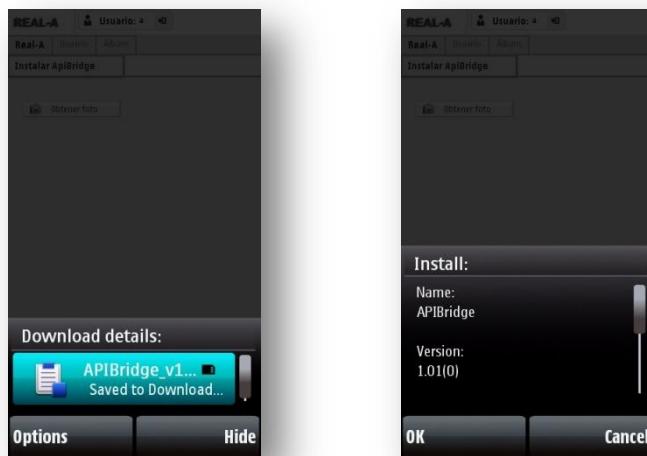


Figura 33. Descarga e instalación APIBridge_v1_1.sis.

El usuario, al pulsar el botón “Obtener foto”, cede el control a la aplicación nativa de la cámara del móvil cuya interfaz de usuario dependerá lógicamente del tipo de móvil concreto utilizado (e.g., figuras 34 y 35).

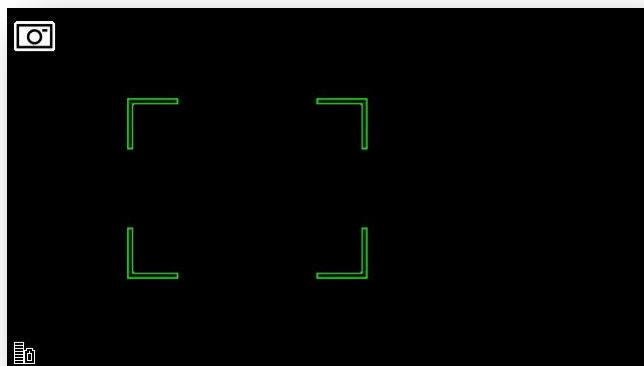


Figura 34.- Real-A. Cámara nativa.

El usuario obtiene la fotografía y debe seleccionarla para que la aplicación nativa de la cámara se cierre y devuelva el control a la aplicación Web.

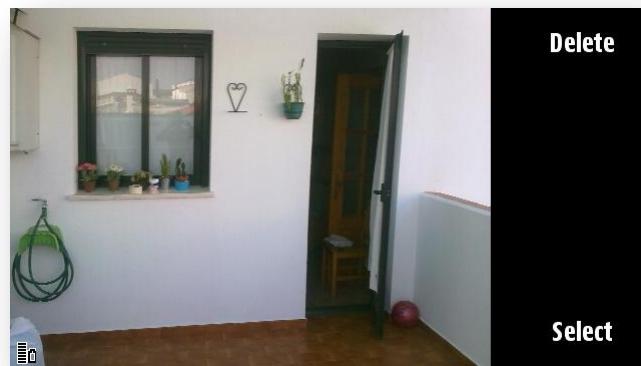


Figura 35.- Real-A. Selección de imagen.

Una vez que se retorna el control a Real-A, ésta se encarga de buscar la imagen obtenida en su base de datos tal como muestra la figura 36.

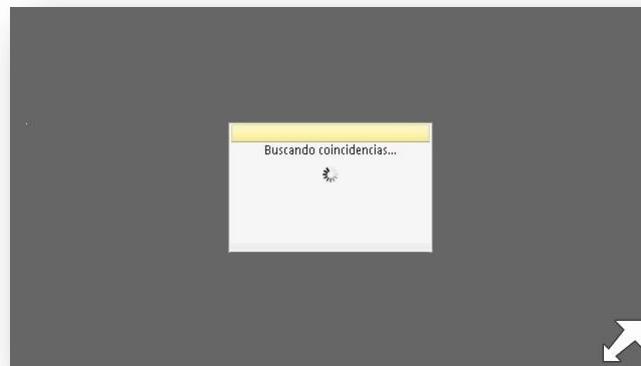


Figura 36.- Buscando coincidencias.

La aplicación muestra al usuario la información relativa a la imagen, en caso de encontrarla en la base de datos, o un mensaje indicando que no ha encontrado coincidencias tal y como se muestra en las diferentes alternativas en la figura 37.

Figura 37.- Real-A. Información de la imagen obtenida.

En todo caso, haya encontrado o no coincidencia, se da la opción al usuario de realizar anotaciones sobre la imagen obtenida y la opción de poder grabar una anotación de voz o cualquier tipo de sonido, a través de la grabadora de sonidos nativa pulsando el botón “Grabar” tal y como se muestra en la figura 38, donde de nuevo la aplicación Real-A ha lanzado y dado el control a la aplicación nativa del móvil que realiza grabación de audio.

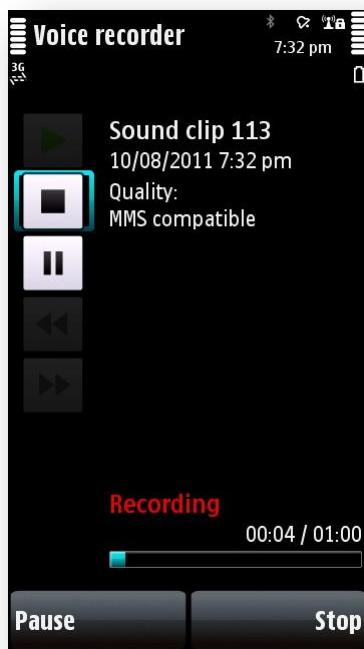


Figura 38.- Real-A. Grabadora Nativa.

Una vez obtenida la grabación, la aplicación nativa retorna el control a Real-A, que ahora muestra un pequeño reproductor en la pantalla de información, como se muestra en la figura 39, que permite reproducir el sonido grabado, además permite grabar la imagen obtenida, junto con la información, anotaciones y sonido, pulsando el botón “Guardar en álbum”:



Figura 39.- Real-A. Información de la imagen obtenida.

Usuario.

Desde esta opción de menú se permite modificar cualquiera de los datos de usuario, esta opción de menú está accesible para cualquier tipo de dispositivo.

Los cambios de correo electrónico o contraseña, obligan al usuario a introducir la contraseña actual.

El cambio de contraseña obliga al usuario a introducir la contraseña nueva por duplicado, para evitar que se comentan errores a la hora de introducirla.

Además con el cambio de correo se genera una nueva contraseña que es enviada al usuario a la nueva cuenta de correo.

También se permite la baja del usuario a través del botón "Borrar el usuario" en el panel de botones generales. El borrado del usuario implica el borrado de las fotos almacenadas en el álbum, por lo que para realizar esta operación es necesaria la confirmación por parte del usuario (figuras 40 y 41). Una vez que se realiza la operación de borrado, se produce la desconexión, volviendo a la pantalla principal en la Zona Pública.

Figura 40.- Mantenimiento de Usuario.

Figura 41.- Mantenimiento de Usuario Móvil.

Álbum.

La opción de menú Álbum, da acceso al usuario al álbum con las fotos obtenidas por él. Cada usuario solamente tiene acceso a sus propias fotos.

La página principal del álbum muestra una tabla con todas las fotos almacenadas por el usuario tal y como se puede observar en las figuras 42 y 43.

