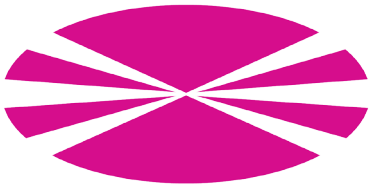
U N I V E R S I D A D E D A C O R U Ñ A

FACULTAD DE INFORMÁTICA

Departamento de Tecnologías de la Información y las Comunicaciones



Tesis Doctoral

ESTUDIO DE TÉCNICAS INMERSIVAS PARA LA FUSIÓN DE DATOS EN TIEMPO REAL Y SU APLICACIÓN EN ENTORNOS DE TELEMONITORIZACIÓN

DOCTORANDO: **ÁNGEL GÓMEZ GARCÍA**

DIRECTORES: **BERNARDINO ARCAY VARELA**
JOSÉ CARLOS DAFONTE VÁZQUEZ

A Coruña, Septiembre de 2015

Estudio de técnicas inmersivas para la fusión de datos en tiempo real y su aplicación en entornos de telemonitorización

Autor: Ángel Gómez García

Tesis Doctoral UDC / 2015

Directores: Bernardino Arcay Varela

José Carlos Dafonte Vázquez

Departamento de Tecnologías de la Información y las Comunicaciones

UNIVERSIDADE DA CORUÑA

D. **BERNARDINO ARCAY VARELA**, Profesor Catedrático de Universidad en el Área de *Ciencias de la Computación e Inteligencia Artificial* de la *Universidade da Coruña*, y D. **JOSÉ CARLOS DAFONTE VÁZQUEZ**, Profesor Titular de Universidad en el Área de *Ciencias de la Computación e Inteligencia Artificial* de la *Universidade da Coruña*, HACEN CONSTAR QUE:

La memoria "**Estudio de técnicas inmersivas para la fusión de datos en tiempo real y su aplicación en entornos de telemonitorización**" ha sido realizada por D. **ÁNGEL GÓMEZ GARCÍA**, bajo nuestra dirección, en el Departamento de *Tecnologías de la Información y las Comunicaciones* de la *Universidade da Coruña*, y constituye la Tesis que presenta para optar al Grado de Doctor en Informática por la *Universidade da Coruña*.

A Coruña, 29 de Septiembre de 2015

D. Bernardino Arcay Varela D. José Carlos Dafonte Vázquez

Agradecimientos

En primer lugar, quiero expresar mi más profundo agradecimiento a mis dos directores, Berni y Dafonte, no sólo por toda la ayuda proporcionada para realizar esta tesis doctoral sino sobre todo por confiar en mí desde el principio. Sólo debido a vuestra perseverancia y aliento he podido culminar este trabajo.

A mis compañeros del grupo de investigación "*Laboratorio Interdisciplinar de Aplicaciones de la Inteligencia Artificial* - l (i a) 2", tanto a los miembros actuales como a los que pasaron en algún momento por sus filas y siguieron su camino.

A mis colegas de la *Facultad de Informática* de la UDC, profesores e investigadores a los que conozco y aprecio. En especial a la gente del departamento de *Tecnologías de la Información y las Comunicaciones*.

A mis compañeros de trabajo en el *Servicio de Informática y Comunicaciones* (SIC) de la UDC, especialmente al CeCaFI y al *Área de Investigación y Biblioteca* en la que llevo trabajando un montón de tiempo. Formar parte del *Personal de Administración y Servicios* como analista informático es una gran responsabilidad pero también me ha permitido participar en proyectos que de otra forma nunca habría podido realizar. Tampoco quiero olvidar a los miembros del antiguo SIAIN (*Servicios Informáticos de Apoyo a la INvestigación*) con los que tuve la suerte de coincidir.

A mi familia, que nunca han entendido demasiado a qué se dedican los "*informáticos*" pero que siempre me han apoyado. Gracias a ellos he podido dedicarme a este loco mundo de la Informática, capaz de reinventarse cada día y dejarte obsoleto cuando menos te lo esperas.

Gracias a todos!!,

Ángel

A mi familia

"We can only see a short distance ahead,
but we can see plenty there that needs to be done"

Alan M. Turing

Computing Machinery and Intelligence. MIND, vol. 49, nº236, pp. 433-460 (1950)

Resumo

Un campo activo de investigación, nos últimos anos, dentro do estudo dos sistemas de supervisión en tempo real, é o da análise e deseño de interfaces de visualización que cumpran cos requisitos asociados ao problema, isto é: grande volume de información; fusión de información simbólica, numérica, e temporal; procesamento distribuído; transmisión e visualización en tempo real; etc.

O obxectivo principal desta tese centrase en demostrar a validez dos traballos realizados dentro dos sistemas de monitorización en tempo real coa finalidade de mellorar os procesos de adquisición, tratamento e transmisión dos datos recollidos para a súa posterior visualización nunha interface adecuada. Así, desenvolveuse un sistema de xestión e supervisión de recursos informáticos, baseado en axentes intelixentes, para mellorar a distribución de procesos nun sistema de Telemedicina. Tamén, se implementaron ferramentas de monitorización de equipamentos informáticos e redes para complementar e mellorar os sistemas de telemonitorización existentes empregando interfaces de visualización avanzadas (interfaces 2D/3D, sistemas inmersivos, etc.), incluíndo aplicacións orientadas a dispositivos móbiles capaces de aproveitar os seus sensores hardware para conseguir interfaces gráficas interactivas con novas funcionalidades: representacións 3D virtuais, realidade aumentada, recoñecemento de códigos gráficos, etc.

Palabras chave

telemonitorización, telesupervisión, entornos inmersivos, sistemas distribuidos, fusión de datos, tempo real, visualización avanzada, telemedicina, monitorización de equipamentos e redes, interface gráfica de usuario.

Resumen

Un campo activo de investigación, en los últimos años, dentro del estudio de los sistemas de supervisión en tiempo real, es el del análisis y diseño de interfaces de visualización que cumplan con los requisitos asociados al problema, esto es: gran volumen de información; fusión de información simbólica, numérica, y temporal; procesamiento distribuido; transmisión y visualización en tiempo real; etc.

El objetivo principal de esta tesis se centra en demostrar la validez de los trabajos realizados dentro de los sistemas de monitorización en tiempo real con la finalidad de mejorar los procesos de adquisición, tratamiento y transmisión de los datos recogidos para su posterior visualización en una interfaz adecuada. Así, se ha desarrollado un sistema de gestión y supervisión de recursos informáticos, basado en agentes inteligentes, para mejorar la distribución de procesos en un sistema de Telemedicina. También, se han implementado herramientas de monitorización de equipamientos informáticos y redes para complementar y mejorar los sistemas de telemonitorización existentes utilizando interfaces de visualización avanzadas (interfaces 2D/3D, sistemas inmersivos, etc.), incluyendo aplicaciones orientadas a dispositivos móviles capaces de aprovechar sus sensores hardware para conseguir interfaces gráficas interactivas con nuevas funcionalidades: representaciones 3D virtuales, realidad aumentada, reconocimiento de códigos gráficos, etc.

Palabras clave

telemonitorización, telesupervisión, entornos inmersivos, sistemas distribuidos, fusión de datos, tiempo real, visualización avanzada, telemedicina, monitorización de equipamientos y redes, interfaz gráfica de usuario.

Abstract

In recent years, an active area of research within the study of real-time monitoring systems has been the analysis and design of visualization interfaces that respond to the specific features of said systems: large amounts of information; the fusion of symbolic, numeric, and temporal information; distributed processing; real-time transmission and visualization; etc.

The main purpose of this thesis is to prove the validity of applications developed under real-time monitoring systems in order to improve the process of acquisition, processing, and transmission of the collected data to be displayed in a suitable interface. As such, we develop a system for the management and supervision of IT resources based on intelligent agents, so as to improve the distribution of processes in a telemedicine system. Also, we implement monitoring tools for computer equipment and networks so as to complement and improve existing telemonitoring systems by means of advanced visualization interfaces (2D/3D interfaces, immersive systems, etc.); this includes mobile applications that use their hardware sensors to create interactive graphical interfaces with new functionalities: virtual 3D representations, augmented reality, recognition of graphic codes, etc.

Keywords

telemonitorization, telesupervision, immersive environments, distributed systems, data fusion, real-time, advanced visualization, telemedicine, IT infrastructure monitoring, graphical user interface.

Índice general

1 Introducción

1.1. CONTEXTUALIZACIÓN

Los avances tecnológicos y el uso de las tecnologías de la información, en prácticamente todas las áreas, han posibilitado la obtención y registro de grandes cantidades de datos con el objetivo de mejorar la realización de tareas específicas. Estos datos serán obtenidos mediante diferentes sistemas de adquisición, en función de su origen, para ser posteriormente tratados por sistemas de procesamiento y visualización adecuados. El procesamiento de los datos recogidos permite realizar una clasificación previa para generar información significativa, y mediante unas interfaces adecuadas se podrá plasmar esta información de tal modo que se adapte a las necesidades funcionales del problema y de los usuarios finales del sistema.

Un caso específico dentro de estas áreas de aplicación de las tecnologías de la información son aquellas que requieren de sistemas de supervisión en tiempo real mediante redes de comunicaciones, incluyendo tanto sistemas centralizados como sistemas distribuidos. En estos casos se tendrá que tener en cuenta la variable temporal para que el procesado y visualización de la información se realice de la forma más rápida y eficiente posible. También habrá que tener en cuenta que este tipo de sistemas requieren del manejo de grandes volúmenes de información y por lo tanto se van a tener que aplicar técnicas como: fusión de información simbólica, numérica, y temporal; procesamiento distribuido en red; contemplar mecanismos de mantenimiento de la coherencia e integridad de la información; compresión de paquetes; transmisión y visualización en tiempo real; etc.

Un campo activo de investigación, en los últimos años, dentro del estudio de los sistemas de supervisión en tiempo real y arquitectura distribuida, es el del análisis y diseño de nuevas interfaces de visualización que ayuden a cumplir con los requisitos asociados a un determinado problema. En estos casos hay que tener en cuenta que las aplicaciones que hagan un tratamiento incorrecto de la información, o dispongan de una visualización inadecuada o con un exceso de datos en pantalla, o con un desfase temporal importante, pueden provocar rechazo por parte de los usuarios finales y como consecuencia afectarán de forma sustancial al uso y rendimiento estimado. Sólo mediante el estudio específico de cada problema y de su ámbito de aplicación se podrá realizar un análisis funcional que permita adaptar el desarrollo del sistema a las necesidades de sus usuarios. Además, serán los prototipos, pruebas, e indicaciones del usuario final las que definirán el nivel de usabilidad de la interfaz implementada, lo cual permitirá conocer el grado de satisfacción alcanzado y ayudará a mejorar el diseño final del sistema.

Como ya se ha indicado, existen múltiples campos de aplicación en los cuales es necesario tener en cuenta la información obtenida en tiempo real para su procesamiento y visualización adecuada. La mayor parte de estos casos de aplicación tienen como misión la supervisión o monitorización de la información y por ese motivo se pueden denominar como "*sistemas de supervisión*" o "*sistemas de monitorización*". Serán estos sistemas, aplicados en diferentes campos, los que centren el desarrollo de esta tesis.

Las técnicas generales que subyacen en estos sistemas de supervisión pueden ser estudiadas y adaptadas para su aplicación en muchos campos debido a que todos ellos comparten similitudes esenciales tales como: procesos de adquisición de grandes cantidades de datos, comunicación y transmisión de información, procesamiento de datos, y visualización avanzada en tiempo real.

El procesamiento de los datos proporcionados por los sistemas de adquisición, dará lugar a la obtención de la información necesaria para el sistema de supervisión, y permitirá detectar y establecer interrelaciones entre datos. Esta información será fusionada y agrupada para generar los parámetros críticos del sistema que serán utilizados por las interfaces de visualización gráfica. Además, será esta información la que alimente a los sistemas de conocimiento encargados

de la toma de decisiones ante las posibles incidencias que se puedan producir, llegando a sugerir las acciones necesarias para solventarlas.

Inicialmente las interfaces de visualización de los sistemas de supervisión se limitaban a mostrar información descriptiva o esquemática de la información procesada por el sistema. Sin embargo, las tecnologías existentes en la actualidad nos permiten experimentar y desarrollar interfaces avanzadas que complementen a las existentes. Entre estas tecnologías destacamos la posibilidad de utilizar entornos 3D interactivos para la representación de la información o para simular el entorno real de la aplicación. Así, se conseguirá mejorar la interacción e interpretación de la información por parte del usuario, acercándolo al entorno real y aumentando el nivel de atención ante incidencias. Este tipo de mejoras pueden limitarse a una visualización 3D avanzada o llegar a utilizar técnicas como la realidad aumentada o los sistemas inmersivos.

1.2. ÁMBITOS DE APLICACIÓN

Una vez contextualizado de forma general el dominio del problema, trataremos de introducir cada campo de aplicación utilizado en el desarrollo de esta tesis doctoral.

Inicialmente, el proyecto se centró en el ámbito de la Telemedicina y la monitorización clínica, más específicamente en el seguimiento en tiempo real de pacientes internados en una Unidad de Cuidados Intensivos (UCI) de un hospital. La selección de este ámbito de trabajo fue realizada debido a los desarrollos y conocimientos previos sobre este tema, ya que se trataba de una de las líneas de investigación principales del grupo de investigación. En esta línea se centraron las investigaciones en la gestión de un sistema de telemonitorización para la ayuda al personal clínico y en la inclusión de un sistema de control de tareas distribuido en los equipos de la red hospitalaria.

Posteriormente, se continuó con el desarrollo en el ámbito de aplicación de la supervisión y monitorización en tiempo real de Centros de Procesamiento de Datos (CPD), concretamente en la supervisión de sistemas y redes. Este cambio se debía principalmente a la posibilidad de aplicar los conocimientos y desarrollos realizados en el ámbito de la telemonitorización de unidades críticas para tratar de

aportar nuevas herramientas que complementasen a los sistemas de supervisión de recursos de un CPD. Además, en este ámbito de aplicación se disponía de una mayor facilidad para la realización de pruebas experimentales en un entorno de trabajo real.

Ambos campos de aplicación comparten características importantes como: manejan una componente temporal crítica, precisan detección y notificación de anomalías en tiempo real, necesitan de una optimización de recursos ante la ejecución de tareas (repartos de carga y procesamiento distribuido), precisan de interfaces de usuario accesibles y adaptadas a diferentes tipos de dispositivos, etc.

1.3. OBJETIVOS

Esta tesis doctoral surge como una rama de desarrollos dirigida a servir de continuación a los trabajos de investigación previamente realizados por el grupo en el que se encuentra integrado el autor. Concretamente, siguiendo una de las líneas de investigación principales del grupo y en la que se han centrado algunos de los trabajos de los directores de esta tesis: *los sistemas de supervisión en tiempo real*.

El objetivo principal establecido en esta tesis se centra en demostrar la validez de los trabajos realizados dentro de los sistemas de monitorización en tiempo real con la finalidad de mejorar los procesos de adquisición, tratamiento y transmisión de los datos recogidos para su posterior visualización en una interfaz adecuada.

Para conseguir este objetivo principal se han planteado una serie de objetivos secundarios:

- Desarrollo de un sistema de gestión y supervisión de recursos informáticos para mejorar la distribución de procesos en un sistema de telemonitorización de unidades críticas. Para ello, se utilizarán tecnologías de agentes inteligentes capaces de dotar al sistema global de mecanismos de planificación de procesos que tengan en cuenta la disponibilidad de recursos y el estado de las redes de datos en tiempo real.

- Mejora de las herramientas de monitorización de equipamientos informáticos y redes de datos mediante el diseño e implementación de sistemas complementarios centrados en la gestión inteligente de procesos.
- Aplicación de técnicas de visualización avanzada (interfaces 2D/3D, sistemas inmersivos, etc.) en el campo de la gestión de sistemas y redes, con el objetivo de complementar y mejorar las funcionalidades que proporcionan los sistemas de supervisión y monitorización existentes.
- Estudio y desarrollo de nuevas aplicaciones basadas en la utilización de dispositivos hardware móviles para su integración con herramientas de monitorización de sistemas. Estas aplicaciones complementarán los sistemas existentes, aprovechando los sensores presentes en estos dispositivos (pantalla táctil, cámara, acelerómetro, GPS, etc.), para dotarlos de interfaces gráficas interactivas con funcionalidades avanzadas: representaciones 3D virtuales del entorno de trabajo real, realidad aumentada, reconocimiento de códigos gráficos, etc.).

1.4. ESTRUCTURA DE LA TESIS

Para la organización de este documento se ha optado por seguir el orden cronológico de los desarrollos realizados, de este modo se facilita el seguimiento de la evolución de la investigación y sus correspondientes interrelaciones. Así, la estructura que se seguirá está dividida en los siguientes capítulos:

- **Capítulo 1**. Dedicado a la introducción del dominio del problema, ámbito de aplicación, problemática, y objetivos planteados con esta tesis doctoral.
- **Capítulo 2**. Describe la metodología de trabajo utilizada y los desarrollos realizados.
- **Capítulo 3**. Se presentan los principales conceptos correspondientes al campo de aplicación de la telemonitorización en tiempo real en el ámbito médico. También se introduce el sistema de Telemedicina desarrollado por nuestro grupo de investigación. En este proyecto se comentan las fases que llevan desde la adquisición de los datos provenientes de los dispositivos

médicos hasta el desarrollo de interfaces adecuadas para la visualización de la información.

- **Capítulo 4**. Estudio de las tecnologías de agentes inteligentes para la gestión de procesos y su aplicación en el desarrollo de un sistema de control de procesos en un sistema de Telemedicina.
- **Capítulo 5**. Presentación de los conceptos correspondientes al ámbito de aplicación de la supervisión de sistemas y redes. Comparación entre las herramientas de monitorización y supervisión existentes. Aplicación de técnicas de gestión inteligente y visualización avanzada en este campo. Estos desarrollos se plantean como una variante en la cual aprovechar los conocimientos adquiridos con los sistemas de telemedicina para su aplicación en la monitorización de equipamientos informáticos.
- **Capítulo 6**. Desarrollo de una aplicación de visualización inmersiva para la ayuda a la monitorización de equipamientos informáticos y redes de datos. Experimentación con nuevas formas de representación de la información de un sistema de telemonitorización.
- **Capítulo 7**. Aplicación de las nuevas tecnologías y dispositivos móviles para su utilización en el entorno de trabajo de la supervisión y gestión de sistemas y redes. Utilización de técnicas de realidad aumentada para el desarrollo de interfaces de monitorización avanzadas.
- **Capítulo 8**. Creación de una herramienta de modelización de infraestructuras bajo plataformas móviles y su integración con un sistema de telemonitorización.
- **Capítulo 9**. En este capítulo se abordarán los resultados y conclusiones finales de esta tesis doctoral. También se plantearán nuevas metas posibles para la realización de futuros trabajos de investigación.

2
Metodología de trabajo

2.1. DESCRIPCIÓN

Los entornos que requieren de sistemas de telemonitorización en tiempo real se caracterizan por la criticidad de las tareas que realizan, siendo de vital importancia la detección y reacción ante posibles anomalías en su funcionamiento normal. Para conseguir este objetivo se desarrollan herramientas específicas que tratan de gestionar adecuadamente una gran cantidad de datos, provenientes de todo tipo de sensores y dispositivos, para servir de ayuda al personal técnico. Uno de los inconvenientes principales será cómo visualizar la información en la interfaz gráfica, de tal modo que se facilite la localización de un problema o que se puedan destacar aquellos valores que están fuera de sus umbrales de seguridad.

En este trabajo se presentan algunos experimentos realizados para tratar de mejorar la forma en que se representa la información, utilizando interfaces gráficas avanzadas para reflejar de forma visual e inmersiva el entorno real de la aplicación. De este modo, se aportan nuevas opciones a los sistemas de monitorización clásicos con interfaces descriptivas o centradas en mostrar el detalle de todos los valores recibidos de cada dispositivo.

Teniendo en cuenta que los objetivos planteados con esta tesis, indicados el capítulo anterior, incluyen el desarrollo de aplicaciones software, se han aplicado técnicas y herramientas de *Ingeniería del Software* (IS) para conseguir desarrollos de software fiables y eficientes. Aunque existen múltiples definiciones del término IS podemos citar la propuesta por Fritz Bauer como una de las principales:

- "*La Ingeniería del Software es el establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales*" [Naur 1969].

Posteriormente, el IEEE [IEEE 1993] ha desarrollado una definición más completa del término:

- "*La ingeniería de software es: 1) La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software. 2) El estudio de enfoques según el punto 1*".

Para conseguir desarrollar software de calidad, la IS proporciona *procesos*, *métodos* y *herramientas* que permiten gestionar la creación de proyectos software [Pressman 2010]. El *proceso* define una estructura que debe establecerse para la obtención eficaz de tecnología de IS. El proceso de software forma la base para el control de la administración de proyectos de software, y establece el contexto en el que se aplican métodos técnicos, se generan productos del trabajo, se establecen puntos de referencia, se asegura la calidad y se administra el cambio de manera apropiada. Los *métodos* de la IS se basan en un conjunto de principios fundamentales que gobiernan cada área de la tecnología e incluyen actividades de modelación y otras técnicas descriptivas. Las *herramientas* de la IS proporcionan un apoyo automatizado o semiautomatizado para el proceso y los métodos.

Se han utilizado modelos evolutivos para el desarrollo de las aplicaciones. Los modelos evolutivos son iterativos y se caracterizan por la forma en la que permiten desarrollar versiones cada vez más completas del software. Así, se han utilizado *prototipos* para ayudar a mejorar el desarrollo y cumplir con los requisitos del proyecto. Mediante estos prototipos se mejora la comprensión del problema y sus características, incluyendo la forma de interacción entre el usuario y la máquina. Esto permite que todo el sistema, o algunas de sus partes, se construyan rápidamente para aclarar ciertos aspectos en los que se aseguren que el desarrollador y el usuario estén de acuerdo en lo que se necesita así como también la solución que se propone para dicha necesidad. Según este modelo el ciclo de vida sigue las siguientes pautas:

- Investigación preliminar. Definición del problema, estudio del ámbito de aplicación y de los sistemas existentes en la organización, etc.
- Especificación de requisitos y prototipado:
    - Análisis y especificación.
    - Diseño y construcción.
    - Evaluación.
    - Modificación.
- Diseño técnico. Diseño detallado, rediseño del prototipo, documentación, etc.
- Programación y prueba. Implementación y pruebas en base al diseño técnico.
- Operación y mantenimiento. Instalación del sistema y tareas de mantenimiento posteriores.

La metodología de trabajo utilizada sigue un enfoque orientado a objetos basado en la identificación y organización de conceptos del dominio de la aplicación, sin centrarse en su representación final en un lenguaje de programación. Entre las características de este tipo de metodologías destacan: son interactivas e incrementales, es fácil dividir el sistema en subsistemas independientes, se fomenta la reutilización de componentes, etc.

La experimentación realizada ha dado lugar al desarrollo de una serie de herramientas de ayuda a la telemonitorización. Estas herramientas mantienen un objetivo común e intentan abordar nuevas formas de gestionar y visualizar la información de monitorización. El desarrollo de cada aplicación ha sido realizado para cubrir una determinada función pero siempre tratando de complementar a los desarrollos anteriores. Así, en la siguiente lista se detallan los desarrollos realizados, coincidiendo con su ordenación cronológica:

1. Desarrollo de una herramienta para la gestión inteligente de procesos en un sistema de Telemedicina en UCIs. Este proyecto tiene como objetivo la gestión y regulación de los procesos ejecutados en los equipos conectados al sistema, consiguiendo una optimización del ancho de banda disponible y

un mejor aprovechamiento de los recursos. Utiliza técnicas de agentes inteligentes para recoger la información sobre el estado de cada equipo y poder distribuir y priorizar la realización de tareas requeridas por el usuario. Cada agente dispone de un sistema de conocimiento basado en reglas para la toma de decisiones y de una interfaz de visualización capaz de mostrar la red de agentes en una representación 3D interactiva. Posteriormente, se realizó una adaptación de esta herramienta de gestión de procesos en UCIs para su uso en el ámbito de los sistemas de monitorización de equipamientos informáticos.

2. Creación de una herramienta Web inmersiva para su integración con un sistema de monitorización de equipamientos y redes de datos. Su función principal es dotar al sistema de monitorización utilizado en el CPD de una herramienta de apoyo basada en la representación 3D virtual del entorno real en el que se ubican los equipamientos bajo supervisión. Proporciona al personal técnico una alternativa para la gestión del estado del sistema centrada en la visualización e interacción por las localizaciones del espacio 3D. Se utilizan técnicas de fusión de datos para interpretar la información a visualizar sobre cada equipo. Además, incluye un módulo de conocimiento basado en reglas para detectar y notificar la existencia de estados de alarma, proponiendo posibles soluciones al personal técnico.

3. Utilización de técnicas de realidad aumentada sobre dispositivos móviles para la ayuda a la monitorización de un CPD. Se centra en la experimentación con nuevas interfaces de visualización interactivas para complementar los sistemas de monitorización de equipamientos informáticos existentes. Utilizando los sensores disponibles en los dispositivos móviles actuales se consigue acceder a la información relevante de un equipo, desde su ubicación real, simplemente utilizando el reconocimiento de códigos gráficos de identificación. Los datos de cada equipamiento se mostrarán de forma resumida y superpuesta sobre la imagen real, facilitando su consulta al personal técnico desplazado al emplazamiento real de los activos.

4. Creación de una herramienta para dispositivos móviles integrada con un sistema de monitorización de un CPD para crear y gestionar una representación gráfica de espacios e infraestructuras. Este proyecto trata de mejorar la localización de los recursos en grandes CPDs mediante la modelización virtual del entorno. Permite modelar los diferentes espacios físicos y posicionar en ellos los equipamientos de que disponen. Estos equipamientos serán representados por objetos 3D interactivos e incluirán la información necesaria para identificarlos. Además, esta herramienta también permite gestionar el estado de los equipamientos bajo supervisión, relacionando los objetos 3D con el dispositivo real que representan a través de la información proporcionada en tiempo real por el sistema de monitorización.

3

Telemedicina y monitorización clínica

3.1. INTRODUCCIÓN

Las mejoras tecnológicas aplicadas al ámbito médico han proporcionado grandes avances en la asistencia a los cuidados de la salud, incluyendo aquellos debidos a la aplicación de las tecnologías de la información y las comunicaciones. Entre estos avances se incluye el desarrollo de las primeras aplicaciones de gestión de los registros electrónicos de los pacientes o los inicios de las transmisiones de información clínica a través de líneas telefónicas para el seguimiento de pacientes a distancia.

En muchos casos, las aplicaciones clínicas se centraron en problemas médicos específicos que requerían de la comunicación con especialistas situados en centros remotos para obtener o confrontar terapias diagnósticas. Posteriormente se utilizaron para proporcionar servicios de atención médica primaria en zonas con una población dispersa. Las telecomunicaciones permitieron también la realización de consultas y diagnósticos remotos de los pacientes, e incluso la educación a distancia del personal médico.

El término Telemedicina surge para referirse a la utilización de las tecnologías de telecomunicaciones para intercambiar información médica y proveer servicios de cuidado de la salud, independientemente de barreras geográficas. Existen múltiples definiciones de Telemedicina, algunas de las más destacadas son:

- "*La Telemedicina es la práctica de la medicina sin la confrontación física normal entre médico y paciente por medio de un sistema de comunicación de audio-vídeo interactivo*" [Bird 1975].
- "*La Telemedicina es el uso de las tecnologías de comunicaciones para asistir en la entrega de cuidados de la salud*" [Conrath 1983].
- "*La Telemedicina es el uso de información electrónica y tecnologías de comunicaciones para proporcionar y soportar cuidados médicos cuando la distancia separa a los participantes*" [CECAT 1996].

Una parte importante de la Telemedicina la constituye la denominada "*informática médica*" o "*informática de la salud*", la cual incluye aquellas aplicaciones de las tecnologías informáticas destinadas a la prestación de servicios médicos, tanto para propósitos administrativos como clínicos. Algunas definiciones de este término son las siguientes:

- "*La ciencia de la informática médica es la ciencia del uso de herramientas de análisis de sistemas para el desarrollo de procedimientos (algoritmos) para la gestión, control de procesos, toma de decisiones y análisis científico de los conocimientos médicos*" [Shortliffe 1984].
- La *American Medical Informatics Association* (AMIA) aporta una definición funcional en base a sus funciones principales: "*representación del conocimiento médico, adquisición y presentación de información clínica, gestión de cambios e integración de información*" [AMIA 1999].

En la actualidad existe un conjunto muy amplio de especificaciones estándares que se encargan de cubrir las distintas especialidades sanitarias para la aplicación de las tecnologías informáticas en el campo de la salud. Este es el caso de los estándares ISO ICS 35.240.80 (*IT applications in health care technology*) [ISOICS 2015].

Las aplicaciones de la Telemedicina pueden agruparse en tres grandes grupos dependiendo del tipo de trabajo al que están destinadas:

- **Administrativas**. Dirigidas a la gestión de información administrativa de los centros hospitalarios. Entre la información que deben manejar se

incluye: información demográfica, datos de facturación, ficha clínica del paciente, etc.

- **Clínicas**. Encargadas de realizar tareas diagnósticas o de ayuda al clínico. Dentro de este grupo distinguiremos los siguientes tipos:
    - **Imágenes diagnósticas**. Conjunto de aplicaciones destinadas al manejo de imágenes radiológicas, anatomía patológica, dermatología, u otras de relevancia clínica. Incluirán funciones de procesado, análisis, almacenamiento e integración con otras aplicaciones médicas y su intercambio a través de redes de comunicaciones.
    - **Exámenes de laboratorio**. Sistemas que permiten la integración de equipos de trabajo, gestionando información administrativa y resultados de exámenes. También incluirán otras tareas como la automatización de procesos y la intercomunicación con otros sistemas de información de salud.
    - **Monitorización de pacientes**. Dedicados al registro de información de monitorización del paciente. Se encargarán de gestionar la evolución en tiempo real de sus signos vitales, registros de enfermería y monitorización electrónica procedente de la instrumentación médica.
- **Educativas**. Aplicaciones y servicios de investigación y educación que permiten el intercambio de información o la actualización educativa del personal clínico.

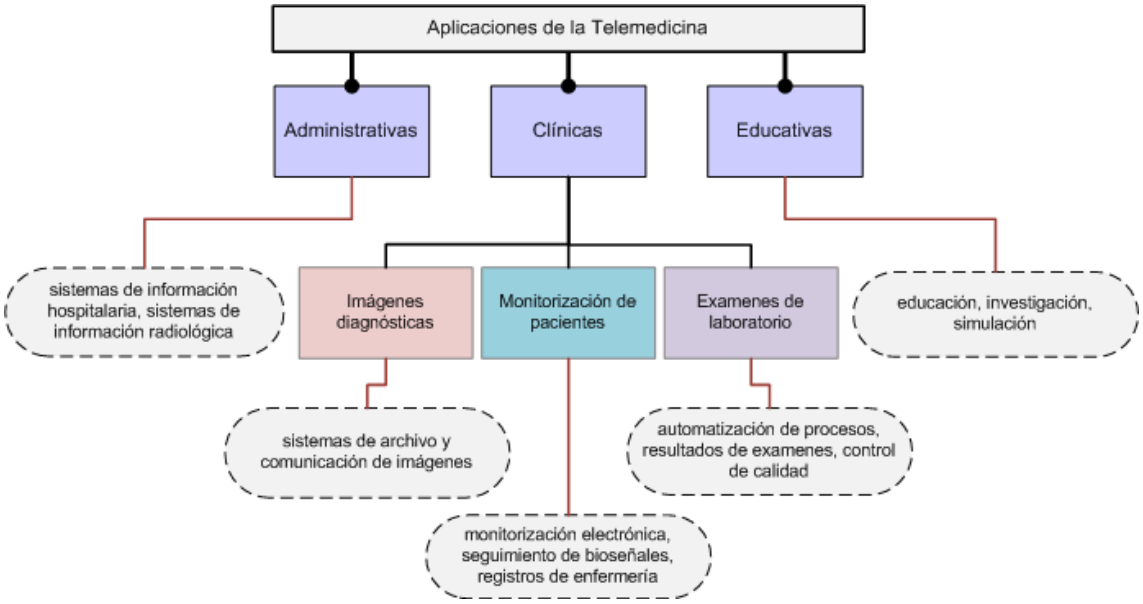


Figura 1. Clasificación de aplicaciones de la Telemedicina

La Telemonitorización y Telesupervisión son servicios de la Telemedicina, centrados en realizar un seguimiento, local o remoto, de la evolución del estado de un paciente. Para ello, estas aplicaciones constarán de una serie de procesos que permitirán obtener los datos procedentes de diferentes dispositivos médicos, para a continuación procesar y transmitir la información de los parámetros críticos a un sistema de supervisión bajo el control del personal clínico. El campo de la telemedicina y la monitorización clínica ha evolucionado significativamente en los últimos años, proporcionando al personal sanitario una gran cantidad de instrumentación biomédica para labores de seguimiento y control del estado fisiopatológico de los pacientes ingresados en un centro hospitalario. Así, podemos citar: respiradores, capnógrafos, monitores hemodinámicos, etc.

Las unidades críticas (UCIs) son una de las áreas del centro hospitalario en las cuales existe una necesidad creciente de incluir constantes evoluciones tecnológicas y sistemas de Telemedicina que proporcionen herramientas para la monitorización y supervisión de los pacientes en tiempo real. Esto ha dado lugar a mejoras importantes en el seguimiento y tratamiento de pacientes críticos en los cuales es vital mantener un control del estado del paciente, con reducidos tiempos de reacción ante cambios. Durante la estancia de estos pacientes en estas unidades se puede considerar que el objetivo funcional es conseguir la transición desde una situación crítica o de inestabilidad fisiológica, hasta un punto de estabilidad y

corrección. Para conseguir esta transición, el paciente pasará por una serie de estados intermedios bajo continua supervisión técnica y humana: adquisición, análisis y validación de la información, definición de niveles de alarma o criticidad, realización de las intervenciones terapéuticas.

Los avances en el campo de la monitorización han llevado a crear completos sistemas informáticos, dotados de ciertos módulos que emplean técnicas de Inteligencia Artificial (IA), para realizar una supervisión inteligente de los parámetros fisiológicos del paciente y contribuir a ayudar al clínico en el diagnóstico. Dichos sistemas permiten el acceso a la información de forma tanto local como remota, utilizando para ello la red del Sistema de Información Hospitalario (SIH). Además, estos sistemas de Telemedicina pueden ampliar su alcance al permitir el acceso e interacción desde el exterior del SIH, utilizando conexiones basadas en protocolos de comunicación seguros, a través de las redes de telecomunicaciones.

3.2. APLICACIONES INMERSIVAS EN TELEMEDICINA

La Realidad Virtual (RV) se puede definir como una funcionalidad que trata de simular, mediante gráficos por computador, entornos dinámicos con una apariencia realista sobre los cuales el usuario puede interactuar en tiempo real. La comunidad científica lleva décadas trabajando en este campo que supone una de las más potentes interfaces hombre-máquina de las que se puede disponer. Este tipo de tecnologías están ligadas al uso de dispositivos hardware que permiten al usuario interactuar con el entorno virtual: cascos, guantes, sensores de posición, etc. Podemos encontrar ejemplos de la aplicación de la RV en multitud de ámbitos como el diseño asistido por computador, simuladores, videojuegos, robótica, medicina, educación, etc.

Los sistemas inmersivos permiten, por medio del uso de técnicas de realidad virtual, que el usuario se introduzca en un entorno sintético en tres dimensiones, en el cual se podrá mover e interaccionar con los objetos incluidos en dicho entorno y en tiempo real. Normalmente este tipo de sistemas requieren del uso de periféricos específicos para realizar acciones sobre el mundo virtual, las cuales pueden llegar a tener su correspondiente acción en el mundo real. Sin embargo,

también es posible disponer de sistemas de este tipo que no requieren de ningún periférico específico para conseguir la inmersión del usuario en la tarea a la que está destinada la aplicación. En estos casos, la interacción con el entorno 3D se limita al uso de dispositivos comunes como: pantalla, teclado, ratón. La ventaja de estos sistemas es en parte económica, al no tener que adquirir nuevos periféricos, pero principalmente permite generalizar los sistemas inmersivos para su funcionamiento en aplicaciones instalables y en entornos Web.

En el campo de la medicina, dentro de la categoría de aplicaciones de Telemedicina que incluyen técnicas inmersivas podemos citar a una gran cantidad de aplicaciones como [Preim 2013]:

- Visualización médica. Incluye las aplicaciones médicas centradas en la visualización anatómica del cuerpo humano, tanto a nivel externo como interno, y permiten el libre desplazamiento por los elementos 3D e imágenes que lo componen. De este modo, se facilita el estudio y detección de las zonas afectadas/dañadas, permitiendo su selección y análisis de forma detallada. Este tipo de aplicaciones también pueden tener como objetivo el uso educativo basado en información gráfica.
- Cirugía virtual (Telecirugía). Consiste en la utilización de aplicaciones de RV destinadas a realizar operaciones de cirugía de forma virtual. Así, se consigue que los especialistas médicos puedan operar de forma remota a un paciente internado en un centro hospitalario situado a gran distancia. Esto permitirá que un clínico, especializado en cierto tipo de operaciones, pueda efectuar una operación sin necesidad de desplazarse físicamente al centro en el que se encuentra el paciente para realizar la intervención, consiguiendo un importante beneficio temporal y económico.
- Visión artificial (visión por computador). Es un campo de la IA destinado a la adquisición, procesamiento, análisis, y reconocimiento de imágenes. Este tipo de técnicas pueden ser dedicadas a aplicaciones que permiten realizar diagnósticos automáticos. En el ámbito médico existen múltiples aplicaciones como por ejemplo: sistemas para la revisión y tratamiento de problemas oftalmológicos, detección de enfermedades dermatológicas, aplicaciones odontológicas, etc.

- Simulación quirúrgica. Las aplicaciones de simulación virtual tratan de representar de la forma más real posible el entorno y la reacción producida al interactuar sobre los objetos que lo componen. La simulación de operaciones quirúrgicas permite al personal clínico el entrenamiento y perfeccionamiento de técnicas quirúrgicas sin ningún peligro para los pacientes. El uso de simuladores virtuales de este tipo proporciona múltiples ventajas y beneficios, algunos ejemplos son los siguientes: mejora el conocimiento de la anatomía de los pacientes, suponen una forma más segura y de menor coste para el aprendizaje de técnicas fundamentales, consigue un ahorro de costes al no precisar de material físico para el entrenamiento, permite realizar pruebas prácticas antes de realizar una operación real consiguiendo mejorar las habilidades del clínico, sirve como alternativa para probar nuevas técnicas de intervención, etc.
- Robótica médica. Se basa en la utilización de robots especialmente diseñados para la ayuda a los clínicos y cuyo control se realiza mediante la interacción con una aplicación inmersivas. Por medio de este tipo de robots es posible la realización de intervenciones quirúrgicas delicadas de forma más segura, proporcionando al personal especializado nuevos procedimientos para mejorar su labor.
- Entornos médicos virtuales. Permiten simular un entorno real para facilitar la consulta y revisión médica diaria. Así, se puede conseguir que el personal clínico realice su operativa habitual, desplazándose por el entorno virtual, sin necesidad de realizar desplazamientos físicos innecesarios. Este tipo de aplicaciones posibilitan el acceso a una representación virtual de un área determinada del centro hospitalario y la consulta del estado actual de cada paciente internado. La información disponible sobre cada paciente puede incluir su monitorización en tiempo real, evolución temporal, historia clínica completa, etc.

La información 2D/3D utilizada por este tipo de aplicaciones clínicas se puede obtener en tiempo real por medio de técnicas no invasivas como sistemas de imagen por resonancia magnética (MRI-*Magnetic Resonance Imaging*), escáneres de tomografía computerizada (CT-*Computed Tomography*, CAT-*Computerized Axial*

Tomography), tomografía por emisión de positrones (PET-*Positron Emission Tomography*), dispositivos de imagen médica por ultrasonidos (ecógrafos), etc. Posteriormente, estos conjuntos de datos pueden ser combinados y renderizados para su utilización en sistemas médicos específicos. Mediante sistemas de este tipo se proporciona al personal especializado herramientas de ayuda para la toma de decisiones diagnósticas. La posibilidad de interactuar con una imagen 2D/3D del paciente facilita tareas como la visualización de la zona afectada, o la planificación de operaciones de cirugía mínimamente invasiva.

Dentro de las aplicaciones de RV en el campo de la Telemedicina destacan las aplicaciones de Telecirugía virtual. Este tipo de aplicaciones utilizan técnicas de RV y robótica para realizar operaciones de cirugía asistida por computador. Al igual que en los simuladores, las sensaciones transmitidas al usuario deben ser similares a las producidas durante una operación en un quirófano real. Así, detalles como el comportamiento simulado de los tejidos o la forma de actuar de los diferentes órganos, deben modelarse como objetos con comportamientos propios. Aspectos como la sensación de peso y rozamiento sobre los órganos del paciente también deben reproducirse en el entorno virtual. Para conseguir reproducir estas sensaciones se utilizan sistemas hápticos diseñados para una función específica. En el caso de las operaciones virtuales estos sistemas se basan en el uso de un conjunto de poleas para manejar el bisturí virtual y en la utilización de una visión estroboscópica.

Figura 2. Ejemplo de sistema de Telecirugía virtual (sistema "da Vinci Si HD Surgical System")

Los denominados sistemas de Cirugía Guiada por la Imagen (IGS-*Image Guided Surgery*) constituyen otro tipo de aplicaciones importantes de la Telecirugía. Estos sistemas se basan en la utilización de la imagen médica obtenida por medio de distintas técnicas y herramientas para guiar al personal clínico durante el procedimiento quirúrgico. Normalmente, las imágenes son obtenidas antes de la intervención y almacenadas en una estación de trabajo conectada a un sensor de posición capaz de registrar la posición 3D del paciente para mostrar las imágenes correspondientes en cada momento de la operación. Durante la operación, el cirujano utiliza instrumentos conectados al sensor de posición para ver las imágenes de la localización que él está tocando. Estos sistemas de ayuda son muy útiles en áreas como la neurología y la traumatología. Además, el procesamiento de las imágenes obtenidas, por ejemplo de tomografías o resonancias, puede ser renderizado para crear recreaciones virtuales interactivas, este es el caso de las endoscopias virtuales.

3.3. SISTEMA DE TELEMEDICINA EN UCIS

En este apartado se describirán las principales características y aportaciones del sistema de Telemedicina desarrollado por nuestro grupo de investigación, y que ha dado lugar a importantes trabajos y proyectos de investigación. En algunos de estos desarrollos también ha participado el autor, pero además este sistema sirve de base para el desarrollo de nuevas funcionalidades que serán descritas en el siguiente capítulo de esta tesis.

El sistema desarrollado se centra en la supervisión inteligente de los pacientes ingresados en unidades críticas y ha dado lugar, entre otros, a los trabajos correspondientes a las tesis doctorales de los investigadores Bernardino Arcay [Arcay 1990] y Carlos Dafonte [Dafonte 2002]. Este sistema de Telemedicina tiene como objetivo principal contribuir a mejorar la atención médica dentro de la UCI de un centro hospitalario. Así, se ha dotado al sistema de las funcionalidades que permiten desde la adquisición en tiempo real de los parámetros fisiológicos del paciente (presión arterial, frecuencia cardíaca, presión intracraneal, presión venosa central, etc.), hasta la visualización y seguimiento de la evolución del paciente por medio de una interfaz gráfica 2D/3D. Además, este sistema incluye

entre sus funcionalidades la posibilidad de establecer comunicaciones mediante videoconferencia para realizar seguimientos a distancia o para la intercomunicación entre expertos clínicos.

El sistema de Telemedicina sigue una arquitectura cliente/servidor conforme al estándar IEEE P1073 MIB (*Medical Information Bus*) [Franklin 1989]. Incluye las aplicaciones locales de monitorización y control, y las aplicaciones correspondientes al sistema servidor encargado de recoger la información procedente de todos los equipos cliente de la UCI.

Teniendo en cuenta que cada uno de los equipos de monitorización conectados a un paciente para registrar sus variables fisiológicas puede proporcionar hasta 30 variables y que cada paciente puede tener conectados hasta 20 dispositivos diferentes [Kampmann 1989], tendremos como resultado la necesidad de gestionar una gran cantidad de información en tiempo real. Además, de esta información también habrá que gestionar otros datos importantes como: los parámetros de configuración (umbrales de alarma, ratios de infusión y de frecuencia respiratoria, etc.), pruebas de laboratorio e información proporcionada por el equipamiento médico para el soporte vital (ventiladores mecánicos, bombas de infusión, etc.). Toda esta información servirá de ayuda al personal clínico especializado, de tal modo que le permita tomar decisiones sobre el diagnóstico y terapia a seguir. Sin embargo, en una UCI, es bastante probable que existan variaciones importantes en el estado general del paciente o que aparezcan complicaciones médicas. Por lo tanto existirá una fuerte dependencia de la patología del paciente, de su evolución y de las reacciones producidas ante la medicación que se le haya proporcionado.