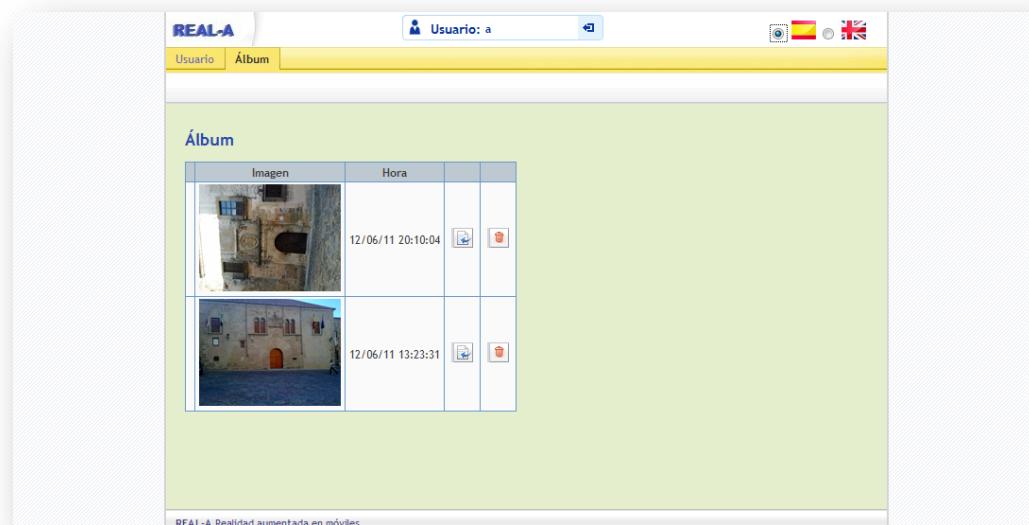


Figura 42.- Álbum.



Figura 43.- Álbum Móvil.

Pulsando el botón de un registro, se permite borrar ese registro.

Todas las operaciones de borrado necesitan ser confirmadas mediante un popup de confirmación, para evitar que el usuario pulse los botones de borrado por error.

Pulsando el botón de un registro se accede al mantenimiento de ese registro, donde se muestra toda la información relativa a la fotografía almacenada, permitiéndose modificar las anotaciones realizadas sobre las

imágenes y también el borrado de la imagen a través del botón “Borrar” en el panel de botones generales.

Además en la vista para dispositivos móviles, también se dispone de los botones de reproducción de sonidos, para el caso de que la fotografía fuera grabada junto a un fichero de sonido obtenido con la grabadora nativa.

Las figuras 44 y 45 corresponden a capturas reales de las pantallas correspondientes a esta funcionalidad.

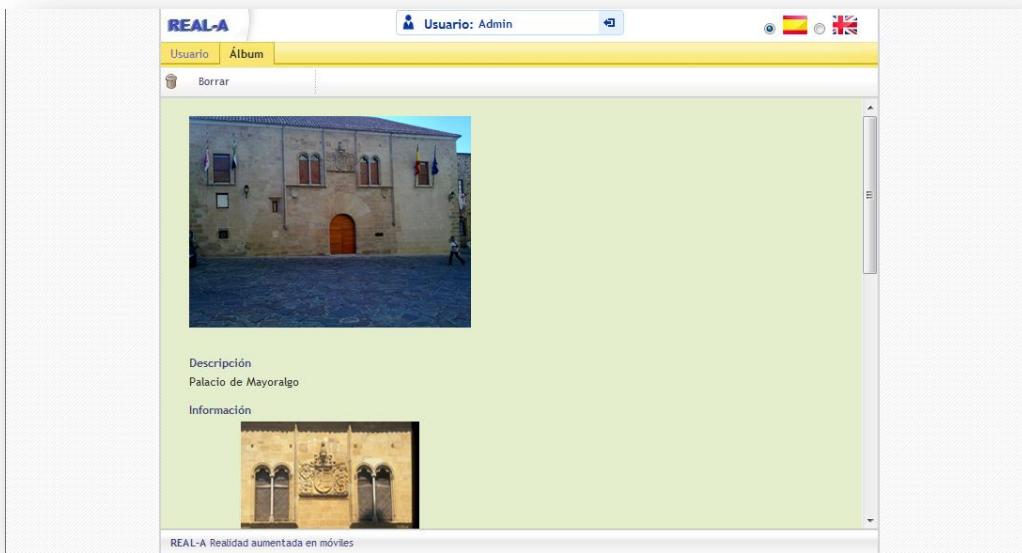


Figura 44.- Mantenimiento de fotos del álbum.



Figura 45.- Mantenimiento de fotos del álbum Móvil.

4.4.4. Zona Administrador.

Una vez que un usuario administrador se conecta en la aplicación accede a la Zona de Administrador. Esta zona permite realizar operaciones de configuración de la aplicación. Se compone de un menú con cuatro opciones:

- Objetos.
- Mantenimiento de usuarios.
- Idiomas.
- Búsquedas.

A continuación se detalla el funcionamiento de cada una de las opciones de menú:

Objetos.

Desde la opción de menú objetos, se permite crear los objetos que se utilizan para el reconocimiento de imágenes, cada objeto contiene una serie de imágenes sobre las que se realiza la búsqueda de las imágenes, y una serie de descripciones, una por cada idioma configurado en la aplicación, en HTML, que será la información que se muestre al usuario.

La página principal de esta opción de menú muestra una tabla paginada, donde se muestran los registros de objetos ya creados, dando la opción al usuario de crear un nuevo objeto, a través del botón “Crear nuevo objeto” en el panel de botones generales, o modificar uno ya existente, pulsando el botón  asociado al registro que se quiere modificar.

La navegación entre las páginas de la tabla se realiza mediante los botones   1 a 2 de 22  . Las figuras 46 y 47 capturan esta funcionalidad.

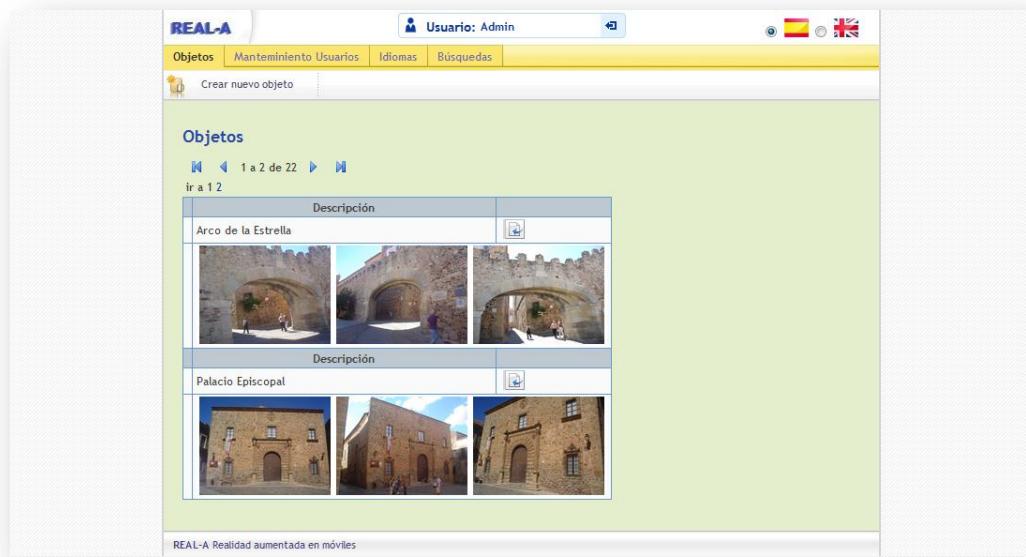


Figura 46.- Mantenimiento de Objetos.



Figura 47.- Mantenimiento de Objetos Móvil.

Si el usuario selecciona crear un nuevo objeto o modificar uno existente, la aplicación llega a las pantallas mostradas en la figuras 48 y 49. Desde aquí se puede mantener un objeto. En esta pantalla existen dos tablas, una con las imágenes asociadas al objeto, y otra con las descripciones.

Permite mantener las descripciones asociadas al objeto, dando la opción al usuario de crear una nueva descripción, a través del botón “Crear descripción” en el panel de botones generales, o modificar una ya existente,

pulsando el botón  asociado al registro que se quiere modificar en la tabla de descripciones.

Para el mantenimiento de las imágenes pertenecientes al objeto, la aplicación permite la eliminación de una imagen, pulsando el botón  asociado a la imagen que se desea borrar.