La problemática anteriormente citada unida a la necesidad de tener en cuenta cada vez más dispositivos conectados al paciente, y por lo tanto incrementando los parámetros a analizar, ha dado lugar a la aparición de sistemas de monitorización y supervisión inteligente específicos. Estos sistemas tendrán como función principal servir de ayuda al personal clínico en la toma decisiones. Así, se encargarán de realizar una monitorización y supervisión adecuada a cada caso en particular, presentando de forma clara y concisa aquella información que es más relevante para el personal médico especializado.

3.3.1. Arquitectura del sistema

El estándar utilizado, IEEE P1073 MIB, describe una red de área local (LAN-*Local Area Network*), que sigue una aproximación en capas compatible con el estándar OSI (*Open System Intercommunication*) de la ISO (*International Standard Organization*). El esquema de la arquitectura que define esta norma ha sido mejorado por nuestro grupo de investigación para añadirle una componente de monitorización inteligente [Dafonte 2002]. A continuación se describe cada uno de los componentes que incluye:

- **DCC (*Device Communication Controller*)**. Tiene como función convertir el formato de los datos que proporciona cada monitor clínico a un formato estándar de intercambio de información médica denominado MDDL (*Medical Device Data Language*). Los DCC son dispositivos secuenciales con hardware específicamente desarrollado para la obtención y comunicación de la información de un dispositivo concreto.
- **IBCC (*Intelligent Bedside Communication Controller*)**. Toda la información procedente de los dispositivos DCC es transmitida al monitor IBCC. La red local que engloba al IBCC y los DCCs conectados a él, con una topología en estrella, tiene capacidad para comunicar más de 20 dispositivos médicos que pueden aparecer asociados a un único paciente.
- **IMCC (*Intelligent MIB Communication Controller*)**. Esta plataforma se encarga de recibir y almacenar toda la información proporcionada por los IBCC. La conexión entre el IBCC y el IMCC se realiza a través de una red Ethernet, y debe soportar la transmisión de información de todos los pacientes, es decir, el ancho de banda debe ser suficiente para realizar esta labor.

Aunque la conexión entre el IMCC y el IBCC permite el funcionamiento centralizado del sistema y potencia las capacidades de los equipos locales, no es imprescindible. Así, el IBCC está capacitado para funcionar autónomamente en la red en estrella. Al funcionar de esta forma, este dispositivo se encargará del almacenamiento de toda la información recogida. Cuando el IMCC está conectado al IBCC la información

fluye desde los dispositivos hacia el IMCC a través de las redes de datos, de tal modo que el IBCC se comporta como una pasarela.

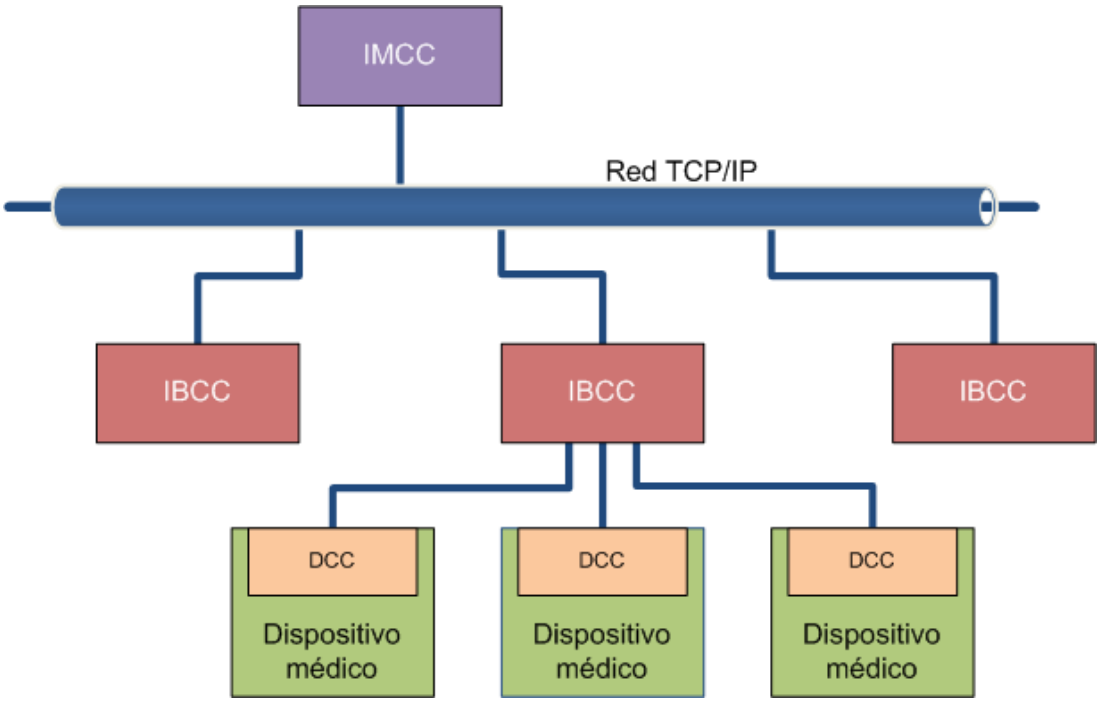


Figura 3. Diagrama de la arquitectura del sistema de Telemedicina basado en el estándar IEEE P1073 MIB

3.3.2. Sistema cliente/servidor

El sistema de Telemedicina en UCIs sigue una arquitectura cliente/servidor clásica [Gómez 1999] y por lo tanto consta de dos subsistemas claramente diferenciados pero interrelacionados por medio de una red de datos:

- **Sistema local**. Estará ubicado cerca del paciente e incluye las aplicaciones necesarias para poder realizar una monitorización local, así como para servir de cliente del sistema de control centralizado. Este subsistema incluirá también las aplicaciones y el hardware necesario para poder realizar la adquisición de los datos proporcionados por la instrumentación médica conectada al paciente.
- **Sistema central**. Equipo situado en una localización distanciada de los pacientes y que se encargará del registro y supervisión de los parámetros correspondientes a cada uno de los sistemas locales de la UCI. Permite al

personal clínico la monitorización y visualización del estado de los pacientes de forma remota.

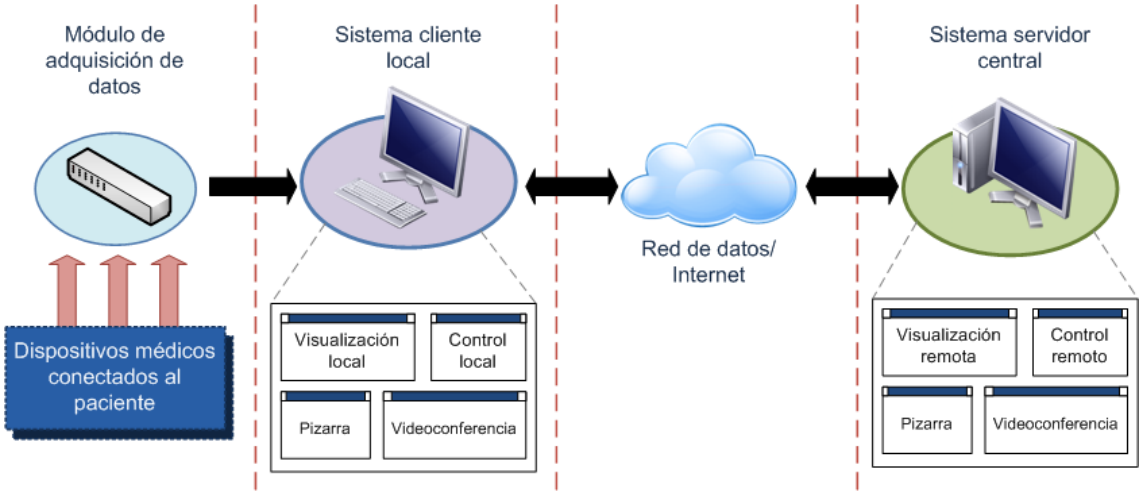


Figura 4. Diagrama general del sistema de Telemedicina

La gestión de las comunicaciones entre el subsistema local y central se realizará por medio de aplicaciones específicas para el cliente y el servidor, basadas en el uso de canales de comunicación mediante *sockets* TCP. Además, con el objetivo de optimizar el ancho de banda disponible, se establecerá una organización mediante canales específicos para cada tipo de información a transmitir/recibir. Así distinguiremos los siguientes canales:

- **Datos de control**. Canal dedicado al envío y recepción de comandos e información relativa al control remoto del subsistema local por parte del subsistema central. Este canal permanecerá abierto durante el tiempo que dure la conexión con el subsistema local.

- **Datos de la adquisición de la instrumentación médica**. Canal dedicado a la transmisión al servidor de las señales procedentes de los dispositivos conectados al paciente. El canal estará abierto mientras exista la conexión con el subsistema local y el envío de los datos se realizará cada vez que se complete la adquisición de un buffer de señales en el subsistema local. El tamaño del buffer puede ser definido en la configuración del sistema pero para mejorar los tiempos de transmisión, y por consiguiente mejorar el ancho de banda disponible, se aplicará un proceso de compresión de datos en tiempo real utilizando las funciones de la librería *zlib* (utilidad de compresión/descompresión de datos sin pérdida) [Gailly 2013].

- **Datos de audio y vídeo**. Canal utilizado para la comunicación entre ambos subsistemas por medio de videoconferencia. Este canal sólo se mantendrá abierto mientras dure la comunicación entre los interlocutores. El envío de esta información se realizará siguiendo estándares de transmisión de audio y vídeo comprimidos. Además, se podrá regular el ancho de banda empleado dependiendo de la carga existente en el sistema.
- **Datos multimedia**. Canal dedicado a la transferencia de todo tipo de archivos médicos (imágenes, documentos, etc.). Este canal será establecido para realizar una tarea y se cerrará al finalizarla. La información puede ser transmitida directamente o aplicando una compresión previa.

La inclusión de capacidades de comunicación por videoconferencia complementa el sistema de Telemedicina pero implica la necesidad de disponer de hardware y software adecuados. Además, habrá que tener en cuenta la forma en que esto afectará al sistema global. Concretamente, el uso de vídeo en tiempo real requerirá un ancho de banda elevado, lo cual puede llegar a afectar a las tareas de monitorización y supervisión inteligente. Para solucionar este problema habrá que establecer una serie de preferencias para la utilización del canal de comunicaciones compartido. Así, la utilización de normas de compresión de audio y vídeo contribuirán a reducir significativamente el ancho de banda utilizado, aunque a costa de una pérdida en la calidad de la transmisión. Por ello, el sistema desarrollado permite que el porcentaje de compresión aplicado pueda ser regulado por software, consiguiendo de este modo que se adapte a las necesidades que se precisen en cada momento. El estándar de transmisión de videoconferencia utilizado originalmente sigue la recomendación ITU-T H.263 [ITU-T 2005].

El sistema utiliza las redes de comunicaciones del centro hospitalario para integrarse en el SIH, pudiendo acceder a la información proveniente no sólo de la instrumentación de la UCI sino también a la información de otros servicios (bases de datos, tomografías, radiografías, analíticas, etc.). Además, proporciona un medio de intercomunicación entre complejos hospitalarios con el objetivo de facilitar la cooperación diagnóstica y la mejora del estado de los pacientes.

Este tipo de sistemas de Telemedicina pueden emplearse en multitud de casos, diseñando las partes específicas para tratar cada patología concreta. El sistema

desarrollado incluyó un módulo específico diseñado para la supervisión de pacientes con traumatismo craneoencefálico (TCE) [Dafonte 2002]. Este tipo de traumatismos suponen un daño mecánico en las neuronas por impacto directo o por aceleración-desaceleración de las estructuras intracraneales y un daño secundario debido a: comprensión de arterias cerebrales; hemorragia subaracnoidea; hipertensión craneal; y factores extracraneales como hipoxia, hipotensión o anemia. Los objetivos para el tratamiento del TCE están dirigidos a prevenir lesiones adicionales por alteraciones secundarias (hipoxia, hipertensión arterial, presión intracraneal elevada, infección, etc.) y proporcionar un medio adecuado para las neuronas potencialmente recuperables.

El tratamiento de pacientes con TCE es muy complejo e involucra a todos los sistemas: hemodinámico, respiratorio, neurológico y renal. En cada uno de estos sistemas habrá que monitorizar un importante número de variables: hemodinámicas (presión arterial, frecuencia cardiaca, presión venosa central, hematocrito, presión capilar pulmonar, presión de perfusión cerebral), respiratorias (pCO2, pO2, SaO2), renales (diuresis, electrolitos), neurológicas (presión intracraneal, SjvO2, medidas de test tipo Glasgow), temperatura, imágenes médicas (tomografías, resonancias, radiografías), etc. La gran cantidad y variedad de tipos de datos utilizados (variables de control, valores numéricos y alfanuméricos, estimaciones, históricos/tendencias, gráficos, etc.) hace que sea necesario implementar los mecanismos adecuados para el acceso a los datos y su posterior procesamiento. Mediante técnicas de fusión de datos se tratará de obtener la información crítica que será visualizada en la interfaz gráfica en tiempo real.

Una de las componentes principales del módulo de supervisión de pacientes con TCE es su sistema de conocimiento basado en reglas. Será este sistema el encargado de interpretar la información para realizar una monitorización inteligente y detectar las alteraciones críticas que puedan afectar a la evolución clínica del paciente, incluyendo la aportación de recomendaciones de terapias adecuadas [Dafonte 2007]. Este tipo de procesos patológicos reúnen características que los hacen muy apropiados para el uso de técnicas de IA: sistemas complejos, fuerte dependencia temporal, adquisición y clasificación de

datos de múltiples fuentes, existencia de correlaciones entre variables, análisis dependiente del contexto fisiológico y de la evolución del paciente, componentes basadas en la experiencia del personal clínico, etc.

3.3.3. Interfaces de visualización

Uno de los desafíos a la hora de diseñar un sistema de Telesupervisión clínica es conseguir una interfaz de visualización y consulta adaptada a las necesidades del personal clínico, que sea ergonómica y fácil de utilizar con una curva de aprendizaje reducida. En nuestro sistema se han implementado distintos tipos de interfaces para los subsistemas locales y remotos, centrados en la monitorización en tiempo real de las señales recibidas de los dispositivos médicos y en la visualización de mensajes ante situaciones de alarma. La apariencia y el funcionamiento general de los sistemas local y remoto serán similares para facilitar su uso por parte del personal clínico.

Figura 5. Interfaz de monitorización local del paciente

Desde los equipos cliente se podrán configurar todas las opciones y parámetros requeridos para cada uno de los dispositivos conectados al paciente. Esta información afectará a las opciones de visualización gráfica, permitiendo

establecer los canales de comunicación y los umbrales de alarma correspondientes a cada dispositivo.

Desde la interfaz del equipo servidor se podrá realizar un control remoto de las funciones de cada uno de los clientes conectados al sistema. Así, se podrá monitorizar de forma remota el estado de cada paciente de la UCI. Posteriormente, se desarrolló un módulo con una interfaz gráfica de visualización 3D interactiva para la representación del paciente y sus órganos internos. Esta interfaz permite seleccionar y mostrar la información correspondiente a cada paciente (datos personales, evolución de las variables fisiopatológicas, historiales clínicos, etc.) e interactuar con el modelo 3D y los objetos 3D interactivos que lo componen. También permite destacar de forma visual la aparición de alarmas y los órganos relacionados con ellas, facilitando su detección para reducir los tiempos de respuesta.

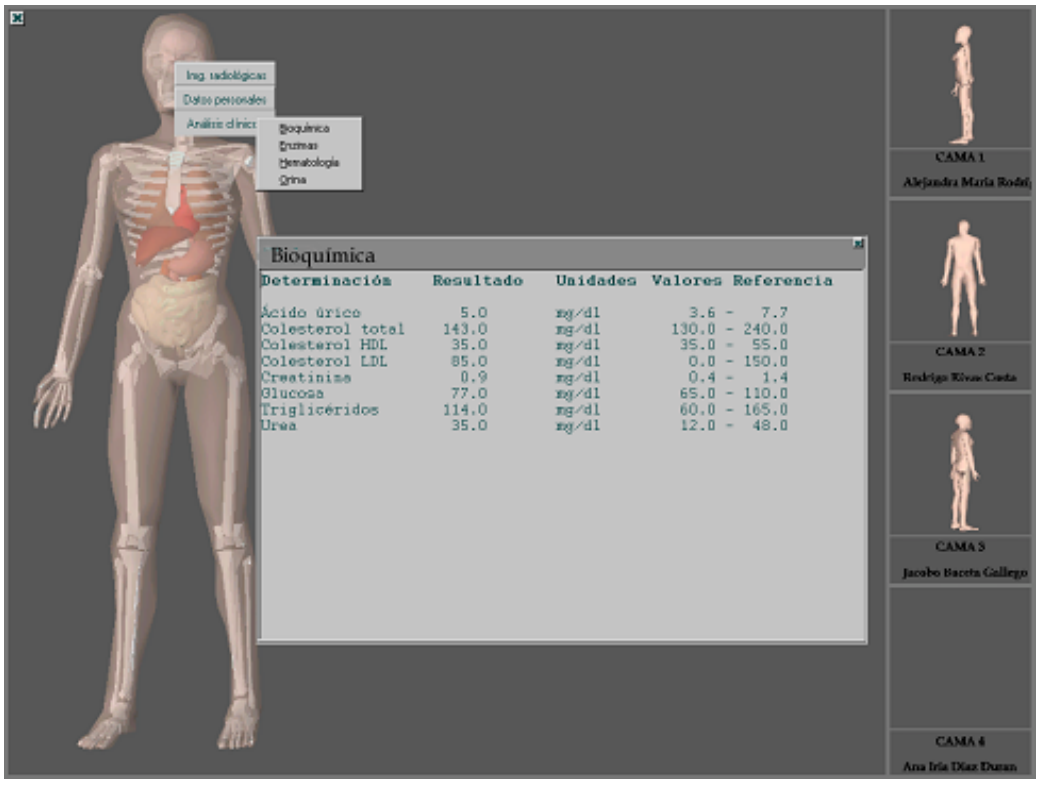


Figura 6. Interfaz de visualización 3D del sistema de Telemedicina

3.3.4. Módulo de visualización 3D

La implementación de un módulo de visualización 3D en este proyecto de Telemedicina trata de aportar una nueva forma de supervisar la evolución de los

pacientes internados en la UCI del centro hospitalario. Esta interfaz se centra en la representación virtual mediante objetos 3D interactivos, creando una nueva capa para la interoperabilidad entre el sistema y el personal clínico.

El desarrollo de este módulo utiliza la especificación OpenGL para crear y gestionar el escenario y los objetos tridimensionales [OpenGL 2015]. Los objetos poseen capacidades interactivas permitiendo acceder a información relativa a un determinado órgano o actuando como indicador ante situaciones de alarma, mediante cambios de tamaño y color. De este modo, se fusiona toda la información recogida por el sistema de monitorización para mostrar una interfaz simplificada y fácil de interpretar.

Desde esta interfaz también es posible acceder a la información detallada de los parámetros bajo monitorización de un paciente o acceder a otro tipo de información proporcionada por el SIH. Así, el personal sanitario puede interactuar con el entorno de visualización para obtener los datos personales de cada uno de los pacientes ingresados en la unidad crítica, consultar la información de análisis de laboratorio que consten en su historial clínico, visualizar imágenes médicas, etc.

La notificación de las alarmas médicas se realiza por medio de cambios visuales en los objetos 3D de la representación del paciente afectado, unido a avisos sonoros. Sin embargo, también se emplean mensajes en pantalla utilizando una simbología intuitiva para indicar el problema detectado. Al pulsar sobre este mensaje se puede obtener una información detallada sobre su origen y parámetros involucrados en su disparo.

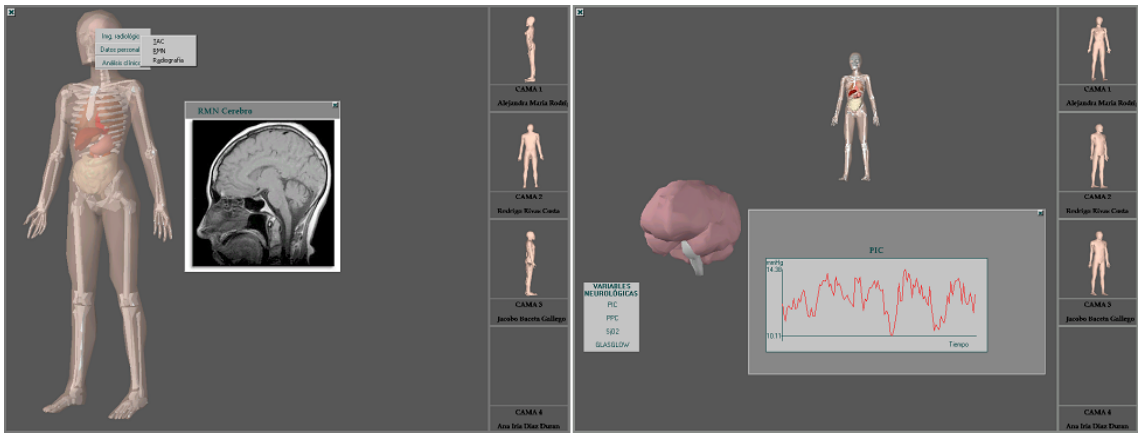


Figura 7. Detalles informativos en la interfaz de visualización 3D del sistema

4

Gestión inteligente de procesos

4.1. INTRODUCCIÓN

El presente capítulo plantea una herramienta que intenta mejorar la gestión del ancho de banda disponible en sistemas conectados mediante redes de datos y que precisan de una comunicación en tiempo real. El protocolo de red TCP/IP no proporciona un mecanismo que permita realizar un reparto del ancho de banda existente mediante la asignación de prioridades lo cual obliga a realizar dicho control a nivel de aplicación.

Tal y como se ha indicado en capítulos anteriores, los sistemas informáticos dedicados a la gestión y supervisión en tiempo real van a requerir que los tiempos de procesamiento, desde que los datos son adquiridos hasta que son adecuadamente visualizados, sean lo más reducidos posible. En ámbitos de aplicación críticos, como los sistemas de telemedicina, se hace imprescindible disponer de mecanismos que permitan mejorar el funcionamiento global de los equipamientos informáticos en los que se están ejecutando los sistemas de supervisión. Estos mecanismos pasan por controlar tanto el estado de los distintos elementos, que repercuten en el rendimiento del equipo, como el estado de las comunicaciones a través de redes de datos.

Realizar un control sobre el estado de los principales componentes de un equipo informático (CPU, memoria, espacio de almacenamiento, etc.) permite disponer de

información en tiempo real de su estado y por lo tanto permite controlar la gestión de sus recursos y de los procesos, en ejecución o pendientes de ser ejecutados, para adaptarse y poder priorizar aquellos procesos que tengan una mayor criticidad.

Para realizar este control y gestión de los recursos se dotará a los sistemas de mecanismos de control y regulación de recursos disponibles y de supervisión del estado de la red de datos.

En nuestro caso, se ha desarrollado un aplicativo multiplataforma basado en agentes inteligentes para la gestión y control del ancho de banda dentro de un sistema de Telemedicina en unidades críticas. Esta herramienta de control se caracteriza principalmente por seguir especificaciones estándar para la implementación de los agentes y por utilizar un sistema de reglas de conocimiento para la toma de decisiones. Así, se dotará a los agentes de inteligencia para gestionar de forma autónoma las tareas o procesos que tengan que ejecutarse y el agente que se encargará de ello. Además, cada agente dispondrá de una interfaz gráfica para su gestión, incluyendo gráficos 3D interactivos para visualizar e interactuar con la red de agentes.

4.2. AGENTES INTELIGENTES

En la bibliografía existen múltiples definiciones sobre los **agentes software** o **agentes inteligentes**, así se pueden definir de forma amplia como: "*un software que utiliza técnicas de inteligencia artificial para asistir al usuario humano para una tarea específica*" [Maes 1994]. Una definición más adecuada del término agente es la que lo define como: "*un sistema computacional situado en un entorno, y que es capaz de realizar acciones de forma autónoma en dicho entorno para cumplir los objetivos con los que fue diseñado*" [Wooldridge 2002]. Según esta definición se puede considerar que los agentes inteligentes son aquellos que cumplen con las siguientes propiedades:

- **Autonomía**. Los agentes operan sin la intervención directa de usuarios u otros agentes y disponen de algún tipo de control sobre sus acciones y su estado interno.

- **Sociabilidad (capacidad social)**. Los agentes pueden interactuar con otros agentes o con usuarios por medio de algún tipo de lenguaje de comunicación de agentes.
- **Reactividad**. Los agentes perciben su entorno (por ejemplo: una red de agentes, Internet, una interfaz gráfica de usuario, etc.) y reaccionan en un intervalo de tiempo adecuado ante los cambios que se producen.
- **Proactividad (iniciativa)**. Además de reaccionar ante el entorno, los agentes son capaces de tomar la iniciativa, exhibiendo un comportamiento dirigido por objetivos.

Los agentes inteligentes se pueden agrupar en cinco clases en función de su nivel de percepción y capacidad [Russell 2010]:

1. **Agentes reactivos simples**. Sólo actúan en base a la percepción actual, ignorando la percepción anterior o histórica. Estos agentes se basan en la regla *condición-acción*.
2. **Agentes reactivos basados en modelos**. Pueden manejar parcialmente el entorno observable y su estado se guarda en una estructura dentro del agente. Al mantener algún tipo de modelo interno dependiente de la percepción histórica puede manejar algunos aspectos no observados en el estado actual. La percepción histórica y su impacto en la acción puede estar determinado por este modelo interno.
3. **Agentes basados en objetivos**. Mejoran las capacidades de los agentes basados en modelos utilizando una información objetivo para describir situaciones deseables. Esto permite que el agente pueda seleccionar entre múltiples posibilidades aquella opción que le lleve a alcanzar el estado objetivo.
4. **Agentes basados en utilidad**. Los agentes basados en objetivos sólo distinguen entre los estados que son objetivo y los que no lo son. En los agentes basados en utilidad se define una medida del nivel de deseabilidad de un estado. Esta medida se puede obtener mediante una función de utilidad encargada de relacionar un estado con una medida de la utilidad de ese estado.

5. **Agentes que aprenden**. El aprendizaje permite a los agentes operar en entornos desconocidos consiguiendo resultados competentes y ampliando su conocimiento inicial. Utiliza un mecanismo de retroalimentación sobre la actuación del agente para determinar las modificaciones que se deben realizar para conseguir mejores resultados en el futuro.

Los entornos en los que se ejecutan los sistemas de agentes pueden ser de diversos tipos [Russell 2010]:

- **Accesibles o inaccesibles**: un entorno es accesible si el agente puede disponer de información completa, precisa y actualizada acerca del estado del mismo.
- **Deterministas o no deterministas**: un entorno es determinista si cada acción puede producir un único resultado, que se puede conocer de antemano.
- **Estáticos o dinámicos**: un entorno es estático si los únicos cambios que sufre son debidos a las acciones del agente. En los entornos dinámicos actúan otros procesos, que producen cambios que no están bajo el control del agente.
- **Discretos o continuos**: un entorno es discreto si hay un número finito fijo de posibles acciones y percepciones.

Atendiendo a su capacidad de colaboración entre agentes, dentro de las tecnologías de agentes, surgen los denominados **Sistemas Multiagente** (SMA) [Ferber 1999]. Un sistema multiagente se puede definir como un sistema compuesto por múltiples agentes inteligentes interactuando entre ellos para realizar una determinada tarea dentro de su entorno. Por lo tanto, estos sistemas disponen de diferentes agentes interconectados capaces de colaborar para conseguir la realización de tareas distribuidas. Los sistemas multiagente son apropiados para resolver problemas complejos y que no pueden ser abordados por sistemas de agentes independientes. Ejemplos de este tipo serían: problemas que requieren la interconexión e interoperación de sistemas autónomos, problemas cuya solución depende de la colaboración de conocimiento experto distribuido,

problemas en los cuales se depende de la fusión de información proveniente de fuentes distribuidas, etc.

4.2.1. FIPA

La fundación FIPA (*Foundation for Intelligent Physical Agents*) es una organización creada con el propósito de promover el desarrollo de estándares para agentes y sistemas basados en agentes heterogéneos y su interoperabilidad con otras tecnologías [FIPA 2002][Poslad 2007]. Actualmente, FIPA forma parte del comité de estándares de la *IEEE Computer Society*.

Mediante las especificaciones FIPA se pretende la estandarización de la sintaxis de los documentos, proporcionando formas estándares de comunicación entre agentes preservando su significado. Para lograr dicho objetivo incluye varios tipos de especificaciones: arquitecturas para soportar la comunicación entre agentes, lenguajes de comunicación y lenguajes de contenido para expresar los mensajes, protocolos de interacción que amplían el ámbito de simples mensajes a transacciones completas.

4.2.1.1. La arquitectura abstracta FIPA

El propósito de la arquitectura abstracta FIPA consiste en permitir el intercambio de mensajes semánticamente significativos entre agentes que pueden estar utilizando distintas capas de transporte, diferentes lenguajes de comunicación de agentes y diferentes lenguajes de contenido. Esta arquitectura incluye:

- Un modelo de servicios y de descubrimiento de servicios disponible para los agentes y para otros servicios.
- Interoperabilidad para el transporte de mensajes.
- Soporte para varias formas de representación ACL (*Agent Communication Language*).
- Soporte para varias formas de lenguajes de contenido.
- Soporte para múltiples representaciones de servicios de directorio.

4.2.1.2. Modelo de referencia de gestión de agentes

El modelo de referencia de gestión de agentes proporciona un marco en el que los agentes existen y operan. Las entidades que constituyen este modelo son servicios lógicos y no implican una configuración física.

En el modelo de referencia distinguimos los siguientes componentes lógicos:

- **Agente**. Proceso computacional que implementa la funcionalidad comunicativa de una aplicación. El agente es el actor fundamental de una plataforma de agentes (AP-*Agent Platform*) y combina uno o más servicios, publicados en una descripción de servicios, en un modelo de ejecución unificado e integrado. Un agente debe tener un propietario y un identificador de agente (AID-*Agent IDentifier*), que permita distinguirlo sin ambigüedad dentro del universo de agentes (AU-*Agent Universe*). Un agente puede registrarse en varias direcciones de transporte, desde las que se podrá contactar con él. Los agentes se comunican mediante el lenguaje de comunicación de agentes (ACL-*Agent Communication Language*).
- **Directorio** (DF-*Directory Facilitator*). Componente opcional que proporciona un servicio de páginas amarillas a los agentes. Permite registrar y buscar servicios ofrecidos por los agentes.
- **Sistema de Gestión de Agentes** (AMS-*Agent Management System*). Componente obligatorio y único para cada agente que se encarga de supervisar el acceso y uso de la plataforma (AP). Mantiene un directorio de identificadores (AID) que incluye direcciones de transporte de los agentes de la AP. Cada agente debe registrarse en un AMS para obtener un AID válido.
- **Sistema de Transporte de Mensajes** (MTS-*Message Transport System*). Sistema que permite la comunicación entre agentes de distintas plataformas (AP).
- **Plataforma de Agentes** (AP-*Agent Platform*). Proporciona una infraestructura física para los agentes. Está formado por las máquinas, sistema operativo, software de soporte para los agentes, componentes de gestión de agentes FIPA (DF, AMS, y MTS) y agentes.

- **Software**. Se refiere al código que no forma parte de los agentes pero que puede ser accedido por ellos.

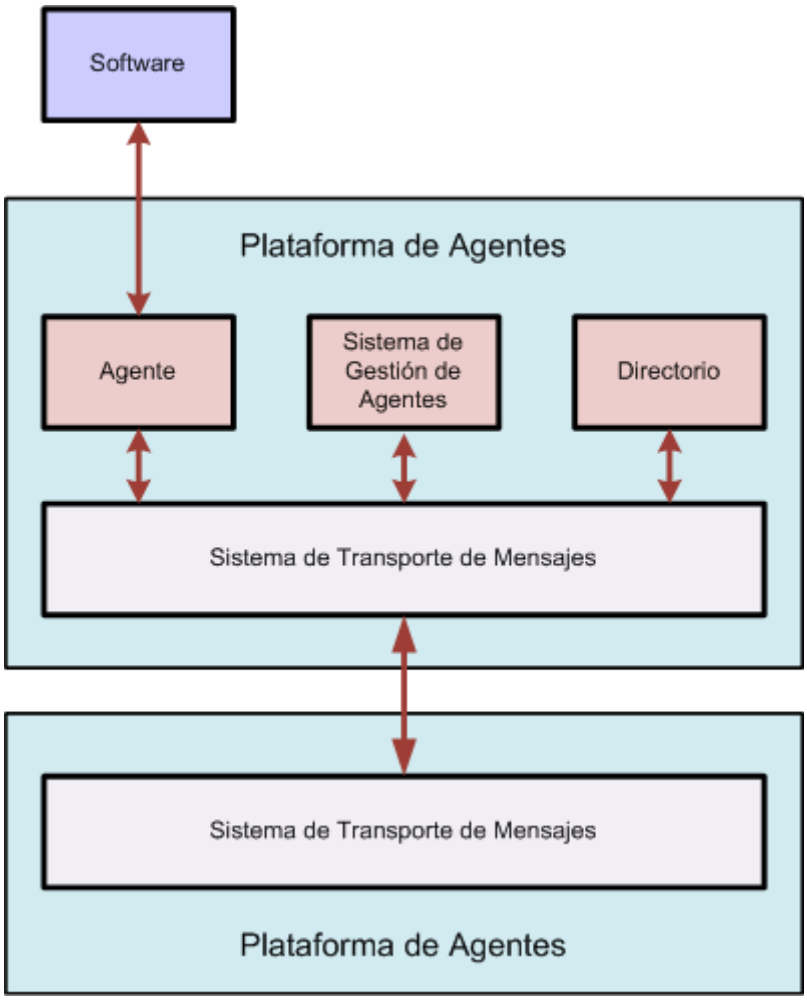


Figura 8. Modelo de referencia de gestión de agentes FIPA

4.2.2. FIPA-OS

FIPA-OS (*FIPA-Open Source*) es un conjunto de herramientas orientado a componentes que facilita el desarrollo de agentes conforme a las especificaciones FIPA. Algunos de estos componentes son obligatorios para la ejecución de los agentes, otros son componentes con implementaciones conmutables, y también dispone de componentes opcionales [FIPA-OS 2003][Poslad 2000].

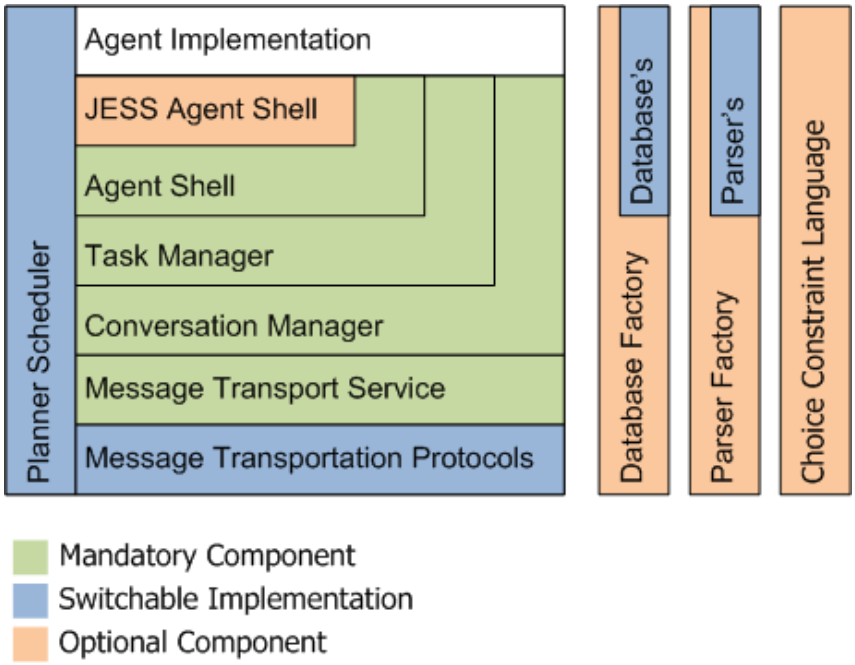


Figura 9. Componentes de la arquitectura de FIPA-OS

Las tareas FIPA-OS se caracterizan por:

- Normalmente encapsulan alguna funcionalidad que forma una unidad lógica de trabajo (por ejemplo: realizar una negociación con otro agente).
- Pueden ser descompuestas en subtareas.
- Están basadas en un procesamiento dirigido por eventos.
- Pueden estar asociadas a varias conversaciones del agente, asegurando que los cambios en estas se propaguen a las tareas.
- Cualquier agente que utilice tareas debe tener una tarea específica para tratar los mensajes recibidos que no formen parte de una conversación que esté siendo realizada por otra tarea.

Los agentes implementados con FIPA-OS no tienen que utilizar obligatoriamente tareas pero su uso proporciona una serie de ventajas:

- Facilitan el desarrollo de nuevos agentes.
- Permiten acceder a numerosas características basadas en tareas del API de FIPA-OS.
- Facilita la estructuración del código de forma lógica.

- Elimina la necesidad de estructuras "*if…else*" complejas, ya que siempre se sabe qué mensajes pueden llegar a una tarea.
- Proporciona a los agentes una estructura modular, permitiendo añadir nuevas funcionalidades al agente sin requerir grandes cambios en el código.
- Sigue un patrón de diseño reconocido.
- Permite compartir código a los desarrolladores de agentes.
- Garantiza la compatibilidad con nuevas versiones de FIPA-OS.

4.3. JESS

El motor de inferencia JESS (*Java Expert System Shell*) se define como un motor de reglas y entorno de *scripting* para la plataforma Java [JESS 2013][Friedman-Hill 2003]. Está implementado totalmente en lenguaje Java y permite el desarrollo de sistemas expertos basados en reglas capaces de interoperar mediante código Java. Así, es posible llamar a funciones Java desde Jess y utilizar Java para extender al propio motor de reglas. También, es posible invocar a Jess desde una aplicación escrita en lenguaje Java para dotarla de un sistema de conocimiento mediante reglas declarativas.

Distinguimos dos formas de utilizar Jess:

- Motor de reglas. Básicamente es un tipo de programa encargado de aplicar de forma muy eficiente reglas a datos. Un programa basado en reglas puede llegar a tener una gran cantidad de reglas definidas, y mediante Jess se estarán aplicando continuamente a los datos, que se almacenan en una base de conocimiento. Normalmente las reglas definidas tratan de plasmar el conocimiento de un experto humano en un dominio concreto, y la base de conocimiento representará el estado de una situación determinada. En este caso, se dice que es un sistema experto. Los sistemas expertos se utilizan en gran variedad de dominios. Entre sus nuevas aplicaciones está la parte de razonamiento de los agentes inteligentes.
- Lenguaje de programación (entorno de scripting). Jess dispone de su propio lenguaje de reglas de propósito general. Además, utilizando este lenguaje de

programación se puede acceder directamente a todas las clases y librerías del lenguaje Java, crear objetos, realizar llamadas a métodos, implementar interfaces, etc.

El motor de reglas de Jess utiliza una versión mejorada del algoritmo Rete [Forgy 1982] para realizar el emparejamiento de las reglas con la base de conocimiento. El algoritmo Rete minimiza el tiempo de ejecución a costa de un mayor uso de memoria pero se ha demostrado como un mecanismo muy eficiente para resolver problemas de emparejamientos muchos-a-muchos.

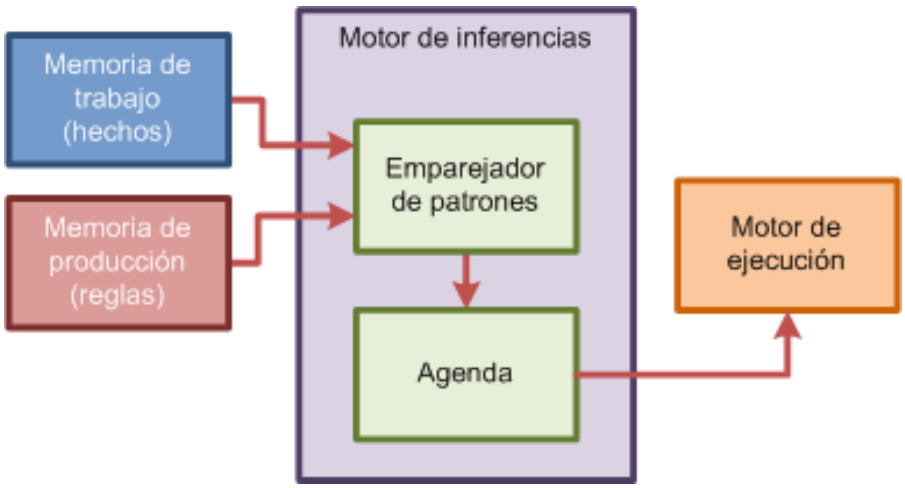


Figura 10. Diagrama de la arquitectura de JESS

4.3.1. Lenguaje de reglas

Las principales características del lenguaje de reglas de Jess son las siguientes:

- Símbolos. Equivalen a los identificadores de otros lenguajes de programación y pueden contener letras, dígitos, y determinados caracteres. Los símbolos de Jess distinguen caracteres en mayúsculas y en minúsculas.
- Números. Siguen una sintaxis similar al lenguaje Java y utilizan funciones de Java para comprobar si contienen valores válidos.
- Cadenas. Representan cadenas de caracteres y se definen entre comillas dobles.
- Listas. Una lista está formada por un conjunto de cero o más símbolos, números, cadenas, u otras listas, encerrados entre paréntesis. El primer elemento se denomina cabeza de la lista.

- Funciones. En el lenguaje de reglas de Jess, una función es una lista. Se emplea una notación prefija; una lista cuya cabeza es un símbolo que es el nombre de una función existente, se considera una llamada a dicha función. Por ejemplo, la función suma aplicada a los sumados 2 y 3 se escribe como:

  (+ 2 3)

- Variables. Una variable de programación es un símbolo que empieza con el carácter "?". Este carácter inicial también es parte del nombre de la variable. Una variable puede hacer referencia a un símbolo, un número, una cadena, o una lista. Mediante la función "*bind*" se le puede asignar un valor a una variable. Por ejemplo el siguiente código asignará el valor "123" a la variable "?a":

  (bind ?a 123)

- Hechos. La base de conocimiento mantiene una colección de elementos de conocimiento a los que denomina hechos (*facts*). Se distinguen tres tipos básicos de hechos en Jess:

  - Hechos ordenados (*ordered facts*). Son simplemente listas en las que el primer elemento actúa como categoría organizadora del hecho. Un ejemplo:

    (computer "Intel" "Pentium 4" 2000)

  - Hechos desordenados (*unordered facts*). Son hechos que pueden tener campos a los que se denomina "slots". Este tipo de hechos requieren de la definición previa de una plantilla (mediante el constructor "*deftemplate*"). Un ejemplo:

    (deftemplate computer "A specific computer." (slot make) (slot model) (slot year (type INTEGER)))

    (computer (make "Intel") (model " Pentium 4") (year 2000))

  - Hechos ocultos (*shadow facts*). Son hechos desordenados que permiten trabajar con objetos Java Beans. Estos hechos sirven de puente con objetos Java y al igual que los hechos desordenados requieren de la definición de una plantilla.

- Reglas. En Jess una regla es similar a una sentencia "*if...then*" de un lenguaje de programación, pero no se utiliza del mismo modo. Mientras que las declaraciones "*if...then*" se ejecutan en un momento y en un orden específicos, de acuerdo con la codificación realizada por el programador, las reglas en Jess se ejecutan siempre que sus partes "*if*" se activan. Para definir una regla se emplea el constructor "*defrule*". El siguiente ejemplo muestra una regla que se activa si hay algún hecho en la base de conocimiento que coincida con el patrón indicado, generando un mensaje de salida:

```
(defrule computer_model
        (computer ?model)
        =>
        (printout t "Model: " ?model))
```

4.4. JAVA 3D

El API Java 3D facilita la creación de aplicaciones Java que utilicen gráficos tridimensionales [Java3D 2008]. Este API es válido para construir aplicaciones independientes y *applets* 3D ejecutadas en un navegador de Internet. Mediante el API Java 3D disponemos de una jerarquía de clases Java que actúan de interfaz con un sistema de renderización de gráficos tridimensionales y de sonido. Proporciona al programador los constructores de alto nivel que le permiten crear y manipular objetos geométricos en 3D, y los que permiten la construcción de estructuras para el renderizado de esos objetos. Los objetos creados residirán en un universo virtual que será renderizado de forma eficiente por este software de desarrollo.

Un programa que utilice el API Java 3D crea instancias de los objetos y los coloca en una estructura de datos denominada grafo de escena. Dicha estructura está compuesta por objetos 3D organizados en forma de árbol, y será la encargada de especificar el contenido del universo virtual y cómo se realizará el proceso de renderizado [Selman 2002].