Para añadir una nueva imagen al objeto, se utiliza el botón “Nueva imagen”, que muestra al usuario un diálogo que le permite subir una fotografía al sistema.

El borrado de un objeto, se realiza pulsando el botón “Eliminar”, el sistema pide confirmación para evitar errores y elimina tanto el objeto, como las fotografías y descripciones asociadas a éste.

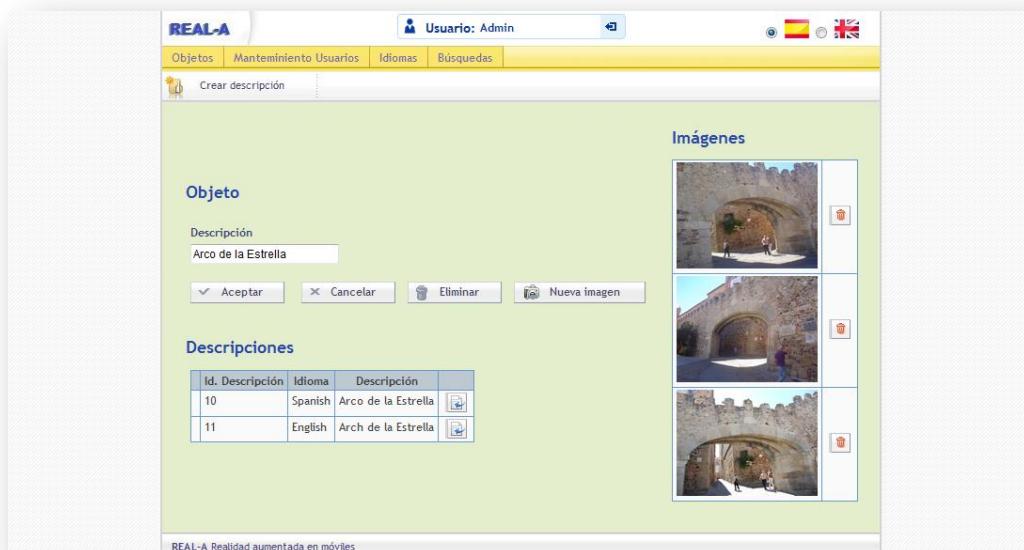


Figura 48.- Mantenimiento de objeto.

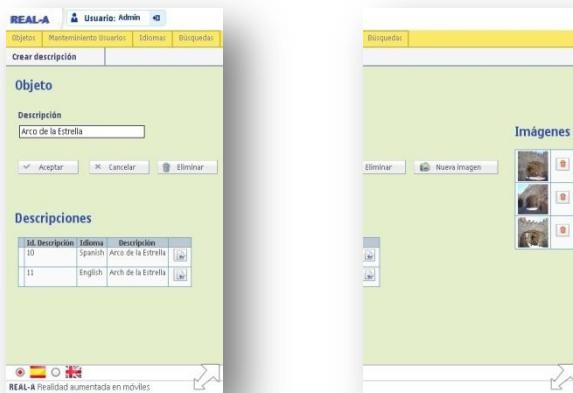


Figura 49.- Mantenimiento de objeto Móvil.

En las figuras 50 y 51 se muestra el diálogo para subir un nuevo fichero y asociarlo al objeto que se está tratando. Pulsando el botón “Examinar...” el usuario accede a su sistema de ficheros local, donde puede seleccionar el fichero que desea subir a la aplicación. Una vez seleccionada, la imagen será almacenada en el sistema asociada al objeto tratado, pulsando el botón “Aceptar”. Por el contrario, si el usuario pulsa el botón “Cancelar”, el diálogo se oculta, sin realizar ninguna operación.

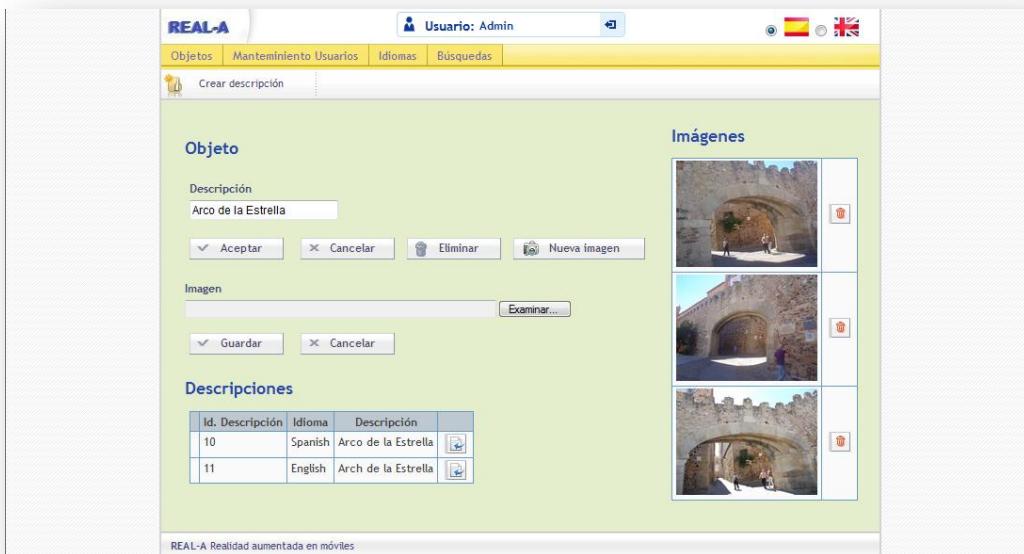


Figura 50.- Mantenimiento de objetos. Nueva imagen.



Figura 51.- Mantenimiento de objetos. Nueva imagen. Móvil.

Si el usuario selecciona crear una nueva descripción o modificar una existente, la aplicación llega a las pantallas mostradas en las figuras 52 y 53.

El formulario de esta pantalla contiene un editor de texto completo, desde el que se puede crear una descripción elaborada sobre el objeto, un combo con los idiomas configurados en la aplicación, y un campo descripción, todos ellos son obligatorios.

Además el formulario contiene los siguientes botones:

“Crear”, que grabará la nueva descripción. Sólo visible si es un alta.

“Modificar”, que grabará las modificaciones realizadas sobre la descripción seleccionada. Sólo visible si es una modificación.

“Cancelar”, que vuelve a la pantalla de mantenimiento de objeto sin realizar ninguna operación.

“Borrar”, pide la confirmación del borrado y en caso afirmativo, borra la descripción. Sólo visible si es una modificación.

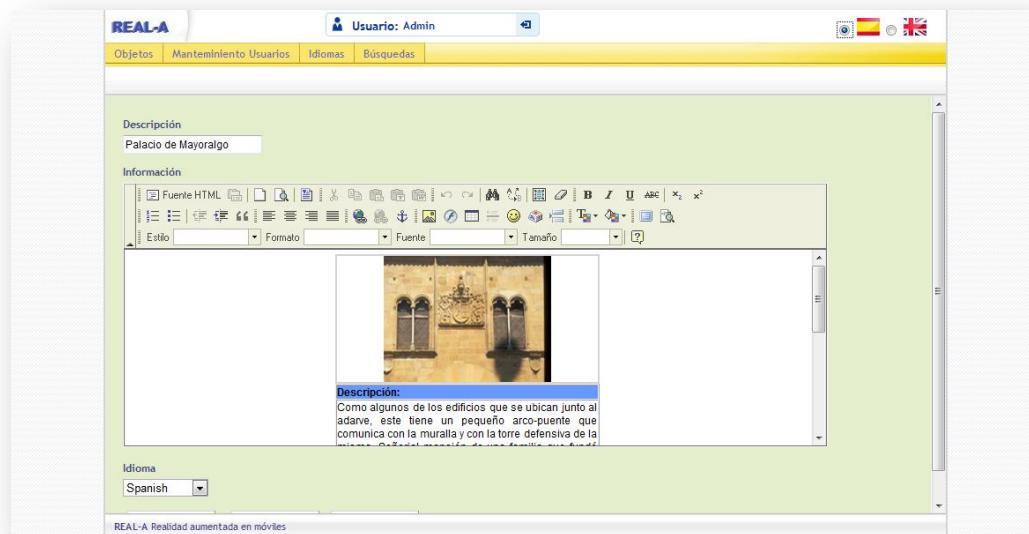


Figura 52.- Mantenimiento de descripciones.



Figura 53.- Mantenimiento de descripciones Móvil.

Mantenimiento de Usuarios.

Esta opción permite el mantenimiento de usuarios.

La pantalla principal (figuras 54 y 55) de esta opción contiene un formulario que sirve para filtrar los datos mostrados en la tabla paginada, donde están cargados todos los usuarios registrados en la aplicación, de forma que informando uno o varios de los campos utilizados como criterio de búsqueda

(usuario, correo electrónico y rol) y pulsando el botón “Buscar” se acotará el resultado de usuarios mostrados en la tabla, haciéndola más manejable.

Permite mantener los usuarios, dando la opción de crear uno nuevo, a través del botón “Crear usuario” en el panel de botones generales, o modificar una ya existente, pulsando el botón  asociado al registro que se quiere modificar en la tabla de usuarios.

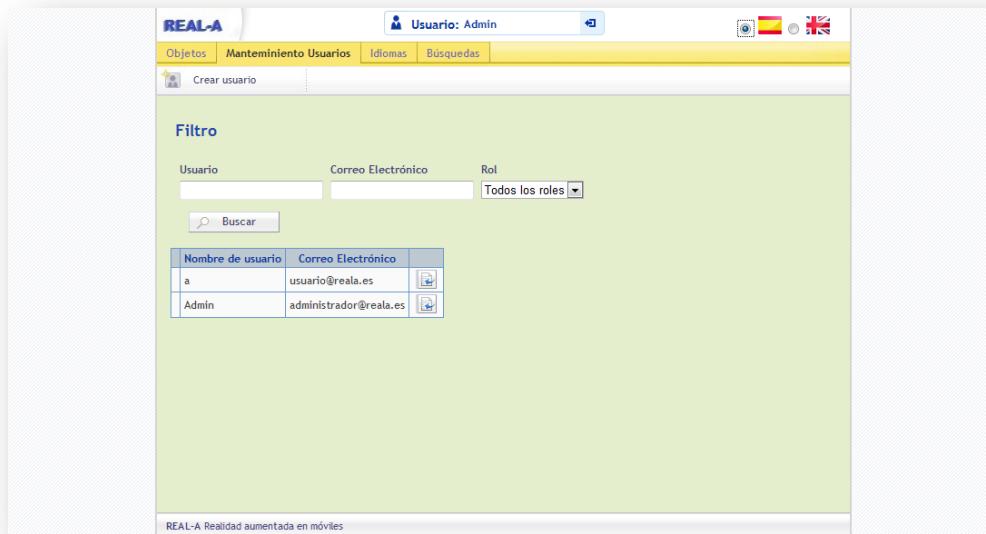


Figura 54.- Mantenimiento de Usuarios.

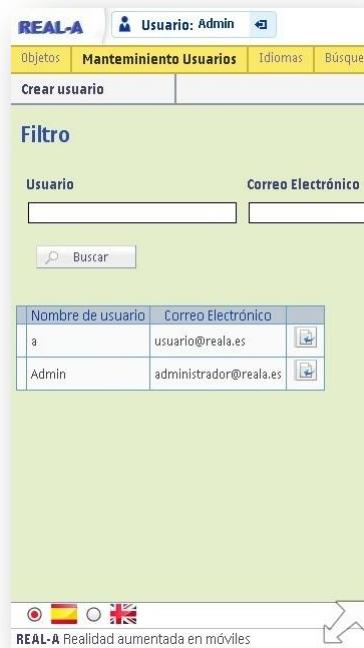


Figura 55.- Mantenimiento de Usuarios Móvil.

Si el usuario selecciona crear un nuevo usuario o modificar uno ya existente, la aplicación llega a las pantallas mostradas en las figuras 56 y 57.

El funcionamiento de esta pantalla es similar a los mantenimientos comentados anteriormente, permitiendo crear, modificar y eliminar el registro tratado en ese instante.

El botón “Generar contraseña” en el panel de botones generales permite generar una contraseña automáticamente.

REAL-A

Usuario: Admin

Objetos Mantenimiento Usuarios Idiomas Búsquedas

Generar contraseña

Mantenimiento

Identificador
9

Nombre de usuario
a

Contraseña

Correo Electrónico
usuario@real-a.es

Grupo
 Admin Usuarios

▼ Modificar Cancelar Borrar

REAL-A Realidad aumentada en móviles

Figura 56.- Mantenimiento de Usuario.

REAL-A

Usuario: Admin

Objetos Mantenimiento Usuarios Idiomas Búsquedas

Generar contraseña

Mantenimiento

Identificador
9

Nombre de usuario
a

Contraseña

Correo Electrónico
usuario@real-a.es

Grupo
 Admin Usuarios

▼ Modificar Cancelar Borrar

REAL-A Realidad aumentada en móviles

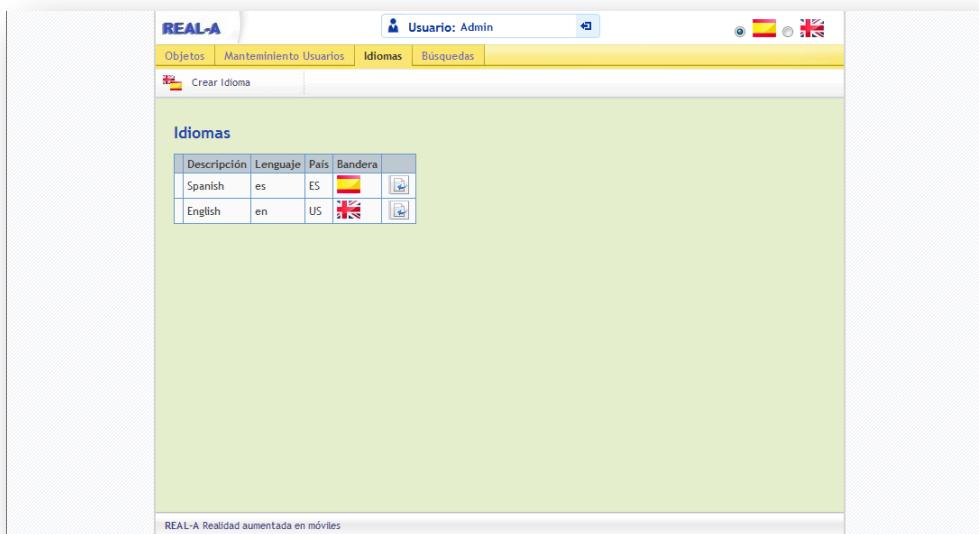
Figura 57.- Mantenimiento de Usuario Móvil.

Idiomas.

Esta opción permite el mantenimiento de los idiomas soportados por la aplicación.

Permite crear un nuevo idioma, a través del botón “Crear Idioma” en el panel de botones generales, o modificar uno ya existente, pulsando el botón  asociado al registro que se quiere modificar en la tabla de Idiomas.

Las figuras 58 y 59 muestran las capturas de la funcionalidad Idioma.



Descripción	Lenguaje	País	Bandera	
Spanish	es	ES		
English	en	US		

Figura 58.- Mantenimiento de Idiomas.



Descripción	Lenguaje	País	Bandera	
Spanish	es	ES		
English	en	US		

Figura 59.- Mantenimiento de Idiomas Móvil.

El funcionamiento de esta pantalla es similar a los mantenimientos comentados anteriormente, permitiendo crear, modificar y eliminar el registro tratado en ese instante.