El motor de renderizado de Java 3D utiliza los hilos de ejecución (*threads*) de Java para realizar un procesamiento en paralelo y conseguir optimizar todo el proceso. También se encarga de realizar automáticamente optimizaciones que mejoran el

rendimiento del renderizado. Además, simplifica el trabajo del programador ya que los detalles del renderizado del universo virtual y de sus objetos son gestionados de forma automática por el motor de Java 3D.

4.4.1. Funcionamiento

El grafo de escena permite crear un universo virtual en Java 3D. Para ello, define la geometría, sonido, luces, localización, orientación, y aspecto de los objetos visuales y de audio.

Un grafo se puede definir como una estructura de datos compuesta por nodos y arcos. Un nodo es un elemento de datos y un arco sirve para establecer una relación entre nodos. Los nodos en el grafo de escena son instancias de las clases Java 3D. Los arcos representan los dos tipos de relaciones entre instancias Java 3D: padre-hijo, referencia. En el primer caso tendremos: un nodo de grupo que está formado por un padre y cualquier número de hijos, y un nodo hoja que tiene un padre y ningún hijo. En el caso de la relación por referencia, tendremos una asociación entre un objeto *NodeComponent* a un *Node* del grafo de escena. Los objetos *NodeComponent* definen los atributos de geometría y aspecto que se emplean en la renderización de los objetos visuales.

Un grafo de escena se construye a partir de objetos *Node*, entre los que se establecen relaciones padre-hijo. Hay un nodo raíz desde el que es posible acceder a otros nodos siguiendo los arcos pero no se pueden formar ciclos. Un grafo de escena está formado por los árboles asociados a los objetos *Locale*. Los objetos *NodeComponent* y los arcos no forman parte del árbol del grafo de escena.

El camino desde la raíz a un nodo hoja se denomina *path* del grafo de escena. Éste especifica completamente la información de estado del nodo hoja. La información de estado incluye la localización, orientación, y tamaño de un objeto visual. Por tanto, los atributos visuales de cada objeto visual sólo dependen de su *path* del grafo de escena. El motor de renderizado de Java 3D aprovecha esto, y renderiza las hojas en el orden que considera más eficiente. El programador normalmente no controla dicho orden.

Las representaciones gráficas de un grafo de escena pueden servir como herramienta de diseño y de documentación. Para construirlas se emplean unos

símbolos gráficos estándar, tal y como se muestran en la siguiente figura. Cada uno de los símbolos mostrados en la columna izquierda de la figura representa un objeto de un grafo de escena. Los símbolos de la columna derecha representan referencias entre objetos.

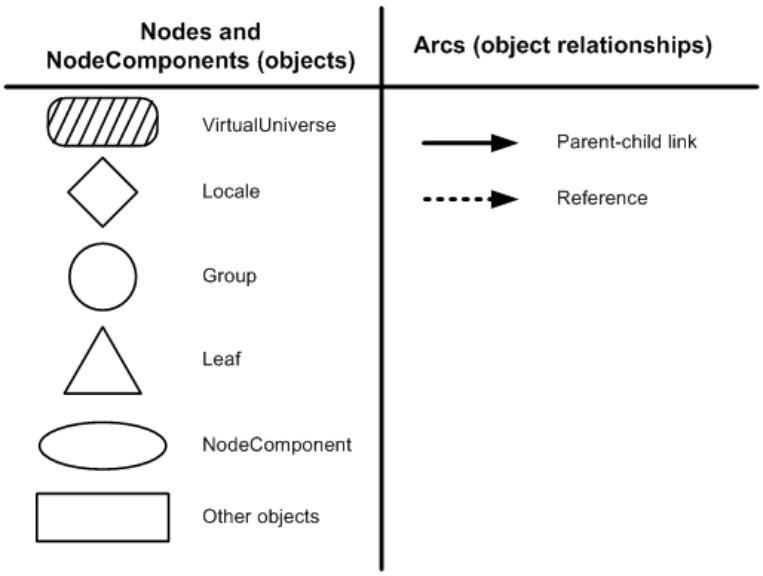


Figura 11. Símbolos para representar los objetos de un grafo de escena

En cada grafo de escena tendremos un único objeto *VirtualUniverse*. Éste tiene una lista de objetos *Locale*, cada uno de los cuales proporciona un punto de referencia en el universo virtual. Es posible que existan varios objetos *VirtualUniverse* en un programa Java 3D, pero se recomienda que haya sólo uno.

La mayoría de los programas emplean un único objeto *Locale*. Cada *Locale* puede servir como raíz de múltiples subgrafos del grafo de escena.

Un objeto *BranchGroup* es la raíz de un subgrafo. Hay dos tipos diferentes de subgrafos de escena: de vista, y de contenido. El subgrafo de contenido especifica la geometría, apariencia, comportamiento, localización, sonido, y luces. El subgrafo de vista especifica los parámetros de visualización, tales como la localización y dirección de la vista.

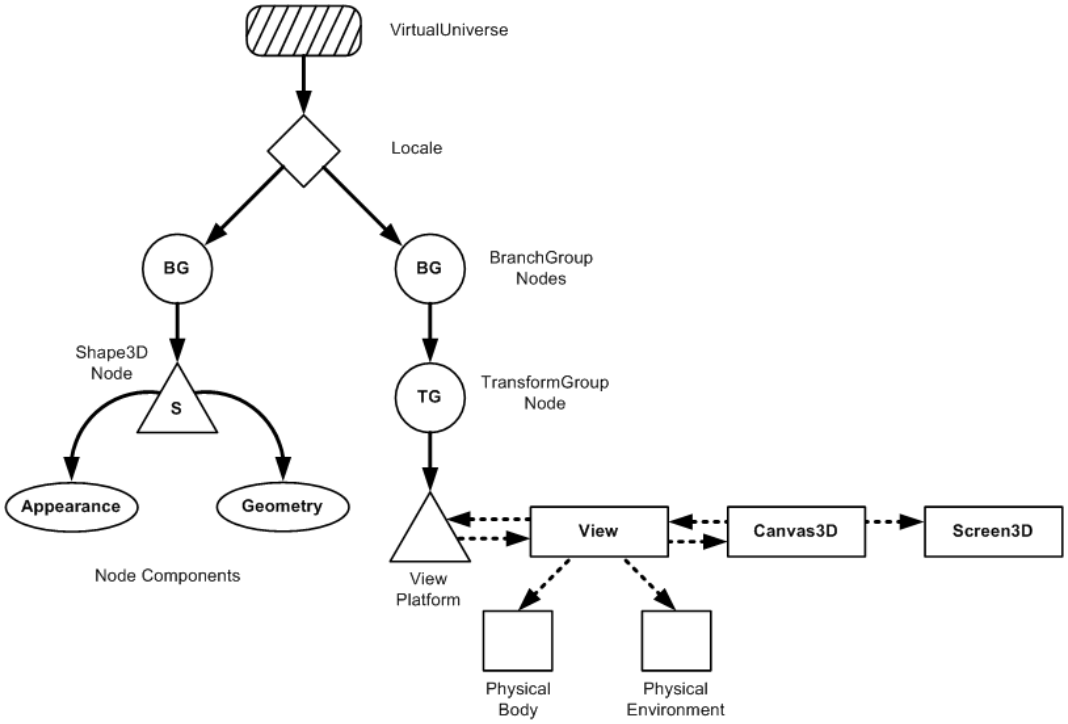


Figura 12. Ejemplo de grafo de escena con Java 3D

4.5. OBJETIVOS

En el capítulo anterior, se han introducido los conceptos y aplicaciones de la Telemedicina. También se ha presentado brevemente el sistema de Telemedicina desarrollado por nuestro grupo de investigación para la monitorización de pacientes internados en una UCI de un centro hospitalario. En este tipo de sistemas es importante conseguir un buen aprovechamiento de los recursos y una gestión adecuada del ancho de banda disponible. Hay que tener en cuenta que en estos sistemas se están actualizando datos de forma continua, como los parámetros de monitorización en tiempo real del paciente, y al mismo tiempo se puede solicitar la ejecución de tareas que ayuden en la mejora de la atención clínica, como la consulta del historial clínico o el envío de una imagen radiológica del paciente. Estas peticiones de información no van a tener siempre la misma prioridad y además no deben interferir con actividades críticas como la monitorización del estado del paciente.

El objetivo principal de esta sección es describir el desarrollo realizado para dotar a nuestro sistema de Telemedicina en UCIs de un mecanismo de control para el reparto inteligente de tareas y recursos entre los dispositivos conectados a través

de la red del sistema de Telemedicina. Para realizar esta labor será necesario disponer de ciertas aplicaciones instaladas en los equipos para poder recoger en tiempo real la información sobre su estado, comunicarla al resto de equipos, y decidir en consecuencia cuál es el equipo más óptimo para realizar una determinada tarea. Finalmente, se ha optado por utilizar las tecnologías de agentes inteligentes para implementar este tipo de aplicaciones interconectadas debido a que se han mostrado muy adecuadas para este tipo de funciones.

Las tareas o procesos implicados pueden ser de múltiples tipos: adquisición y tratamiento de señales de dispositivos clínicos, procesado de imágenes biomédicas, generación de informes sobre el estado del paciente, etc. Sus principales características vienen definidas por las necesidades de recursos que precisan para su realización, tanto hardware como software, y por su criticidad temporal.

La amplia variedad de equipos disponibles en la red de la UCI impone la necesidad de desarrollar un módulo multiplataforma, de tal modo que pueda ser instalado en la mayor cantidad posible de equipos. Por este motivo se decidió utilizar la plataforma Java para su implementación.

Así pues podríamos definir como objetivos de este módulo del sistema los siguientes:

- Desarrollo de un sistema de control del ancho de banda y de los recursos disponibles sobre una red de equipos multiplataforma conectados a través de la red hospitalaria. Concretamente, al tratarse de un sistema destinado a UCIs habrá que tener en cuenta importantes restricciones temporales y por lo tanto será necesario que el sistema de control intente optimizar los tiempos de respuesta.
- Utilización de estándares de tecnologías de agentes inteligentes como base para el desarrollo. De este modo, el sistema de control podrá adaptarse a los equipos de la red en funcionamiento y mejorar la distribución de los procesos de forma transparente para el usuario final.
- El gestor del sistema podrá realizar un control y supervisión en tiempo real de la red de agentes disponibles en cada momento. Entre la información

proporcionada por cada agente del sistema se incluirá el estado del equipo en el que reside y los procesos que tenga en su lista de tareas.

- Se realizará una gestión completa de las tareas: pudiendo definir prioridades, órdenes de cancelación, órdenes de redistribución, tratamiento de mensajes entre agentes, etc.
- Incluir un mecanismo de ayuda a la toma de decisiones mediante un motor de reglas de conocimiento encargado de gestionar la distribución de tareas entre los agentes de la red.
- Se diseñará una interfaz gráfica avanzada para la gestión de la red de agentes que facilite la visualización y control en tiempo real. Para este diseño se probarán técnicas de visualización 3D.

4.6. DESARROLLO

Siguiendo los objetivos planteados con este proyecto, se desarrolló un módulo para la gestión de tareas distribuidas para un sistema de Telemedicina. Mediante este módulo se controlará la distribución de procesos entre los dispositivos en función del estado de cada equipo y del estado de la red (ancho de banda disponible para su transmisión). También se definirán valores de prioridad de las tareas para reflejar la criticidad o importancia de realizar un determinado proceso en un período de tiempo concreto.

La base para el desarrollo de este módulo será la utilización de tecnologías de agentes inteligentes. Los agentes se encargarán de la recopilación de información sobre el estado de cada equipo y su comunicación al resto de equipos para poder decidir dónde realizar una determinada tarea. Estos agentes se instalarán en los equipos de la red hospitalaria por lo que será necesario que pueda adaptarse al mayor espectro posible de sistemas operativos existentes. Por este motivo, se optó por realizar un módulo multiplataforma ejecutado bajo la máquina virtual Java.

Los agentes inteligentes seguirán una de las especificaciones más utilizadas para este tipo de aplicaciones, FIPA. Con el objetivo de facilitar la implementación de los agentes, garantizando que cumplen con la especificación FIPA, se ha utilizado el software de desarrollo FIPA-OS.

La incorporación de un motor de reglas de conocimiento en los agentes desarrollados, permite dotar a los agentes de capacidades de inteligencia. Estas capacidades pueden ser ampliadas/mejoradas para adaptarse a las necesidades de la aplicación de Telemedicina mediante la implementación de nuevas reglas de conocimiento. De esta forma, se podrá mejorar la toma de decisiones para la distribución de las tareas entre los agentes de la red, consiguiendo optimizar el ancho de banda y los recursos disponibles en cada momento. Para la implementación del sistema de decisión de los agentes se utilizó el motor de inferencia JESS.

Además, con este desarrollo se ha tratado de probar nuevas formas de gestionar y representar la información de los agentes inteligentes, incluyendo la información descriptiva proporciona por los parámetros más relevantes de cada agente y añadiendo una interfaz gráfica avanzada para la supervisión del correcto funcionamiento de la red de agentes. Esta interfaz utiliza objetos 3D interactivos para modelar la red de agentes y permite controlar la ejecución de tareas de forma simple. Para la implementación de esta interfaz se ha optado por utilizar el API Java 3D, que permite crear y manejar gráficos tridimensionales bajo la plataforma Java.

En el desarrollo de este proyecto se ha utilizado la metodología destinada a la creación de sistemas basados en el conocimiento CommonKADS (*Common Knowledge Acquisition and Documentation Structuring*). Esta metodología nos permite realizar el análisis y diseño del sistema de gestión de agentes siguiendo una aproximación estructurada [Schreiber 1999].

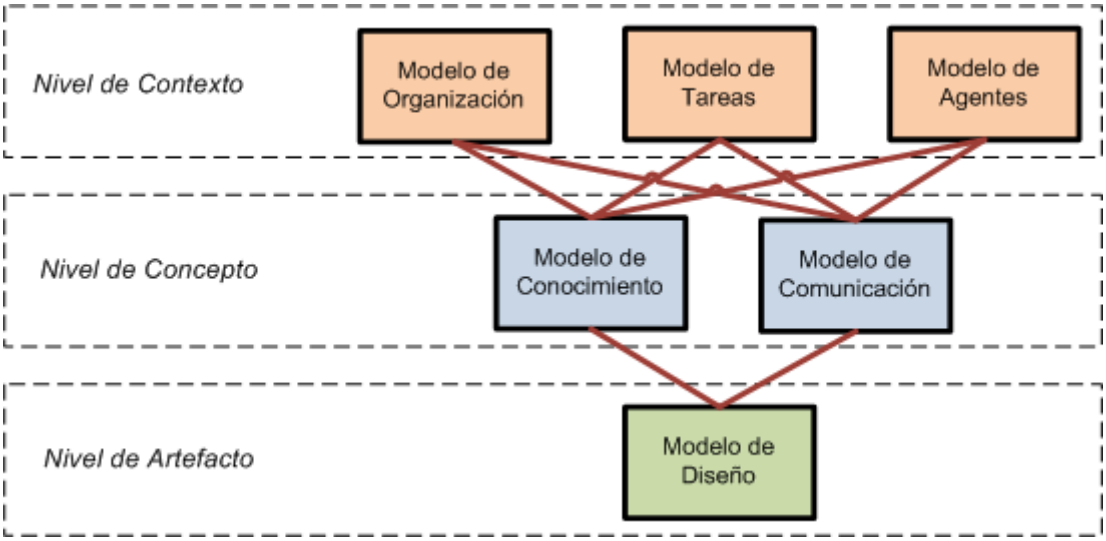


Figura 13. Modelos de la metodología CommonKADS

La metodología CommonKADS, para la fase de análisis de este proyecto, está compuesta por la definición de las tablas correspondientes a cada uno de los siguientes modelos [Schreiber 1999][Akkermans 2004]:

1. **Modelo de organización** (OM-*Organization Model*). En estas tablas se plasma el análisis de la organización sobre la cual se va a introducir el sistema basado en el conocimiento, con la finalidad de descubrir los problemas y oportunidades que plantea y las soluciones que proporciona. En un detalle posterior se definen los procesos necesarios y sus componentes para poder completar las tareas.

2. **Modelo de tareas** (TM-*Task Model*). En este modelo se crean las tablas que definen las partes relevantes del proceso describiendo a nivel general cada una de las tareas que son realizadas en el entorno organizativo.

   En nuestro caso, las tareas que deben ser definidas son:

   - Selección del equipo en el cual se ejecutará la tarea teniendo en cuenta los recursos disponibles en cada equipo y los recursos necesarios para ejecutar la tarea.
   - Solicitud de eliminación de una tarea por cualquier motivo (por ejemplo: tiempo de ejecución elevado, exceso de carga del equipo, prioridad de otras tareas, etc.).

3. **Modelo de agentes** (AM-*Agent Model*). En esta fase se describen los agentes implicados en la realización de las tareas definidas en el modelo anterior. Con esta información, podremos obtener los casos de uso identificados en el desarrollo del sistema de agentes.

   En nuestro caso, los casos de uso que se obtienen para el proyecto son:

   - Ejecutar una tarea de usuario, tanto de forma local como remota.
   - Eliminar una tarea de usuario, tanto de forma local como remota.
   - Obtener agente ejecutor de una tarea.
   - Monitorizar la red de agentes.
   - Comprobar comunicación entre agentes.
   - Obtener información de estado entre agentes.

4. **Modelo de conocimiento** (KM-*Knowledge Model*). Se modela el conocimiento empleado por un agente para realizar una tarea que resuelva un problema. El conocimiento de la aplicación se divide en tres subniveles:

   - Nivel del dominio: conocimiento declarativo sobre el dominio.
   - Nivel de inferencia: biblioteca de estructuras genéricas de inferencia.
   - Nivel de tarea: orden de las inferencias.

   En el proyecto desarrollado distinguiremos los siguientes conceptos del dominio:

   - Tarea de usuario (*user-task*). Tareas solicitadas por el usuario al sistema de agentes.
   - Computadora (*computer*). Equipo dentro de la red de la UCI disponible para el sistema de agentes.
   - Relación computadora-tarea (*user-task-computer*). Representa la relación entre una tarea y un equipo de la red.
   - Niveles de recursos (*resources-levels*). Indica los umbrales (bajo, medio, alto) para los diferentes valores que figuran en las reglas del modelo. De este modo, no es necesario cambiar las reglas del modelo

cuando se incorporan nuevos equipos a la red con características hardware (CPU, memoria, almacenamiento, gráfica, red, etc.) más recientes.

5. **Modelo de comunicación** (CM-*Communication Model*). Se encarga de detallar el intercambio de información entre los diferentes agentes involucrados en la ejecución de las tareas descritas en el modelo de tareas. Este modelo consta de tres niveles:

    - Plan de comunicación: gobierna el diálogo completo entre dos agentes.
    - Transacciones: enlazan dos tareas llevadas a cabo por distintos agentes.
    - Especificación de intercambio de información: describe la estructura interna de los mensajes de una transacción.

    Por ejemplo, dentro del modelo de comunicación se define el diagrama de diálogo correspondiente a la ejecución de una tarea de usuario. Este diagrama refleja los siguientes pasos:

    - El usuario solicita iniciar una tarea al agente local, indicando una estimación de recursos que necesitará.
    - El agente local decide cuál es el agente de la red más adecuado para ejecutar la tarea solicitada. Para ello, se encargará de monitorizar a los agentes del sistema para obtener la información necesaria sobre las capacidades de los equipos y el estado del sistema. Utilizando comandos del sistema (*ping*) es posible comprobar la conectividad con otros agentes e incluso medir el ancho de banda disponible.
    - Envío de los datos necesarios para ejecutar la tarea del usuario al agente seleccionado. En el caso de ejecutar la tarea en el agente local no será necesario realizar ningún tipo de comunicación con otro agente remoto.

- El usuario podrá comprobar el estado de la tarea cuando la interfaz de usuario del agente actualice el estado de la red de agentes. Finalmente, se obtendrá el resultado de la tarea.

6. **Modelo de diseño** (DM-*Design Model*). Se utiliza para describir la arquitectura y el diseño técnico del sistema basado en el conocimiento como paso previo a su implementación. El proceso de diseño se divide en cuatro etapas:
    - Diseño de la arquitectura global del sistema.
    - Identificación de la plataforma software y hardware de implementación.
    - Especificación de los componentes arquitecturales.
    - Especificación de la aplicación dentro de la arquitectura seleccionada.

La arquitectura de la herramienta desarrollada sigue el patrón MVC (Modelo, Vista, Controlador) donde:

- Modelo: contiene el sistema de reglas que permiten seleccionar un agente para ejecutar una determinada tarea.
- Vista: se corresponde con la interfaz de usuario. Periódicamente invoca al controlador para obtener el estado de la red de agentes.
- Controlador: se encarga de gestionar la comunicación entre agentes.

El modelado de los paquetes de la aplicación desarrollada se estructura de la forma siguiente:

- *configuration*: clases destinadas al tratamiento de la información de configuración de las tareas de usuario.
- *usertasks*: clases encargadas de gestionar las tareas de usuario (creación, eliminación, obtención de su ID, comprobación del estado, estimación de recursos necesarios).
- *model*: clases del modelo de la aplicación encargadas de interactuar con el sistema de reglas.

- *view*: clases de la interfaz gráfica de usuario. También incluirá las clases correspondientes para la visualización con Java 3D (*j3d*).
- *controller*: clases del controlador de la aplicación encargadas de gestionar la comunicación entre los agentes. Aquí tendremos:
    - *agent*: contiene las clases que implementan las funcionalidades principales del controlador.
    - *fipaostasks*: contiene las clases para las tareas FIPA-OS.
    - *messages*: contiene las clases relacionadas con los mensajes que un agente puede enviar o recibir.
    - *protocol*: contiene las clases asociadas con el protocolo *FIPARequest* para el envío y recepción de mensajes.
- *util*: clases que permiten simplificar la complejidad del núcleo de la aplicación y que se pueden utilizan en otros paquetes.

A continuación se incluyen algunos de los diagramas UML más relevantes del sistema:

- Diagrama de clases que permite gestionar las tareas de usuario (*usertasks*). La clase *UserTask* representa una tarea de usuario y la clase *UserTaskParameter* representa un parámetro de una tarea de usuario. Este parámetro puede ser una cadena de texto o una referencia a un fichero.

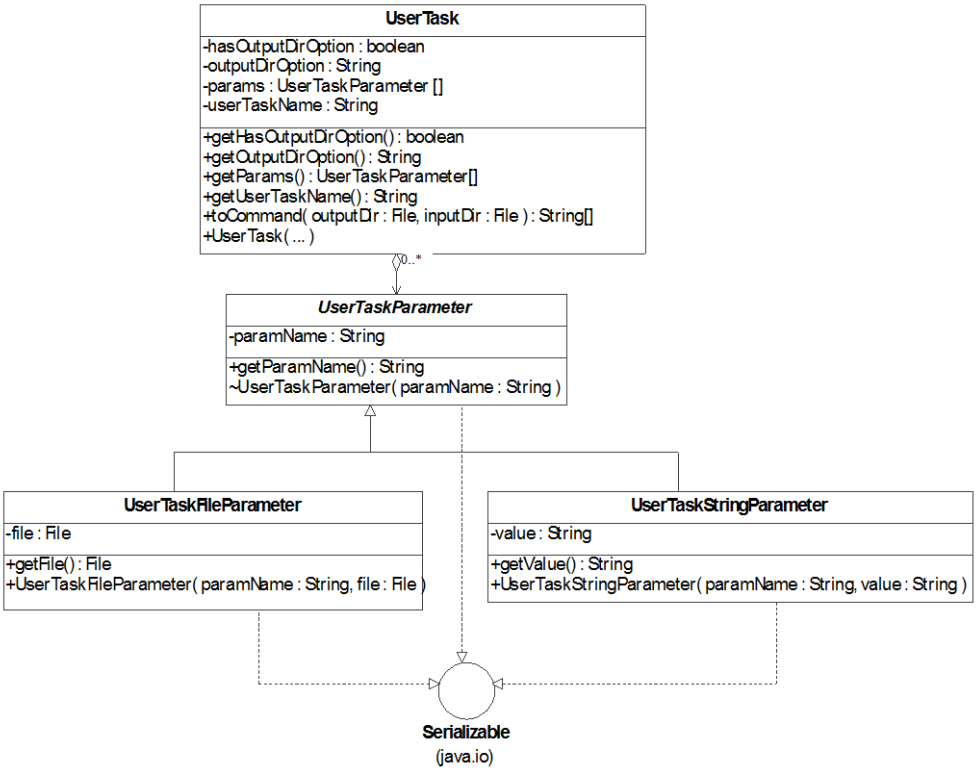


Figura 14. Diagrama de clases para una tarea solicitada por el usuario

- Diagrama del modelo (*model*). La clase *JessAgent* se encarga de la comunicación con el sistema de reglas Jess. La clase *BMASAbstractAgent* realiza la selección del agente que ejecutará una tarea de usuario específica. La clase *Computer* representa el estado de un equipo de la red. La clase *ResourcesLevels* obtiene los valores de los atributos de la configuración de la aplicación.

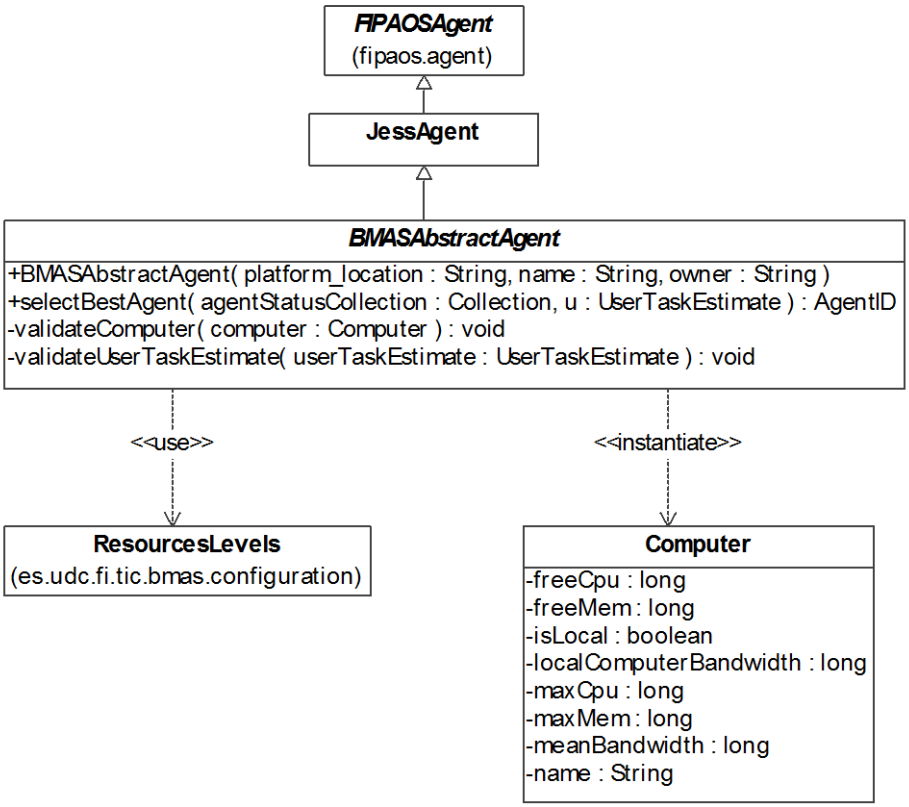


Figura 15. Diagrama de clases del modelo

- Diagrama de la vista (*view*). Las clases del controlador no crean directamente la clase que implementa la interfaz de usuario, sino que se obtiene una instancia de forma dinámica en función de la configuración de la aplicación. De este modo, se consigue desacoplar las clases de la vista y del controlador, facilitando el mantenimiento.

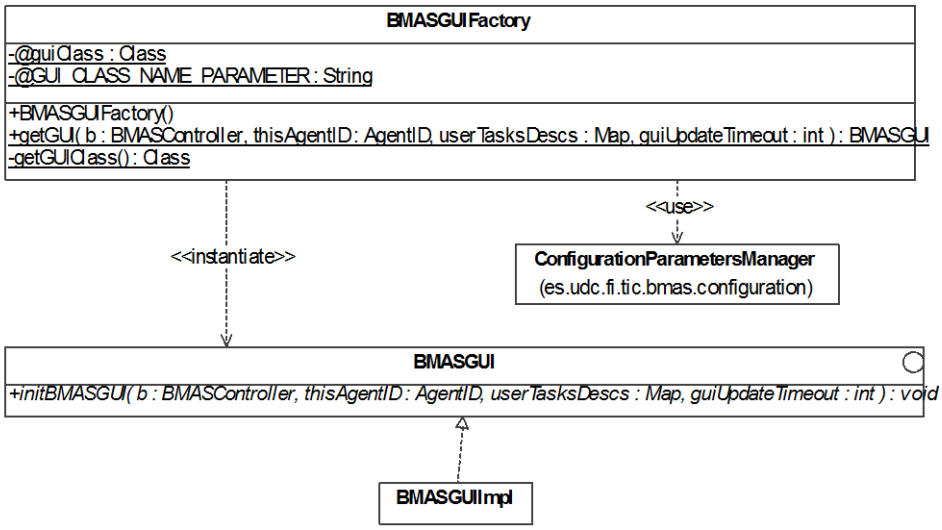


Figura 16. Diagrama de clases de la vista

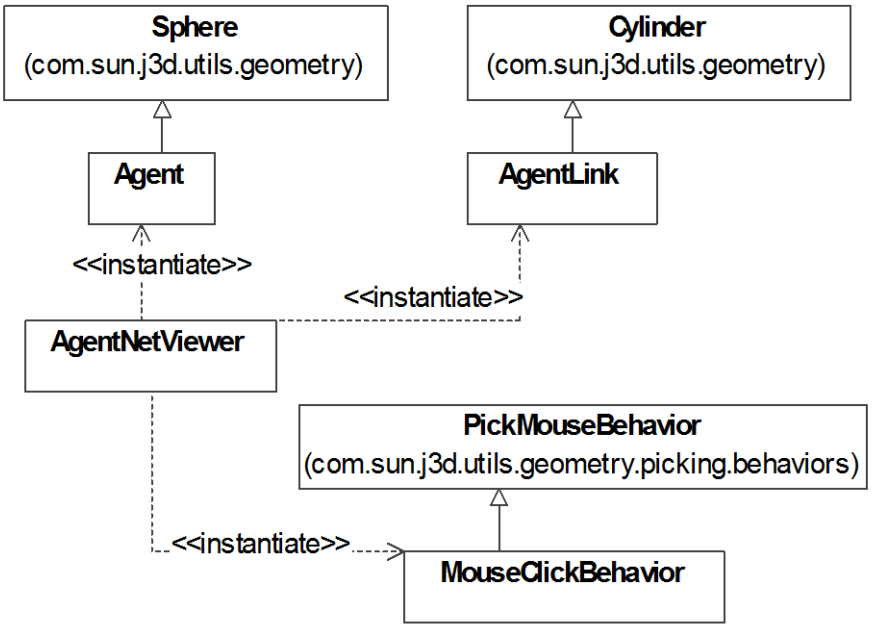


Figura 17. Diagrama de clases de la representación Java 3D

- Diagrama de clases del núcleo del controlador (*controller*). La clase *BMASAgent* es la clase principal del controlador. Las clases de la interfaz de usuario no trabajan directamente con la clase que implementa el controlador, sino con la interfaz *BMASController*. Así, se consigue desacoplar ambas capas, facilitando cambios de implementación futuros y mejorando el mantenimiento.

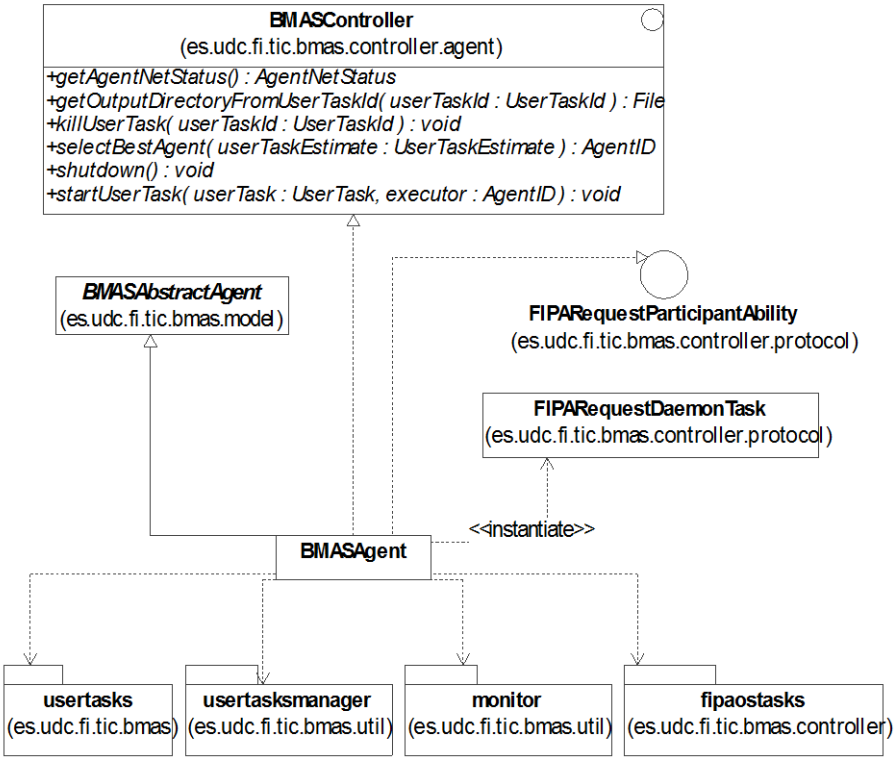


Figura 18. Diagrama de clases del núcleo del controlador

En el diseño final de la interfaz del sistema se ha intentado adaptar la información visualizada para mejorar la experiencia del usuario, simplificando y agrupando los datos mostrados. Así, en la pantalla principal de cada agente instalado tendremos una serie de pestañas, cada una de ellas con su correspondiente funcionalidad:

- Controlar el estado del equipo local en el que está instalado el agente inteligente. Se mostrarán los principales parámetros que definen su estado y de forma gráfica se podrá monitorizar la evolución del rendimiento del equipo.
- Mostrar la información correspondiente a las tareas solicitadas por el usuario local.
- Mostrar la información de todas las tareas en ejecución del sistema de agentes y de las tareas pendientes de selección.
- Lanzar la solicitud de ejecución de nuevas tareas o procesos en la red de agentes.
- Visualizar una representación esquemática de la red de agentes presentes en el sistema. Esta visualización se realiza mediante objetos 3D y permite

representar el estado de los agentes en tiempo real e interaccionar con ellos de forma simple.

La información correspondiente a cada agente resumirá los parámetros, hardware y software, más relevantes para establecer el estado del equipo en el que reside. Esta información provendrá de diferentes fuentes y será obtenida mediante llamadas a aplicaciones propias del sistema operativo y a programas específicos instalados en el equipo. Por ejemplo, en sistemas GNU/Linux podremos utilizar herramientas como: dmesg (*display message*), netstat (*network statistics*), top (*top CPU processes*), df (*disk free*), lshw (*list hardware*), hwinfo (*hardware information*), lspci (*list PCI devices*), lshal (*list HAL devices*), dmidecode (*DMI table decoder*), etc. También es posible obtener la información de cada agente mediante la consulta a un sistema de telemonitorización de equipamientos informáticos establecido en la red hospitalaria (por ejemplo: Nagios, Zabbix, etc.).

Los principales parámetros que se tendrán en cuenta para cada agente instalado en un equipo serán los siguientes:

- Rendimiento de la CPU (*CPU performance*). Es un valor numérico que permite realizar comparaciones entre las CPUs de los equipos conectados a la red. Este indicador se obtiene de la configuración de la aplicación y se establece en función de diferentes características de la CPU (arquitectura, MIPS, frecuencia de reloj, modelo, número de núcleos/cores, etc.).
- Uso de CPU (*CPU usage*). Indica el porcentaje de uso de la CPU en un determinado momento.
- Memoria física libre (*free physical memory*). Indica la cantidad de memoria RAM disponible en Bytes en un determinado momento.
- Memoria virtual libre (*free virtual memory*). Indica la cantidad de memoria virtual libre en Bytes en un determinado momento.
- Carga de memoria (*memory load*). Es el porcentaje de memoria RAM en uso en un momento determinado.
- Número de CPUs (*number of CPUs*). Indica el número de CPUs disponibles en un equipo.

- Tamaño de página (*page size*). Es el tamaño de página en Bytes.
- Memoria física (*physical memory*). Es la cantidad de memoria RAM total en Bytes.
- Memoria virtual (*virtual memory*). Es la cantidad de memoria virtual total en Bytes.
- Ancho de banda medio (*mean bandwidth*). Indica el valor del ancho de banda medio en Bytes/segundo.

Además, cada agente de la red gestionará una tabla que le permita conocer el ancho de banda disponible en cada momento entre el agente seleccionado y cada uno de los demás agentes del sistema.

En la implementación de la interfaz de la aplicación distinguimos dos funcionalidades principales:

- Consulta de la red de agentes. Permite consultar el estado de la red de agentes distribuidos. Se puede visualizar la red mediante una representación gráfica 3D interactiva o consultar una lista de ítems seleccionables.
- Gestión de tareas. Incluye todas las opciones que permiten gestionar las tareas en ejecución y la solicitud de nuevas tareas por parte del usuario.

A continuación se comentarán cada una de las funcionalidades que proporciona la interfaz de los agentes desarrollados para su gestión.

Red de agentes

La interfaz dispone de una pestaña dedicada a la visualización e interacción con la red de agentes. Este espacio permite al usuario la monitorización gráfica y en tiempo real de la red de agentes, utilizando objetos 3D simples. Los objetos principales son los que representan los agentes instalados en cada equipo de la red. Son figuras esféricas cuyo color identifica claramente el estado del agente. Así tendremos:

- Agente remoto activo: objeto con color verde.
- Agente remoto inactivo: objeto con color rojo.

También se diferencia al agente local de los agentes remotos estableciendo el color azul para el objeto que lo representa.

La comunicación entre los agentes de la red se indica mediante un objeto de forma cilíndrica que conecta dos agentes. Mediante la modificación del grosor de este objeto se representa el ancho de banda estimado entre los agentes. Así, cuanto mayor sea el grosor del objeto cilíndrico, mayor será el ancho de banda disponible para la comunicación entre los agentes. Además, estos enlaces de comunicación entre agentes utilizan un código de colores para facilitar la visualización:

- Rojo: indica un nivel de ancho de banda bajo.
- Azul: indica un nivel de ancho de banda normal.
- Verde: indica un nivel de ancho de banda alto.
- Blanco: indica un nivel de ancho de banda muy alto.

El escenario virtual que representa la red de agentes conectados al sistema, también permite al usuario realizar acciones interactivas:

- Acercar/alejar (zoom). Permite cambiar el punto de vista del usuario mediante la acción de acercarse o alejarse, para ello se utiliza el botón central del ratón.
- Desplazar: Permite desplazarse por la red de agentes y se realiza con el botón izquierdo del ratón.
- Rotar: Permite realizar rotaciones dentro del espacio virtual y se realiza utilizando el botón derecho del ratón.
- Seleccionar un agente de la red: La selección de un agente de la red se resaltará en el objeto seleccionado y se realiza pulsando con el ratón sobre la esfera que representa el agente. Además, esta acción permite acceder directamente a la información detallada sobre el estado de ese agente. Dicha información será mostrada en el panel informativo situado en la parte derecha de esta pantalla de la aplicación.
- Mostrar los enlaces entre un agente y el resto de agentes: Al seleccionar un agente se resaltan también sus enlaces con otros agentes de la red y la información correspondiente a estos enlaces también se mostrará de forma

detallada en la aplicación. Así, el usuario puede comprobar el estado de las comunicaciones entre un determinado agente y los otros agentes interconectados a través de la red.

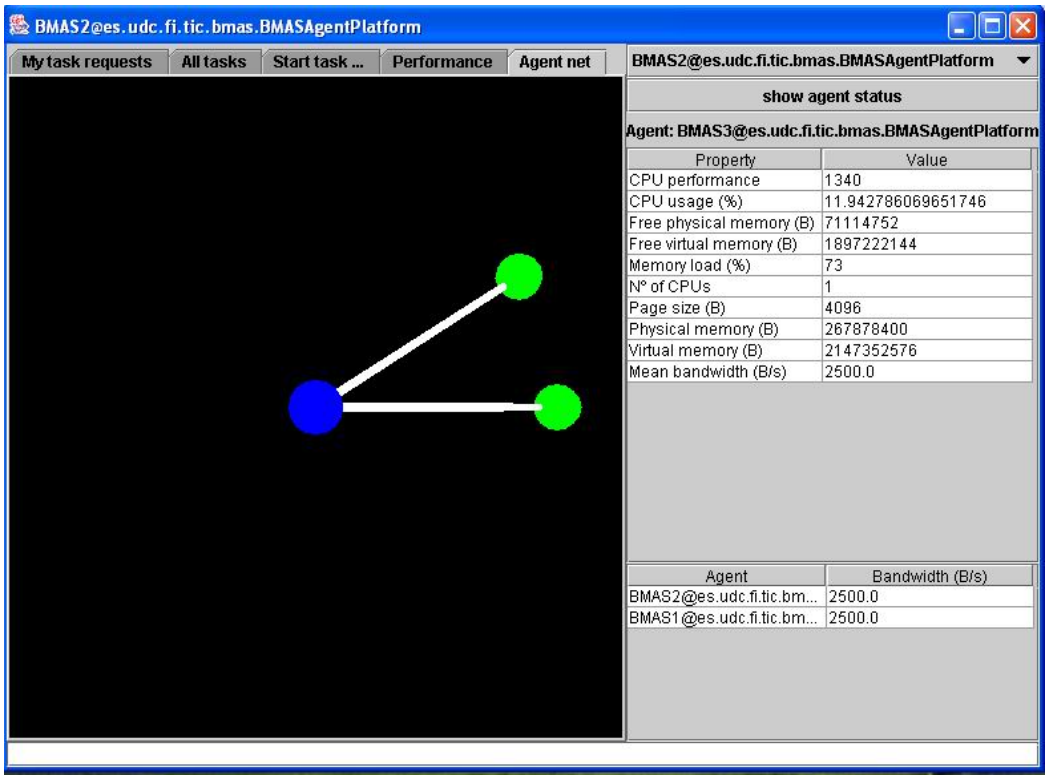


Figura 19. Visualización 3D interactiva de la red de agentes

Rendimiento del agente local

Se ha definido una pestaña dedicada a mostrar la información detallada sobre el estado del agente local. En esta pantalla de la interfaz se muestran en tiempo real los valores de los parámetros que componen el rendimiento de un equipo. Así, podremos seguir de forma gráfica la evolución temporal del uso de la CPU y la carga de memoria. Al mismo tiempo, se mostrarán los valores actuales del equipo en la parte derecha de la pantalla, incluyendo los valores que definen los enlaces de comunicación entre el agente local y el resto de agentes conectados.

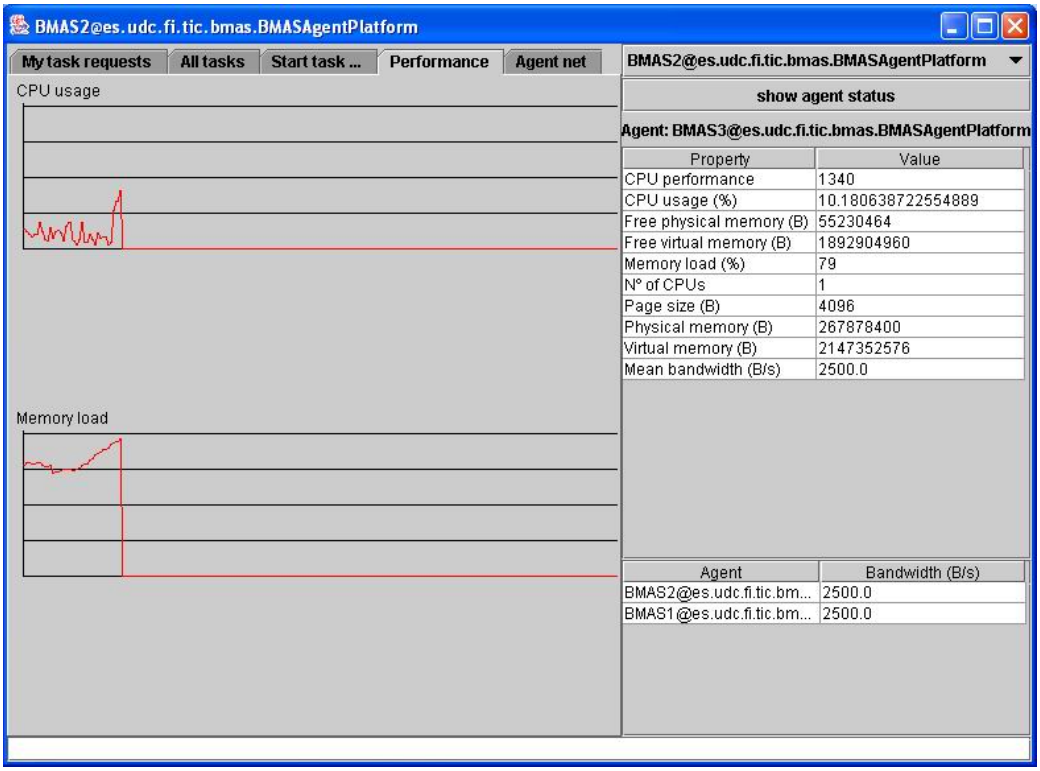


Figura 20. Visualización del estado del equipo local

Solicitud de ejecución de una tarea

Una de las funciones principales de esta aplicación es la realización de tareas que deben ser ejecutadas en alguno de los equipos conectados y gestionados a través de la red de agentes, atendiendo a criterios de rendimiento y optimización de recursos. Así, en la interfaz se incluye una pestaña que permite al usuario solicitar la ejecución de una tarea al sistema, de tal modo que la selección y envío de datos al equipo seleccionado sea totalmente transparente. Esta pantalla está formada por los siguientes componentes:

- Panel de tareas (*Tasks*): muestra una lista con todos los tipos de tareas definidas y que serán los tipos de tareas disponibles para ser solicitadas. Se identifican por una etiqueta con el nombre de la tarea y un campo de descripción de la misma.
- Panel de parámetros (*Parameters*): muestra una lista con los parámetros de la tarea seleccionada en el panel de tareas. Para cada parámetro se solicita cubrir los valores correspondientes a los campos: nombre, descripción, tipo (fichero o cadena), valor.

- Panel de opciones (*Options*): mediante la selección de las siguientes opciones se permite configurar detalles de funcionamiento de la tarea solicitada y servirán de ayuda en la toma de decisiones del sistema de agentes. Estas opciones son:
    - Prioridad de la tarea
    - Estimación del uso de CPU de la tarea
    - Estimación del uso de memoria de la tarea
    - Estimación del uso de comunicaciones
    - Estimación del tiempo que necesitará la tarea para finalizar
    - Agente que ejecutará la tarea
- Botón de selección del agente (*Select executor agent*): este botón solicita al sistema la selección de un agente de la red para ejecutar la tarea del usuario. El resultado de esta selección será proporcionado por el sistema de conocimiento, el cual utilizará la información sobre el estado del sistema de agentes, los recursos de los equipos conectados, y las características propias de la tarea para decidir el equipo más indicado para la ejecución de la tarea solicitada por el usuario.
- Botón de inicio (*Start task*): permite ejecutar la tarea solicitada por el usuario. Iniciará el proceso seleccionando el equipo en el que será ejecutada la tarea y le enviará la información sobre la tarea y los datos necesarios para su realización. Una vez terminada la tarea, el agente que la ha realizado le devolverá la información al agente solicitante con el resultado de su ejecución y los datos de salida obtenidos. Cuando se produzca algún tipo de problema, como una pérdida de comunicación o un error en el envío de la información, el agente solicitante podrá decidir si realiza un reenvío o si cancela la petición para seleccionar otro agente que se encargue de ejecutar la tarea.

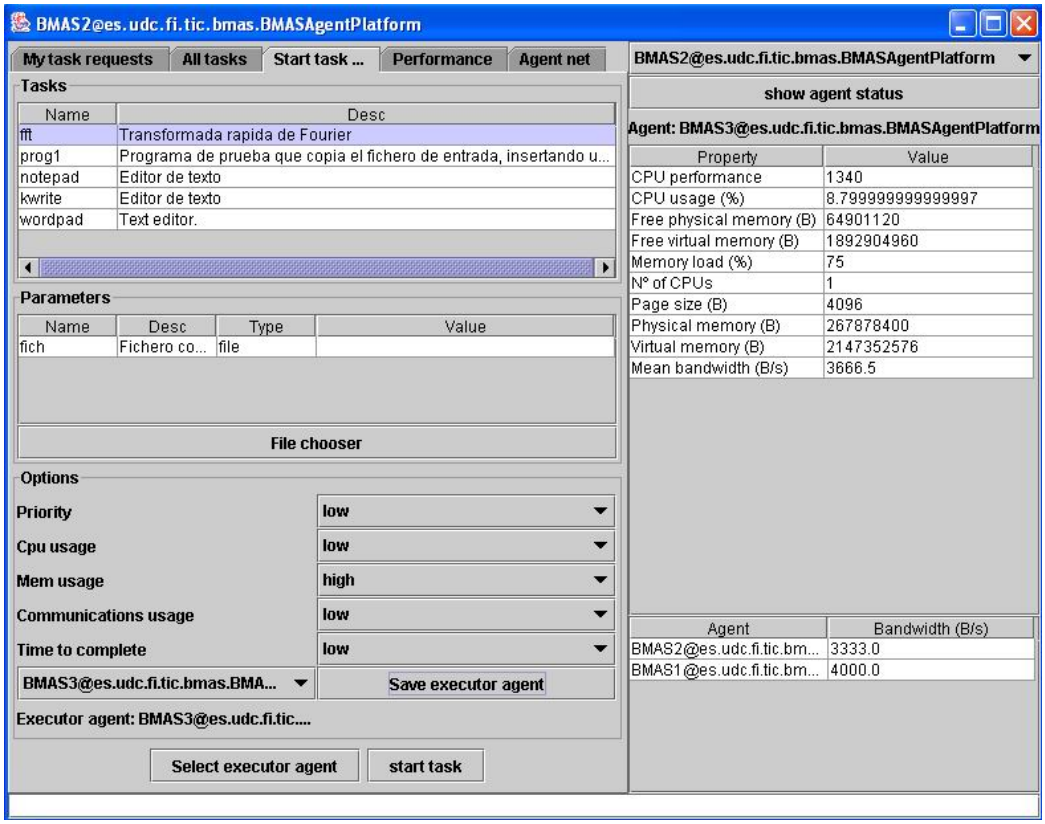


Figura 21. Solicitud de ejecución de una tarea

Lista de todas las tareas ejecutadas en el sistema de agentes

En la interfaz se dispone de una pestaña que permite ver la lista de tareas solicitadas al sistema de agentes. De este modo podemos visualizar la información básica sobre las tareas y el estado en el que se encuentran en tiempo real. La información que se muestra en esta opción es la siguiente:

- Agente iniciador (*Initiator*): indica el agente desde el que se ha solicitado la ejecución de una tarea de usuario.
- Agente ejecutor (*Executor*): indica el agente seleccionado para la ejecución de la tarea solicitada.
- Número (*Nº*): valor numérico asignado de forma automática por el sistema como parte del identificador de la tarea.
- Nombre de la tarea (*Name*): indica el nombre asignado a la tarea y es una etiqueta que permite al usuario reconocer el objetivo de la tarea.
- Estado de la tarea (*Status*): etiqueta que muestra el estado de ejecución de una tarea. Tendrá tres posibles valores:

- Ejecución (*Running*): la tarea se está ejecutando y por lo tanto aún no ha sido terminada.
- Completada (*Done*): la tarea ha sido ejecutada correctamente y por lo tanto está terminada.
- Cancelada (*Killed*): la tarea no ha finalizado correctamente o ha sido cancelada por el usuario.

Además de visualizar la lista de tareas ejecutadas por el sistema de agentes, desde esta pantalla de la interfaz el usuario podrá cancelar una tarea. Para ello, tendrá que seleccionar la tarea de la lista y pulsar el botón encargado de realizar esta función. La orden de cancelación será enviada al agente encargado de su solicitud para que se lo notifique al agente que la está ejecutando.

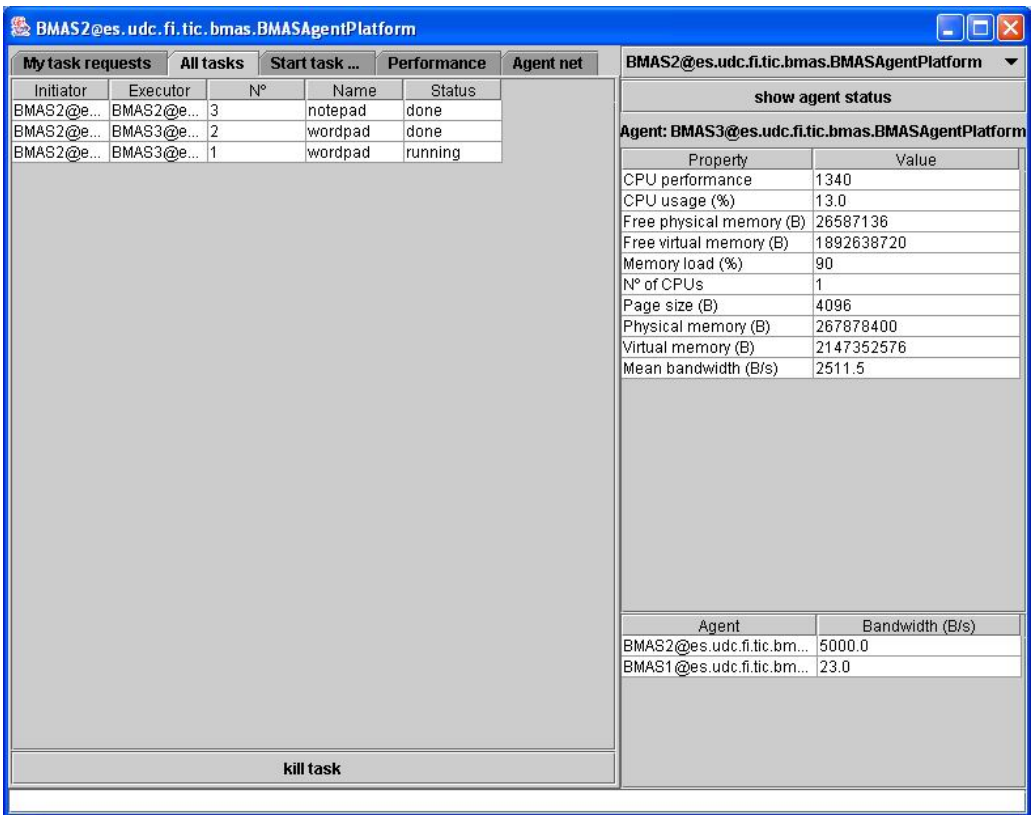


Figura 22. Visualización del estado de las tareas del sistema de agentes

Lista de las tareas solicitadas por el agente local

El seguimiento de las tareas solicitadas por el agente local se registrará en una pestaña específica de la interfaz. De este modo se podrá consultar la información y

el estado de aquellas tareas solicitadas desde el equipo local. La información que se muestra en este caso está compuesta por los siguientes valores:

- Agente ejecutor (*Executor*): indica el agente seleccionado para la ejecución de la tarea solicitada.
- Número (*Nº*): valor numérico asignado de forma automática por el sistema como parte del identificador de la tarea.
- Nombre de la tarea (*Name*): indica el nombre asignado a la tarea y es una etiqueta que permite al usuario reconocer el objetivo de la tarea.
- Estado de la tarea (*Status*): etiqueta que muestra el estado de ejecución de una tarea. Tendrá tres posibles valores:
    - Ejecución (*Running*): la tarea se está ejecutando y por lo tanto aún no ha sido terminada.
    - Completada (*Done*): la tarea ha sido ejecutada correctamente y por lo tanto está terminada.
    - Cancelada (*Killed*): la tarea no ha finalizado correctamente o ha sido cancelada por el usuario.
- Directorio de salida (*OutputDirectory*): especifica la dirección completa del directorio local que contendrá los archivos de salida obtenidos como resultado de la ejecución de una tarea.

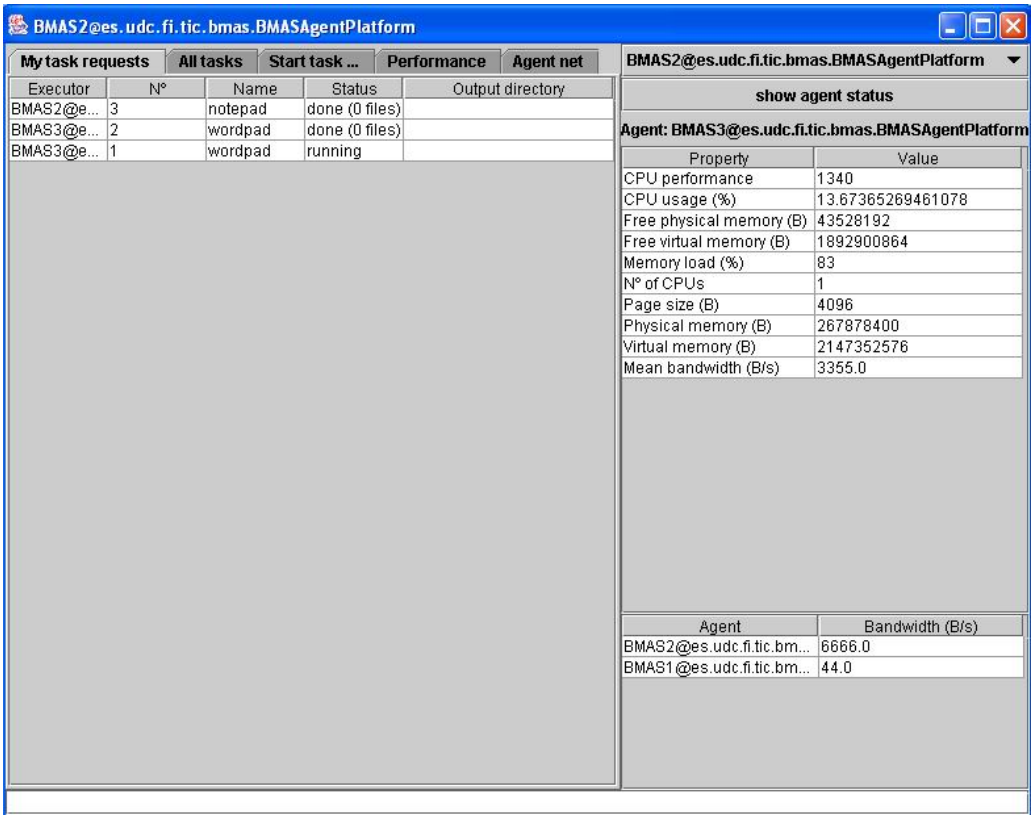


Figura 23. Visualización de la lista de tareas solicitadas por el usuario del agente local

Gestión de tareas de usuario

Será necesario gestionar cualquier tipo de tarea que el usuario pueda solicitar para su ejecución en el sistema de agentes. Por ello, la aplicación permite realizar desde la interfaz las operaciones típicas necesarias para una tarea: añadir, modificar, eliminar. Estas operaciones serán realizadas desde la pantalla principal de la aplicación, donde se puede consultar y gestionar la lista de tipos de tareas ya existentes.

Después de seleccionar una de las tareas ya definidas en la lista, se pueden realizar las operaciones de eliminación o modificación de la tarea pulsando en los botones correspondientes. La primera de las operaciones borrará la tarea de la lista, previa solicitud de confirmación por parte del usuario. La segunda de las operaciones permite acceder y modificar las propiedades y valores que definen la tarea.

Mediante el botón de alta de una nueva tarea se accede a las pantallas que permiten definir la tarea, solicitando la información básica para su ejecución. La información que se solicitará será la siguiente:

- Nombre de la tarea: indica el nombre de la tarea que se ejecutará cuando un usuario la solicite.
- Descripción: indica de forma breve el objetivo que se pretende conseguir con la ejecución de la tarea.
- Directorio de salida: parámetro que permite indicar que la tarea requiere un directorio de salida para contener los resultados de la ejecución de la tarea.
- Ruta de instalación: indica la ruta completa y el nombre del programa que se ejecutará al solicitar esta tarea. Normalmente será una aplicación ejecutable pero también puede ser un script del intérprete de comandos del sistema.
- Parámetros de entrada: permite gestionar los parámetros de entrada de la aplicación a ejecutar. Con esta opción se pueden añadir todos los parámetros necesarios indicando el nombre y el tipo de cada argumento. Se han definido dos tipos posibles de argumentos: cadena de caracteres y fichero.

4.7. CONCLUSIONES

En este proyecto se ha desarrollado un sistema de regulación y control del ancho de banda en las transmisiones de datos en red en un sistema de Telemedicina por medio de una red de agentes multiplataforma. Este desarrollo forma parte de un proyecto más amplio de Telemedicina, y trata de conseguir un mejor aprovechamiento de los recursos facilitando la gestión distribuida de tareas.

La heterogeneidad de los equipos disponibles en el sistema de Telemedicina requiere que esta herramienta pueda contemplar diferentes sistemas operativos, de tal modo que sea fácilmente extensible a nuevos entornos. Así, se ha implementado como una aplicación multiplataforma en lenguaje Java.

Se ha empleado la tecnología de agentes inteligentes, siguiendo las especificaciones estándar FIPA, para conseguir un sistema tolerante a fallos (si un agente tiene algún problema, el sistema sigue funcionando), y que permite distribuir la carga computacional entre varias máquinas de forma transparente para el usuario.

Las tareas que se pueden ejecutar en el sistema de agentes pueden ser definidas y configuradas por el usuario desde la interfaz de la aplicación. Esto proporciona un elevado nivel de adaptabilidad a las necesidades de un problema concreto.

La gestión de las tareas es capaz de tratar eventos asíncronos (recepción de nuevos mensajes, cancelación de una tarea de usuario, etc.) y utiliza hilos de ejecución para mejorar el rendimiento. Se han tenido en cuenta los posibles problemas de concurrencia sincronizando los accesos a recursos compartidos, creando copias de los mismos, o estableciendo la política de que ciertos objetos no se pueden modificar después de que hayan sido creados e inicializados.

Cuando una tarea determinada se está ejecutando en un agente remoto, el sistema gestiona la transferencia de los ficheros de entrada y de salida de forma transparente al usuario, incluyendo técnicas de compresión de datos en tiempo real.

El mecanismo de toma de decisiones basado en reglas, implementado con JESS, permite determinar el agente más adecuado para ejecutar una determinada tarea en el momento en que se solicita. Para ello, se tendrá en cuenta el estado de la red de agentes y las características de cada uno de los equipos disponibles en la red. Los umbrales que definen la clasificación de los componentes de un equipo pueden ser configurados en la aplicación, lo que permite adaptar el sistema a las necesidades de cada entorno y tener en cuenta las evoluciones tecnológicas. Así, se evita tener que modificar el código de la aplicación para añadir nuevas reglas o modificar las existentes.

El desarrollo de la interfaz gráfica de cada agente trata de conseguir un completo mecanismo de control y gestión. Sin embargo, se ha intentado que su diseño sea simple y fácil de utilizar por el usuario. Para conseguir este objetivo se incluido una representación 3D interactiva de la red de agentes implementada con el API Java 3D.

Este sistema se puede adaptar para su utilización en otros ámbitos de aplicación en los que sea necesario realizar una gestión distribuida de tareas con una optimización de los recursos disponibles en la red. Para ello, las principales modificaciones necesarias afectarían a las reglas definidas en el sistema. Aunque las tareas solicitadas por el usuario serán diferentes para cada ámbito, el soporte

para poder crear tareas desde la interfaz de los agentes evita tener que implementar nuevas funcionalidades.

5

Supervisión de sistemas y redes de comunicaciones

5.1. INTRODUCCIÓN

La **gestión de sistemas** es la actividad de integración e identificación de diversos productos y procesos con el objetivo de proporcionar un entorno de tecnologías de la información estable y eficaz [Scheisser 2010]. Bajo la gestión de sistemas se cubren específicamente las siguientes disciplinas:

- Gestión de incidentes (gestión de eventos, recuperación ante fallos, etc.).
- Gestión de problemas (análisis y gestión de problemas de funcionamiento de equipamientos por el administrador de sistemas: actualizaciones, parches, etc.).
- Gestión de disponibilidad (comprobación de funcionamiento de servicios a usuarios finales y a otros servicios, gestión de alertas, etc.).
- Gestión de rendimiento (pruebas de rendimiento y ajuste de configuración, identificación de cuellos de botella en el sistema y en la conexión, etc.).
- Gestión de la configuración (descubrimiento de dispositivos de red y componentes del centro de datos, topología de la red de datos, etc.).

Con estas actividades se pretende mejorar los esfuerzos y tiempos dedicados para la gestión de sistemas.

Podemos entender la **monitorización de un Centro de Proceso de Datos** (CPD) como el proceso encargado de la monitorización, gestión, y funcionamiento de un centro de datos para cumplir con los requisitos operativos y organizacionales establecidos. Este proceso utiliza técnicas y herramientas, manuales y automáticas, para asegurar el mejor funcionamiento posible del CPD.

Figura 24. Ejemplo de equipamientos de un CPD (CeCaFI)

La supervisión de sistemas informáticos y redes de comunicaciones ha sido desde sus inicios una de las funciones principales del personal técnico de un CPD. Su misión se puede resumir en comprobar, de forma periódica, el correcto funcionamiento de todos los equipamientos informáticos y de redes que estén bajo supervisión. Para ello, se utilizarán diversas herramientas de monitorización con el objetivo de ayudar al personal técnico a garantizar la prestación adecuada de servicios a los usuarios finales, e incluso detectar lo más rápidamente posible la

aparición de incidencias y la búsqueda de soluciones ante todo tipo de problemáticas (tanto hardware como software).

En la actualidad, existen multitud de sistemas de monitorización de sistemas dotados de interfaces de usuario accesibles mediante terminal o mediante interfaz Web. Estas herramientas se centran en mostrar de forma más o menos estructurada un conjunto de parámetros que definen el estado de un determinado dispositivo, pero también pueden generar información resumida (mensajes sobre alertas, alarmas, errores, etc.) o generar estadísticas basadas en datos históricos. En algunos casos estas herramientas también incluyen representaciones esquemáticas de las redes y equipamientos que se están monitorizando con el fin de ayudar al personal técnico a localizar posibles incidencias.

5.2. HERRAMIENTAS DE MONITORIZACIÓN Y SUPERVISIÓN

Las herramientas de supervisión de sistemas informáticos y redes de datos son aplicaciones informáticas cuyo objetivo es proporcionar un mecanismo de ayuda al personal técnico para verificar el correcto funcionamiento de los equipamientos. Estas herramientas realizan tareas de monitorización en tiempo real, mediante las cuales se encargan de recoger y visualizar toda la información relevante para conocer el estado actual y la evolución temporal de los activos conectados al sistema. Además de las tareas de monitorización, realizan tareas de supervisión mediante las cuales se interpreta la información recogida con la finalidad de detectar cualquier problema que se pueda producir en el funcionamiento normal de los activos. En estos casos, el sistema de supervisión debe notificar al personal técnico la aparición de incidencias y alarmas para que puedan ser tratadas con la mayor brevedad posible [Campi 2008].

Estos sistemas realizan una función de vigilancia completa del funcionamiento de un dispositivo (parte hardware) y de los servicios que proporciona (parte software). Para realizar este control se seleccionan inicialmente los parámetros que definen su correcto funcionamiento y se comprueba en tiempo real que se mantengan dentro de unos valores adecuados. Algunos de los parámetros que pueden definir el estado del hardware de un dispositivo son: carga del procesador, uso de memoria, capacidad y ocupación de los discos de almacenamiento,

periféricos conectados, etc. En el caso de los parámetros software a supervisar, dependerán de los servicios de red proporcionados por el equipo siguiendo protocolos como: HTTP, HTTPS, POP3, SMTP, SNMP, etc.

Las tareas de monitorización y supervisión se apoyan en la utilización de alguna de las múltiples herramientas disponibles en el mercado para realizar estas funciones. Estas herramientas pueden ser específicas, para un determinado dispositivo o fabricante, o genéricas, para la supervisión de una amplia variedad de dispositivos y fabricantes de forma unificada. Además, dentro de los sistemas existentes podemos encontrar tanto herramientas propietarias como herramientas de código abierto (*Open Source*). En muchos casos es posible ampliar o mejorar las funcionalidades proporcionadas por las herramientas de supervisión mediante el desarrollo de aplicaciones específicas (*plugins*).

5.2.1. Monitorización de sistemas

La monitorización de sistemas o equipamientos informáticos se basa en la recopilación de datos sobre el estado de los activos y en su procesado posterior para su visualización mediante una interfaz adecuada (tanto de forma gráfica como en texto). Los datos recopilados pueden obtenerse de forma distribuida a través de una red de datos o pueden provenir de un único equipo.

Los datos que definen el estado de un equipamiento dependerán del tipo y funcionalidad de cada equipo (por ejemplo: servidores Web, servidores de almacenamiento, servidores de copias de seguridad, etc.) y por lo tanto será preciso establecer qué parámetros deben ser tenidos en cuenta por el sistema de monitorización y dónde obtenerlos. Así, estos datos serán obtenidos mediante la consulta de:

- **Ficheros de configuración** (tanto del sistema operativo como de las aplicaciones y servicios relevantes). Archivos en los cuales se reflejan los parámetros de configuración de un proceso.
- **Ficheros de registro de eventos** (ficheros de *log*). Archivos en los que se registran los eventos (mensajes de aviso, alerta, y error) producidos por el sistema operativo y las aplicaciones en ejecución.

- **Comandos del sistema operativo y aplicaciones instaladas**. Ejecución de aplicaciones, incluidas las propias de la herramienta de monitorización, que permiten obtener información sobre el estado del equipo.

Es importante que la herramienta de monitorización tenga un rendimiento adecuado y que no influya en el funcionamiento normal del equipo monitorizado, consumiendo el menor número posible de recursos del sistema. Además, la monitorización debe ser eficiente para poder mostrar/transmitir los datos del equipo en tiempo real.

Otro aspecto a tener en cuenta en este tipo de herramientas será la escalabilidad que permita gestionar de forma correcta un amplio grupo de equipamientos distribuidos por un CPD.

Dentro de los sistemas de monitorización, para la obtención de los datos relevantes de un equipo distinguimos tres modos principales:

- *Monitor poll*: Forma de recolección de datos basada en la petición de órdenes de ejecución de comandos encargados de recopilar los datos del sistema monitorizado.
- *Agent push*: Método de recolección basado en la instalación de una aplicación (agente de monitorización) en cada sistema monitorizado dedicado a la recopilación periódica de datos del sistema.
- *Hybrid mode*: Aproximación intermedia entre los dos métodos anteriores donde la configuración del sistema determina el método a utilizar en cada caso o servicio del equipo.

En cualquiera de los modos citados anteriormente, aunque exista la posibilidad de hacer una supervisión local en cada equipo, la información obtenida podrá ser transmitida por medio de una red de datos a un servidor central en el cual esté instalado el software de gestión de la monitorización.

5.2.2. Monitorización de redes de datos

La monitorización de redes de datos es similar a la de sistemas pero específicamente dedicada a comprobar el correcto funcionamiento de los

dispositivos de red y de los enlaces de conexión existentes para detectar posibles incidencias y errores.

Detectar incidencias en las comunicaciones de datos se centra en la comprobación de parámetros como: tiempos de respuesta, disponibilidad, tiempo de funcionamiento, etc. El sistema de monitorización de redes debe detectar la existencia de errores de conexión, *timeouts*, latencias elevadas, pérdidas de paquetes, etc. Estos problemas deben ser notificados al sistema de monitorización para intentar solventarlos lo más rápidamente posible ya que pueden provocar la inaccesibilidad a equipamientos y sistemas conectados a la red.

5.3. PRINCIPALES SISTEMAS DE MONITORIZACIÓN

Atendiendo a la forma en la que se realiza la monitorización distinguiremos entre sistemas de monitorización remota y sistemas de monitorización local. Los primeros se basan en la existencia de agentes o aplicaciones, en ejecución en cada máquina a monitorizar, encargados de recopilar y enviar la información a un servidor que los gestiona de forma remota. El segundo tipo de sistemas trabajan de forma local utilizando protocolos de comunicaciones, generalmente el protocolo SNMP [Mauro 2005], para obtener la información de cada equipo monitorizado.

5.3.1. Sistemas de monitorización remota

Los sistemas de monitorización remota son herramientas de supervisión que están basadas en la recolección de datos locales mediante aplicaciones software (agentes) y en su envío a un servidor central para su procesado y posterior visualización. Este tipo de sistemas son los más utilizados en equipos servidores ya que permiten realizar una supervisión centralizada y en tiempo real de todos los parámetros importantes de los dispositivos.

Existen multitud de alternativas a la hora de implantar una herramienta de monitorización de este tipo pero entre las opciones más utilizadas destacamos:

- **Nagios**. Es una herramienta de código abierto que está considerada como unos de los principales sistemas de monitorización de redes, infraestructuras y software. Desarrollado por Ethan Galstad, su primera versión se publicó en el año 1998, con el nombre de NetSaint, que

posteriormente fue modificado al actual Nagios (*Nagios Ain't Gonna Insist On Sainthood*) debido a problemas legales. Actualmente cuenta con una gran difusión y una amplia comunidad de desarrolladores que lo convierten en el sistema por excelencia para realizar funciones de monitorización y supervisión.

Nagios ofrece capacidades de monitorización completa y sistemas de alerta para equipos, dispositivos de red, aplicaciones y servicios [Nagios 2015]. Su función principal será supervisar los equipamientos y servicios con el objetivo de detectar y notificar incidencias, incluyendo la notificación de las resoluciones de problemas encontrados. Puede utilizar diferentes tipos de vías de notificación al usuario como: correo electrónico, mensaje SMS, mensaje instantáneo, etc.

El estado de los equipamientos conectados al sistema Nagios se obtiene mediante agentes capaces de monitorizar los activos instalados, independientemente de su localización física, utilizando diferentes tipos de canales de comunicación como túneles SSL o SSH. Estos agentes son los encargados de recoger y enviar los datos al servidor central Nagios.

Las funcionalidades de esta herramienta pueden ser ampliadas fácilmente mediante módulos específicos (*plugins*), tanto oficiales como no oficiales. De este modo, se consigue mejorar y potenciar el sistema Nagios para adaptarse a las necesidades propias un determinado CPD. La implementación de estos módulos es sencilla pero debe seguir la documentación y recomendaciones proporcionadas por el desarrollador del sistema Nagios.

Aunque su configuración se puede considerar poco amigable, basada en ficheros de configuración en texto plano, ofrece una interfaz Web donde se puede consultar el estado actual de los activos, revisar los datos históricos o generar informes de funcionamiento. Por medio de esta interfaz también se pueden detectar y revisar las alarmas que se produzcan dentro de la red bajo supervisión.

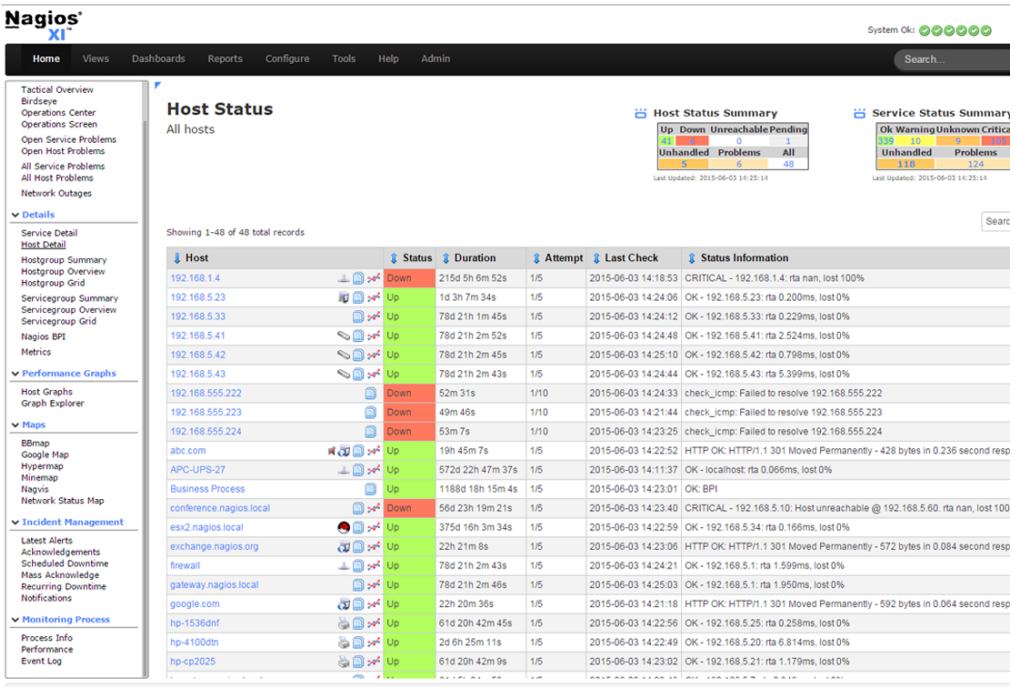


Figura 25. Nagios. Ejemplo de interfaz del sistema

- **Zabbix.** Es una herramienta diseñada para monitorizar el estado software/hardware de los diferentes servicios de red y equipos servidores [Zabbix 2015]. Desarrollado por Alexei Vladishev, surge como proyecto con licencia libre GPL en el año 2001, como una solución de monitorización de bajo coste. Está basado en el uso de agentes parametrizables capaces de recoger los datos de los equipamientos y enviarlos al servidor central Zabbix. Los datos obtenidos son almacenados en un gestor de bases de datos (por ejemplo: MySQL, PostgreSQL, SQLite, Oracle, IBM DB2).

  Esta herramienta dispone una gran variedad de opciones de monitorización: permite verificar el estado de servicios como SMTP o HTTP sin necesidad de instalar un agente específico, mediante agentes instalados puede obtener información sobre diversos parámetros como la carga de CPU, utilización de la red, ocupación de los sistemas de almacenamiento, etc.

  Dispone de una interfaz Web que permite obtener información de monitorización y gestionar eventos, tanto en formato texto como gráfico, mediante un completo sistema de configuración.

El API de desarrollo de Zabbix utiliza JSON RPC para el intercambio de información con aplicaciones de terceros. De este modo, se puede realizar cualquier función propia de la interfaz Web mediante el envío de mensajes JSON.

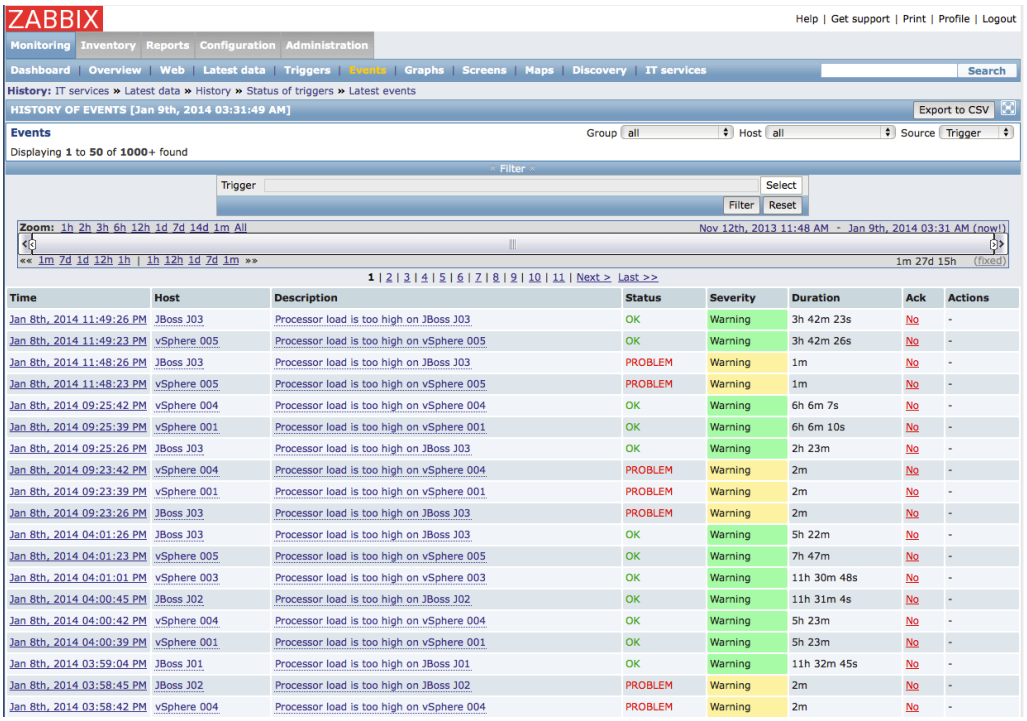


Figura 26. Zabbix. Ejemplo de interfaz del sistema

- **Pandora FMS (Pandora Flexible Monitoring System)**. Es un sistema de monitorización de infraestructuras informáticas iniciado en el año 2005 [PandoraFMS 2015]. Esta herramienta se basa en el uso de agentes específicos para obtener la información que permita analizar el estado y rendimiento de los diferentes equipamientos. La intercomunicación entre los activos y el servidor central Pandora FMS se realiza a través de SSH, FTP, NFS, o un contenedor XML.

  Este sistema de monitorización tiene una estructura basada en servidores múltiples (servidor de datos, servidor de plugins, servidor de red, etc.), una completa interface Web y una base de datos. Utiliza una base de datos MySQL para guardar todos los datos de configuración, informes, auditorías, etc. La información de monitorización, obtenida de los agentes del sistema, también se almacena en la base de datos.

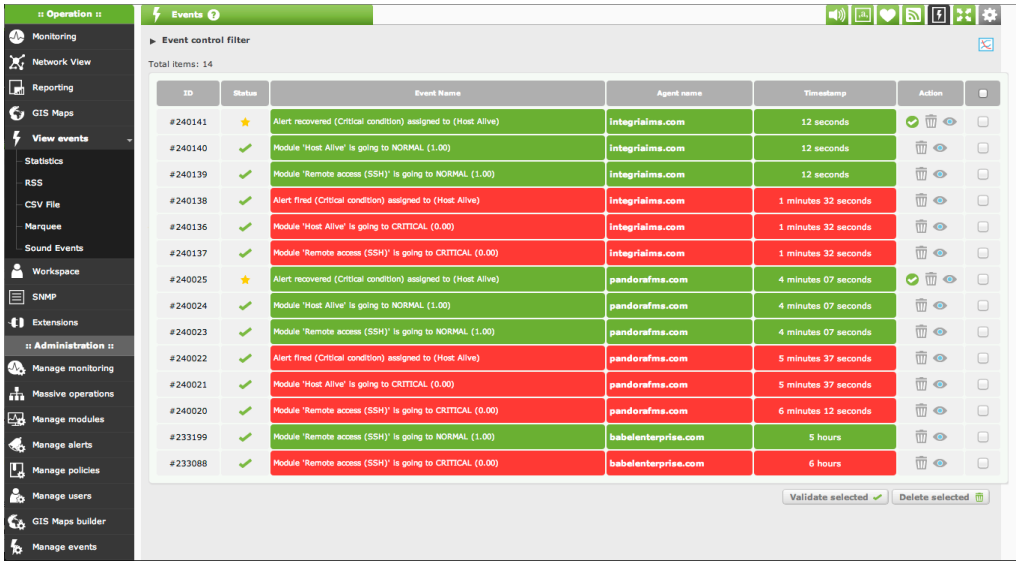


Figura 27. PandoraFMS. Ejemplo de interfaz del sistema

- **Zenoss.** Esta herramienta de código abierto está diseñada para la gestión de servidores y redes de datos. Proporciona una completa interfaz Web que permite monitorizar la disponibilidad, configuración, rendimiento, y eventos de los activos bajo supervisión [Zenoss 2015]. El proyecto comenzó en 2002, desarrollado por Erik Dahl, y actualmente dispone de una importante comunidad de desarrolladores.

  La interfaz de Zenoss es similar a la del sistema Nagios. Además, existe la posibilidad de importar los plugins de Nagios para su funcionamiento con Zenoss.

  Mediante esta herramienta se pueden obtener de forma automática algunos parámetros de los activos bajo supervisión y aunque no utiliza un agente específico puede obtener información utilizando protocolos estándar como SNMP, SSH, o WMI (*Windows Management Instrumentation*) en equipos bajo S.O. Microsoft Windows.

  En muchos casos Zenoss no necesita agentes en las máquinas remotas, ya que mediante conexiones SSH puede ejecutar de forma segura cualquier comando disponible en el equipo para extraer todo tipo de información. En el caso de máquinas bajo S.O. Microsoft Windows, este sistema puede modelar y monitorizar sus servicios utilizando aplicaciones propias para la intercomunicación a través de WMI.

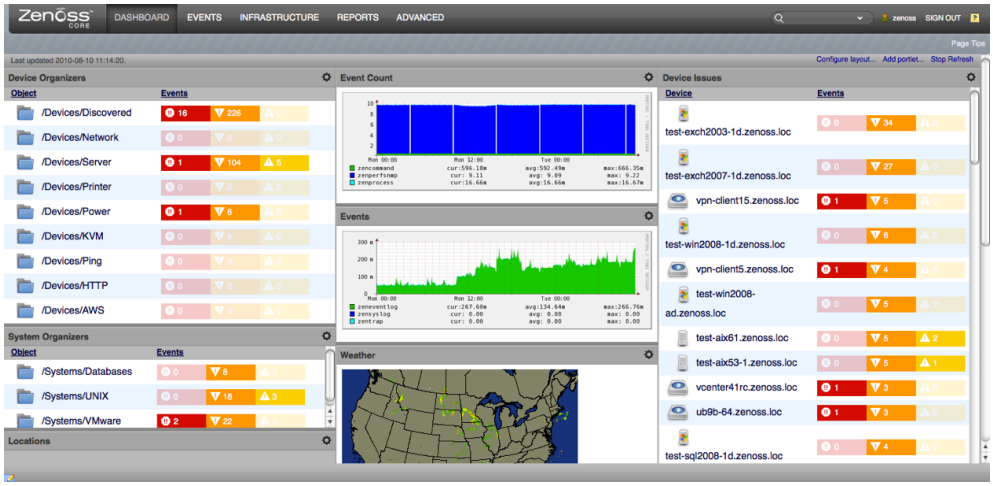


Figura 28. Zenoss. Ejemplo de interfaz del sistema

- **Cacti.** Herramienta de monitorización diseñada para proporcionar una interfaz Web del estándar de monitorización y recopilación de registros RRDtool [Cacti 2015]. Se utiliza principalmente para la monitorización de tráfico de red y para la comprobación de las interfaces de comunicación de dispositivos de red por medio del protocolo SNMP. Mediante una interfaz Web caracterizada por un aspecto visual muy cuidado se puede realizar el seguimiento de los equipamientos e incluso generar gráficos con información estadística.

  Dispone de ciertas capacidades de configuración que facilitan el establecimiento de los parámetros principales sin necesidad de configurar manualmente RRDtool [RRDtool 2015]. Entre las opciones de configuración propias de la interfaz Web destacan la gestión de permisos y usuarios, o la personalización de las gráficas generadas.

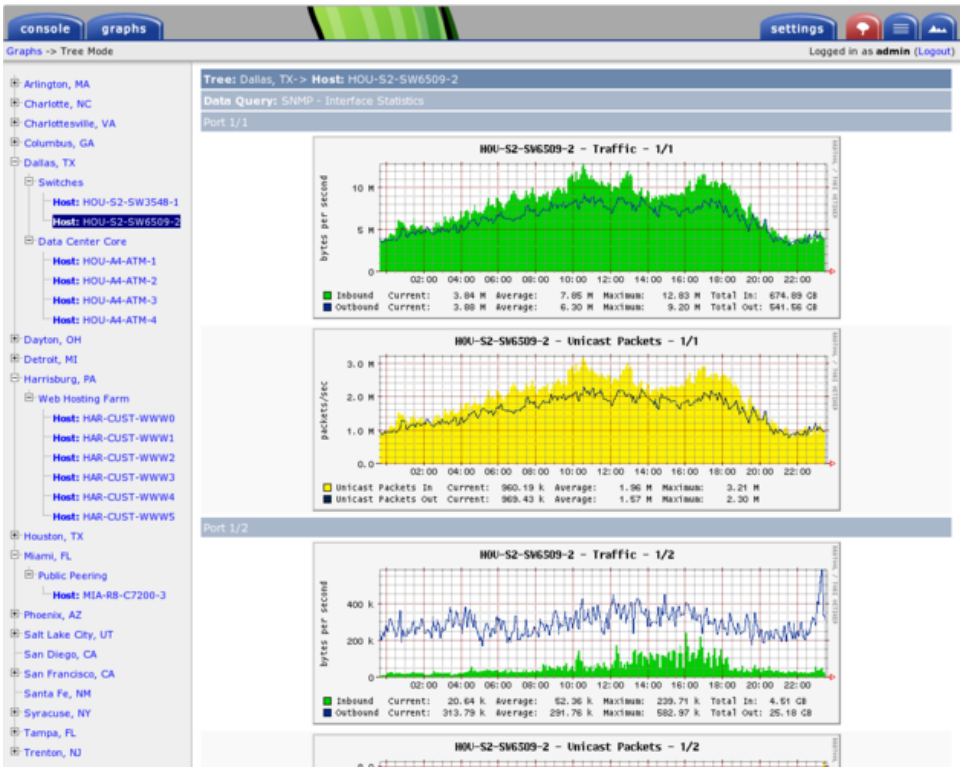


Figura 29. Cacti. Ejemplo de interfaz del sistema

5.3.2. Sistemas de monitorización local

Este tipo de sistemas de monitorización son controlados de forma local, desde el dispositivo en que ha sido instalado. Se basan en el uso de protocolos de comunicación estándar para obtener los parámetros de cada equipo monitorizado.