Para añadir nuevos idiomas a la aplicación, además de crear el nuevo registro idioma con sus parámetros a través de las páginas mostradas en las figuras 60 y 61, hay que tener en cuenta que habrá que crear también las descripciones asociadas a los objetos para el nuevo idioma, así como un nuevo fichero ApplicationResources_idioma_pais.properties.

The screenshot shows a web-based application interface for managing languages. At the top, there's a header bar with the REAL-A logo, user information ('Usuario: Admin'), and language selection buttons for Spanish and English. Below the header is a navigation menu with tabs: 'Objetos', 'Mantenimiento Usuarios', 'Idiomas' (which is currently selected and highlighted in yellow), and 'Búsquedas'. The main content area has a light green background and contains a form for editing a language entry. The form fields are: 'Descripción' (with 'Spanish' typed in), 'Lenguaje' (with 'es' typed in), 'País' (with 'ES' typed in), and 'Bandera' (with a file input field showing a preview of the Spanish flag). At the bottom of the form are three buttons: 'Modificar' (Modify), 'Cancelar' (Cancel), and 'Borrar' (Delete). The footer of the page includes the text 'REAL-A Realidad aumentada en móviles' and a small logo.

Figura 60.- Mantenimiento de Idioma.

This screenshot shows the same 'Idioma' maintenance screen as Figure 60, but with some visual differences. The flag preview in the 'Bandera' field shows the British flag instead of the Spanish one. The footer of the page also includes a small icon of a mobile phone with a signal.

Figura 61.- Mantenimiento de Idioma Móvil.

Búsquedas.

Desde la opción de menú Búsquedas se permite al usuario realizar búsquedas de imágenes sobre la base de datos.

El formulario (figuras 62 y 63) contiene un campo que permite cargar la imagen que se desea buscar. Pulsando el botón “Examinar...” el usuario accede a su sistema de ficheros local, donde puede seleccionar la imagen a buscar. Una vez seleccionada, pulsando el botón “Buscar” se inicia la búsqueda.

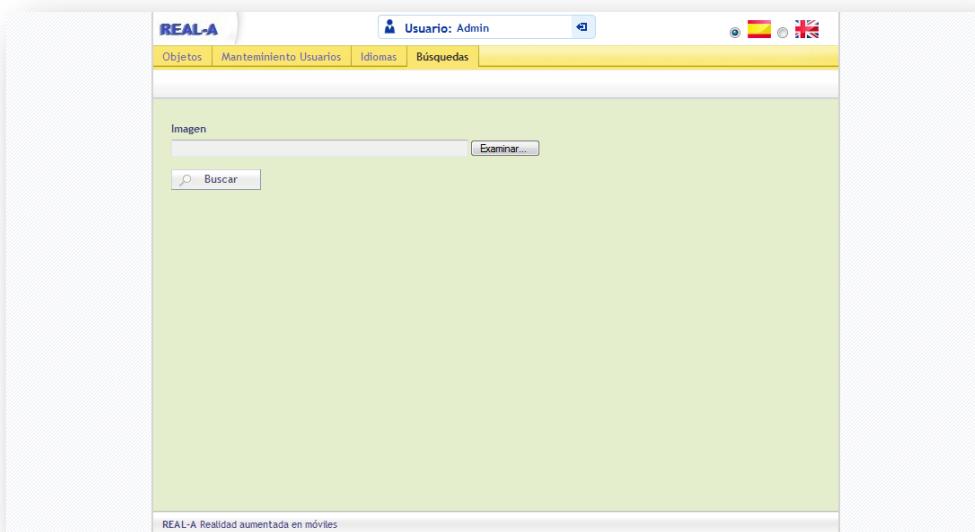


Figura 62.- Búsqueda de imágenes.

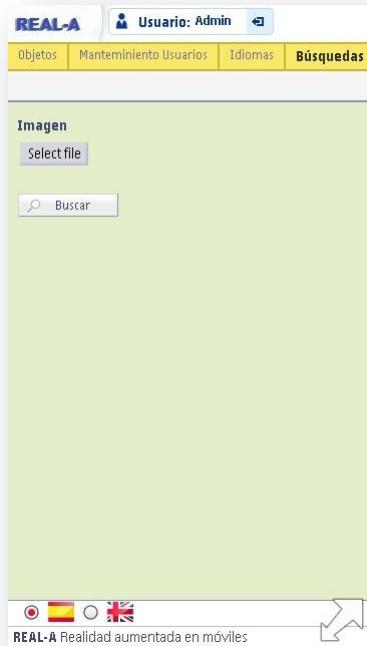


Figura 63.- Búsqueda de imágenes Móvil.

Tras realizar la búsqueda, el sistema muestra los resultados (figuras 64 y 65) con las 10 imágenes de la base de datos con más puntos coincidentes respecto a la imagen buscada. No todas las imágenes encontradas son coincidencias, en general todas las imágenes con menos de 10 puntos coincidentes son falsos positivos. Asociados a cada registro de la tabla existen dos botones, que muestra una comparativa de la imagen buscada y la imagen de base de datos, y que muestra la información relativa a la imagen de base de datos.

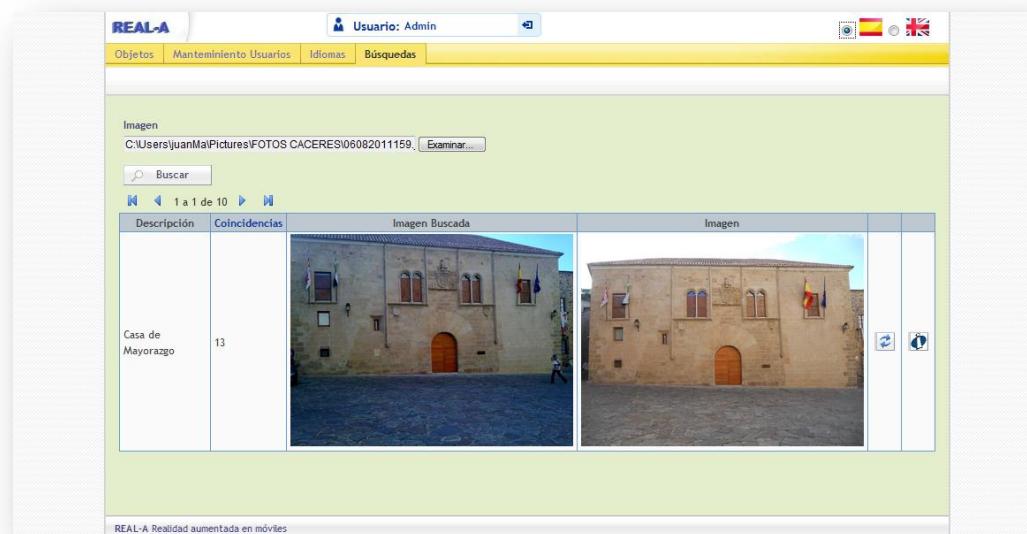


Figura 64.- Resultados.

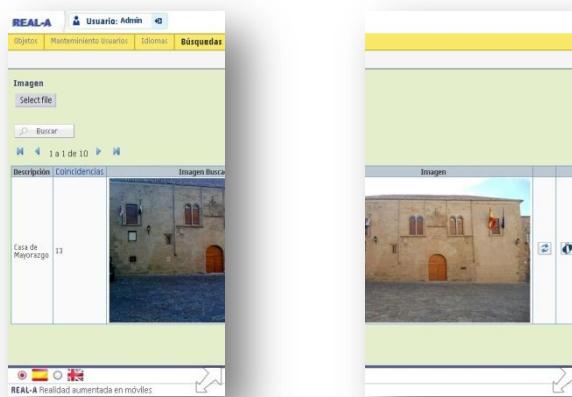


Figura 65.- Resultados Móvil.

Al pulsar el botón el sistema muestra un popup con la comparativa de la imagen buscada y la imagen del sistema, mostrando los puntos coincidentes entre ambas (figuras 66 y 67). En ella se puede comprobar tanto los puntos coincidentes reales, como los falsos positivos.