Dentro de este tipo de herramientas se pueden destacar los siguientes proyectos:

- **OpenNMS**. Se define como una plataforma abierta de gestión y monitorización de redes. La obtención de la información de monitorización de equipamientos se realiza principalmente a través del protocolo SNMP, pero también puede utilizar otros protocolos (HTTP, JMX, WMI, XMP, XML, etc.) para obtener la información de servicios [OpenNMS 2015]. También dispone de un potente sistema de notificaciones y gestión de eventos, y de capacidades de descubrimiento automático de equipamientos. OpenNMS se gestiona por medio de una interfaz Web simple que incluye opciones de generación de informes y estadísticas.

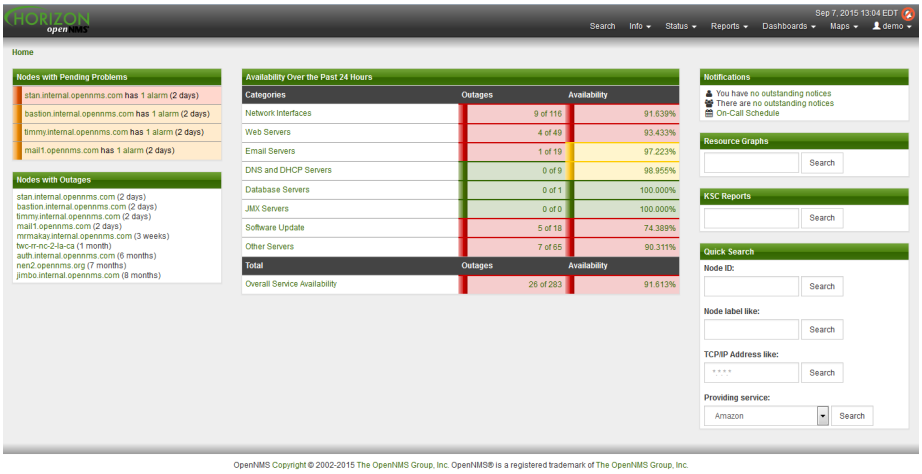


Figura 30. OpenNMS. Ejemplo de interfaz del sistema

- **NINO**. Es una herramienta de gestión de red que permite gestionar dispositivos como *routers*, *switches*, servidores y aplicaciones [NINO 2015]. Está destinada a su uso en redes locales, ya que no dispone de agentes que puedan enviar la información de monitorización a un servidor central. NINO utiliza SNMP, HTTP/TCP y WMI para gestionar la información de los dispositivos conectados.

  Dispone de una interfaz Web simple para visualizar el estado de los activos y realizar una monitorización gráfica en tiempo real. Entre sus funcionalidades destaca la capacidad para el descubrimiento automático de redes y equipamientos a partir de una dirección IP o de un rango de direcciones, o a través de la información de las tablas de enrutamiento de un *router*.

Figura 31. NINO. Ejemplo de interfaz del sistema

- **MRTG (Multi Router Traffic Grapher)**. Es un software de referencia para la monitorización y medición de la carga de tráfico de los enlaces de red [MRTG 2015]. MRTG es una herramienta que utiliza el protocolo SNMP para recoger los datos proporcionados por los *routers* y crear representaciones gráficas del tráfico dentro de la red bajo monitorización. Estas gráficas son incluidas en páginas Web para su consulta por el usuario.

  La información utilizada por MRTG incluye los registros históricos de tal modo que también puede generar gráficas que representen la evolución temporal del tráfico de red. Mediante esta herramienta también es posible monitorizar otro tipo de datos proporcionados por SNMP. Para la generación de gráficas temporales utiliza la herramienta RRDtool [RRDtool 2015].

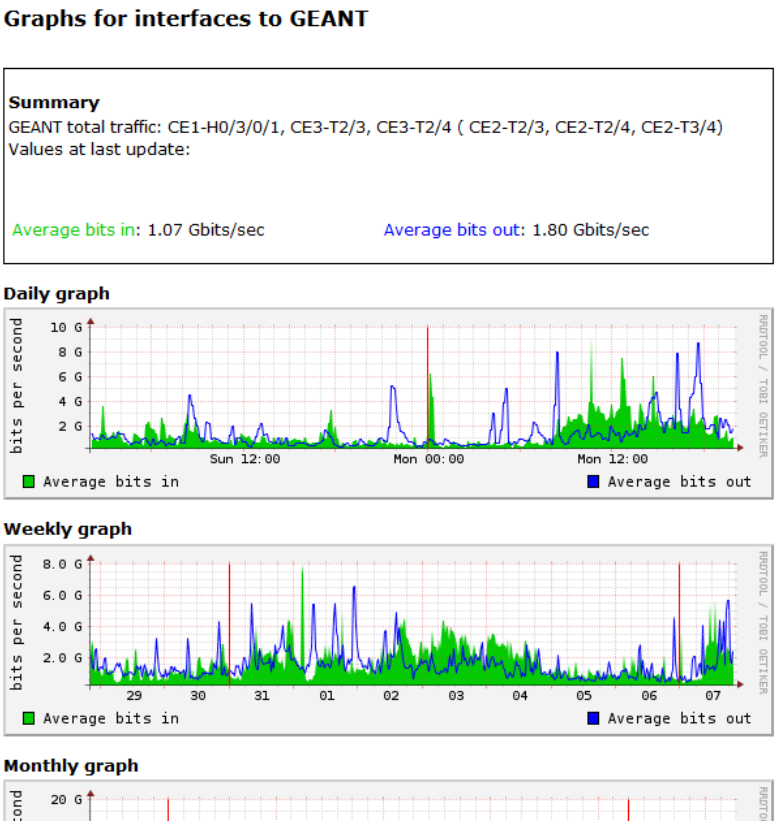


Figura 32. MRTG. Ejemplo de interfaz del sistema

6

Herramienta Web de monitorización inmersiva

6.1. INTRODUCCIÓN

Los sistemas de monitorización y supervisión de equipamientos informáticos y redes de comunicaciones se centran principalmente en interfaces descriptivas, basadas en mostrar una gran cantidad de información en pantalla y en tiempo real. Esto implica disponer de personal técnico dedicado a un trabajo que requiere un alto nivel de atención para tratar de responder en el menor tiempo posible ante la detección de problemas. Además, en CPDs que gestionan un elevado número de activos o en aquellos que se encuentran distribuidos en múltiples localizaciones, encontrar la ubicación física de un determinado activo se complica enormemente.

Uno de los aspectos que permiten mejorar la utilización de una aplicación informática es el diseño de su interfaz. Hacer que la visualización del programa se adapte a las necesidades funcionales del usuario traerá como consecuencia una optimización de su empleo y por lo tanto redundará en una mejor interacción y en un mejor aprovechamiento del tiempo dedicado a la supervisión. Además de los motivos puramente ergonómicos y funcionales es muy importante que sea sencillo interactuar con el sistema y que proporcione capacidades de ayuda para facilitar la detección de anomalías.

Aunque el desarrollo de interfaces avanzadas ya está presente en aplicaciones de escritorio, su utilización en Web estaba muy limitada debido a su elevado consumo

de recursos y a su bajo rendimiento. Actualmente la proliferación de nuevas tecnologías software, unidas a mejoras en las capacidades hardware, ha permitido crear interfaces de este tipo para sistemas basados en Web consiguiendo un rendimiento aceptable. De este modo podemos implementar aplicaciones Web que hagan uso de técnicas de visualización 3D, sistemas inmersivos e interactivos, sistemas de realidad aumentada, etc.

El objetivo principal de este desarrollo será experimentar con nuevas formas de representación visual, más cercanas a la realidad física de los dispositivos monitorizados, para complementar y mejorar la operativa diaria de la supervisión de sistemas informáticos. Esta visualización avanzada permitirá abstraer la información proporcionada por los sistemas de monitorización en una representación 3D del entorno real, teniendo la posibilidad de interactuar con los elementos existentes (equipamientos conectados al sistema), mostrar las incidencias producidas y su localización física en el entorno virtual, y obtener un detalle completo del estado de los activos pulsando sobre el objeto 3D que lo identifica. Además, se dotará a esta aplicación de un módulo de reglas de conocimiento con el objetivo de interpretar las incidencias producidas en el sistema de monitorización para tratar de aportar posibles soluciones al personal técnico.

6.2. INTERFACES DE VISUALIZACIÓN WEB AVANZADA

La utilización de tecnologías avanzadas en el desarrollo de interfaces busca dotar a las aplicaciones informáticas de características gráficas capaces de añadir una nueva dimensión a la interfaz (tridimensionalidad, interactividad, inmersividad, accesibilidad, etc.) con el objetivo de mejorar la interacción con el usuario [Bowman 2005]. La utilización de este tipo de mejoras están vinculadas con las capacidades tecnológicas disponibles y la viabilidad de su empleo dependerá en gran medida de las características hardware y software de los dispositivos.

6.2.1. Tecnologías de visualización Web

Las tecnologías de visualización gráfica para aplicaciones Web han evolucionado de forma significativa, pasando de representaciones simples en lenguaje HTML,

basadas en formato texto y gráficos 2D, hasta la utilización de plugins específicos que permiten realizar complejas representaciones 3D interactivas.

Los sistemas de visualización 3D tradicionales requieren de la existencia de hardware específico capaz de optimizar los resultados de la visualización, proporcionando representaciones muy realistas y con una capacidad de interacción muy elevada (por ejemplo: juegos de ordenador, simuladores, sistemas de realidad virtual y sistemas inmersivos, etc.). Sin embargo, en determinadas aplicaciones no es preciso disponer de altos niveles de detalle o de determinados efectos gráficos, si no que se busca aprovechar la facilidad de interacción tridimensional unida a la independencia hardware del dispositivo desde el que se accede a la aplicación. Este es el caso de las interfaces gráficas 3D en entornos Web en las cuales habrá que tener en cuenta que el rendimiento proporcionado por las aplicaciones debe ser lo suficientemente ajustado como para que sea posible la interacción normal (tiempos de carga reducidos, consumos de memoria y disco ajustados, ausencia de retardos, fluidez, etc.).

Dentro de las tecnologías de visualización Web que nos permiten añadir capacidades de gráficos 3D podemos destacar dos tipos: aquellas que proporcionan un mayor rendimiento al requerir de soporte hardware y las que no precisan de hardware específico. En el primer grupo se incluyen tecnologías como: OpenGL (*Open Graphics Library*) [OpenGL 2015], Microsoft DirectX [DirectX 2015], Java3D [Java3D 2008]. Entre las tecnologías que no requieren de hardware específico figuran: VRML (*Virtual Reality Modeling Language*) [VRML 1997], X3D (eXtensible 3D) [X3D 2015], Adobe Flash [Flash 2015], Microsoft Silverlight [Silverlight 2015].

Para facilitar el desarrollo de aplicaciones con altas prestaciones y requisitos gráficos, principalmente orientados al mercado de los videojuegos y simuladores, disponemos de plataformas que incluyen su propio motor gráfico. Mediante estas herramientas se proporciona al desarrollador un mecanismo para simplificar tareas como el renderizado de gráficos 2D/3D, simulación de eventos físicos, detección de colisiones entre objetos, sonidos, animaciones, inteligencia artificial, gestión de memoria y gráficos, capacidades de *streaming*, etc. Los principales motores gráficos actuales se basan en las especificaciones de OpenGL y DirectX.

Algunos ejemplos de este tipo de plataformas son los siguientes: Unity [Unity 2015], Unreal Engine [Unreal 2015], Crytek CryENGINE [Crytek 2015].

Revisando las tecnologías existentes y sus características principales, se puede considerar que actualmente la plataforma Adobe Flash es una de las opciones más interesantes para el desarrollo de aplicaciones interactivas complejas y con bajos requisitos de hardware [Reinhardt 2009]. Para su utilización es necesario instalar el plugin correspondiente pero debido a su amplia difusión este plugin ya se encuentra incluido en la mayoría de navegadores Web actuales. Por defecto, Adobe Flash en sus últimas versiones proporciona algunas primitivas 3D pero no un motor gráfico suficiente para determinados proyectos complejos [Allen 2008]. Para cubrir estas necesidades se pueden utilizar motores 3D Flash independientes. La principal ventaja que proporcionan los motores 3D para Flash es que permiten al programador combinar el espacio 3D con el espacio 2D de animaciones de Flash, reutilizando todos los controles y objetos (*widgets*) existentes. Entre estos motores 3D podemos destacar por su nivel de difusión:

- **Papervision3D** [Papervision3D 2009]. Librería de clases 3D desarrollada en lenguaje ActionScript. Este motor 3D permite cargar modelos 3D en diferentes formatos: 3DStudio, MD2, Collada y SketchUp. Es una de las librerías más utilizadas y dispone de abundante documentación.

- **Away3D** [Away3D 2013]. Basado originalmente en Papervision3D, ofrece características similares a éste, incluso ambas librerías comparten desarrollos. Su rendimiento es ligeramente inferior al proporcionado por Papervision3D, debido a una implementación más compleja del algoritmo de *Z-Sorting*.

- **Alternativa3D** [Alternativa3D 2012]. Es una librería 3D para Adobe Flash que permite crear juegos y aplicaciones 3D para el navegador Web. Aprovecha las capacidades de las tarjetas gráficas actuales para proporcionar altos rendimientos basados en aceleración por hardware (GPU). Entre los formatos de los modelos 3D que soporta se incluye 3DS Max, disponiendo de un plugin específico para esta conocida herramienta de modelado (Autodesk 3DS Max).

- **O3D** [O3D 2009]. Es un API web que sirve para crear aplicaciones 3D completas e interactivas en el navegador. Se ha desarrollado como un plugin Web experimental para los principales navegadores Web (Internet Explorer, Firefox y Google Chrome). Permite cargar modelos en formato Collada.

El mayor inconveniente que tienen todos los motores 3D sobre Flash es la carencia de aceleración hardware, aunque esto puede ser considerado una gran ventaja para las aplicaciones Web al no tener este tipo de dependencias. Esta carencia repercute en la calidad del renderizado en comparación con el rendimiento alcanzado por OpenGL o DirectX. Como consecuencia los modelos que pueden cargarse deben ser reducidos, con la menor cantidad de polígonos posible, y con texturas sencillas. Además, para aumentar el rendimiento, todos estos motores utilizan algoritmos de *Z-Sorting* (*depth sorting*) muy conservadores, que provocan que muchas veces las caras de los objetos 3D no se ordenen correctamente en el espacio, perdiendo calidad y quitando realismo a las escenas [Foley 1990][Ver Hague 2006].

Como alternativa al uso de la plataforma Adobe Flash también existen otras herramientas que proporcionan opciones de visualización 3D en navegadores Web. Estas herramientas precisan de la instalación de plugins específicos en el navegador para poder reproducir el contenido 3D, este es el caso de O3D de Google [O3D 2009], Cortona3D Viewer para modelos VRML [Cortona3D 2015] y Octaga Player para modelos XSD [Octaga 2015]. El primero de estos ejemplos utiliza JavaScript como lenguaje de programación. Los restantes son plugins comerciales y están orientados a aplicaciones de realidad virtual.

En 2009 se crea la especificación estándar WebGL, para desplegar gráficos en 3D en navegadores Web, permitiendo activar gráficos en 3D acelerados por hardware en páginas Web, sin la necesidad de plugins en cualquier plataforma que soporte OpenGL (está basada en la especificación OpenGL ES 2.0) [WebGL 2015].

Hasta la versión 4 de la especificación HTML no se consideraba el ámbito de desarrollo 3D y por este motivo se necesitaban plugins específicos. Los nuevos elementos y atributos añadidos por el nuevo estándar HTML5 permitirán la inclusión de grandes mejoras en el apartado de la visualización para nuevos

navegadores Web, facilitando el diseño y desarrollo de interfaces Web avanzadas. Por medio del elemento *Canvas* de HTML5, y utilizando las nuevas especificaciones y estándares como WebGL, se conseguirá integrar en el navegador Web las capacidades gráficas 3D con aceleración por hardware (CPU, GPU) cuando el dispositivo soporte la especificación OpenGL.

6.2.2. Plataforma Adobe Flash

La plataforma Adobe Flash (anteriormente conocida como Macromedia Flash o Shockwave Flash) es una plataforma software multimedia utilizada para la creación de gráficos vectoriales, animaciones, aplicaciones y juegos para navegadores Web, aplicaciones de Internet enriquecidas, aplicaciones de escritorio, aplicaciones y juegos para dispositivos móviles [Flash 2015]. Además de capturar acciones de ratón y teclado, mediante Flash se pueden realizar transmisiones por *streaming* de audio y video (utiliza dispositivos de entrada como cámaras y micrófonos).

El motor de visualización del cliente Flash se basa en gráficos vectoriales, que se ajustan fácilmente a cualquier resolución de pantalla, para permitir transiciones fluidas y todo tipo de efectos gráficos dinámicos. De este modo, también se consiguen contenidos con tamaños más reducidos que los basados en gráficos con mapas de bits, optimizando el ancho de banda necesario para su transmisión.

La reproducción de audio en Flash utiliza su soporte nativo del protocolo MP3, permitiendo la sincronización con otros eventos o la reproducción de audio y música de alta calidad. El cliente Flash soporta el estándar H.263 para la reproducción de vídeo, utilizando el formato de vídeo de Flash (FLV).

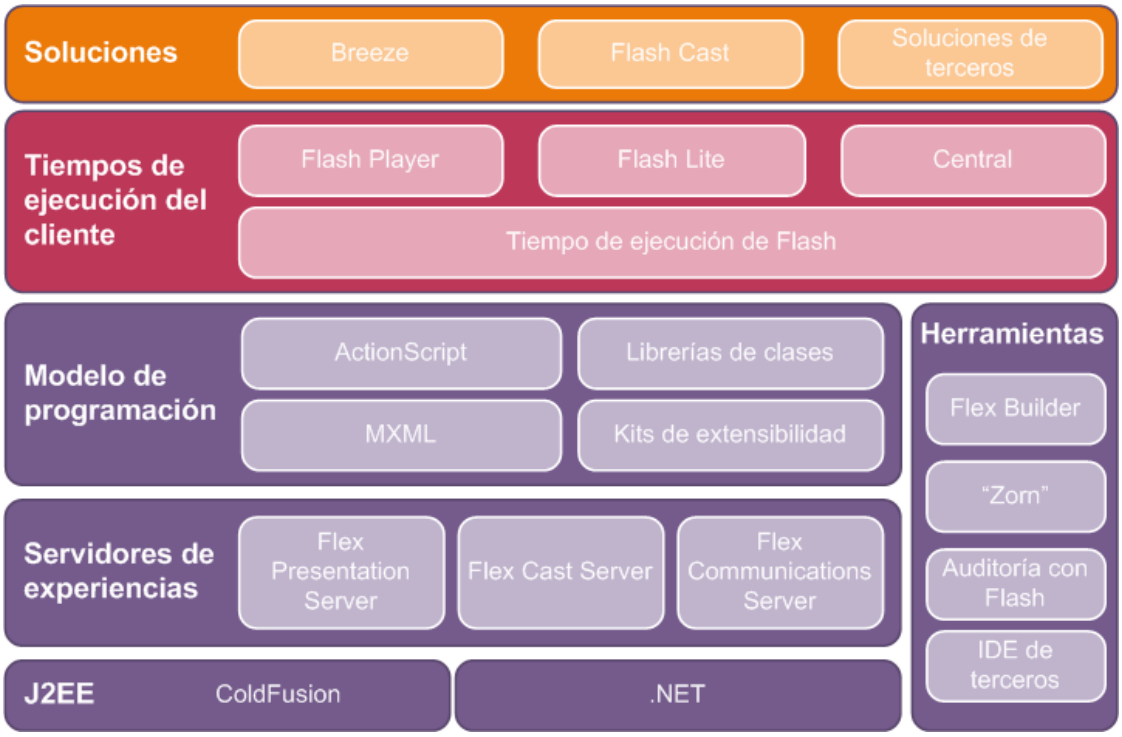


Figura 33. Diagrama de la arquitectura de la plataforma Adobe Flash

La arquitectura que constituye la plataforma Adobe Flash abarca los siguientes elementos [Flash 2015]:

- **Cliente dinámico (Flash Player)**. La aplicación cliente es el núcleo central de esta plataforma. Es el reproductor de Adobe Flash y se distribuye como un plugin del navegador Web. El contenido y las aplicaciones que se ejecutan en el reproductor de Flash funcionan de forma coherente, independientemente de los diferentes sistemas operativos y navegadores utilizados. Las aplicaciones se ejecutan en una zona protegida, segura y basada en dominios, y no es posible acceder libremente a ellas desde otros dominios. Al igual que en otras aplicaciones Web, Flash cuenta con soporte para comunicaciones nativas por medio de estándares HTTP y HTTPS.
- **Modelo de programación**. Incluye un modelo coherente para desarrolladores que combina el lenguaje de programación ActionScript y el lenguaje MXML, ambos creados originalmente por la compañía Macromedia:

- **ActionScript**: es un lenguaje de programación orientado a objetos para el desarrollo de aplicaciones en Flash y sigue las especificaciones ECMAScript (ECMA-262) [ECMA 2015].
- **MXML**: es un lenguaje declarativo basado en la especificación XML para el desarrollo rápido de interfaces de usuario y enlaces a datos. Este lenguaje cumple con el esquema de la especificación XML del W3C [XML 2015].

Para crear una aplicación en Flash, las declaraciones MXML se convierten en código ActionScript y se compilan junto con los archivos ActionScript del programa para generar un archivo SWF (*ShockWave Flash*). Este archivo será ejecutado en el navegador Web por medio del reproductor Flash Player. Como los componentes y la lógica del nivel de presentación se ejecutan dentro del reproductor, en la máquina del usuario en lugar de en el servidor, la aplicación resultante ofrece una rápida respuesta y proporciona una mejor experiencia de usuario.

- **Servidores de experiencias**. Un conjunto de tecnologías de servidor capaces de proporcionar servicios que funcionan con los sistemas centrales existentes. Entre este tipo de tecnologías tenemos J2EE, .NET, o servidores Web estándar (HTML).
- **Herramientas**. La plataforma Adobe Flash dispone de una amplia colección de herramientas para crear contenido interactivo y aplicaciones interactivas. Dentro de esta colección destacan: herramientas de autoría de contenido dinámico e interactivo, herramientas de desarrollo como Adobe Flash Builder y Adobe Flex para la creación de aplicaciones dinámicas de Internet, herramientas de desarrollo integradas en el entorno Eclipse como FDT (*Flexible Development Toolkit*).
- **Soluciones**. Incluye todo tipo de aplicaciones completas sobre la plataforma Adobe Flash.

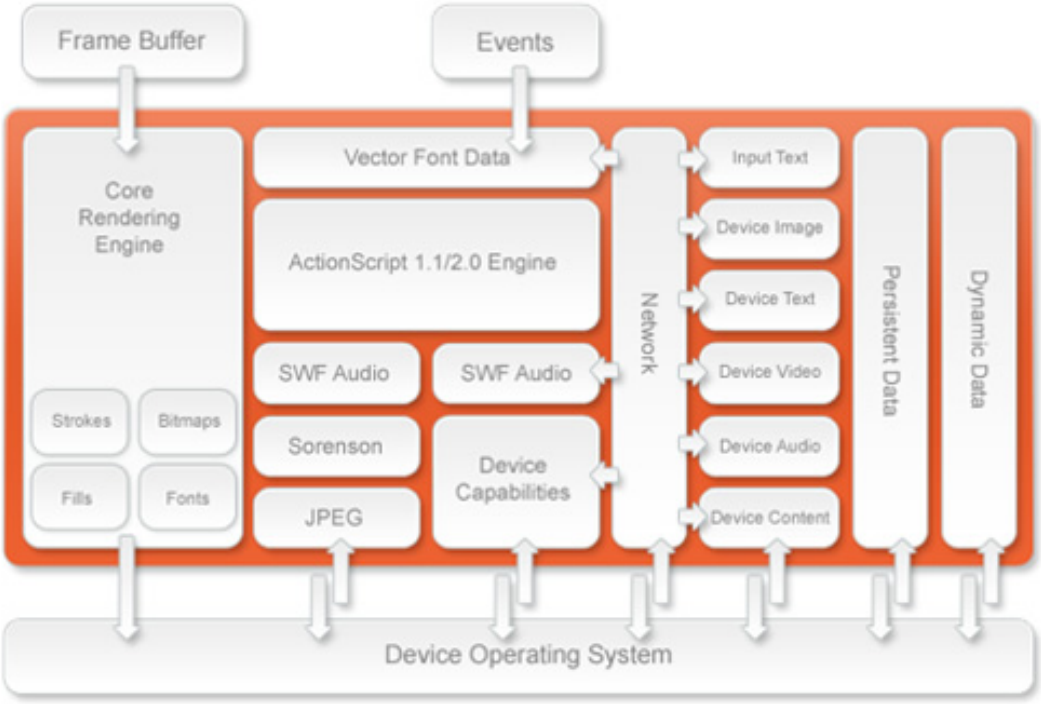


Figura 34. Diagrama general de la arquitectura de cliente Adobe Flash Lite v.3

6.2.3. Papervision3D

El software Papervision3D es un motor gráfico para el renderizado en tiempo real de contenidos 3D interactivos para la plataforma Adobe Flash. Esta librería de clases fue lanzada inicialmente en el año 2005, por el desarrollador Carlos Ulloa, como una de las primeras alternativas para dotar de un motor gráfico 3D a las aplicaciones Flash [Papervision3D 2009].

Aunque en Papervision3D el renderizado de objetos 3D está destinado únicamente a la CPU, sin aprovechar las capacidades específicas de aceleración gráfica por GPU, su nivel de completitud y su elevado rendimiento lo situaron como la mejor alternativa para su utilización en aplicaciones Flash 3D interactivas. Para el desarrollo de aplicaciones con Papervision3D se utiliza, al igual que en Adobe Flash, el lenguaje de programación ActionScript.

Como en la mayor parte de tecnologías de visualización de entornos tridimensionales, en una representación 3D con Papervision se manejan los conceptos básicos siguientes:

- Escena. Representa la composición completa de objetos en el espacio 3D.
- Cámara. Define el punto de vista desde el cual se visualiza la escena. Cambiando la posición o el ajuste de la cámara se cambiará la visualización de la escena.
- Ventana de visualización (*viewport*). Muestra la ventana rectangular correspondiente a la visión de la cámara en cada momento. De este modo, según el ajuste de la cámara se visualizará la escena completa o sólo una parte determinada.
- Luces. Permiten iluminar la escena y pueden ser direccionales o focales.
- Objetos 3D. Representan figuras 3D que pueden ser colocadas en cualquier posición del espacio 3D y que también pueden ser escaladas o rotadas en cualquiera de los ejes del espacio de coordenadas cartesianas (x,y,z).
- Material. Es una textura que puede ser aplicada sobre un objeto 3D para definir su apariencia. Si un objeto 3D no tiene ningún material aplicado, pasará a ser un objeto invisible en la escena. Existen múltiples tipos de materiales utilizables como pueden ser colores simples, gráficos de mapas de bits, o incluso videos.
- Motor de renderizado. Se encarga de generar la ventana de visualización registrada por la cámara sobre la información de la escena. Esta tarea supone un trabajo intensivo para la CPU del equipo, ya que necesita representar cada objeto con su material correspondiente, recalculando y posicionando los elementos dentro de la escena para generar la ventana de visualización.

6.3. OBJETIVOS

El objetivo principal será la creación de una herramienta Web de apoyo a la monitorización de equipamientos informáticos dentro de una red de datos, que muestre una representación real de los dispositivos monitorizados dentro del entorno en el que se encuentran ubicados. Para ello, se realizará un modelado en 3D de la representación del espacio físico del edificio (plantas y salas) y de los equipamientos disponibles (armarios de datos y de comunicaciones, equipos y

dispositivos informáticos, etc.), permitiendo al usuario visualizar e interactuar con este espacio virtual.

Con este desarrollo se busca estudiar nuevas formas de visualización que complementen a las herramientas de monitorización existentes. Así, se plantea una herramienta de ayuda a la supervisión del estado de los equipamientos informáticos conectados al sistema, facilitando la detección de anomalías y la localización de la posición física de los activos afectados utilizando una representación virtual del entorno.

Esta aplicación de visualización se conectará y obtendrá la información sobre los parámetros de los dispositivos por medio de la herramienta de supervisión de sistemas utilizada (Nagios, Zabbix, etc.). La información será convenientemente procesada para ajustarse a las necesidades de la aplicación e influirá en la representación virtual del entorno en tiempo real.

Se ha planteado que esta herramienta pueda ser adaptada a cualquier entorno real pero teniendo en cuenta la posibilidad de aplicar el desarrollo sobre un contexto controlado y con una amplia información disponible, se ha seleccionado como entorno de trabajo del sistema de supervisión a la Facultad de Informática de la UDC (FIC). Así, se partirá de los planos arquitectónicos del edificio para crear el modelo adecuado del espacio virtual y que dará lugar a una interfaz 3D que permita interactuar con las plantas y salas del edificio para obtener la información correspondiente al estado de los dispositivos bajo supervisión.

La representación virtual de los equipamientos se realizará mediante objetos 3D basados en los dispositivos físicos reales. Cada uno de estos objetos tendrá unas características determinadas en función del dispositivo que representa y que afectarán en su visualización e interacción dentro de la interfaz.

En el diseño de la interfaz se ha intentado centrar la visualización en la representación virtual de un determinado espacio pero también se ha incluido un panel de control que permite acceder de forma rápida a los equipamientos existentes en cada espacio. De este modo, al navegar por una determinada sala se podrá consultar, en una estructura jerárquica, el listado actualizado de salas y dispositivos. Además, utilizando esta misma estructura también se podrá indicar

de forma visual la existencia de incidencias producidas en los equipamientos bajo supervisión y la sala en la que están ubicados.

Aunque esta herramienta se centra en intentar abstraer la gran cantidad de parámetros controlados por las herramientas de supervisión de sistemas mediante una visualización avanzada, también será posible consultar la información detallada de un determinado dispositivo desde la aplicación. De este modo, el personal técnico puede consultar todos los parámetros bajo monitorización.

Los activos bajo supervisión pueden ser dispositivos físicos o virtuales, de tal modo que se pueda monitorizar la información de un equipo servidor o de las máquinas virtuales existentes en un equipo servidor. Desde el punto de vista de la interfaz desarrollada, esto supone que se tendrá en cuenta la posibilidad de que un determinado equipamiento pueda disponer de máquinas virtualizadas y por lo tanto se tendrá que mostrar su información adecuadamente.

Ante fallos en la monitorización de un dispositivo, tanto por una pérdida de conexión con el sistema supervisor como por un fallo en alguno de los servicios ofrecidos, se notificará al usuario modificando parámetros visuales como el color del objeto que lo representa o mediante la presentación de un mensaje de alerta en pantalla. Utilizando un sistema de reglas de conocimiento se puede llegar a proponer posibles soluciones según la naturaleza del problema.

6.4. DESARROLLO

Para llegar a implementar la herramienta de supervisión descrita en apartados anteriores, en la fase de análisis se ha realizado un estudio previo sobre las tecnologías existentes, tanto a nivel de sistemas de monitorización como de herramientas de desarrollo más adecuadas para su implementación.

Teniendo en cuenta que la información a visualizar provendrá de un sistema de telemonitorización existente en el CPD, es importante conseguir que esta herramienta sea diseñada para ser independiente de este tipo de sistemas. Así, se podría adaptar y configurar de forma simple para poder trabajar con cualquier sistema de este tipo. La principal limitación que nos encontramos es disponer de algún mecanismo (plugin, servicio Web, API de desarrollo, etc.) capaz de proporcionar la información obtenida por el sistema de monitorización.

Finalmente, las pruebas de experimentación fueron desarrolladas para el sistema de monitorización Nagios, debido a que es uno de los más ampliamente utilizados por su nivel de difusión, documentación y bibliografía, plugins existentes, y la posibilidad de implementar aplicaciones propias.

La elección de la plataforma de desarrollo estaba condicionada en gran medida por la necesidad de conseguir una interfaz avanzada con rendimientos gráficos aceptables y que pudiese ser utilizada en una amplia gama de navegadores Web. Entre las opciones posibles se optó por emplear Adobe Flash, por ser un estándar de hecho pensado para el desarrollo de aplicaciones enriquecidas en entornos Web y por disponer de soporte para los navegadores más utilizados. Además, como ya se ha indicado en apartados anteriores, la existencia de una gran variedad de motores 3D que utilizan la tecnología Flash con buenos resultados reforzaba esta elección. Adobe Flash utiliza como lenguaje de programación ActionScript y será este lenguaje el que se utilizará para la implementación de la herramienta. Mediante ActionScript también se pueden implementar todos los componentes bidimensionales de la aplicación (menús, listas, paneles, botones, etc.), lo cual facilitará su reutilización, su posicionamiento dinámico y su capacidad de interactividad.

Una vez definida la tecnología de desarrollo se seleccionó la librería Papervision3D como el motor 3D más adecuado para la implementación de la herramienta debido a su alto rendimiento, elevada difusión y amplia documentación disponible.

En estudios iniciales, se realizaron una serie de pruebas experimentales con la tecnología seleccionada para comprobar el potencial y limitaciones que proporcionaba para la implementación de una interface Web 3D interactiva. Principalmente, se centraron las pruebas en comprobar las posibilidades y rendimiento ofrecidos por la librería de representación Papervision3D, tanto para el modelo correspondiente al espacio virtual como para los objetos existentes en la escena (armarios, servidores, computadoras, etc.). Estas pruebas incluían funcionalidades básicas para la herramienta como la interactividad con los objetos o el desplazamiento de la cámara siguiendo el movimiento del usuario. También se contempló la capacidad para poder incluir y situar objetos dentro del entorno y dotarlos de interactividad ante eventos del usuario. Este tipo de objetos serán los

que representen a los elementos del sistema de telemonitorización que se encuentran bajo supervisión.

Uno de los primeros pasos consistió en convertir los modelos 3D del edificio de la FIC, en formato Autodesk Autocad y Autodesk Maya, en un formato que pudiese ser utilizado por una aplicación 3D interactiva con Papervision3D. Para ello, se optó por un formato estándar denominado COLLADA (*COLLAborative Design Activity*), creado y gestionado por el consorcio Khronos Group y posteriormente adoptado como una especificación ISO (ISO/PAS 17506) [COLLADA 2012]. COLLADA define un esquema estándar XML para facilitar el intercambio digital de activos 3D entre aplicaciones. Normalmente los documentos que siguen este formato se identifican con ficheros que tienen la extensión *DAE* (*Digital Asset Exchange*).

Existen herramientas que facilitan la conversión de formatos gráficos para obtener un archivo en formato COLLADA y que se han utilizado para conseguir adecuar el modelo arquitectónico a la aplicación desarrollada. Concretamente, se han utilizado las siguientes herramientas:

- **ColladaMaya**. Es una herramienta integrada en el software de modelado y animación Autodesk Maya. Su función es exportar los modelos creados en Maya al formato COLLADA [ColladaMaya 2009]. Mediante esta herramienta se han exportado los modelos originales eliminando cámaras, luces, y sombras. Además, se han convertido los polígonos en triángulos ya que mejora el rendimiento en Papervision3D.
- **Polygon Cruncher**. Es una plugin para Autodesk Maya que permite realizar una simplificación y optimización de la escena 3D sin afectar a su apariencia [PolygonCruncher 2009]. Durante las primeras pruebas del proyecto se comprobó que el modelo 3D utilizado era muy complejo, del orden de los 25000 triángulos por cada planta del edificio. Con este software se consiguió reducir la complejidad del modelo 3D, mediante una reducción automática de polígonos, permitiendo su carga con la fluidez necesaria para realizar las pruebas sobre él.

Después de realizar pruebas de carga con los modelos 3D obtenidos se pudo comprobar que era necesario reducir considerablemente el número de polígonos

ya que de lo contrario el rendimiento de los renderizados se degradaba progresivamente, provocando una disminución importante de la frecuencia a la que se muestran los fotogramas (FPS-*Frames Per Second*). Para mejorar este rendimiento también será preciso disminuir la resolución de las texturas utilizadas. Hay que tener en cuenta que aunque un objeto sea creado con pocos triángulos, si la imagen de la textura posee una alta resolución, el funcionamiento del motor de Papervision3D lo renderizará generando dinámicamente más triángulos para adaptarse a las dimensiones de la textura [Tondeur 2009]. En algunos casos, también se comprobó que los efectos de iluminación y sombreado con Papervision3D no producían los resultados esperados y por ese motivo era preferible generar estos efectos directamente en las texturas de los objetos.

Las pruebas realizadas mostraron que la gestión del recolector de basura de Adobe Flash utilizando Papervision3D es bastante ineficiente. Esto es debido a una característica propia del motor 3D: durante la carga de cualquier modelo 3D se registran sus materiales (texturas) en una colección interna de datos y cuando posteriormente se destruye el objeto las referencias a sus materiales no se eliminan, evitando que sean marcados para recolección y conservándose innecesariamente en memoria [Lively 2010]. Por tanto, es necesario implementar los mecanismos que permitan mejorar esta gestión de los objetos, eliminado todas sus referencias internas y destruyendo todos los materiales y estructuras de almacenamiento.

El diseño de la interfaz se centró en visualizar el modelo 3D del edificio de la FIC y en gestionar la navegabilidad del usuario por este espacio virtual. La navegación se basa en el desplazamiento de la cámara en todas las direcciones del plano horizontal, teniendo en cuenta los límites establecidos en la geometría de los objetos, incluyendo además la posibilidad de cambiar el ángulo de visión del observador.

En cada representación virtual de un espacio real se incluirán los objetos 3D que se correspondan con los equipamientos reales bajo supervisión (armarios, servidores, etc.) y se los dotará de funciones interactivas, incluyendo su relación con el sistema de monitorización.

En toda herramienta dedicada a la monitorización de sistemas es importante que se visualice en tiempo real el estado de los equipamientos y que se resalte de alguna forma cualquier incidencia que se produzca para intentar solventar el problema con la mayor rapidez posible. La aplicación desarrollada tratará estas incidencias mediante cambios en la interfaz gráfica, con el propósito de destacar el estado de cada elemento en tiempo real. Además, se definirán mensajes de alerta, con diferentes niveles de criticidad, para mantener informado en todo momento al personal técnico ante posibles fallos hardware/software o ante fallos en el acceso por red a un determinado activo.

Otra de las funciones de la herramienta será intentar orientar al usuario ante las incidencias que se produzcan en los activos para tratar de volver a una situación estable lo más rápidamente posible. Así, se puede incluir un módulo de ayuda a la toma de decisiones que defina un conjunto de reglas de conocimiento para realizar sugerencias al usuario en función de los parámetros bajo supervisión y sus posibles causas. Un ejemplo de este tipo sería el provocado por fallos de conexión con un conjunto de activos debido a un fallo en el dispositivo de red al que están conectados (por ejemplo: la caída de un *router* provocará un error de conexión en todos los activos que tenga conectados). Para proporcionar estas funcionalidades se han incluido opciones en la interfaz como paneles informativos con el estado resumido de un conjunto de activos y parámetros que permitan conocer el esquema de conexión entre los equipamientos y los dispositivos de red.

El desarrollo en Adobe Flash se centra en la definición de una clase como punto de partida, será una subclase del componente *Sprite*, que representa la interfaz gráfica del usuario y que contiene todo el control sobre las acciones interactivas. En nuestro caso, la clase principal que actúa como elemento de inicio se hereda de *PaperBase*, una clase que proporciona la librería de Papervision3D, que se encarga de inicializar el motor de renderizado. Esta clase principal define y mantiene el estado de dos utilidades:

- *Controls*: es la clase que controla el comportamiento de la cámara, respondiendo a movimientos del ratón o a pulsaciones del teclado, escuchándolos de la clase principal. La clase *Bounds* define los límites espaciales de movimiento de la cámara.

- *SceneManager*: se encarga de crear la escena a representar. Cuando se inicia, crea un objeto *AbstractFloor*, que cargará el modelo 3D de la planta. Además, esta clase es la encargada de controlar los cambios de planta, realizando las tareas de destrucción de la planta actual y ejecutando la creación de la nueva planta a visualizar.

Al inicializar una planta (*AbstractFloor*), además de cargar el modelo 3D correspondiente a ella, se crean los objetos a monitorizar (*Host3D*) y se posicionan en el escenario. A su vez, también se definen los límites de navegación correspondientes a su geometría.

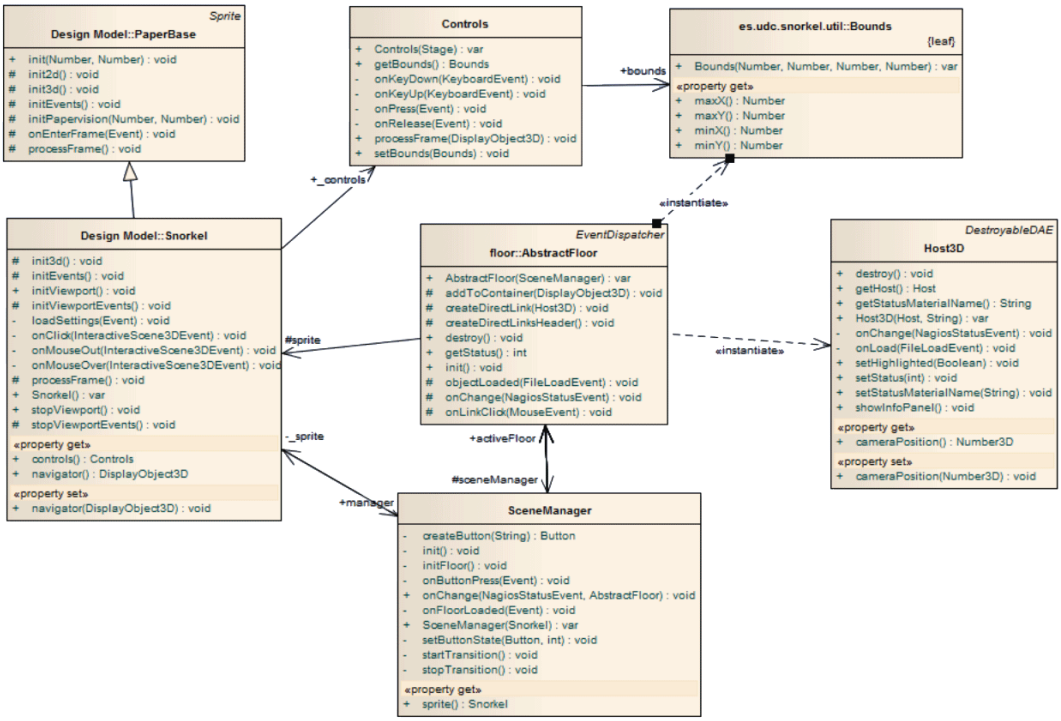


Figura 35. Diagrama de clases del funcionamiento básico de la vista

Los objetos interactivos, correspondientes a los sistemas bajo supervisión, están modelados por la clase *Host3D*. Esta clase hereda de la clase *DAE*, que es una clase de la librería Papervision3D que permite cargar los objetos en formato Collada DAE. Para visualizar la información de los activos, proveniente del sistema de monitorización, se utilizará la clase *InfoPanel* que se encarga de crear un panel tridimensional en la vista del usuario con los datos correspondientes al activo seleccionado.

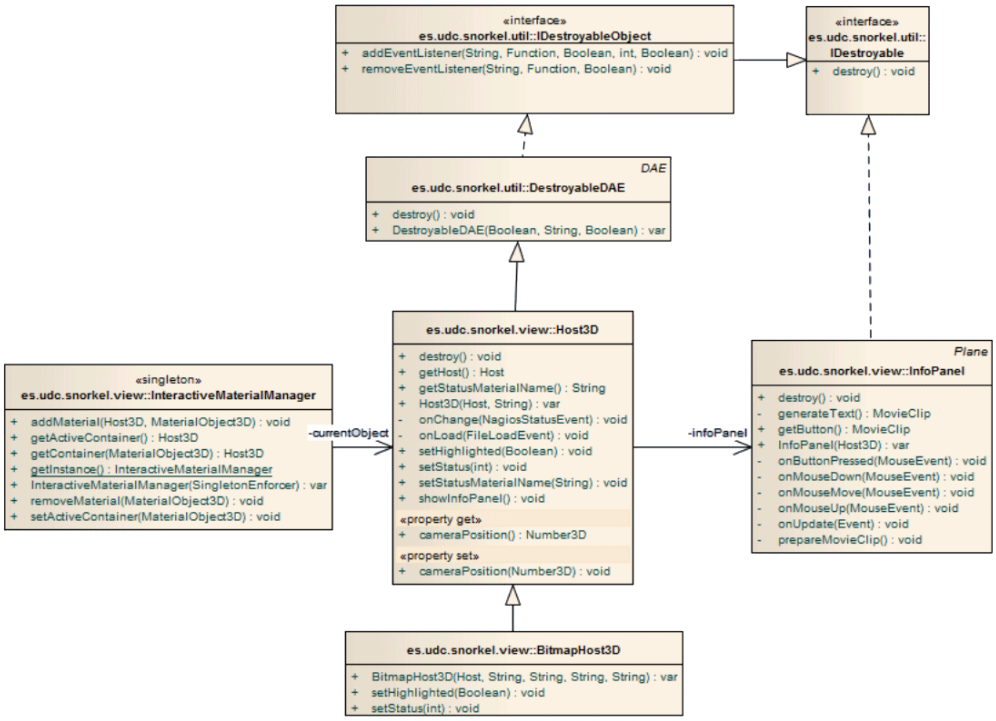


Figura 36. Diagrama de clases del modelado de los objetos interactivos

Para facilitar la detección de incidencias en un determinado espacio se realizan agrupaciones de grupos de activos. También existirá una agrupación lógica entre todos los activos incluidos en un armario (*rack*), pero en este caso también será necesario incluir la información técnica del armario y la posición física que ocupa cada elemento. La jerarquía de agrupaciones y activos se almacena en un fichero de configuración en formato XML, donde también se incluirán las posiciones de los componentes en el espacio físico.

El acceso a los datos proporcionado por el sistema de monitorización se realiza mediante una clase que define la fuente de datos, en este caso el sistema Nagios. Periódicamente, se establecerá una conexión con el servicio Web XML de Nagios para obtener los datos de los equipamientos. El período de tiempo entre conexiones es un parámetro configurable, permitiendo que sea el personal técnico el encargado de establecer su valor. Los datos recibidos serán procesados para crear los objetos de transferencia de datos que contienen la información sobre el estado de un activo y de sus servicios bajo supervisión. Cuando los objetos están disponibles se notifica, utilizando el API de Flash, a los escuchadores (*listeners*) para que se actualice el estado de los activos y se refleje en la vista.

La utilización de agrupaciones entre activos facilita añadir en la interfaz un menú con estructura de árbol donde seguir la jerarquía existente en un determinado espacio. Así, se utilizará este menú para:

- Mostrar la distribución completa de los activos bajo supervisión en cada escenario. Para ayudar al usuario en la tarea de identificación de incidencias se indicará mediante códigos de colores el estado de los activos incluidos en esta estructura jerárquica. También se resaltará la rama correspondiente al escenario que se está observando.
- Mostrar la lista de activos pertenecientes al espacio por el que se está navegando. Además, al pulsar sobre un activo de esta lista la cámara se dirigirá automáticamente hasta una posición centrada en el objeto 3D que lo representa. De este modo se mejora el acceso a la representación 3D de un equipamiento determinado.

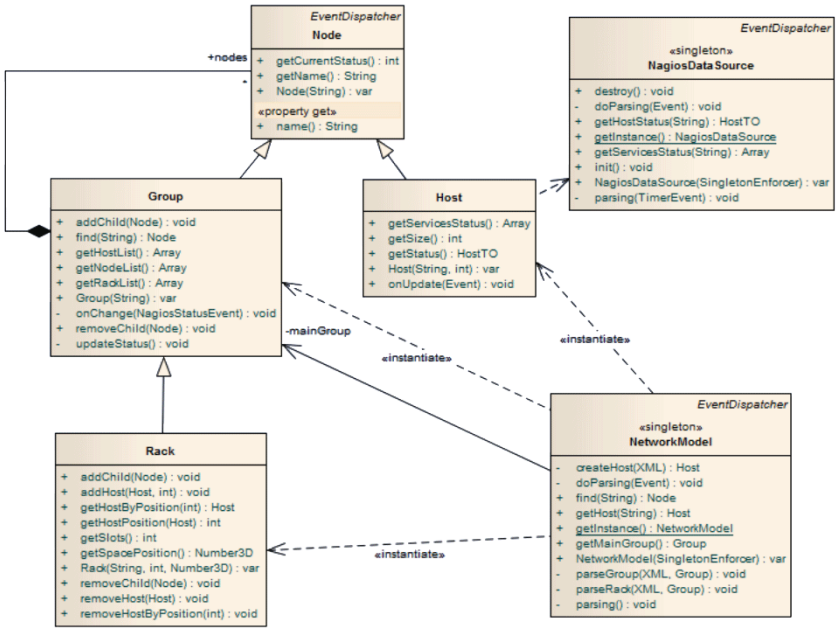


Figura 37. Diagrama de clases de la capa modelo

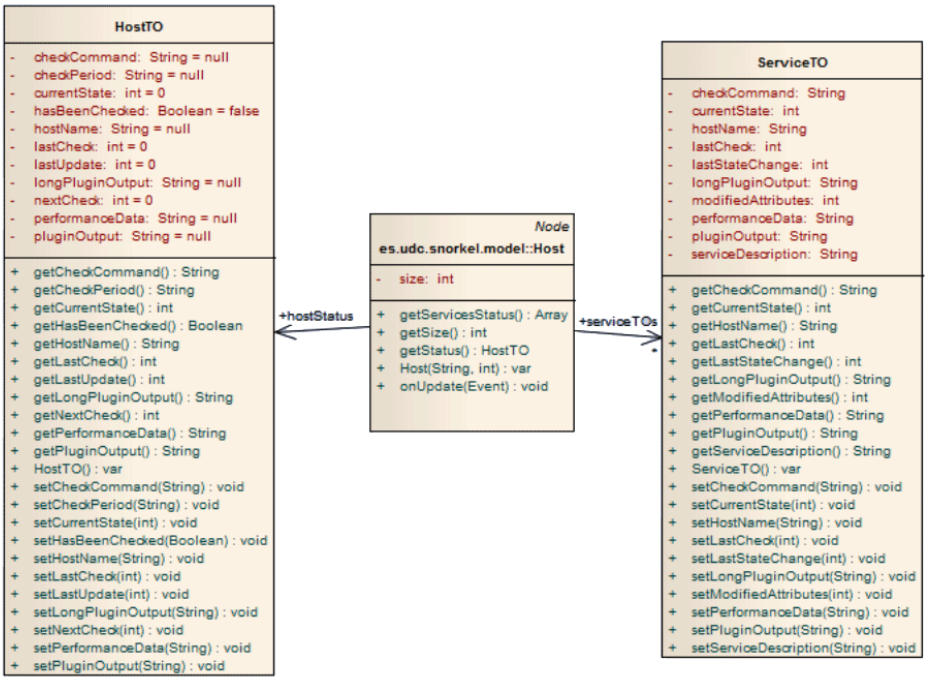


Figura 38. Diagrama de clases del modelo de datos

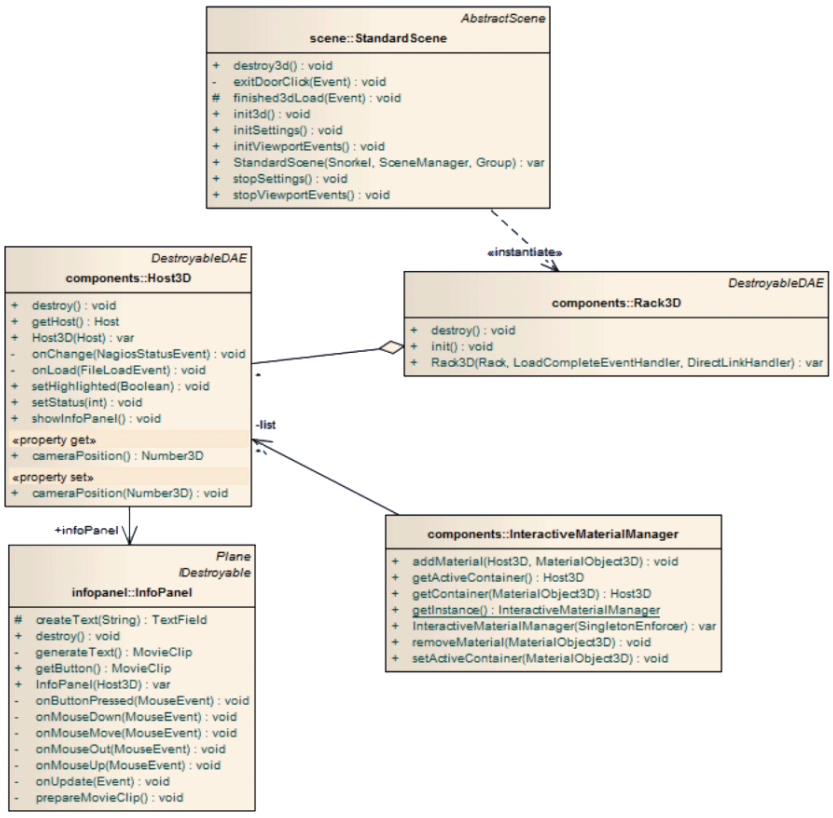


Figura 39. Diagrama de clases de los objetos 3D

Además de los plugins estándar que acompañan a la distribución de Nagios, se ha utilizado el plugin denominado *Status XML Generator*, que puede encontrarse en el

portal Nagios Exchange [Nagios-plugins 2014]. Este plugin actúa como un servicio Web que proporciona la información de estado de Nagios en formato XML.

Para implementar la interacción con el sistema Nagios se ha creado un plugin específico que se encarga de obtener la información relativa al hardware de cada máquina. Este plugin realizará un análisis de la información obtenida en formato XML para su utilización posterior, y en su implementación se han seguido los criterios de diseño marcados en la guía de estilos de este software de monitorización [Nagios-plugins 2014]. La información detallada del hardware de los activos se obtiene por medio de llamadas a comandos estándar del sistema operativo. Este es el caso de utilidades como *lshw* (*list hardware*) que permite obtener información sobre la configuración y tamaño de la memoria del sistema, firmware, configuración de la placa base, velocidad y descripción del procesador, etc. Funciona en máquinas con arquitectura x86 y x64 compatibles con DMI (*Desktop Management Interface*).

Así, la salida del plugin desarrollado indicará el estado del servicio en formato texto, delimitando los valores obtenidos por un carácter separador. Se devolverá una línea para cada dispositivo bajo supervisión y en cada línea figurarán los valores actuales de cada uno de los parámetros definidos en el formato de salida.

Los parámetros que conforman cada línea de salida de este plugin se centran en la información sobre el procesador, la memoria, y las particiones de disco. Sin embargo, se podría extender la funcionalidad de este plugin de forma simple a otros campos que se consideren significativos para los equipamientos bajo supervisión. Concretamente, los parámetros que se están registrando son los siguientes:

- *processor_desc*: descripción del procesador (modelo, fabricante, etc.).
- *processor_freq*: frecuencia máxima de trabajo del procesador en Hz.
- *memory_desc*: descripción del tipo de memoria RAM.
- *memory_size*: capacidad de la memoria en bytes.
- *memory_clock*: frecuencia de reloj de la memoria en Hz.
- *disk_dev*: nombre del dispositivo de almacenamiento.

- *disk_mount_point*: punto de montaje del dispositivo de almacenamiento.
- *disk_desc*: descripción del sistema de ficheros utilizado por el dispositivo de almacenamiento (ejemplos: ext3, ext4, swap, ntfs).
- *disk_capacity*: capacidad del espacio de almacenamiento en bytes.

Este plugin permitirá obtener los datos tanto de máquinas físicas como de máquinas virtuales. Este es un detalle importante ya que en muchos casos los servicios proporcionados por un CPD provienen de activos virtualizados. A diferencia de las máquinas físicas, donde no es habitual que se produzcan cambios en el hardware, en las máquinas virtuales es bastante común que se asignen nuevos recursos hardware para cubrir las necesidades crecientes de un servicio. Esto hace que sea necesario indicar todos los parámetros definidos en el plugin en cada llamada al sistema de monitorización.

El sistema de monitorización Nagios será el encargado de obtener la información proporcionada por los agentes instalados en los equipamientos bajo supervisión. Estos agentes incluirán el plugin desarrollado para adquirir y procesar la información sobre el estado de un activo. Utilizando estos agentes o por medio de llamadas remotas, el servidor central Nagios obtendrá y proporcionará la información que será utilizada por la herramienta de visualización inmersiva.

El desarrollo de los agentes Nagios puede ser ampliado utilizando un sistema de agentes inteligentes multiplataforma como el definido en esta tesis doctoral en un capítulo anterior. Así, los agentes implementados en lenguaje de programación Java, siguiendo las especificaciones FIPA, pueden encargarse de recoger la información proporcionada por los dispositivos monitorizados. Además, utilizando el módulo de conocimiento basado en reglas se puede procesar e interpretar la información obtenida para su envío a la herramienta de telemonitorización y visualización. El mecanismo de reglas de conocimiento sigue la implementación ya comentada anteriormente y está implementado con el motor de reglas JESS.

Como ejemplos de reglas de decisión genéricas dentro del sistema tendremos:

- Detección de problemas de acceso al activo provocados por cortes de red o por cortafuegos intermedios.

- Detección de errores en el funcionamiento de los servicios proporcionados por un activo, provocados por problemas internos (caídas de servicios, cortafuegos internos, etc.).
- Detección de fallos hardware en los dispositivos mediante la recepción de señales del sistema operativo.
- Detección de sobrecargas en la utilización de los componentes del activo (memoria, CPU, GPU, etc.), provocados por procesos que consumen más recursos de los aceptables dentro de los umbrales de funcionamiento definidos.

Dentro de la herramienta de visualización 3D, se ha definido un módulo de ayuda ante fallos que se encarga de escuchar las notificaciones de estado de todos los activos del sistema y de visualizar adecuadamente en la interfaz las incidencias producidas. Se utiliza un sistema de reglas para interpretar la información sobre el equipamiento en el que se produce el estado de alarma y así proponer una acción para resolver la incidencia. En el caso de obtener una acción posible, se crearía un nuevo mensaje *Advice* y se notificaría a los observadores. Mediante el componente visual *FailureAdvisorViewer* se escuchan estos cambios y se presentan las sugerencias en la interfaz de la herramienta.

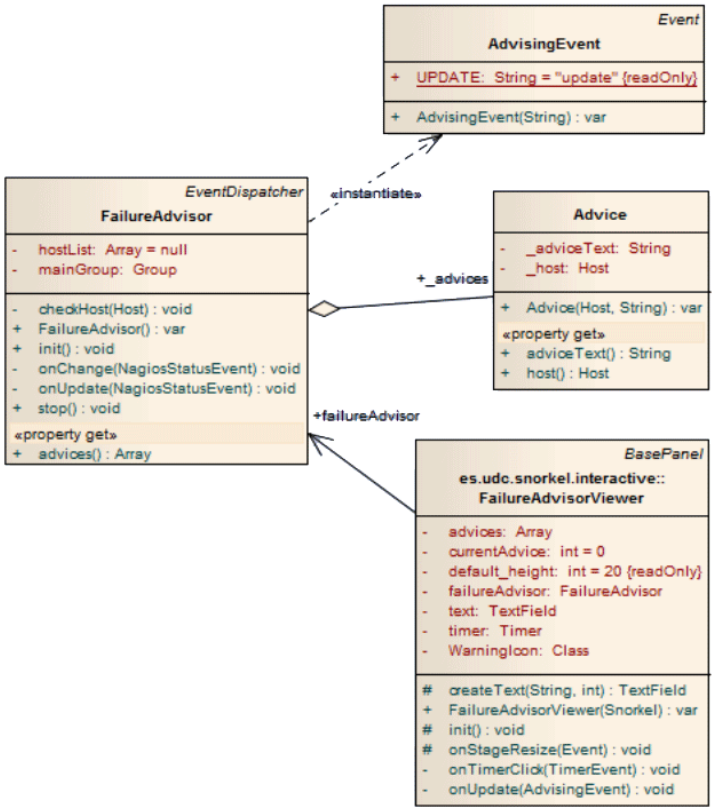


Figura 40. Diagrama de clases del módulo de ayuda a la toma de decisiones

Con las funcionalidades descritas, este sistema de visualización 3D interactiva trata de simplificar la gestión, facilitando la detección y ubicación concreta de la incidencia. Para ello, se facilita la identificación visual por múltiples medios: cambios de color y tamaño de los objetos 3D, mensajes descriptivos de la notificación y posible resolución, enlaces directos a la visualización 3D de los espacios virtuales y de los activos afectados, avisos sonoros en caso necesario, acceso a información detallada de los datos a través del servidor de telemonitorización Nagios, etc.

En las siguientes figuras se puede observar la apariencia y funcionamiento final de esta herramienta. Así, podremos ver el modelo 3D global del edificio para la selección de las plantas y al mismo tiempo la estructura jerárquica de selección rápida. Se pueden observar también la visualización de alertas producidas en el sistema de telemonitorización, mediante notificaciones y cambios de color en la interfaz gráfica.

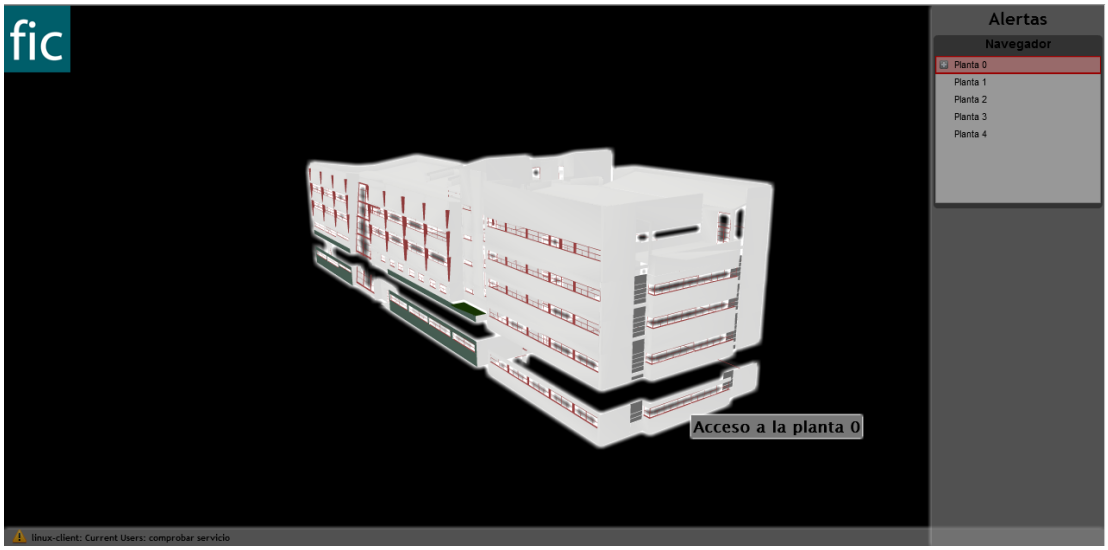


Figura 41. Interface principal para la selección de plantas del edificio

Posteriormente, al seleccionar una de las plantas del edificio se accede al plano interactivo de los espacios que lo componen. En este plano, la estructura jerárquica de equipamientos se centra en los activos presentes en la planta actual del edificio, facilitando la localización y acceso rápido.

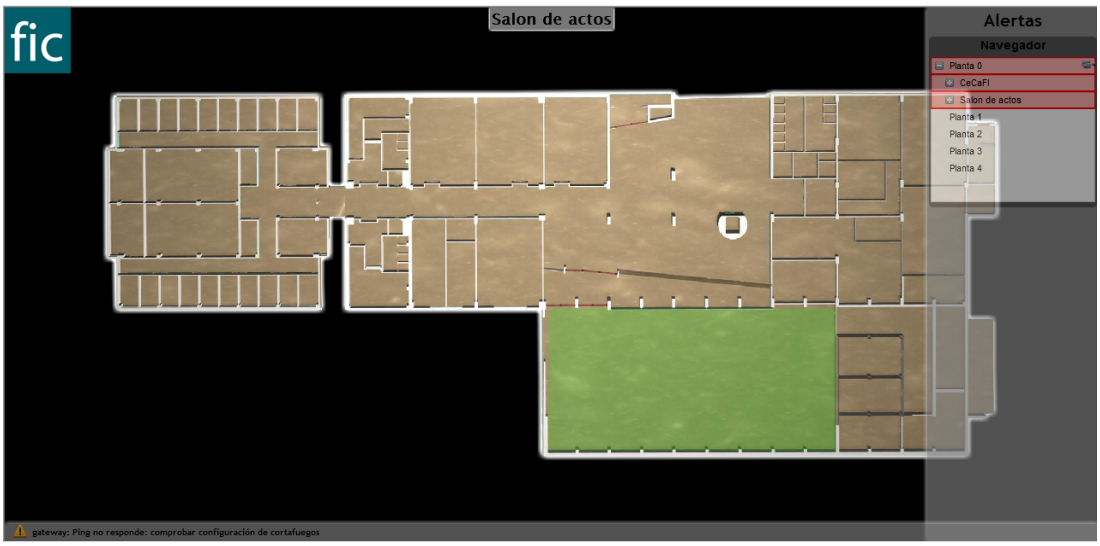


Figura 42. Visualización interactiva para la selección de espacios dentro de la planta actual del edificio

A continuación se puede acceder a un determinado espacio dentro de la planta actual del edificio. Ahora la interfaz visualizará una representación 3D interactiva en la cual se muestran los equipamientos incluidos en la sala. De forma similar a los casos anteriores, el panel con la estructura jerárquica muestra una información

más específica y se compone por los activos disponibles, incluyendo los distintos servidores que están integrados en los armarios (*rack*).

Figura 43. Visualización 3D interactiva de una sala y sus equipamientos

Finalmente, al seleccionar un determinado activo se accede a su información básica y de monitorización en tiempo real. Utilizando el panel jerárquico, al seleccionar uno de los activos, la cámara se posicionará automáticamente ante su modelo 3D para facilitar el acceso a su información sin tener que desplazarse por el entorno virtual hasta su localización.

Figura 44. Visualización de paneles informativos con la descripción y estado de los equipamientos seleccionados

En caso necesario, mediante el botón correspondiente del panel informativo se puede acceder a la información completa e histórica de la monitorización del activo.

6.5. CONCLUSIONES

En este desarrollo se ha implementado una herramienta Web de apoyo a los sistemas de telemonitorización de infraestructuras informáticas con el objetivo de estudiar nuevas tecnologías de visualización avanzada para asistir al personal técnico en la realización de sus tareas. Concretamente, se han probado técnicas de representación 3D interactivas e inmersivas para simular el espacio físico y los equipamientos bajo supervisión. De este modo, se aporta un nuevo mecanismo que se muestra válido para interactuar con los activos, facilitando la localización real de un dispositivo dentro del CPD. Además, la vinculación entre el sistema de supervisión y los objetos 3D, que representan a los activos físicos, permite acceder a su estado en tiempo real y comprobar de forma visual la aparición de incidencias mediante notificaciones y cambios en los objetos de la interfaz gráfica.

La monitorización de los activos se realiza por medio de la herramienta Nagios y los datos adquiridos son procesados y visualizados de forma adecuada antes de ser presentados dentro del entorno 3D. Dado que estas herramientas proporcionan una gran cantidad de datos de los activos monitorizados, es preciso realizar esta etapa de procesado previa pero sin perder la información relevante.

Además, se ha implementado un sistema de ayuda a la toma de decisiones basado en reglas con el objetivo de proporcionar al administrador de sistemas un mecanismo de soporte para la resolución de fallos en los activos o para detectar posibles incidencias en el funcionamiento de los mismos. Utilizando este sistema se podrá sugerir al personal técnico las posibles actuaciones a realizar cuando se produzca una incidencia en un activo bajo monitorización.

Una de las características destacables de la interfaz 3D implementada es su capacidad de generalización. Así, se pueden realizar tareas como añadir, eliminar y reposicionar activos dentro de una sala de forma simple a través de ficheros de configuración XML. De la misma forma también se pueden dar de alta nuevos

espacios físicos o modificar los ya existentes para adaptarse al entorno real a monitorizar.

La herramienta realizada fue implementada bajo la plataforma Adobe Flash, ya que en el momento de su desarrollo, era la tecnología más adecuada para el desarrollo de aplicaciones Web 3D interactivas debido a su amplia generalización multiplataforma y a precisar de unos bajos requisitos hardware y software. En las pruebas realizadas se detectaron algunas limitaciones que obligaron a reducir la complejidad y nivel de detalle de los objetos 3D utilizados. Sin embargo, estas limitaciones no afectaban el resultado final y la herramienta cumplía con el objetivo planteado inicialmente, consiguiendo tiempos de carga reducidos.

Actualmente, año 2015, los graves problemas de seguridad que se descubren, casi a diario, en el plugin de Adobe Flash han provocado la necesidad de realizar actualizaciones urgentes de forma continuada en los navegadores Web, llegando incluso a no recomendarse su uso por parte de algunos fabricantes. A esto hay que añadir el soporte de la especificación HTML5 en la mayor parte de los navegadores actuales, permitiendo desarrollar aplicaciones Web avanzadas sin necesidad de recurrir a plugins o tecnologías específicas. Por estos motivos, se está produciendo una reducción progresiva del uso de la plataforma Adobe Flash en los desarrollos actuales y como consecuencia también se han paralizado proyectos como los motores gráficos 3D para Adobe Flash. Al mismo tiempo surgen herramientas de desarrollo que facilitan la conversión de aplicaciones implementadas en Adobe Flash en aplicaciones basadas en HTML5 y JavaScript (por ejemplo: Adobe Flash Professional CC, Google Swiffy), aunque con bastantes limitaciones.

7

Realidad aumentada para la gestión de sistemas

7.1. INTRODUCCIÓN

Como ya se ha indicado en capítulos anteriores, el personal técnico dedicado a la gestión de sistemas y redes de un CPD, tiene entre sus funciones principales la supervisión del correcto funcionamiento del equipamiento existente. Esta tarea es de vital importancia ya que tiene como misión detectar y solucionar las incidencias que se presenten con la mayor rapidez posible. Aunque en el mercado existen múltiples herramientas de monitorización y supervisión, normalmente sus interfaces son bastante rígidas: muestran un exceso de información que dificulta la detección de anomalías e implica un elevado nivel de atención por parte del personal, sólo permite el acceso a la interfaz desde un puesto de control lo cual puede ser un problema en CPDs de gran tamaño o muy distribuidos, no disponen de interfaces adecuadas para dispositivos móviles inteligentes, etc.

En este capítulo se describirá una herramienta desarrollada para complementar y mejorar a los sistemas de telemonitorización existentes, con el objetivo de experimentar con nuevas formas de visualización e interacción con el usuario. Esta herramienta ha sido diseñada para su uso en dispositivos móviles inteligentes y utiliza interfaces gráficas avanzadas para intentar mejorar el trabajo habitual de los técnicos de sistemas. Así, se aprovechan las capacidades de los sensores de las plataformas móviles para dotar a la interfaz de técnicas de visualización como la

Realidad Aumentada. Con este desarrollo se pretende facilitar las actividades de supervisión de equipamientos dentro del CPD al poder acceder al estado en tiempo real de un activo simplemente con la lectura de un código gráfico situado físicamente en el activo. La información sobre el estado del activo se mostrará de forma superpuesta sobre la imagen real capturada por la cámara del dispositivo móvil.

7.2. TECNOLOGÍAS DE REALIDAD AUMENTADA

El término **Realidad Virtual** (RV) se basa en la generación de entornos sintéticos o virtuales en un espacio 3D que pueden ser explorados de forma interactiva por medio de diferentes dispositivos hardware (por ejemplo: visores, cascos, guantes, sensores, etc.) utilizando los sentidos de los seres humanos. Una definición formal podría ser la siguiente [Burdea 2003]: "*La Realidad Virtual es una interfaz usuario-computador de alto nivel que involucra la simulación e interacción en tiempo real a través de múltiples canales sensoriales como la visión, el oído, el tacto, el olfato, y el gusto*".

El concepto de RV está integrado por el conocido como "*triángulo de las tres I*" de la realidad virtual [Burdea 2003]: **Inmersión**, **Interacción**, e **Imaginación**.

- La **inmersión** se refiere a la capacidad de conseguir que el usuario se sienta como parte de la acción del entorno virtual.
- La **interacción** se refiere a la capacidad de un computador para detectar las acciones del usuario (gestos, comandos verbales, etc.) y modificar el entorno virtual de forma simultánea.
- La **imaginación** se refiere a la capacidad de la mente para percibir cosas que no existen realmente.

El término **Realidad Aumentada** (RA) está relacionado con el término de RV y se basa en la combinación de elementos sintéticos o virtuales sobre un entorno físico del mundo real. Una definición aproximada de este término es [Azuma 1997] [Azuma 2001]: "*Generación de una imagen virtual superpuesta sobre una imagen real, permitiendo la interacción en tiempo real, mezclando perfectamente objetos virtuales 2D/3D con objetos reales*". Esta superposición de elementos genera una

realidad mixta en tiempo real sobre la cual se puede interaccionar, aportando interfaces de visualización inmersivas al usuario. De este modo, objetos físicos pueden ser reconocidos para proporcionar nuevas capacidades de interacción mediante la superposición de elementos sintéticos, generados por ordenador, sobre la imagen real.

Figura 45. Ejemplo de representación del funcionamiento de las técnicas de realidad aumentada

Dentro de las aplicaciones de RA, para realizar esta superposición entre información virtual y real se utilizan diferentes métodos, tanto de forma individual como combinada:

- **Patrones de disparo del software de RA**. Se basan en la utilización de algún tipo de patrón o código para seleccionar la información virtual que hay que mostrar en cada momento. Dentro de este método podemos definir los siguiente tipos:
    - **Reconocimiento sobre imagen real**. Se utiliza la imagen real como marcador de la información a mostrar.
    - **Reconocimiento de marcadores especiales**. Se utilizan marcadores especiales reconocibles mediante sistemas de visión artificial para seleccionar la información virtual a mostrar. Estos marcadores pueden ser imágenes con marcas específicas (*marker*) o dibujos e imágenes registrados en el sistema (*tag*).

- **Reconocimiento de códigos gráficos**. Se utilizan códigos gráficos 2D (códigos QR, *Data Matrix*, códigos de barras, etc.) para obtener la información vinculada al código.
- **Realidad aumentada sin marcadores**. Se utiliza la propia escena real para el reconocimiento y visualización de los elementos virtuales (*markerless*).

- **Geolocalización**. Se centra en la utilización de los datos provenientes de los sensores de los dispositivos (por ejemplo: GPS) para obtener la información sobre la posición actual en el entorno real. Esta localización estará relacionada con *puntos de interés* (POI) definidos en el programa de RA y que se comportarán como marcadores especiales. Los POI se pueden especificar con múltiples formatos, como por ejemplo XML, y se encargan de registrar la información correspondiente a la ubicación de un punto de localización específico. Así, incluirán información propia del punto al que hacen referencia y también la información que permite geolocalizarlo (Latitud, Longitud, Altitud).
- **Interacción con redes de datos**. Se basan en la utilización de bases de datos en las que buscar y relacionar la imagen real para obtener información virtual.

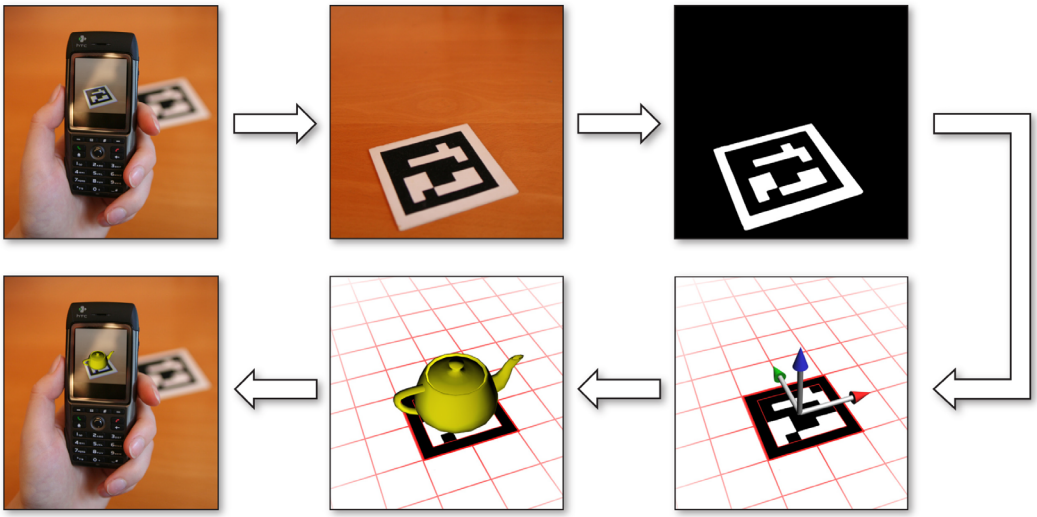


Figura 46. Ilustración de funcionamiento de un sistema de realidad aumentada utilizando marcadores gráficos [Handheldar 2015]

La creciente disponibilidad de múltiples dispositivos móviles dotados de capacidades gráficas suficientes para utilizar interfaces avanzadas unido a la inclusión de elementos hardware adecuados han hecho posible el desarrollo de aplicaciones de realidad aumentada enfocadas a este tipo de dispositivos. En la mayor parte de estos dispositivos (teléfonos móviles, tabletas y ultra-portátiles) dispondremos de sensores integrados como acelerómetros o GPS y de cámaras con una resolución suficiente para su aplicación en proyectos de RA.

A la hora de diseñar e implementar funcionalidades de RA en aplicaciones podremos utilizar librerías de desarrollo (SDK) ya existentes para crear aplicaciones independientes o podremos utilizar navegadores de realidad aumentada para el acceso a las funciones de la aplicación. En el primer caso, dispondremos de librerías de código nativo que nos facilitarán el desarrollo al disponer de funciones de alto nivel para la comunicación con el hardware del dispositivo. En el segundo caso, dispondremos de una aplicación multiplataforma similar a un navegador Web pero capaz de soportar técnicas de RA.

7.2.1. Librerías de desarrollo de Realidad Aumentada

Existen múltiples librerías de desarrollo (SDK) de Realidad Aumentada en el mercado, tanto propietarias como de código abierto, pero por su difusión actual y funcionalidades podemos destacar las siguientes:

- **ARToolKit** [DAQRI 2015]. Es una de las primeras librerías de desarrollo de RA y ha dado lugar a multitud de proyectos de SDKs de este tipo. ARToolKit es una librería de posicionamiento por computadora para la creación de aplicaciones de RA capaz de superponer imágenes virtuales sobre un entorno real. Utiliza funcionalidades de seguimiento de video para calcular la posición real de la cámara y la orientación relativa a los marcadores físicos o reales en tiempo real. Una vez establecida la posición real de la cámara, se posiciona una cámara virtual en el mismo punto para superponer los modelos gráficos 3D sobre el marcador físico. De esta forma, ARToolKit consigue resolver los dos problemas clave de la RA: el seguimiento del punto de vista y la interacción con los objetos virtuales.

Las características principales de este SDK son: desarrollo de fuente abierta y basado en módulos, soporte para las principales plataformas (iOS, Android, GNU/Linux, Microsoft Windows, Mac OS), centrado en dispositivos móviles (soporte de OpenGL ES, integración con GPS y brújula, herramientas de calibración de la cámara, etc.), permite distintos tipos de marcadores, soporte para múltiples formatos de objetos 3D, etc.

- **ARToolKitPlus** [ARToolKitPlus 2006]**.** Versión extendida de la librería ARToolKit. Es un SDK de código abierto basado en el reconocimiento de marcadores, principalmente imágenes simples con grandes contrastes entre colores para facilitar su reconocimiento, y que permite realizar una interacción con los elementos representados.

  Esta librería destaca por su elevado rendimiento, incluso con dispositivos de baja potencia o con recursos de características limitadas. Su principal desventaja son sus limitaciones a la hora de reconocer marcadores complejos como *ID-markers*. El control sobre la cámara o sobre el motor de renderizado 3D se realiza mediante librerías externas.

- **Studierstube Tracker** [Studierstube 2014]**.** Considerado el sucesor de ARToolKitPlus, utiliza un concepto muy similar a esta librería pero ha sido completamente rediseñado para mejorar su rendimiento y su consumo de recursos. Está basado en módulos de tal modo que se pueden añadir nuevas funcionalidades de forma simple.

  Este SDK acepta distintos tipos de marcadores como *ID-markers*, matrices de datos, etc. y es capaz de detectarlos con condiciones de luz cambiantes. Al igual que en ARToolKitPlus no existe control propio sobre la cámara o sobre el motor de renderizado.

- **Qualcomm Vuforia** [Qualcomm 2015]**.** Es un SDK desarrollado por uno de los principales fabricantes de procesadores de dispositivos móviles lo cual le permite obtener un gran rendimiento. Entre sus características destaca el reconocimiento de una gran variedad de marcadores: marcadores especiales, imágenes de referencia, objetos 3D simples, e incluso imágenes sin marcas (*markerless*).

Mediante este SDK los desarrolladores pueden posicionar y orientar modelos virtuales, modelos 3D y otros tipos de información en tiempo real, colocar botones virtuales, etc. Además, esta librería soporta múltiples lenguajes de programación (C++, Java, Objective-C, .Net) y es portable a múltiples sistemas operativos móviles como Android e iOS.

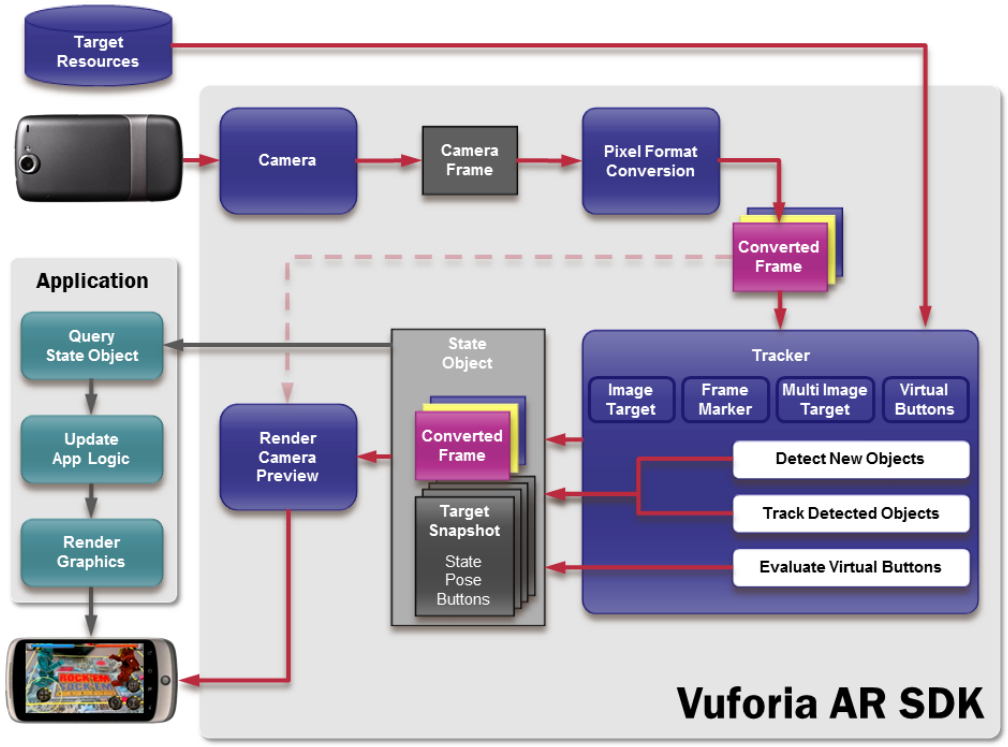


Figura 47. Esquema de funcionamiento con Vuforia AR SDK (Qualcomm)

- **Metaio SDK framework** [Metaio 2015]. Es un marco de trabajo que incluye: componente de captura, componente de sensores de la interfaz, componente de renderizado, componente de posicionamiento, y la interfaz de Metaio SDK. La interfaz proporciona interacción entre la aplicación y los cuatro componentes modulares. Esta configuración encapsula los detalles de implementación de forma transparente para el usuario. Las principales funcionalidades son realizadas a través del API del SDK, simplificando la implementación de aplicaciones de RA.

  Entre sus características incluye soporte para múltiples tipos de marcadores de posicionamiento 2D y 3D, posicionamiento de POIs, soporte para lectura de códigos gráficos como QR y barras, renderizado 3D,

optimización para hardware de dispositivos móviles, posicionamiento con marcadores LLA (Latitud, Longitud, Altitud), etc.

El Metaio SDK es compatible con las principales plataformas de desarrollo software: Microsoft Windows, Mac OS, Android, iOS. Utilizando las interfaces de Metaio SDK se puede interactuar fácilmente con el entorno de desarrollo para crear completas aplicaciones con soporte de técnicas de RA.

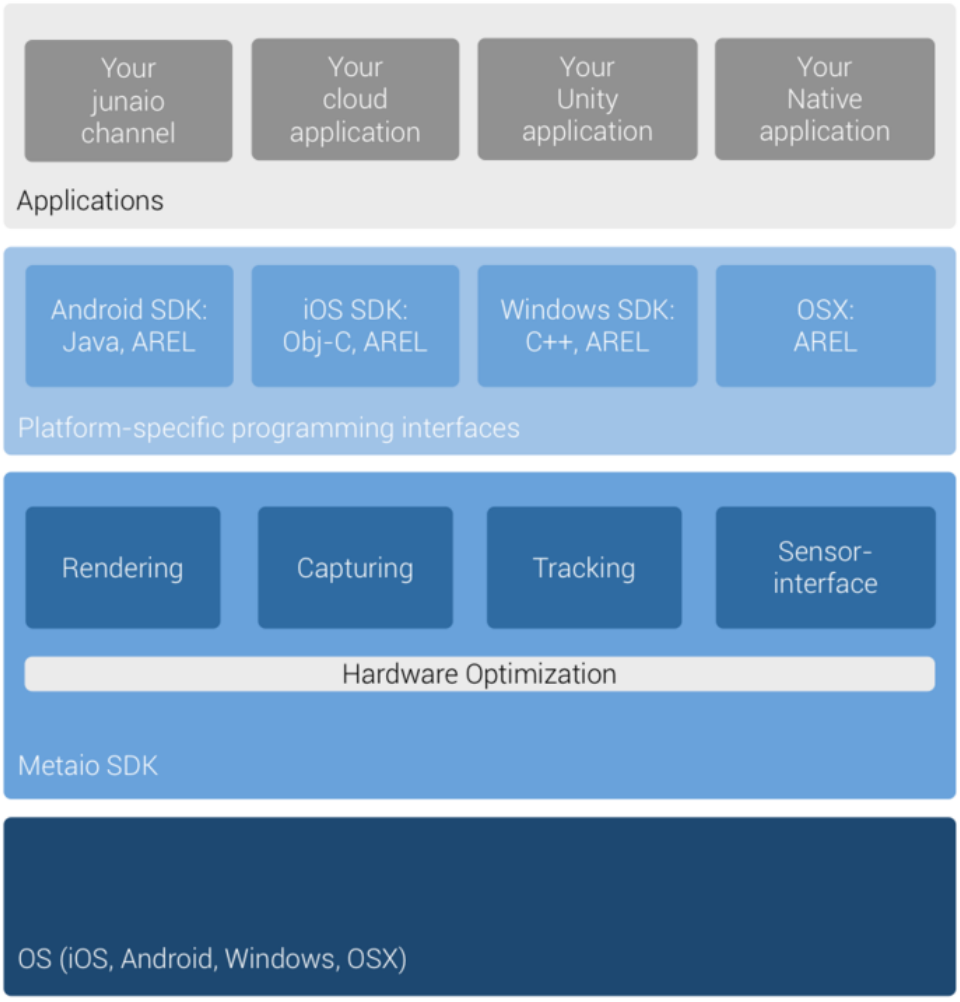


Figura 48. Arquitectura del framework Metaio SDK (Metaio)

- **ARmedia 3D SDK** [AR-media 2015]. Es un SDK que permite desarrollar aplicaciones de RA basadas en el posicionamiento de objetos 3D en tiempo real. Reconoce imágenes planas y objetos 3D complejos, independientemente de su tamaño y geometría. La arquitectura del SDK consta del motor de renderizado de modelos 3D, posicionamiento de objetos, captura de imágenes desde la cámara del dispositivo, y una interfaz

nativa para Android e iOS. ARmedia está implementado en C/C++ y es multiplataforma.

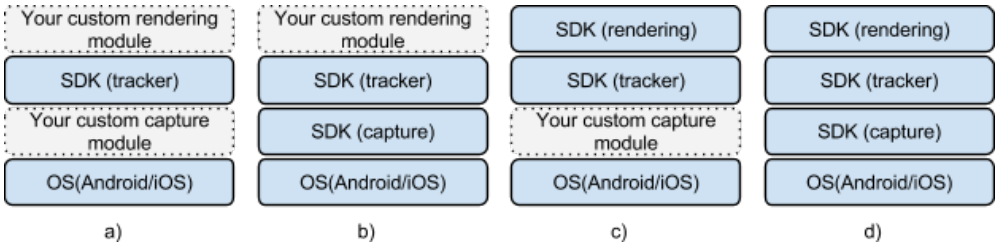


Figura 49. Módulos del software ARmedia 3D SDK

7.2.2. Navegadores de Realidad Aumentada

Los navegadores de realidad aumentada son programas con un funcionamiento similar a los navegadores Web pero dotados de capacidades específicas para soportar las funcionalidades de RA. Para ello, proporcionan APIs de desarrollo que mediante servicios Web permiten el intercambio de información entre la aplicación desarrollada y el navegador. Normalmente se utilizan formatos como JSON o XML para la comunicación entre ambos.