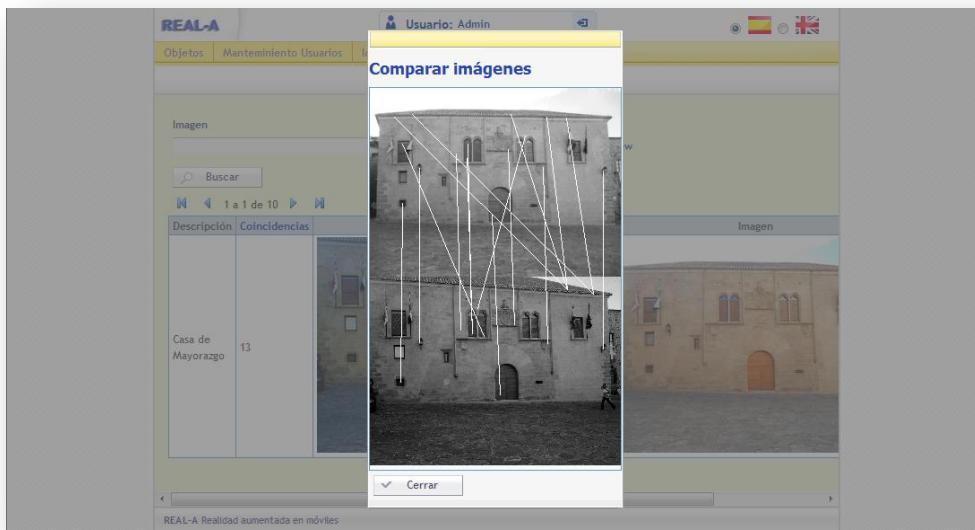


Figura 66.- Comparación de imágenes.

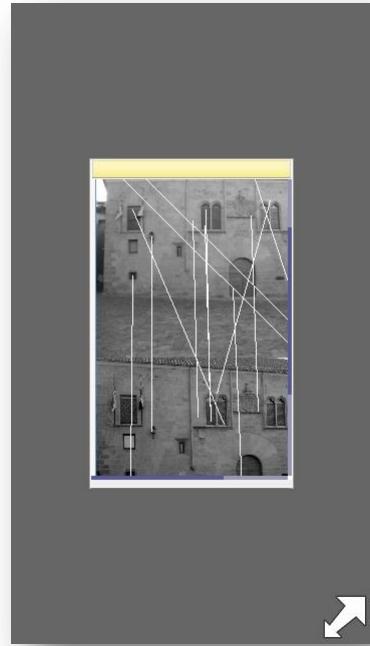


Figura 67.- Comparación de imágenes Móvil.

Al pulsar el botón  el sistema muestra un popup con la información relativa a la imagen de base de datos, es decir, la descripción del objeto al que pertenece dicha imagen (figuras 68 y 69).

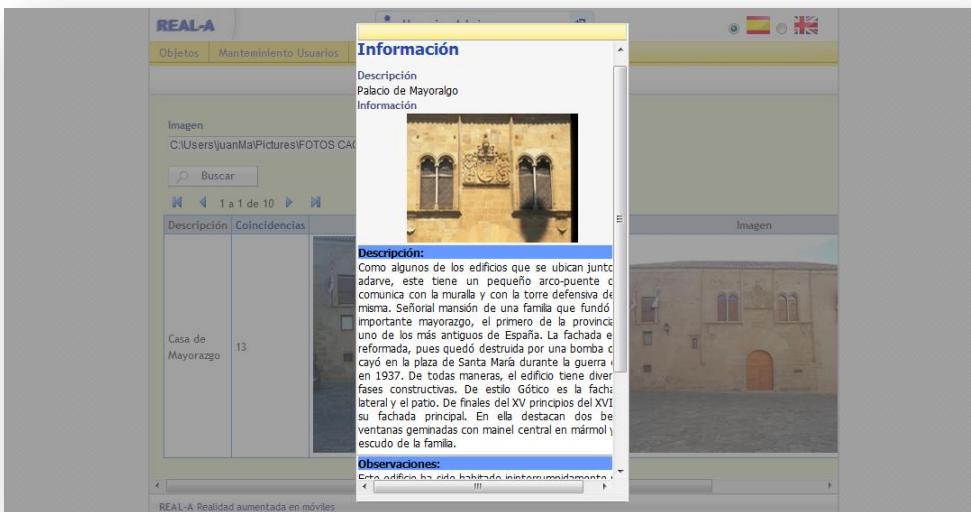


Figura 68.- Información.



Figura 69.- Información Móvil.

5. Resumen y conclusiones.

La tecnología móvil ha conseguido abrirse un hueco muy importante en la sociedad. La aparición de los teléfonos inteligentes y tablets PC, ha hecho que un alto porcentaje de usuarios se decanten por este tipo de dispositivos, siendo actualmente uno de los principales medios de acceso a Internet.

La Realidad Aumentada ha tomado gran impulso en los últimos tiempos gracias, en parte, a los dispositivos móviles y a las herramientas que generalmente disponen, como cámaras fotográficas, GPS, brújulas y demás sensores. Esto los convierte en los medios adecuados para dar soporte a este tipo de aplicaciones, dada la necesidad de la Realidad Aumentada de estar en contacto con el mundo que la rodea.

En el actual proyecto ha unido la tecnología móvil, la Realidad Aumentada y las aplicaciones Web. Actualmente para solventar las limitaciones tecnológicas, se han tenido que combinar la solución con componentes adicionales que requieren instalación previa, pero se prevee que esto no sea necesario a medio-largo plazo. Este tipo de enfoque de desarrollo proporciona los siguientes beneficios respecto al desarrollo de aplicaciones nativas e híbridas:

- Reduce el coste de construcción de aplicaciones.
- Reduce los tiempos de desarrollo.
- Simplifica el desarrollo.
- Reduce los tiempos de aprendizaje.

Para el desarrollo se ha utilizado WebRatio como herramienta de Desarrollo Dirigido en Modelos lo que mejora los puntos anteriores y proporciona los siguientes beneficios adicionales:

- Generación de código automático que cumple con el estándar Java/JSP 2.0.
- Independencia del sistema de gestión de bases de datos
- Soporte completo de SOA y Servicios Web.
- Interoperabilidad con sistemas heredados.
- Gestión de la seguridad y autenticación.
- Fácil desarrollo de aplicaciones multilenguaje.

- Interfaz e interacción enriquecidas (RIA).
- Alto rendimiento.
- Trabajo colaborativo y control de versiones.
- Código abierto y librerías estándar.

Todo esto hace que los paradigmas MDD y en concreto WebRatio se adapten adecuadamente al desarrollo de aplicaciones Web, ofreciendo la rapidez de desarrollo y adaptación a un mundo en continua evolución como son las tecnologías móviles.

Para realizar una comparación de las líneas escritas manualmente frente a las líneas generadas por WebRatio, se ha utilizado Cloc [31], que es un contador de líneas de código, junto con líneas en blanco y líneas de comentario, obteniendo las estadísticas que se muestran en la tabla 4 para las líneas de código generadas y en la tabla 5 para las líneas de código escritas manualmente, a la que se ha añadido las líneas escritas en Flex para las películas Flash, que han sido contadas manualmente, ya que Cloc no soporta este tipo de ficheros.

Lenguaje	Ficheros	Líneas en blanco	Líneas de comentario	Líneas de código
JavaScript	248	9.106	10.959	57.532
JSP	229	7	328	43.292
CSS	19	586	434	3.254
Java	15	380	27	1.607
Groovy	19	231	95	961
C	1	117	52	415
SQL	3	96	67	321
C/C++ Header	1	16	29	45
SUMA:	535	10.539	11.991	107.427

Tabla 4.- Estadísticas líneas de código generadas.