Los navegadores de RA se encargan de proporcionar sistemas de entrada para el intercambio de llamadas con el servicio Web. Estos sistemas reciben nombres diferentes dependiendo del navegador de RA utilizado, así encontramos denominaciones como "*channels*", "*layers*", "*overlays*", etc. Las llamadas al servicio Web están basadas en la interacción del usuario con el navegador y será el propio navegador el encargado de interpretar la información devuelta por estos servicios Web siguiendo un formato predeterminado.

Dentro de los navegadores de RA existentes en el mercado podemos destacar, por su nivel de difusión y funcionalidades, los siguientes:

- **Wikitude** [Wikitude 2015]**.** Es el primer navegador de RA en integrar información de geolocalización y realidad aumentada. Wikitude es un navegador Web en el cual la localización juega un papel muy importante, ya que la información mostrada dependerá de la posición del usuario. Así, se puede considerar que su mayor utilidad es la búsqueda de POIs en base a información obtenida a través de Internet.

Las entradas en este navegador de RA se denominan mundos (*worlds*) y son páginas Web que utilizan un servicio Web para la transferencia de información dinámica. Dependiendo del sistema operativo del dispositivo Wikitude utilizará diferentes lenguajes de programación para componer las páginas web: HTML5 y Javascript en sistemas Android e iOS, KML (*Keyhole Markup Language*) y ARML (*Augmented Reality Markup Language*) en el resto de los sistemas soportados. Además, Wikitude dispone de su propia API de desarrollo.

Las principales características de Wikitude son:

- Navegación de RA basada en geolocalización mediante GPS que permite la superposición de POIs sobre la imagen real en base a coordenadas preestablecidas.
- Representación de POIs utilizando KML y ARML.
- Dispone de un SDK comercial integrable en aplicaciones propias que permite desarrollar aplicaciones dotadas de capacidades de RA.
- Capacidad de creación de interfaces gráficas de usuario (GUIs) en base a la información de RA y utilizando lenguajes HTML y Javascript.
- Representación de imágenes, objetos 3D, sonido y video.
- Disponible para sistemas Android, iOS, Bada, Blackberry OS y Windows Phone.

Figura 50. Ejemplos de aplicación del navegador Wikitude

- **Junaio** [Junaio 2015]**.** Desarrollado por la compañía Metaio, es un navegador que dispone de una gran cantidad de opciones y que implementa una API de desarrollo basada en intercambio de mensajes XML entre la aplicación y el servicio Web. Aunque se puede utilizar de forma gratuita, requiere que las comunicaciones entre la aplicación desarrollada y el servidor que proporciona los datos se realice a través del propio servidor corporativo de Junaio.

  Mediante este navegador se dispone de características como la navegación GPS y la posibilidad de utilizar múltiples tipos de marcadores (marcadores especiales, códigos QR, códigos de barras, etc.). La información se mostrará mediante unas entradas en la propia aplicación de Junaio denominadas canales (*channels*), en las que se puede disponer de diferentes tipos de información (por ejemplo: imágenes 2D y 3D, capas superpuestas en formato HTML con información adaptada a la captura de la cámara en cada momento, etc.).

  Las características principales de Junaio son:

  - Navegación de RA basada en geolocalización mediante GPS, que permite la superposición de POIs sobre el entorno real en base a coordenadas establecidas con antelación.

- Navegación de RA basada en geolocalización utilizando marcadores LLA, que son marcadores que emulan una geolocalización GPS.
- Representación de objetos 2D/3D, tanto estáticos como dinámicos, utilizando diferentes tipos de disparadores (*triggers*) para su representación (por ejemplo: reconocimiento de imágenes, posición, múltiples tipos de marcadores como los *ID-markers*).
- Reconocimiento de marcadores de diferentes tipos como imágenes predefinidas, *markerless*, reconocimiento de códigos QR y códigos de barras.
- Representación de imagen, sonido y video.
- Diseño basado en la representación de objetos 2D/3D, tanto estáticos como dinámicos, representación de POIs, y posibilidad de superponer una capa basada en HTML5 y Javascript, que será la encargada de gestionar los eventos en tiempo real.
- Proporciona una aplicación Web, denominada "*Channel Creator*", que permite crear canales sencillos sin necesidad de disponer de conocimientos de programación ni de la sintaxis del formato XML.
- Dispone de librerías de desarrollo en lenguajes de programación PHP y Javascript. Mediante PHP se pueden manejar las llamadas al servidor de Junaio y en JavaScript se pueden gestionar los cambios en tiempo real en el lado del usuario.
- Disponible para sistemas Android e iOS.

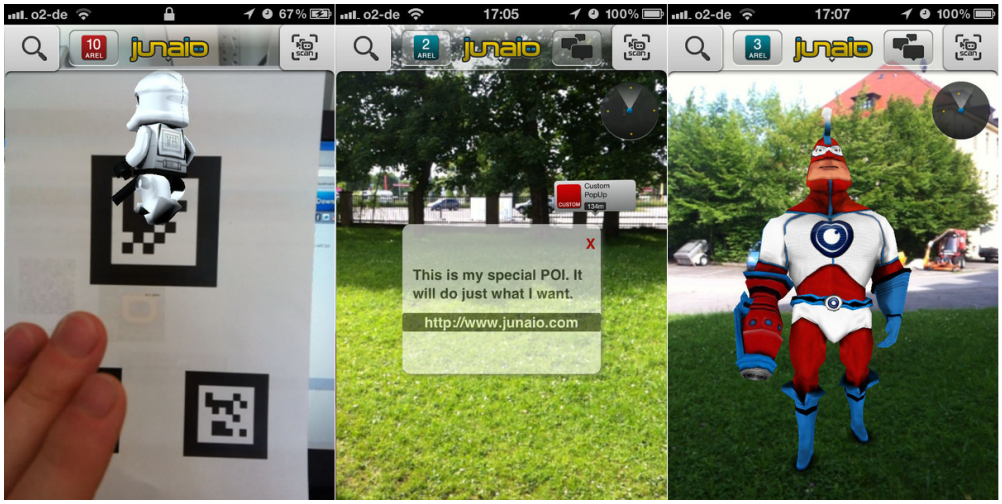


Figura 51. Ejemplos de aplicación del navegador Junaio

- **Layar** [Layar 2015]**.** El navegador Layar es uno de los navegadores de RA más extendidos en los dispositivos móviles. Se basa en el intercambio de información entre el navegador de RA y el servicio Web, utilizando como mecanismo de comunicación entre ambos al servidor corporativo de Layar. Dispone de una API de desarrollo gratuito y para el intercambio de mensajes entre la aplicación y el servidor se utiliza el formato JSON. Entre las características principales de Layar destaca la posibilidad de utilizar la navegación de RA basada tanto en información de geolocalización como en el reconocimiento de imágenes, con la capacidad de superponer una imagen real, POIs, imágenes 2D/3D, etc.

  La forma de mostrar la información de RA es mediante unas entradas en la propia aplicación denominadas capas (*layers*). Dependiendo del tipo de capa se mostrarán imágenes (2D/3D) o POIs de geolocalización.

  Entre las características más destacadas de este navegador de RA tendremos:

  - Navegación de RA basada en geolocalización mediante GPS que permite la superposición de POIs sobre la imagen real en base a coordenadas preestablecidas.
  - Representación de objetos 2D/3D, tanto estáticos como dinámicos, basándose en información de geolocalización.

- Reconocimiento de imágenes utilizando una aplicación denominada "*Layar Vision*".
- Representación de POIs basados en *Layar Vision* (objetos 2D/3D), tanto estáticos como dinámicos.
- Dispone de un SDK integrable en aplicaciones Android e iOS que comparte API con Layar y permite dotar de sus capacidades a aplicaciones propias.
- Capacidades de creación de GUIs con botones personalizados.
- Representación de imagen, sonido y video.
- Dispone de una aplicación web denominada "*Layar Creator*" para crear *Layers* de forma rápida y sin necesidad de conocer el API ni de tener conocimientos de programación.
- Disponible para sistemas Android e iOS.

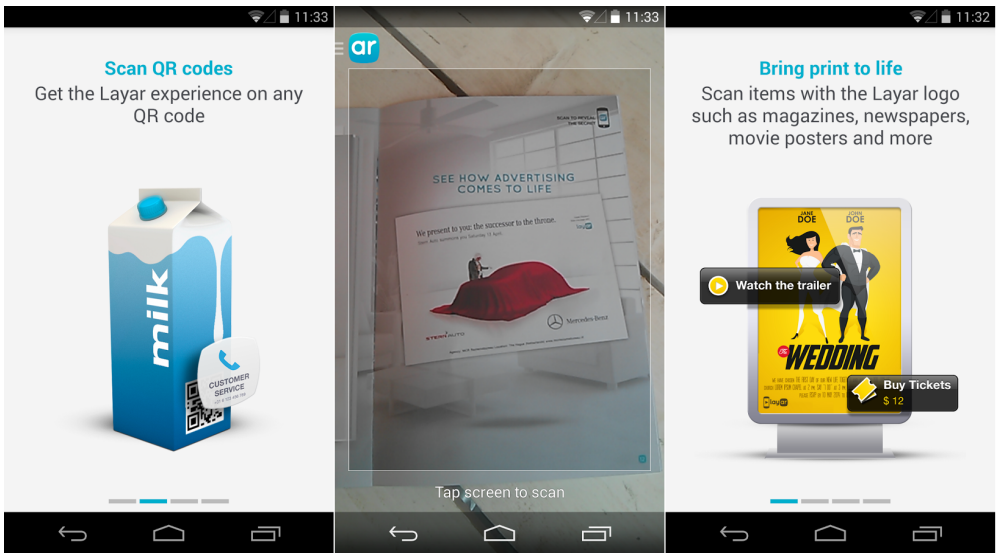


Figura 52. Ejemplos de aplicación del navegador Layar

- **Aurasma** [Aurasma 2015]. Aplicación que permite crear y compartir experiencias basadas en RA. Define una tecnología de reconocimiento de imágenes que utiliza la cámara del dispositivo móvil para reconocer imágenes del mundo real y solapar sobre ella todo tipo de contenidos como animaciones, videos, objetos 3D, o páginas Web. Entre sus características tenemos:

- Reconocimiento de marcadores basados en imágenes.
- Permite solapar múltiples tipos de elementos: imágenes, videos, modelos 3D (incluye soporte para objetos 3D en formato Collada).
- Dispone de un SDK comercial para el desarrollo de aplicaciones con tecnologías de RA.
- Proporciona la aplicación "*Aurasma Studio*" para simplificar la creación de experiencias de RA (denominadas "*auras*").
- Disponible para sistemas Android e iOS.

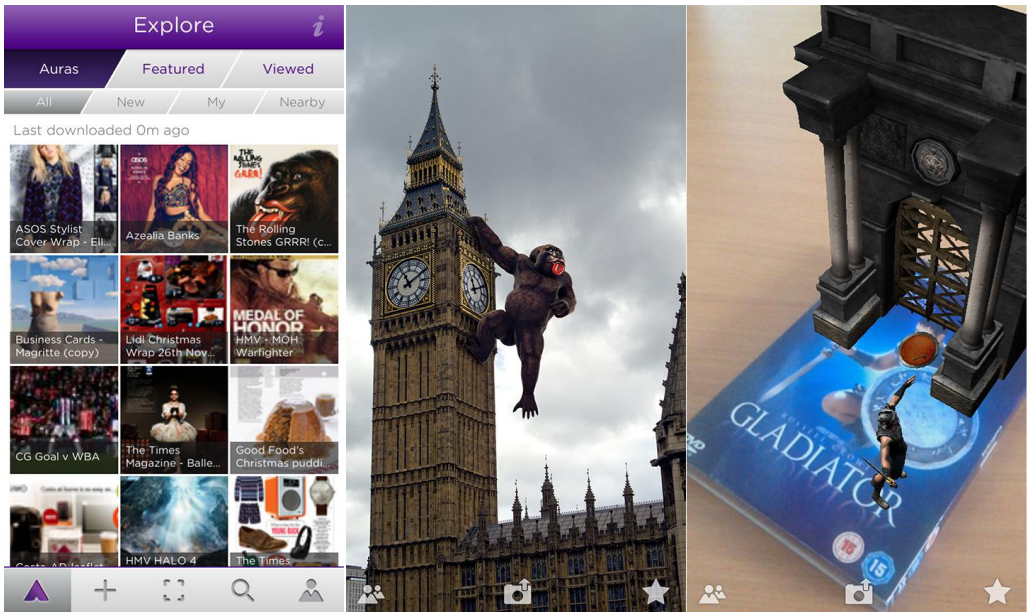


Figura 53. Ejemplos de aplicación del navegador Aurasma

7.2.3. Creación de aplicaciones con Junaio

El funcionamiento de las aplicaciones desarrolladas con Junaio se basa en el uso de canales (*Junaio Channel*) como mecanismo de comunicación e intercambio de información entre el servidor de Junaio y una aplicación que utilice esta tecnología de RA. Estos canales son creados en el servidor corporativo de Junaio por el desarrollador registrado, y entre la información que incluyen figurará el ID que identifica a un determinado canal. Cuando una aplicación cliente abre un canal se seguirán los siguientes pasos [Junaio 2015][Madden 2011]:

1. El cliente envía una petición HTTP desde la aplicación Junaio, instalada en un dispositivo móvil (navegador de RA), y como consecuencia la aplicación

le envía la petición al *Junaio backend* o servidor corporativo de la empresa Junaio para obtener el contenido de un canal especificado con un valor de ID.

2. El servidor (*Junaio backend*) busca la URL contenida en el canal y envía una petición HTTP a esa dirección URL que referencia a un servidor del desarrollador de la aplicación de RA. Esta petición puede contener los parámetros necesarios para responder adecuadamente a la petición. Así, se puede incluir la localización del usuario y el tipo de dispositivo utilizado.
3. El servidor del desarrollador interpreta la petición y responde al servidor *Junaio backend* con un documento XML específico, siguiendo un formato que cumple con las especificaciones indicadas por Junaio. Dependiendo de la forma de generar este tipo de documento XML distinguimos dos tipos de canales de interacción:
    - Canales estáticos. En estos canales el documento XML es un fichero estático y se proporciona al servidor sin necesidad de interpretar su contenido. Este tipo de canales son simples y rápidos pero el que se denominen estáticos no significa que no dispongan de lógica. La lógica de un canal que utilice este tipo de interacción se implementa en lenguaje JavaScript.
    - Canales dinámicos. En estos canales el documento XML es un fichero generado dinámicamente en función de la interacción del usuario. Un ejemplo común de este tipo de canales son los que necesitan devolver POIs basados en la geolocalización del usuario. La lógica de este tipo de canales se implementa principalmente en lenguaje PHP y puede obtener la información necesaria para generar el documento XML realizando una consulta a una base de datos del desarrollador.
4. El servidor Metaio se encarga de redirigir este documento XML al cliente, y será el cliente el que realice su análisis para obtener la información.
5. El cliente se encarga de descargar todo el contenido necesario para su representación en la interfaz (archivos HTML, JavaScript, modelos 3D, imágenes, vídeos, etc.).

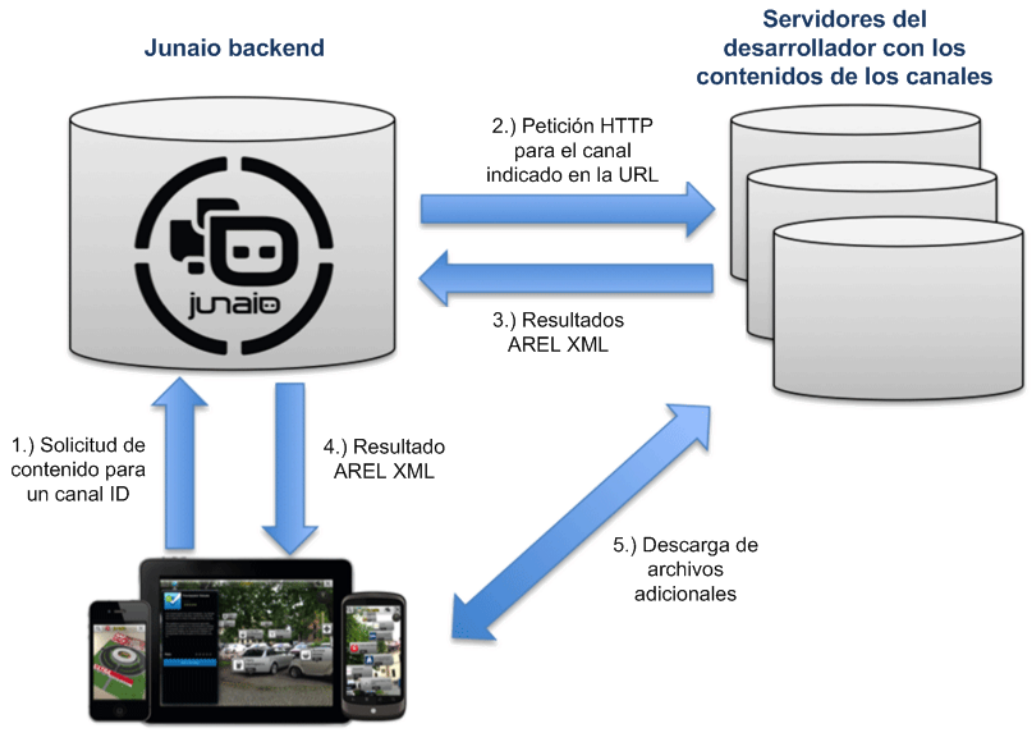


Figura 54. Diagrama de flujo de funcionamiento de los canales de Junaio

7.3. OBJETIVOS

En la actualidad disponemos de una amplia gama de aplicaciones software de monitorización y supervisión de sistemas que permiten al personal técnico realizar un control de anomalías en el funcionamiento hardware y software de los activos. En muchos casos estas aplicaciones serán específicas para un determinado hardware, compañía, marca, etc. y en otros casos serán genéricas para facilitar la supervisión de equipamientos heterogéneos.

Como se ha comentado, uno de los problemas de estas herramientas reside en las restricciones propias de cada una de ellas, lo cual puede dificultar aspectos como la detección de incidencias o el acceso remoto desde un dispositivo que no sea el establecido en la sala de control del CPD.

La generalización de nuevas plataformas móviles (teléfonos inteligentes, tabletas, etc.) capaces de proporcionar posibilidades de conexión y visualización avanzada en cualquier lugar hacen que estos dispositivos sean perfectos para su utilización en tareas de monitorización y supervisión de equipamientos informáticos. Sin embargo, este es un campo que aún no ha sido convenientemente explotado: gran

parte del software de monitorización existente no dispone de interfaces y funcionalidades específicas para este tipo de plataformas, y en los casos en los que sí disponen de ellas no aportan un valor añadido.

El objetivo de la herramienta desarrollada será plantear una aproximación para intentar solventar algunos de los problemas indicados anteriormente, incluyendo la realización de pruebas sobre el desarrollo de nuevas interfaces alternativas para el acceso a los datos de monitorización de forma rápida, cómoda, y accesible desde dispositivos móviles.

Entre las ventajas de los dispositivos móviles como elemento para la monitorización de sistemas y redes tenemos:

- Peso y tamaño reducidos que facilitan la movilidad incluso comparados con un ordenador portátil.
- Múltiples capacidades de conexión a redes de datos (Wi-Fi, Bluetooth).
- Almacenamiento local que permite el trabajo aunque no se disponga de conexión a la red de datos.
- Sensores proporcionados por estos dispositivos (cámara, acelerómetro, pantalla táctil, GPS, etc.) que pueden ser utilizados por la interfaz de monitorización.

En todo el desarrollo realizado se ha intentado simplificar al máximo posible la interfaz y la forma de interaccionar con el usuario final para mejorar la usabilidad y la adaptación a la herramienta.

Mediante esta aplicación se accederá a la información proporcionada por el software de monitorización para un equipamiento concreto desde el propio CPD mediante la lectura del marcador que identifica el activo. Para realizar estos marcadores se optó por utilizar el estándar de codificación QR (*Quick Response*), el cual es un tipo de código fácil de generar y fácil de reconocer mediante dispositivos móviles dotados de una cámara. Este procedimiento permitirá que el usuario obtenga la información del activo de forma casi inmediata y sin necesidad de indicar ninguna orden, tan sólo tendrá que utilizar el dispositivo móvil para leer el código de la etiqueta.

Después de reconocer un activo mediante su código QR, utilizando técnicas de realidad aumentada se mostrará al técnico una ventana superpuesta sobre la imagen de la cámara en la que figurará la información más relevante sobre el activo y su estado actual. Si precisa obtener un mayor detalle se podrá utilizar la pantalla táctil del dispositivo para pulsar sobre la ventana flotante y acceder a una nueva ventana con los parámetros de supervisión del equipamiento.

Dispondremos también de marcadores QR especiales lo cual nos permitirá obtener datos de más de un equipamiento. Incluso se podrá obtener el total de activos disponibles en el CPD lo cual permitirá buscar entre la lista el activo concreto del cual necesitamos monitorizar su información.

Además de visualizar la información de cada activo será preciso destacar las incidencias que se están produciendo en tiempo real en el sistema para mejorar los tiempos de respuesta en la detección y prevención de errores. No sólo se debe hacer un seguimiento del estado del hardware sino también de las actividades y procesos software ejecutados en el sistema, para poder decidir cómo actuar ante una posible vulnerabilidad. Estas actuaciones se incluyen dentro de las recomendaciones de los principales estándares de gestión de seguridad de la información (SGSI-*Sistema de Gestión de la Seguridad de la Información*) con normas como ISO/IEC 27000, ISO/IEC 27001, ISO/IEC 27002 [ISOIEC 2015].

Un aspecto importante a tener en cuenta al implementar este tipo de herramientas de monitorización es la seguridad, ya que se está accediendo a información técnica sensible de los equipamientos del CPD. Aunque las salas, normalmente, dispondrán de medidas de seguridad como controles de acceso (personal de control, sistemas biométricos, sistemas de teclado, puertas y cerraduras de seguridad, etc.) todas las aplicaciones que accedan a datos sensibles deben disponer de mecanismos de seguridad y autenticación que eviten accesos no autorizados. En la herramienta desarrollada se incluye un mecanismo de seguridad mediante usuario/contraseña y se ha creado una jerarquía de permisos de acceso gestionada por el administrador del sistema.

La herramienta implementada ha sido diseñada de forma que sea lo más independiente y flexible posible, de tal modo que la personalización para su funcionamiento en un CPD puede realizarse mediante las opciones de

configuración de la aplicación y mediante la generación de los correspondientes marcadores de los equipamientos. La conexión con el servidor en el cual está instalado el software de monitorización no precisará de otros cambios salvo los realizados en los archivos de configuración.

Para la realización de las pruebas se han empleado activos disponibles en el CPD de la FIC, el cual disponía de un número de recursos adecuados y heterogéneos. Además, en este CPD ya se disponía de un servidor en funcionamiento encargado de las tareas de telemonitorización del cual poder extraer la información necesaria para las pruebas.

7.4. DESARROLLO

El objetivo principal será desarrollar una herramienta de ayuda a la monitorización de sistemas y redes de comunicaciones para plataformas móviles, dotada de capacidades de RA para facilitar las tareas de supervisión. Esta herramienta hará uso de los sensores hardware proporcionados por estos dispositivos (acelerómetro, GPS, pantalla táctil, cámara, etc.) y de las tecnologías de RA con el objetivo de facilitar la supervisión en tiempo real de los equipamientos de un CPD desde su ubicación física.

Mediante este sistema de ayuda a la monitorización se podrá observar el estado de los activos simplemente visualizando su imagen identificativa para obtener automáticamente la información proporcionada por el sistema de monitorización de forma superpuesta en la pantalla.

Esta herramienta debe ser lo más independiente posible de la plataforma de monitorización de sistemas existente en el CPD. Así, se conseguirá que pueda ser utilizada con las principales aplicaciones de supervisión de sistemas y redes disponibles en el mercado. Para conseguir esto se diseñó la aplicación con un driver de acceso específico, adaptado a las características de cada sistema, y se centró la implementación en el diseño de la interfaz para mostrar la información sobre los equipamientos.

Una parte importante del desarrollo consistirá en la experimentación con las tecnologías actuales de RA para dotar a la interfaz de una nueva forma de interacción con los elementos bajo supervisión. Para ello, se utilizó un navegador

de RA cuya estructura de funcionamiento es similar al de un servicio Web, reconociendo el objeto situado en la posición actual y superponiendo una capa con la información correspondiente.

El estudio del problema planteado con esta herramienta de ayuda a la monitorización nos lleva a desarrollar una aplicación que permita realizar las siguientes funciones:

- Visualización de la información sobre el estado de los activos bajo supervisión, de forma individual o agrupada (armario/*rack*), a través del reconocimiento de patrones visuales. Estos patrones pueden ser imágenes o códigos contenedores de información (por ejemplo: códigos QR).
- Visualización de los equipamientos disponibles como objetos 2D sobre un plano 3D.
- Permitir visualizar los valores de funcionamiento de los diferentes componentes hardware/software monitorizados en cada activo. También se incluirá la representación gráfica del historial de funcionamiento de cada uno de estos componentes.
- Integración de notificaciones ante posibles incidencias, tanto en formato gráfico como en formato texto, para tratar de optimizar los tiempos de respuesta del personal técnico.
- Opciones de autenticación y personalización de las preferencias de la aplicación (por ejemplo: los datos del perfil de acceso de cada usuario, el idioma de la interfaz, personalización de los servidores con información complementaria, etc.).

A la hora de tratar de implementar esta herramienta de ayuda hay que destacar cuál será el origen de los datos que se utilizarán. Tal y como se ha indicado, estos datos provendrán de un sistema de supervisión instalado en un servidor central del CPD que será el encargado de la adquisición de la información de cada uno de los equipamientos disponibles. Esta herramienta será independiente del sistema de supervisión y únicamente será necesario establecer los conectores necesarios para la comunicación entre ambos.

Aunque ya se habían realizado implantaciones anteriores utilizando el sistema Nagios, en este caso se decidió utilizar la herramienta de monitorización Zabbix para las pruebas de la aplicación. Además de los motivos técnicos que nos llevaron a seleccionar este sistema de monitorización, esta elección nos permitió realizar pruebas utilizando equipamientos y datos procedentes de un servidor Zabbix instalado en máquinas del CeCaFI (Centro de Cálculo de la Facultad de Informática) de la UDC.

El API de Zabbix utiliza JSON RPC como lenguaje de intercambio y permite el acceso de forma similar a un servicio Web, a través de una dirección predefinida [Vacche 2013][Vacche 2015]. Para gestionar el acceso al API estándar de Zabbix se ha utilizado como base una librería implementada en lenguaje de programación PHP [Farley 2011].

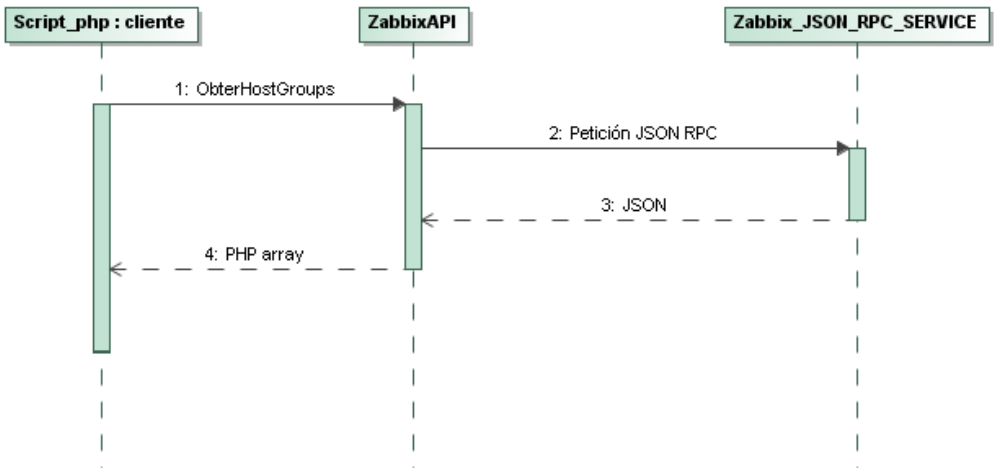


Figura 55. Diagrama de secuencia de las interacciones de Zabbix

La componente que define esta herramienta de ayuda es la utilización de técnicas de RA para tratar de simplificar la interacción entre el usuario y los elementos bajo supervisión. Para el soporte de funcionalidades de RA, disponíamos de la posibilidad de utilizar un SDK específico o de utilizar un navegador de RA. Aunque los SDK permiten un mayor control sobre los flujos de datos y los sensores de los dispositivos móviles, tienen como gran inconveniente la dependencia de la plataforma de implantación, su mayor complejidad y coste. Por ello, se optó por utilizar un navegador de RA para la implementación de la herramienta. Concretamente, se seleccionó Junaio debido a sus características y a disponer de

un API de desarrollo muy completo [Junaio 2015]. Además, en este navegador no es necesario mantener la información en los servidores propietarios de Junaio.

Junaio implementa técnicas de RA basadas en marcadores y en un mecanismo de comunicación cliente-servicio Web transparente. Aunque Junaio recibe los mensajes en formato XML, y por lo tanto sería independiente del lenguaje de programación en que se generen, la herramienta se ha desarrollado en el lenguaje de programación PHP por ser este lenguaje el utilizado por las librerías de Junaio.

Una de las limitaciones de Junaio es su débil sistema de identificación basado en un identificador único, enviado al usuario sin cifrar mediante una variable codificada con SHA-1 (*Secure Hash Algorithm*). Para fortalecer este mecanismo de seguridad se implementó un sistema de autenticación clásico, lo cual permitió añadir diferentes perfiles de acceso de los usuarios autorizados (administrador, usuarios con acceso a todos los activos, usuarios con acceso limitado a unos activos determinados, etc.).

Toda la información necesaria para esta herramienta (usuarios, preferencias, equipamientos, etc.) será almacenada en una base de datos relacional. Aunque la aplicación se ha implementado de tal modo que sea independiente del sistema de gestión de base de datos (SGBD) utilizado, para las pruebas se utilizó la solución más extendida entre los SGBD de código abierto: MySQL. Además, MySQL destaca por su rendimiento y por disponer de drivers de conexión que permiten su implantación en proyectos desarrollados en múltiples lenguajes de programación [MySQL 2015].

El diseño de la interfaz es una parte importante de esta experimentación ya que debe aportar simplicidad y facilidad de uso al usuario. Así, para proporcionar funciones dinámicas y mejoras visuales en la capa HTML que Junaio superpone sobre la imagen real se han utilizado librerías de funciones implementadas en el lenguaje de programación Javascript.

El desarrollo de la aplicación sigue el patrón de diseño MVC (*Modelo, Vista, Controlador*), separando en tres componentes los datos de la aplicación, la interfaz de usuario, y la lógica de negocio.

El modelo es la parte del servicio Web que se dedica a permitir el acceso a los recursos de los equipamientos del sistema (hardware/software) a la vista (interfaz gráfica), a través de peticiones realizadas por el controlador, y ofrecerle una fachada de acceso a estos recursos.

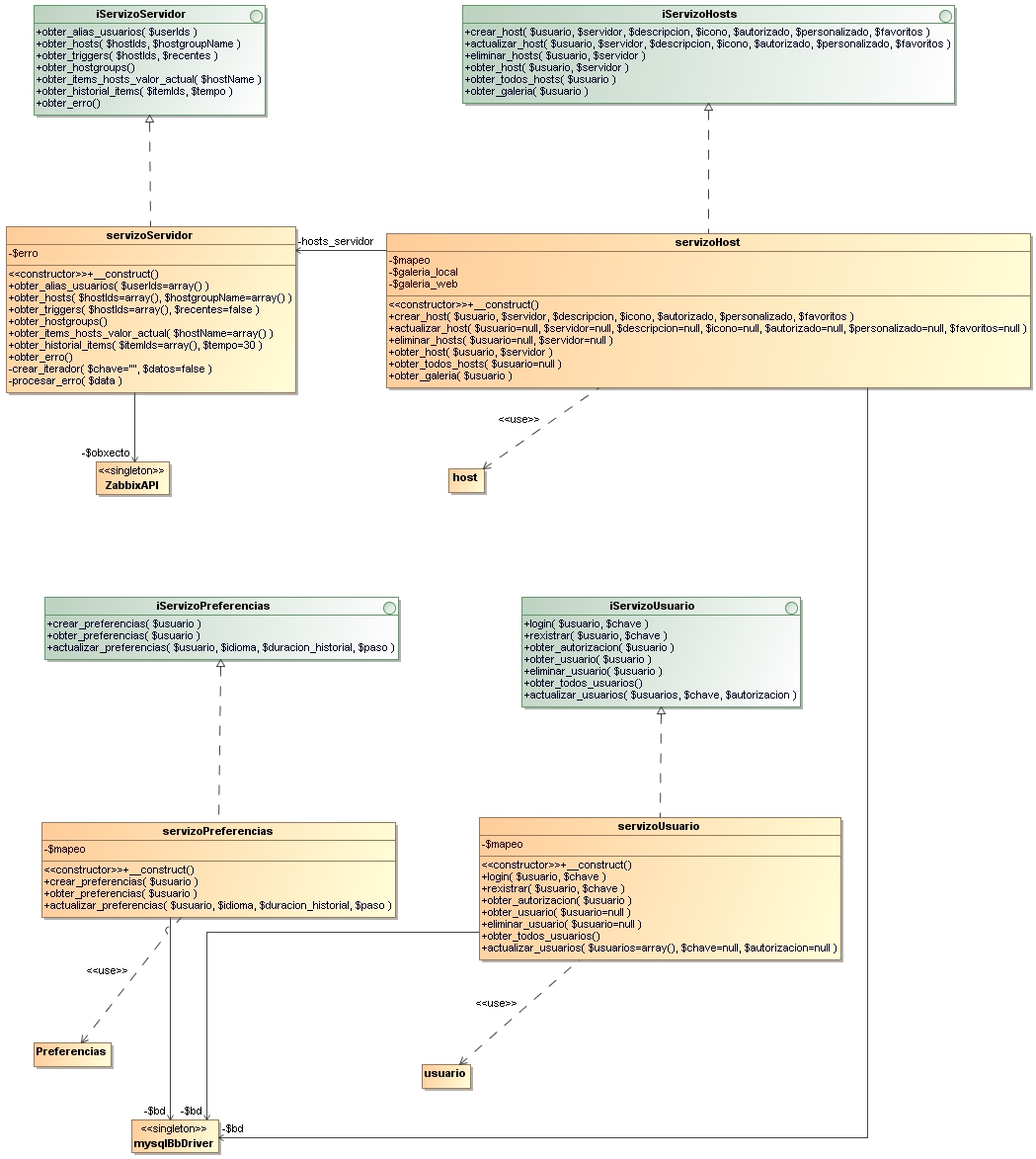


Figura 56. Diagrama de servicios de la capa modelo

El controlador es la parte del servicio Web que se encarga de la comunicación entre la vista y el modelo, proporcionando a la vista los datos que necesite del modelo, y facilitando los datos y peticiones al modelo ante las acciones que el usuario efectúe en la vista. El controlador está constituido por una serie de programas en lenguaje PHP a los que la capa vista, implementada en HTML y

JavaScript, enviará y solicitará información. Estos programas ofrecerán a la vista el acceso a las capacidades del modelo.

La vista del servicio Web es la parte encargada de proporcionar al usuario la funcionalidad gráfica y de acceso a los datos. Las tecnologías utilizadas en este caso son HTML y JavaScript (incluyendo librerías como jQuery y jQuery Mobile). El uso de la librería jQuery Mobile obliga a utilizar un diseño HTML basado en plantillas multipágina, en la cual una única página HTML contiene varias subpáginas. Cada una de estas páginas tiene detrás un determinado código JavaScript que le proporciona la funcionalidad requerida.

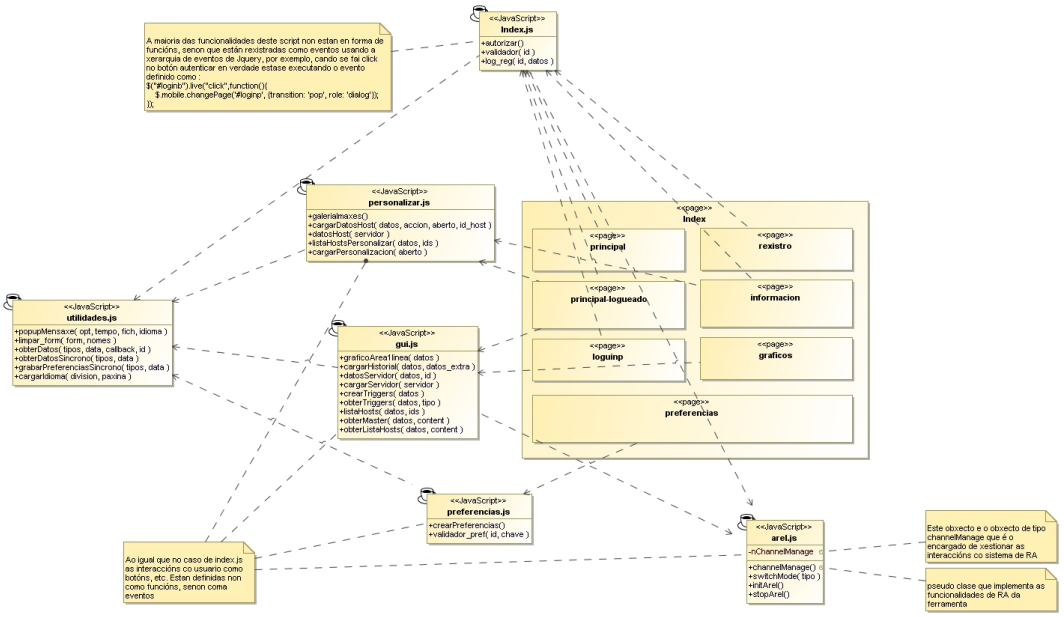


Figura 57. Diagrama de la implementación de la interfaz gráfica

Para dotar a la herramienta de ayuda a la monitorización de funcionalidades de RA se utilizan las librerías de desarrollo proporcionadas por Junaio. Estas librerías se denominan AREL (*Augmented Reality Experience Language*) y utilizan tecnologías Web comunes (HTML5, XML, JavaScript) para servir de enlace con los canales de la plataforma Junaio o para interactuar con el SDK del desarrollador de Junaio (Metaio SDK API) [Metaio 2015]. Con AREL se pueden añadir interfaces gráficas superpuestas, en formato HTML5, para crear proyectos que utilicen los canales de Junaio y conseguir experiencias de RA. Las librerías AREL se dividen en dos: AREL XML y AREL JS [Junaio 2015].

- **AREL XML**: librería para el manejo de eventos en el lado del servidor. Se centra en la parte estática, definiendo todo el contenido y los enlaces a la parte dinámica.
- **AREL JS**: librería para el manejo de eventos en el lado del cliente. Define las interacciones y comportamientos de los objetos dentro de la escena.

El esquema de funcionamiento en la aplicación desarrollada es el siguiente:

1. El cliente envía una petición al servidor Junaio y este a su vez la remite al servicio Web. En el servicio Web se define el fichero que se utilizará como manejador del lado del cliente y se pasará la petición al fichero encargado de cargar la información de los POIs existentes. Los POIs son creados de forma dinámica utilizando la información que se almacena en un fichero de configuración de la aplicación. Estos POIs son enviados como respuesta al servidor Junaio, en formato XML, y este se encargará de reenviarlos al cliente.

```
- <POI valor="argos" id="argos">
    <propiedade valor="imaxe" id="tipo"/>
    <propiedade valor="[redacted]Monitor/html/recursos/imaxes/servidores/argos.png" id="imaxe"/>
    <propiedade valor="0" id="translacion_x"/>
    <propiedade valor="0" id="translacion_y"/>
    <propiedade valor="0" id="translacion_z"/>
    <propiedade valor="1" id="escala_x"/>
    <propiedade valor="1" id="escala_y"/>
    <propiedade valor="1" id="escala_z"/>
    <propiedade valor="0" id="rotacion_x"/>
    <propiedade valor="0" id="rotacion_y"/>
    <propiedade valor="0" id="rotacion_z"/>
    <propiedade valor="100" id="cosID"/>
    <propiedade valor="91" id="tam_x"/>
    <propiedade valor="30" id="tam_y"/>
  </POI>
```

Figura 58. Ejemplo de estructura de un fichero XML con los datos de los POIs

2. El cliente carga el manejador del lado del cliente, que equivale a la página Web de la interfaz gráfica, incluyendo las funciones JavaScript que llaman a AREL JS. Mediante las clases implementadas, el usuario podrá interactuar con las funcionalidades del cliente Junaio e incluso hacer llamadas al servidor Junaio. En este caso, toda la interacción se realiza del lado del usuario mientras que el lado del servidor se limita únicamente a la precarga de los POIs.

La interfaz final desarrollada intenta ser lo más simple e intuitiva posible y está organizada de tal modo que la mayor parte del espacio disponible sea utilizado para la visualización de la información de los equipamientos, en base a los POIs reconocidos por RA. También, dispone de un botón para el menú principal de la

aplicación, que nos permite acceder a todas las opciones de configuración y gestión. Entre estas opciones tenemos:

- Preferencias generales y del perfil del usuario. Incluyen preferencias de idiomas, rangos de tiempos de monitorización, datos del perfil del usuario, datos propios del perfil de un usuario administrador.
- Personalización de los activos. Se incluyen parámetros de personalización como: etiquetas y valores descriptivos, imagen o icono representativo, pertenencia a la lista de dispositivos favoritos, datos de conexión con el sistema de monitorización.
- Clientes conectados. Disponemos de un acceso directo que permite al usuario visualizar, en una lista seleccionable, todos aquellos clientes/activos que se encuentren conectados en un momento determinado.
- Lista de favoritos. Mediante esta opción facilitamos el acceso a determinados activos bajo supervisión, sin necesidad de escanear los marcadores del sistema de RA.

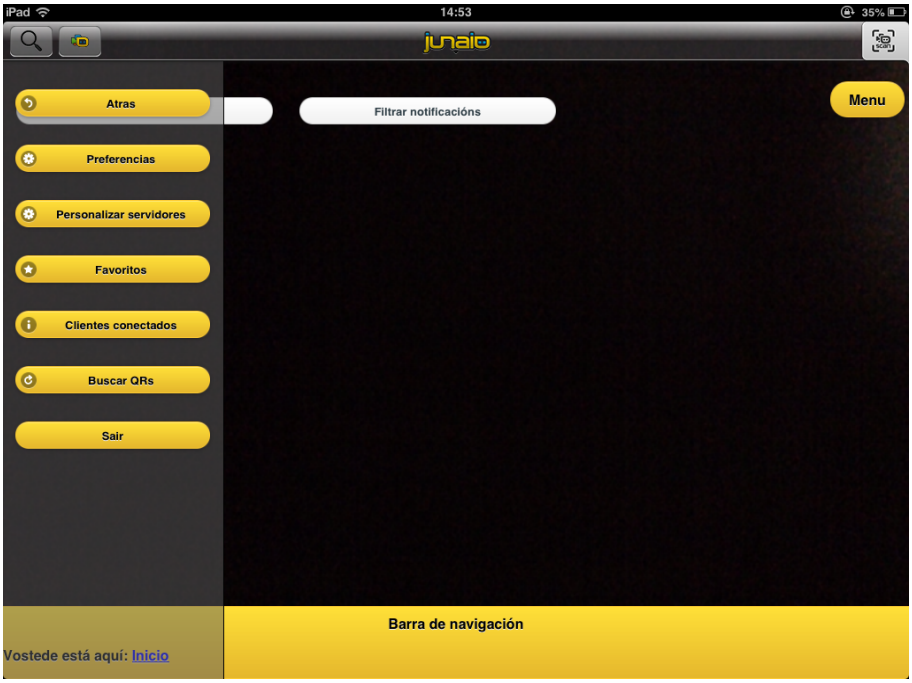


Figura 59. Interfaz de la pantalla principal de la aplicación

Figura 60. Ejemplo de visualización de la personalización de equipamientos

La funcionalidad principal de la herramienta se basa en utilizar las técnicas de RA para el reconocimiento de los dispositivos bajo supervisión. Dependiendo del tipo de marcador utilizado distinguiremos dos tipos de flujos de trabajo:

- Modo imagen. En este caso los marcadores son códigos QR que son reconocidos como imágenes y el cliente Junaio se encarga de asociar los POIs correspondientes a los elementos que representan. Los POIs quedan superpuestos a la imagen real de la cámara en forma de imágenes 2D sobre un plano 3D. Cada uno de estos POIs es interactivo, de tal modo que cuando se pulsa sobre un POI, la aplicación detecta esta interacción y reacciona en consecuencia. Esta reacción ante una selección cerrará la vista de la cámara y la sustituirá por una ventana con la información detallada del equipamiento referenciado por ese POI. Entre la información que se mostrará sobre un activo tendremos: valores actuales de los parámetros críticos en tiempo real y una opción de acceso a la visualización del historial reciente de los valores del dispositivo.
- Modo QR. En este tipo el sistema de RA lee directamente la información codificada en el marcador QR y desactiva la vista de la cámara de forma similar al modo anterior. Sin embargo, en este caso la vista con la

información mostrará la lista de los equipamientos incluidos en la metainformación de la etiqueta QR. Al seleccionar uno de los activos de esta lista se accede a la información detallada, con la información capturada para los parámetros críticos en tiempo real y el acceso al histórico reciente.