Lenguaje	Ficheros	Líneas en blanco	Líneas de comentario	Líneas de código
Groovy	19	231	95	961
C	1	117	52	415
SQL	3	96	67	321
Java	2	49	14	176
CSS	1	7	0	59
C/C++ Header	1	16	29	45
JSP	8	7	0	29
Flex	3	42	49	187
SUMA:	38	565	306	2.193

Tabla 5.- Estadísticas líneas de código manuales.

Como se puede desprender de las tablas 4 y 5, aproximadamente un 2% del código final se ha implementado a mano, para dar soporte al reconocimiento de imágenes y los accesos a la cámara y al micrófono del dispositivo móvil, mientras que el resto se genera a partir de los modelos de WebRatio.

Aunque parece un porcentaje insignificante respecto al resto de la aplicación, tiene un coste significativo, dada la cantidad de lenguajes y tecnologías utilizadas, que requieren el conocimiento de varios lenguajes y entornos por parte del desarrollador, coste que no sería necesario en otro tipo de aplicaciones. Para reducir este coste, probablemente, en el futuro, herramientas como WebRatio deberán hacer un esfuerzo para adaptarse a las nuevas formas de adquisición de datos que proporcionan dispositivos como los móviles. WebRatio maneja el tipo binario Blob, de forma que sería relativamente sencillo utilizar este tipo para generar campos desde los que obtener datos de la cámara, del posicionamiento del dispositivo, etc. Lo que está claro es que los campos Imput de HTML deberán sufrir una evolución hacia estas nuevas tecnologías. En cuanto esta evolución se produzca, será relativamente sencilla y rápida la adaptación por parte de herramientas como WebRatio.

Aunque tanto las aplicaciones Web como MDD se adaptan bien para el desarrollo de aplicaciones, y en general también para aplicaciones móviles, podemos decir que cuando se trata de usar todo el potencial de capacidades que ofrecen los dispositivos móviles, como son el acceso a la cámara, GPS, brújula, sensores de movimiento, etc., mediante tecnologías Web aún es altamente complicado, y no resuelve el problema de la fragmentación de plataformas, siendo necesarias implementaciones dependientes de los dispositivos donde se va a ejecutar la aplicación. Si nos limitamos al uso de tecnologías RIA únicamente basadas en estándares, esto es actualmente imposible, aunque se espera que los añadidos que se hagan a HTML5 faciliten el desarrollo de este tipo de soluciones.

Así, las aplicaciones nativas se convierten en la posibilidad más atractiva actualmente para el desarrollo de aplicaciones que necesitan realizar un uso intensivo de las capacidades de los dispositivos móviles. Esto queda patente en la aplicación de Realidad Aumentada presentada, que necesitan utilizar

necesariamente estas capacidades, las aplicaciones nativas son la principal alternativa, a pesar del coste que conlleva el desarrollo de este tipo de aplicaciones.

Sin embargo, las aplicaciones de Realidad Aumentada conllevan un alto coste en requerimientos Hardware, dadas las operaciones en tiempo real que conllevan. La Realidad Aumentada necesita de operaciones como el reconocimiento de imágenes, renderizado de imágenes en 3D, uso de base de datos con grandes cantidades de datos, etc., que necesitan un uso intensivo de memoria, procesador, etc., y dadas las limitaciones de los dispositivos móviles, pueden requerir del uso de un servidor externo, razón por la que puede ser necesaria la utilización de tecnologías híbridas que permitan desplazar las operaciones más costosas fuera del dispositivo móvil.

Las opciones de desarrollo en cuanto al tipo de aplicación para dispositivos móviles que se han comentado en este documento aparecen resumidas en la tabla 4 junto con sus principales ventajas y desventajas.

		Desarrollo de aplicaciones de Realidad Aumentada		
		Nativas	Hibridas	Web
A favor	Son la alternativa más atractiva, ya que permiten obtener el máximo rendimiento de las capacidades del dispositivo.	En una aplicación híbrida se puede desarrollar la parte que accede a las funcionalidades del dispositivo de la misma forma que en las aplicaciones nativas, con lo que tendríamos tanto las ventajas como los inconvenientes de éstas.	Reduce el coste de aprendizaje y de desarrollo, aun siendo estos mayores que en el desarrollo de una aplicación Web que no requiera el acceso a las características del dispositivo.	
	Alto coste de desarrollo. Dependencia de dispositivo. Son necesarios conocimientos de la plataforma y del lenguaje nativo.	O por el contrario optar por un desarrollo similar al de las aplicaciones Web con lo que se obtendrían las ventajas e inconvenientes de éstas. Pudiéndose llegar a un compromiso entre ambos tipos de desarrollo atendiendo a los requisitos de la aplicación concreta a desarrollar.	El acceso a las capacidades de los móviles sigue siendo dependiente de dispositivo. Además este acceso es limitado.	

Tabla 6.- Alternativas de desarrollo de aplicaciones de Realidad Aumentada en móviles.

Actualmente es difícil poder crear una interfaz de usuario Web que permita el acceso a todas las funcionalidades de los dispositivos móviles, y por tanto, es complicado realizar una aplicación Web de Realidad Aumentada que permita una iteración con el mundo en tiempo real, al menos, de una forma tan sofisticada como se podría realizar mediante aplicaciones nativas.

En el actual proyecto se ha conseguido obtener el acceso a la aplicación nativa de la cámara y obtener un fichero de imagen para su tratamiento intercalando varias tecnologías, como son la API de Nokia ApiBridge, Flash lite y JavaScript, siendo soportado por el navegador Web S60, por tanto, la parte de captura de datos es dependiente del sistema Symbian S60.

Además, al ceder el control desde la aplicación a la aplicación nativa de la cámara, hace que se pierda uno de los puntos importantes de la Realidad Aumentada, que es la iteración entre el mundo real y la información en tiempo real.

En cuanto a la forma de obtener información relativa al mundo que rodea al dispositivo, quizá la opción que menos coste de computación requiere, y por tanto más óptima, hubiese sido posicionar el dispositivo mediante GPS y conocer su orientación mediante la brújula, en dispositivos que cuenten con ambos sensores, y calcular la posición de éste respecto a los puntos de interés cercanos, de forma que es posible representarlos sobre la imagen obtenida con la cámara. Para el proyecto, aunque no es la forma más óptima, se ha optado por el reconocimiento de imágenes, mediante el acceso a la cámara y el acceso a sensores adicionales, como el micrófono, no recogidos en ningún estándar, con el objetivo de comprobar las limitaciones de desarrollo actuales para este tipo de aplicaciones.

Además, se ha considerado más importante el acceso a la cámara para obtener imágenes, que dan al usuario una mayor sensación de iteración con el mundo real, se ha optado por la utilización de ésta, junto con el reconocimiento de imágenes como la vía para obtener la información virtual que mejore o aumente la realidad del usuario.

En definitiva, éste es el momento tanto de la tecnología móvil, como de la Realidad Aumentada, pero aún es complicado unificar esto en una aplicación Web siendo las aplicaciones nativas la principal alternativa de desarrollo para este tipo de aplicaciones. Sin embargo, para aplicaciones en las que el acceso a las funcionalidades de los propios dispositivos no sea necesario, el desarrollo mediante tecnología Web se convierte en una opción factible y atractiva actualmente. Además la velocidad de cambio de la tecnología móvil, hace del paradigma MDD una solución más que aceptable para aplicaciones Web para móviles, dados los beneficios que ofrece.

6. Trabajo futuro.

Para aplicaciones de Realidad Aumentada es necesario un uso intensivo de las capacidades propias de los teléfonos inteligentes actuales, como son GPS, brújulas, cámaras, y demás sensores. Para aplicaciones Web, el acceso a estas capacidades aún está muy restringido, siendo las aplicaciones nativas las que consiguen mejores resultados.

HTML5 proporciona las definiciones necesarias para este tipo de accesos, la mayoría de proveedores de sistemas operativos móviles tiende a la inclusión de implementaciones de HTML5 en sus navegadores, por lo que en un futuro cercano la mayoría de los navegadores soportarán el acceso a las capacidades de los móviles.

Otra opción para el desarrollo Web que requieren altas prestaciones multimedia es Flash. En las plataformas móviles Flash no ha conseguido la misma penetración que en los sistemas de escritorio, no siendo soportado por todos los móviles, y existiendo gran cantidad de versiones que hacen complicado un desarrollo que funcione en varios dispositivos a la vez. Adobe trabaja desde hace tiempo en AIR para móviles pero es solo compatible con unos pocos Sistemas Operativos.

La poca aceptación de flash en dispositivos móviles se debe, en parte, a que Apple no permite runtimes de terceros, por lo que no está disponible ni en iPhone ni en iPad, y porque, como se ha comentado, las principales plataformas móviles están apostando por HTML5.

La posibilidad de acceso a todas las funcionalidades de los dispositivos móviles, desde aplicaciones Web, ya sea mediante HTML5, Flash o cualquier otro tipo de plataforma de desarrollo, convertiría el desarrollo de aplicaciones Web como un serio candidato frente a las aplicaciones nativas, quizás no para aplicaciones con altos requerimientos de hardware, pero si para desarrollos como el afrontado en el actual proyecto. Por tanto, en la actualidad, aún es necesaria una plataforma de desarrollo que permita el acceso a todas las capacidades de los dispositivos móviles.

En cuanto a la aplicación Real-A desarrollada en el actual proyecto, permite varias líneas de trabajo y mejoras:

- Un punto crítico de la aplicación es el rendimiento del reconocimiento de imágenes, una línea de investigación es el estudio de distintos algoritmos de reconocimiento, para mejorar los resultados de utilizado actualmente.
- En el mismo sentido, la búsqueda de imágenes en base de datos es actualmente bastante pesada en cuanto a tiempos de ejecución, cosa que mejoraría sustancialmente estudiando distintos métodos de indexación.
- La utilización del GPS como apoyo al reconocimiento de imágenes, e incluso para su total sustitución a la hora de obtener informaciones relevantes para el usuario, sería otra forma de mejorar el rendimiento.
- Otro aspecto crítico, es el acceso a la cámara del dispositivo, siendo esta parte dependiente de dispositivo, siendo sólo funcional en dispositivos Symbian S60, por lo que se puede intentar extrapolar el acceso a la cámara y al micrófono a otras plataformas, de forma que la aplicación detectaría la plataforma y mostraría la página con la implementación adecuada, siendo el resto de la aplicación Web totalmente funcional e independiente de la plataforma.
- Además el actual proyecto, puede servir como base para el estudio de otros sistemas de Realidad Aumentada, como la superposición de imágenes tridimensionales sobre la imagen real de la cámara, reconocimiento de caracteres y traducción, reconocimiento de rostros, etc.
- Otra línea de trabajo sería el desarrollo de una interfaz nativa para el soporte de la Realidad Aumentada, como el acceso a la cámara, GPS, etc., mientras que la aplicación Web seguiría soportando los procesos de reconocimiento de imágenes y obtención de información, dando soporte a la aplicación nativa mediante servicios Web.

En general, tanto las aplicaciones de Realidad Aumentada como el desarrollo de aplicaciones Web sobre plataformas Web, así como la utilización y evolución de paradigmas MDA y MDD, tienen grandes perspectivas de futuro.

7. Bibliografía.

- [1] ComScore, Inc. (2010, Marzo) UK Leads European Countries in Smartphone Adoption with 70% Growth in Past 12 Months. [Online].
http://www.comscore.com/Press_Events/Press_Releases/2010/3/UK_Leads_European_Countries_in_Smartphone_Adoption_with_70_Growth_in_Past_12_Months
- [2] TAPTAP Networks & The Nielsen Company. Estudio de Mercado Mobile internet. [Online]. <http://www.slideshare.net/sorejon/estudio-mercadomobileinternetq42010-7445315>
- [3] P. Milgram and F. Kishino, A TAXONOMY OF MIXED REALITY VISUAL DISPLAYS.: IEICE Transactions on Information Systems, Vol. E77-D, No. 12., 1994.
- [4] Julie Carmignani et al., "Augmented reality technologies, systems and applications," Multimedia Tools and Applications, vol. 51, no. 1, pp. 341-377, Jan. 2011.
- [5] Quest Visual. Word Lens de Quest Visual. [Online]. <http://questvisual.com/>
- [6] Layar. Augmented Reality Browser: Layar. [Online]. <http://www.layar.com/>
- [7] Vito Tecnology. star-walk. [Online]. <http://vitotechnology.com/star-walk.html>
- [8] Wikipedia, Mobile operating system. [Online].
http://en.wikipedia.org/wiki/Mobile_operating_system
- [9] Worklight, Native, Web or Hybrid Mobile App.: Worklight Webinar, 2011.
- [10] Nitobi. Phonegap. [Online]. <http://www.phonegap.com/>
- [11] Rhomobile. Rhomobile Rhodes. [Online]. <http://rhomobile.com/products/rhodes/>
- [12] AppCelerator. AppCelerator Titanium Mobile. [Online].
<http://www.appcelerator.com/products/titanium-mobile-application-development/>
- [13] Worklight. Worklight. [Online]. <http://www.worklight.com/>
- [14] Maxa Blog. Mobile apps choices: Native Apps vs Web Apps. [Online].
<http://mkblob.exadel.com/2011/04/mobile-apps-choices-native-apps-vs-web-apps/>
- [15] Nokia, APIBridge Web Runtime API. APIBridge Web Runtime API. [Online].
http://www.developer.nokia.com/Community/Wiki/APIBridge_Web_Runtime_API
- [16] Nokia, Flash Lite API Bridge Interface. (2009, Dec.) Flash Lite API Bridge Interface. [Online].
http://www.developer.nokia.com/Community/Wiki/Flash_Lite_API_Bridge_Interface
- [17] WebRatio. [Online]. www.webratio.com
- [18] Giovanni Toffetti, Sara Comai, Juan Carlos Preciado, and Marino Linaje, "State-Of-The-Art and Trends in the Systematic Development of Rich Internet Applications," Journal of Web Engineering, Rinton Press, Vol.10 No.1 March 15, 2011 , pp. 070-086, 2011.
- [19] Stefano Ceri et al., Designing Data-Intensive Web Applications. USA: Morgan Kaufmann, 2002.
- [20] ScientiaMobile, Inc. ScientiaMobile, Inc. [Online]. <http://www.scientiamobile.com/>
- [21] D. G. Lowe, "Object recognition from local scale-invariant features," in Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, Kerkyra, Greece, 1999, pp. 1150--1157 vol.2.
- [22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vision, vol. 60, no. 2, pp. 91--110, 2004.
- [23] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "SURF: Speeded Up Robust Features," in 9th European Conference on Computer Vision, vol. 110, 2008, pp. 346-359.

- [24] J. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in Conference on Computer Vision and Pattern Recognition, Puerto Rico, 1997, pp. 1000–1006.
- [25] opencv. [Online]. <http://opencv.willowgarage.com/wiki/Welcome>
- [26] Raúl Igual and Carlos Medrano. (2008) Tutorial de OpenCV. [Online]. http://docencia-eupt.unizar.es/ctmedra/tutorial_opencv.pdf
- [27] The PostgreSQL Global Development Group. PostgreSQL 8.3.15 Documentation. [Online]. <http://www.postgresql.org/docs/8.3/interactive/index.html>
- [28] Roberto Andrade Fonseca, Programación de funciones en PL/pgSQL para PostgreSQL.: ABL Consultores, S.A., 2002.
- [29] Caceres Joven. [Online].
http://www.caceresjoven.com/paginas/turismo/monumentos/monumentos_list.asp
- [30] Portal de Turismo del Ayuntamiento de Cáceres. [Online].
<http://turismo.caceres.es/plano-ciudad>
- [31] Northrop Grumman Corporation. (2010, Dec.) Cloc - Count Lines of Code. [Online].
<http://cloc.sourceforge.net/>