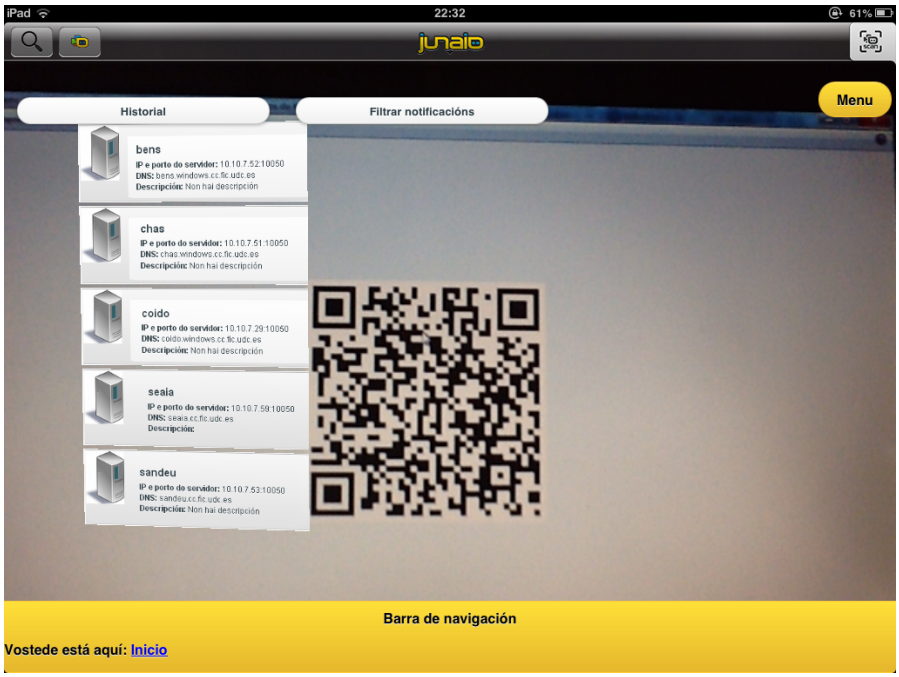


Figura 61. Detalle de funcionamiento mediante el reconocimiento de imágenes superponiendo la información correspondiente a los equipamientos

Figura 62. Detalle de la interfaz con la interacción y visualización de información de un activo seleccionado

Finalmente, esta herramienta de ayuda también permite la detección y respuesta ante alertas o incidencias en el sistema de telemonitorización. Entre la información sobre los parámetros críticos de los activos bajo supervisión también se incluyen los umbrales que definen su correcto funcionamiento, y cuando estos parámetros se salen de estos límites es posible que sea debido a algún tipo de incidencia. El objetivo de la aplicación es procesar esta información y en caso necesario visualizar en la interfaz la anomalía producida para facilitar la detección y corrección del problema. Las alertas serán clasificadas y ordenadas en función de su nivel de importancia, generando accesos directos en la interfaz para permitir la visualización de la información detallada de los equipamientos implicados.

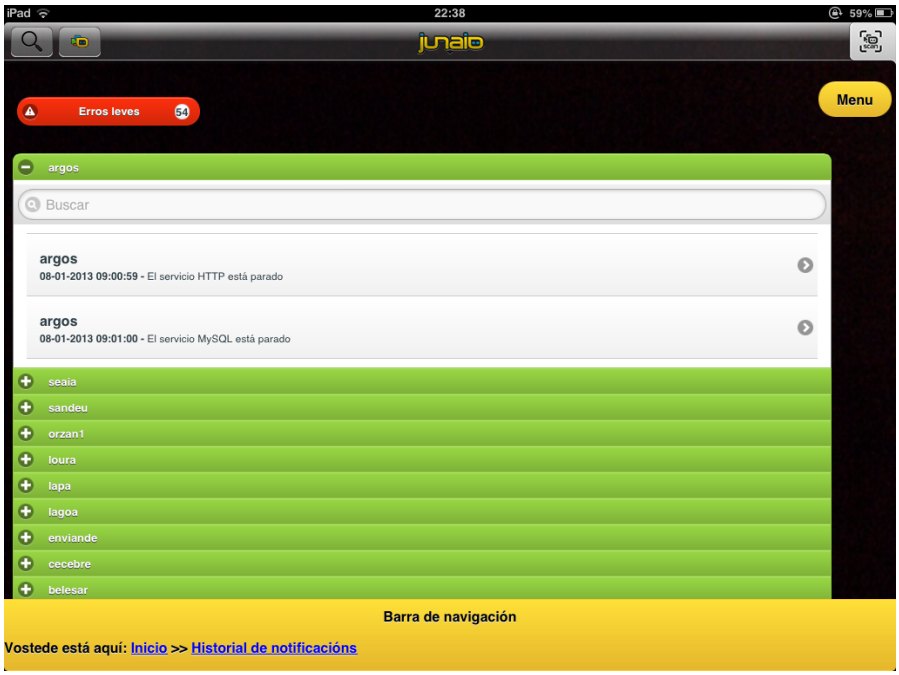


Figura 63. Detalle de visualización de incidencias y consulta del historial de notificaciones

Podemos clasificar las alertas detectadas en dos tipos:

- Alertas recientes. Son alertas producidas recientemente y que aún pueden requerir de la atención del personal técnico para su revisión. En este caso se mostrará una lista desplegable para cada activo afectado, indicando las incidencias detectadas y permitiendo mostrar la información detallada del activo y su ubicación.
- Alertas registradas. El operador dispone de un botón que le permite acceder al histórico de alertas producidas en el sistema. Al igual que en el caso

anterior, se puede mostrar más información sobre cada una de estas alertas registradas en el sistema.

7.5. CONCLUSIONES

La generalización de los dispositivos móviles inteligentes, como teléfonos inteligentes (*smartphones*) y tabletas (*tablets*), proporcionan nuevas posibilidades para la implementación de todo tipo de aplicaciones. En este sentido, las herramientas de supervisión de sistemas informáticos han tratado de adaptar sus interfaces de visualización a dispositivos móviles, principalmente modificando las interfaces descriptivas existentes a pantallas táctiles de tamaño reducido. Sin embargo estos sistemas no aprovechan las nuevas funcionalidades que caracterizan a los dispositivos móviles actuales: pantallas táctiles, cámaras, GPS, acelerómetros, etc.

El proyecto desarrollado ha tenido como objetivo principal la implantación de una herramienta de ayuda complementaria a los sistemas de supervisión actuales, orientada a dispositivos móviles, y capaz de aprovechar los sensores disponibles para facilitar la monitorización de los equipamientos. Concretamente, con este sistema se ha planteado el uso de técnicas de realidad aumentada para el reconocimiento y visualización en tiempo real del estado de un activo desde su ubicación física en el CPD. La visualización de la imagen real del equipamiento unido a la superposición de una capa informativa con los datos proporcionados por el sistema de supervisión dota al sistema de una interfaz sencilla e inmersiva.

Con este tipo de herramienta se consigue facilitar el trabajo del personal técnico desplazado al CPD ya que puede identificar y consultar la información de monitorización de un activo sin necesidad de conectarse a un terminal o de desplazarse a la sala de control. Mediante el uso de la cámara del dispositivo móvil para leer los marcadores de los equipamientos (códigos QR) se simplifica enormemente la interacción entre el sistema de telemonitorización y el técnico. De este modo, se consigue reducir el tiempo de respuesta ante incidencias, facilitando la localización del activo y visualizando en tiempo real sus parámetros actuales y las notificaciones de alerta detectadas.

Las funcionalidades proporcionadas por esta herramienta la convierten en una aplicación que puede ser utilizada como una interfaz alternativa para la telemonitorización en un CPD, ya que permite el acceso a la información crítica y también a la información detallada e histórica de un equipamiento. Además de la sencillez de manejo, con su movilidad se consigue simplificar la localización física de los equipamientos, especialmente en CPDs de gran tamaño o altamente distribuidos.

8

Modelización y supervisión de infraestructuras

8.1. Introducción

Dentro de la monitorización y supervisión de infraestructuras informáticas, uno de los problemas con los que se encuentra el personal técnico es la localización física de los activos para su mantenimiento o reparación ante incidencias. Este problema se produce sobre todo en centros de datos de gran tamaño o distribuidos en múltiples estancias o edificios. Además, afectará tanto a técnicos internos como a personal técnico externo que desconozca la distribución de las salas y equipamientos. En algunos casos, los CPDs cuentan con algún tipo de ayuda como sistemas de inventarios, representaciones esquemáticas, etiquetado de elementos, etc. que ayudan a localizar un determinado activo pero carecen de representaciones detalladas e integradas con el sistema de telemonitorización existente.

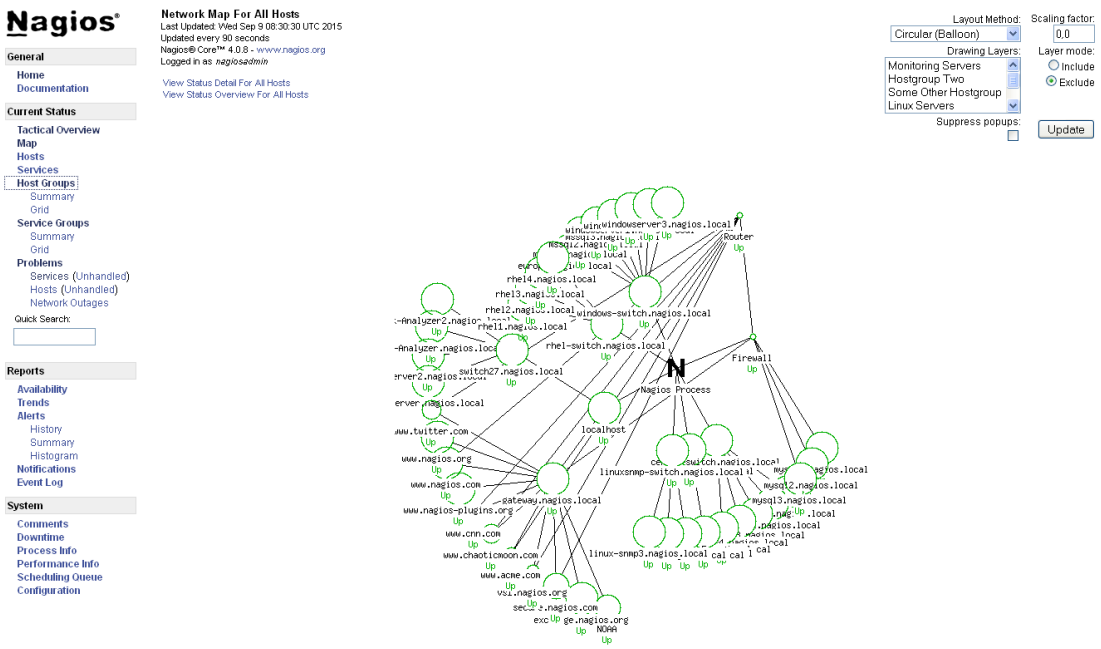


Figura 64. Ejemplo de visualización del mapa de red de equipamientos bajo supervisión con el sistema Nagios

El objetivo del siguiente desarrollo es crear una herramienta para dispositivos móviles, principalmente tabletas pero también teléfonos inteligentes, que permita a los administradores de sistemas crear una modelización virtual de los espacios físicos reales, incluyendo el posicionamiento de los activos en el escenario virtual. Además, una vez creado este plano de equipamientos se le dotará de las correspondientes relaciones con los activos físicos por medio de una conexión con el sistema de monitorización central. De este modo, no sólo se conseguirá registrar de forma interactiva la ubicación de los activos existentes, creando un plano de referencia, sino que también se podrá utilizar este plano como una interfaz complementaria para realizar una supervisión en tiempo real de los equipamientos. Así, mediante la selección de un determinado elemento se podrá llegar a visualizar la información actual de sus parámetros críticos.

8.2. TECNOLOGÍAS DE DESARROLLO MÓVIL

El desarrollo de aplicaciones para dispositivos móviles viene condicionado por la plataforma utilizada (fabricante, tipo de dispositivo, sistema operativo, SDK, etc.). Para conseguir aprovechar las características de estos dispositivos será necesario utilizar herramientas de desarrollo específicas, principalmente orientadas al

sistema operativo del dispositivo móvil. Actualmente, las plataformas móviles más utilizadas y sus opciones para el desarrollo de aplicaciones son las siguientes:

- **Android**. Permite el desarrollo de aplicaciones desde diferentes plataformas: Microsoft Windows, Mac OS, GNU/Linux. Las aplicaciones nativas son implementadas en el lenguaje de programación Java.
- **iOS**. Permite el desarrollo de aplicaciones desde la plataforma Mac OS. La implementación de las aplicaciones se realiza en el lenguaje de programación Objective-C.
- **Windows Phone**. Permite el desarrollo de aplicaciones desde la plataforma Microsoft Windows. Las aplicaciones nativas son implementadas en el lenguaje de programación C#.

Aunque con estos desarrollos específicos para cada plataforma se consiguen aplicaciones optimizadas el coste de desarrollo multiplataforma es muy elevado, ya que implica la conversión e implementación en cada uno de los sistemas.

En algunos casos, estas aplicaciones móviles no precisan de un alto nivel de integración con la plataforma del dispositivo. Así es posible crear aplicaciones multiplataforma basadas en Web utilizando la especificación HTML5. Además, este tipo de aplicaciones pueden utilizar interfaces de Diseño Web Adaptativo (RWD-*Responsive Web Design*). Este concepto de diseño fue introducido por Ethan Marcotte en el año 2010 y aunque era específico para dispositivos móviles también puede englobar a cualquier otro dispositivo para conseguir interfaces accesibles desde cualquier equipo (sobremesa, portátil, teléfono, tableta, etc.) [Marcotte 2010]. Estas interfaces son sensibles al dispositivo en el que se reproduce el contenido de la página y por lo tanto tratan de adaptarse para que sea visualizado de forma correcta independientemente de las dimensiones o resolución de la pantalla [Marcotte 2011].

Figura 65. Diseño Web Adaptativo

El soporte de las especificaciones HTML5 y CSS3 en los principales navegadores Web actuales permite que los diseños adaptables puedan ser interpretados correctamente. A nivel técnico, este tipo de diseños se basan en adaptar la capa del entorno de visualización para aportarle flexibilidad mediante tres características:

- Organización y división del contenido en cuadrículas fluidas basadas en proporciones.
- Uso de imágenes con dimensiones flexibles o escalables a las proporciones de la interfaz.
- Extensión CSS3 de las reglas "*@media*" para incorporar "*media queries*" para renderizar el contenido de la página de forma sensible a las dimensiones de la pantalla.

8.2.1. Plataforma Android

Actualmente, la plataforma Android es la más utilizada en dispositivos móviles de todo tipo (teléfonos, tabletas, cámaras, relojes, etc.). Esto es debido a sus características técnicas y a su amplia difusión en dispositivos de múltiples fabricantes. El Sistema Operativo Android es el bloque principal de esta plataforma y está basado en el núcleo (*kernel*) del sistema Linux [Android 2015].

La arquitectura del sistema operativo Android está compuesta por los siguientes componentes:

- Núcleo Linux. Se encarga de proporcionar una capa de abstracción entre el hardware y el software del dispositivo. También proporciona servicios básicos del sistema como gestión de memoria, gestión de procesos, pila de red, controladores de componentes hardware, etc. La versión específica del núcleo (*kernel*) dependerá del dispositivo y de su hardware.
- Android runtime. Es el encargado de la gestión en tiempo de ejecución de las aplicaciones y de algunos servicios del sistema Android.
- Librerías del sistema. Consiste en un conjunto de librerías escritas en lenguaje C/C++ y utilizadas por los componentes básicos del sistema. Estas librerías también pueden ser utilizadas por los desarrolladores de aplicaciones.
- Framework de aplicaciones. Es el marco de desarrollo para la creación de aplicaciones móviles en el entorno del lenguaje Java. Proporciona acceso completo a los mismos APIs utilizados por las aplicaciones base del sistema.
- Aplicaciones. Incluye las aplicaciones base de la plataforma Android (contactos, mensajería, calendario, navegador, correo electrónico, etc.) y cualquier otra aplicación nativa instalada en el dispositivo. Estas aplicaciones están desarrolladas en el lenguaje de programación Java.

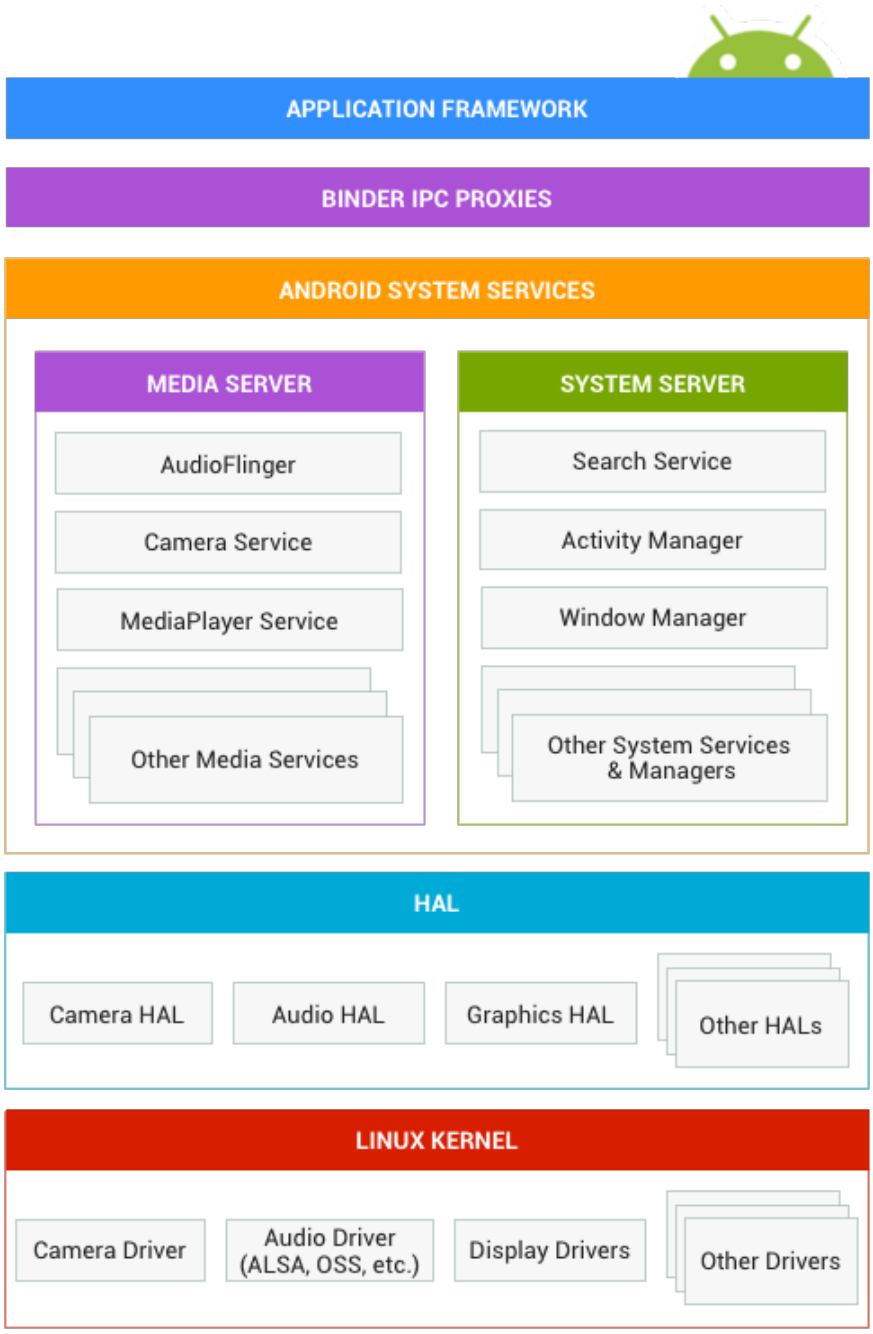


Figura 66. Diagrama de la arquitectura de la plataforma Android (Google)

El desarrollo de aplicaciones nativas para la plataforma Android se fundamenta en su SDK de desarrollo (Android SDK). Este SDK es multiplataforma (puede ser instalado en las plataformas: Microsoft Windows, GNU/Linux, Mac OS) y se compone de un conjunto de herramientas de desarrollo: depurador de código, librerías de programación, emulador de un dispositivo móvil, documentación completa y ejemplos. El desarrollador también dispone de varios IDE de desarrollo específico, como Android Studio, que se integran con el Android SDK por medio de un plugin. Así tenemos el plugin Android Development Tools (ADT) que es un

complemento que permite la integración del SDK en el IDE Eclipse para el desarrollo de aplicaciones desde este entorno de programación.

8.2.2. Framework PhoneGap

PhoneGap es un marco de desarrollo basado en estándares para la construcción rápida, simple, y multiplataforma de aplicaciones móviles [PhoneGap 2015]. Las aplicaciones desarrolladas con PhoneGap pueden ser desplegadas en cualquier dispositivo móvil sin perder las potencialidades que caracterizan a una aplicación nativa para una determinada plataforma. Este framework es compatible con las principales plataformas móviles actuales (Android, iOS, Windows Phone, Palm, Symbian, BlackBerry, etc.) y se basa en el desarrollo de aplicaciones mediante lenguajes HTML5, CSS3, y JavaScript.

Las aplicaciones desarrolladas con PhoneGap son capaces de interactuar con el hardware del dispositivo móvil y aprovechar sus funcionalidades, tales como el acelerómetro, GPS, brújula, cámara, etc. Además, también permite acceder a información del sistema como la agenda de contactos, sistema de ficheros locales, archivos de multimedia, etc.

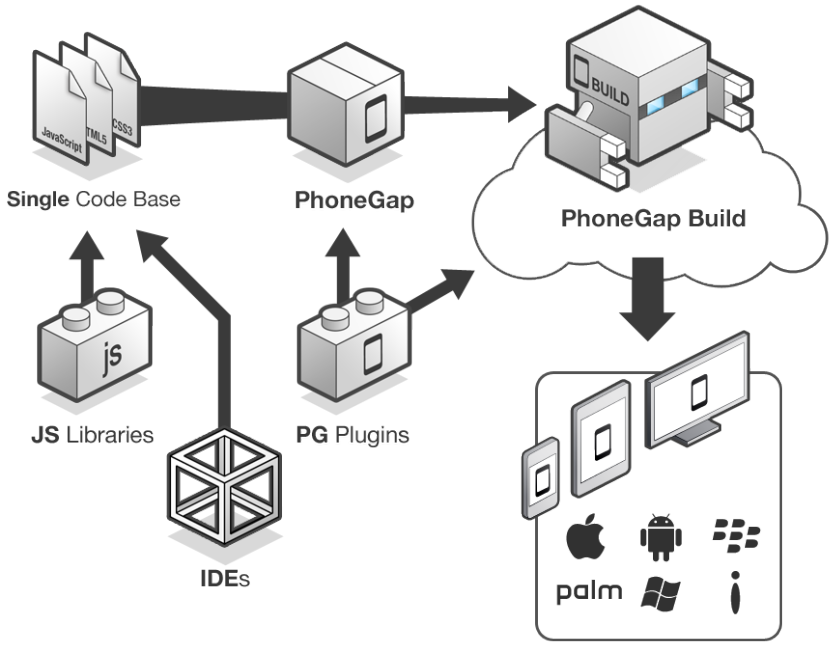


Figura 67. Diagrama de funcionamiento del framework PhoneGap (Adobe)

Mediante PhoneGap también se pueden construir aplicaciones empaquetadas de forma nativa para una determinada plataforma móvil, de tal modo que puedan ser distribuidos por los canales habituales (por ejemplo: Apple App Store, Android Market, etc.).

La creación de proyectos con PhoneGap requiere de la instalación de su SDK para poder compilar las aplicaciones que utilicen este framework. El SDK de PhoneGap proporciona un API que permite al programador de aplicaciones disponer de una capa de abstracción para el acceso a las características hardware específicas de cada plataforma móvil, permitiendo utilizar prácticamente el mismo código para múltiples sistemas. Las principales características que soporta el API son las siguientes:

- Acelerómetro. Permite que la aplicación sea sensible a cambios en la orientación del dispositivo.
- Brújula. Utilizado para adaptarse a la posición a la que apunta el dispositivo en aplicaciones de navegación y mapas.
- Cámara. Permite utilizar la cámara del dispositivo para la captura de imágenes fotográficas. También permite acceder a los archivos almacenados en la galería de imágenes.
- Geolocalización. Ayuda a obtener la localización geográfica en la que está situado el dispositivo por medio de un sistema GPS.
- Contactos. Proporciona acceso para lectura y escritura de los contactos del dispositivo.
- Ficheros. Permite el acceso para lectura y escritura al sistema de ficheros. Se incluyen operaciones sobre archivos y directorios.
- Redes. Utilizado para comprobar el estado de la red o para conocer el tipo de conexión utilizada (2G, 3G, 4G, Wi-Fi, etc.).
- Media. Facilita el control de los sensores y aplicaciones multimedia del dispositivo. Así, permite la grabación y reproducción de video.

- Notificaciones. Permite realizar notificaciones al usuario cuando se produce algún suceso. Las notificaciones pueden ser mediante sonido, vibración, o por medio de alertas en pantalla.
- Almacenamiento. Incluye la construcción y uso de bases de datos para las aplicaciones. Utilizando el lenguaje de consultas SQL (*Structured Query Language*) proporciona el mecanismo para realizar las operaciones básicas sobre la base de datos.

Hay que resaltar que PhoneGap también dispone de software complementario, o plugins, que le permiten añadir funcionalidades extra de forma simple. Un ejemplo de este tipo es el plugin para el IDE Eclipse.

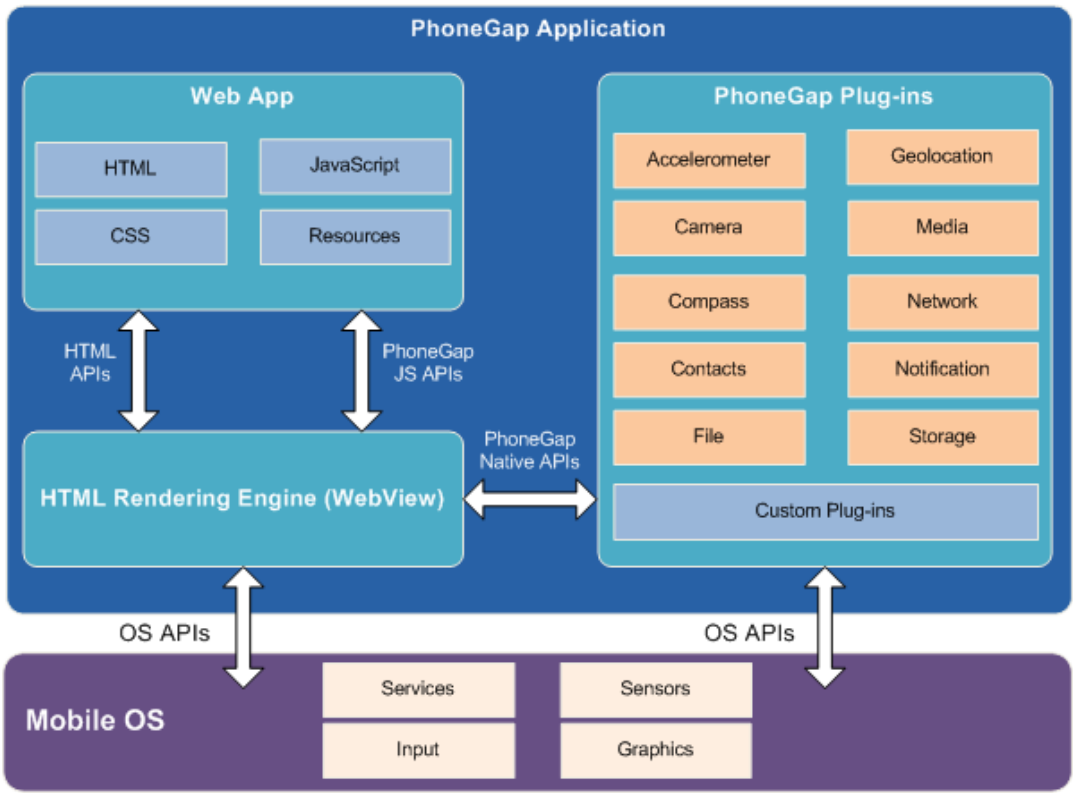


Figura 68. Esquema de la arquitectura del framework PhoneGap

8.3. OBJETIVOS

La universalización de los dispositivos móviles, tanto teléfonos como tabletas, dotados de características hardware y software cada vez más avanzadas abren un nuevo campo de experimentación en el desarrollo de nuevas herramientas para la realización de tareas de trabajo diarias. En el caso de la supervisión de sistemas

informáticos disponemos de la capacidad de proporcionar al personal técnico un dispositivo manejable para la gestión en estancias de procesamiento de datos.

El objetivo principal de este desarrollo será la creación de una aplicación instalable en un dispositivo móvil que sirva de ayuda a la administración de los activos existentes en los diferentes espacios que conforman el CPD de una compañía. Esta aplicación tendrá como función servir de modelador y visor interactivo de los equipamientos bajo supervisión. Además, tratará de complementar las funcionalidades proporcionadas por el sistema de supervisión central instalado en el CPD, obteniendo la información de los parámetros más críticos de cada dispositivo en tiempo real y visualizando su estado directamente en el visor de la herramienta.

Hay que tener en cuenta que en muchos casos los CPDs, de tamaños mediano/grande o muy distribuidos en diferentes espacios físicos, necesitan disponer de un plano de situación de los equipamientos para facilitar el trabajo de los técnicos y mejorar la respuesta ante incidencias. Las herramientas de telemonitorización actuales no incluyen este tipo de representación gráfica, limitándose a diagramas esquemáticos de la red y de los equipamientos bajo monitorización.

Esta herramienta dispondrá de dos funcionalidades principales: edición y visualización. La primera de estas funcionalidades permitirá realizar una edición gráfica e interactiva de un espacio virtual, colocación de objetos presentes en una sala y establecimiento de las características de cada equipamiento, incluyendo la vinculación con el sistema de monitorización central del CPD. La segunda de las funcionalidades está orientada al acceso a la visualización e interacción con los espacios virtuales generados.

Para el modelado de las salas se utilizará una información básica para crear la estancia (nombre, tamaño, forma, etc.) y posteriormente se podrán añadir los diferentes tipos de dispositivos ubicados en ella. Estos dispositivos podrán ser manejados de forma interactiva, aprovechando las capacidades táctiles proporcionadas por los dispositivos móviles, de tal modo que se podrán realizar acciones básicas como: posicionar, mover, eliminar, editar. También se podrá modelar el esquema del cableado de la red de datos de la sala. Para ello, se

permitirá crear una capa específica, superpuesta sobre la capa de equipamientos, en la cual será posible añadir información como: localización, conexiones con otros objetos (activos), número de tomas, tipo de cableado.

La información registrada mediante esta herramienta de modelización de salas será almacenada en una base de datos local, en el propio dispositivo móvil. Así, se conseguirá que la aplicación pueda ser utilizada sin necesidad de conexión permanente a la red de datos. Por ejemplo, cuando se está realizando el modelado de las salas no es necesario disponer de conexión a la red para realizar el trabajo. Cuando el dispositivo móvil está conectado a la red, se habilitará una opción que le permita al usuario realizar la sincronización de los datos almacenados en local en un equipo servidor específico. De este modo, se podrá consolidar toda la información registrada en la herramienta de modelado. Esta información consolidada podrá ser utilizada por cualquier dispositivo móvil que incluya esta herramienta de modelización y supervisión.

Las funciones de supervisión de esta herramienta se basan en el seguimiento en tiempo real de los valores de los parámetros críticos de cada activo que son proporcionados por el sistema de telemonitorización central. Cuando se detecten valores inadecuados en estos parámetros se notificará la incidencia en la interfaz de la aplicación, mediante mensajes de aviso o mediante cambios visuales en los objetos correspondientes (colores, iconos, etc.).

8.4. DESARROLLO

Una vez conocidos los objetivos marcados con esta herramienta, y después de realizar un análisis de los requisitos que debe cumplir, establecemos las siguientes funcionalidades básicas que debe incluir:

- Creación y eliminación de espacios. Podremos definir nuevas zonas de trabajo con sus etiquetas identificativas y sus dimensiones, o eliminar espacios existentes. Para facilitar el modelado de los espacios se utilizará una medida de referencia básica, similar a las baldosas/losetas existentes en los suelos técnicos de las salas de un CPD. Además, esta medida también nos servirá como guías para facilitar la colocación de objetos sobre el plano.

- Edición y visualización de un espacio. Permitirá editar o visualizar los espacios creados por la aplicación. En la opción de edición se permitirá ejecutar acciones de inserción, modificación, borrado de activos dentro de un determinado espacio. En la opción de visualización se podrán ejecutar acciones como desplazamientos o zoom, y también se podrá interactuar con los objetos existentes.
- Detección de espacios y equipamientos mediante códigos QR. Facilita la localización y carga de planos mediante el uso de marcadores gráficos.
- Gestionar y visualizar detalles de los equipamientos. Se realizará una gestión completa de los activos para permitir la visualización de la información correspondiente a sus parámetros críticos. Entre estos equipamientos distinguiremos dos tipos principales: dispositivos simples y dispositivos compuestos. En el primer caso son equipamientos que se corresponden con un dispositivo (por ejemplo: servidor, computadora, *router*, etc.) y en el segundo son activos que incluyen a otros dispositivos (por ejemplo: armario/*rack*). Las tareas de gestión soportadas para dispositivos compuestos incluye las correspondientes opciones de inclusión, edición, y eliminación de dispositivos dentro del armario y la colocación en una determinada posición (unidad dentro del armario).
- Gestión de una capa para el modelado del cableado de red de un espacio. Servirá como esquema general de la red de datos, pudiendo conectarse con los equipamientos existentes y reflejando el estado de los enlaces en función de dichos equipamientos.
- Generación de informes. Proporcionan al personal técnico una función para poder visualizar el estado de un espacio y la posibilidad de exportar este informe a un documento en formato PDF.

La representación gráfica de todos los equipamientos intentará reflejar de la forma más fiel posible el modelo real del dispositivo y por este motivo dispondremos de una colección de archivos gráficos con los activos clasificados por tipo de producto y marca del fabricante. Esta colección puede ser ampliada de forma simple para incorporar nuevos modelos o editar los existentes. Desde la aplicación podremos

vincular la imagen correspondiente a un activo seleccionándolo de la colección existente.

A la hora de abordar la implementación de esta herramienta para dispositivos móviles disponemos de dos opciones: realizar una aplicación nativa para el sistema operativo del dispositivo móvil o realizar una aplicación multiplataforma basada en una interfaz Web con HTML5+CSS3+JavaScript. La primera opción implica utilizar tecnologías y lenguajes de programación específicos para cada plataforma en la que se pretenda implantar pero con la ventaja de conseguir una aplicación optimizada para el dispositivo de destino. En la segunda opción conseguiremos una aplicación prácticamente portable a una gran cantidad de dispositivos siempre que dispongan de un navegador Web que acepte el estándar HTML5.

Teniendo en cuenta las ventajas de conseguir una aplicación portable y el hecho de que también se puedan conseguir rendimientos elevados con una aplicación basada en HTML5, se ha seleccionado esta opción para la implementación de la aplicación. Además, la especificación HTML5 también dispone de funcionalidades gráficas avanzadas como el soporte para el renderizado de gráficos 2D/3D.

Por tanto, la aplicación se desarrollará basándose en los lenguajes HTML5, CSS3, y JavaScript. Entre las librerías JavaScript utilizadas destacan jQuery y su versión optimizada para dispositivos móviles jQueryMobile. Estas librerías facilitan la programación de aplicaciones con Javascript ya que proporcionan funcionalidades como la gestión de eventos, desarrollo de animaciones, interacción mediante AJAX, etc. Además, jQuery dispone de multitud de plugins que se integran para proporcionar nuevas capacidades.

La utilización del framework de desarrollo Phonegap, nos permite adaptar y construir el empaquetado de la aplicación Web en HTML5 para el sistema operativo del dispositivo de forma simple. Phonegap nos proporciona un API que abstrae las funcionalidades nativas de los dispositivos móviles para que el desarrollador se pueda centrar en la aplicación.

El diseño de la aplicación sigue el modelo MVC (Modelo, Vista, Controlador), donde:

- Modelo: está formado por los archivos JavaScript que interactúan con la base de datos local y transforman la información en objetos que puedan ser utilizados por otros módulos.
- Vista: está compuesta por los archivos HTML y CSS que proporcionan la interfaz de usuario y los elementos necesarios para su interacción.
- Controlador: está formado por archivos JavaScript que se encargan de controlar los eventos producidos por el usuario y actuar en consecuencia.

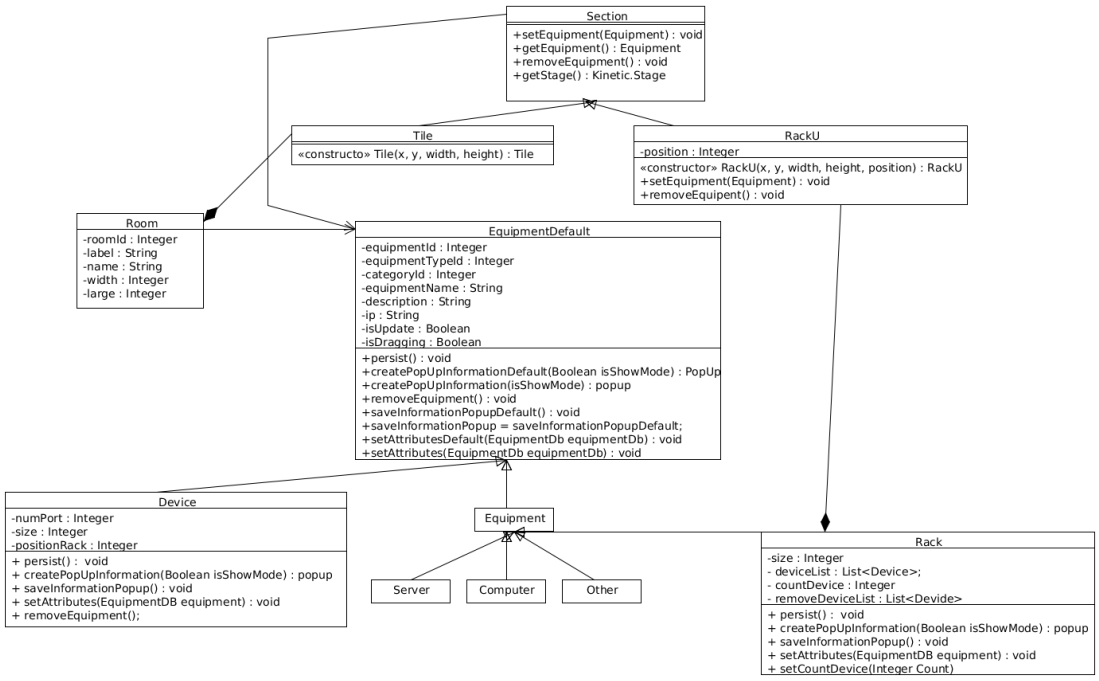


Figura 69. Diagrama de clases de la aplicación

Uno de los objetivos de este proyecto es la experimentación con interfaces de visualización avanzadas. Así, es importante que la interfaz sea sencilla y fácil de utilizar por el usuario, pero también conseguir que su diseño se adapte al dispositivo móvil en el que se está ejecutando. La aplicación desarrollada utiliza un diseño Web adaptativo (RWD) para ajustarse adecuadamente a las dimensiones del dispositivo o al tamaño de la ventana. Además, el usuario dispone de soporte de gestos multitáctiles para aprovechar las capacidades táctiles proporcionadas por los dispositivos móviles.

Se ha utilizado la librería Hammer.JS para implementar el reconocimiento de gestos realizados por medio de diferentes dispositivos de entrada (pantallas

táctiles, ratones, punteros) [HammerJS 2015]. Esta librería JavaScript permite gestionar eventos como: *Tap*, *DoubleTap*, *Swipe*, *Drag*, *Pinch*, y *Rotate*. Además, esta librería dispone de un pequeño plugin para su integración con la librería jQuery.

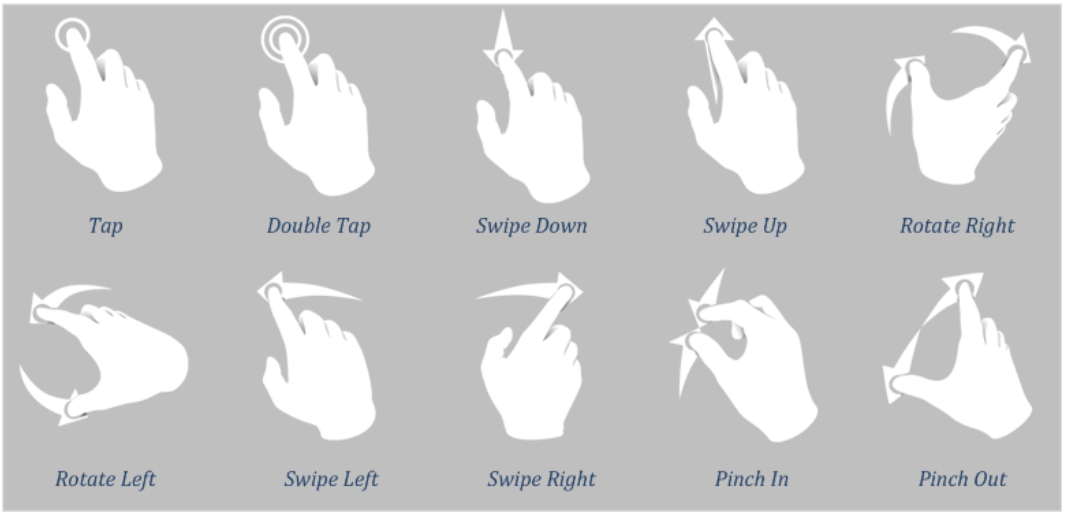


Figura 70. Representación de gestos multitáctiles en una pantalla táctil

La aplicación incluye un menú principal que permite acceder a las funciones de creación de un nuevo espacio/sala, mostrar la lista de espacios disponibles, seleccionar un espacio mediante la lectura de un código QR, y gestionar las opciones de configuración.

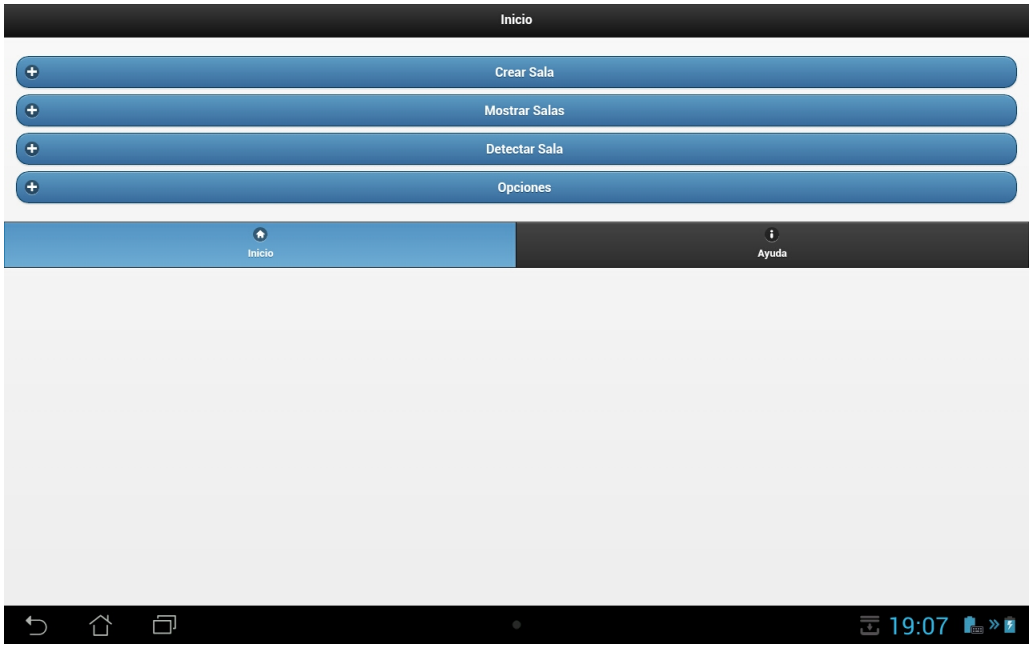


Figura 71. Pantalla de inicio de la aplicación de modelado de infraestructuras

La creación de un nuevo espacio de equipamientos permite definir los datos básicos para generar e identificar la nueva sala. Entre la información necesaria para esta función tendremos: etiqueta o identificador, nombre del espacio, dimensiones (ancho y largo), tamaño de la baldosa/loseta del suelo técnico, textura de la baldosa. La baldosa será utilizada como medida de referencia para la creación del suelo técnico y permite definir una rejilla para facilitar la colocación posterior de los equipamientos en la representación virtual del espacio.

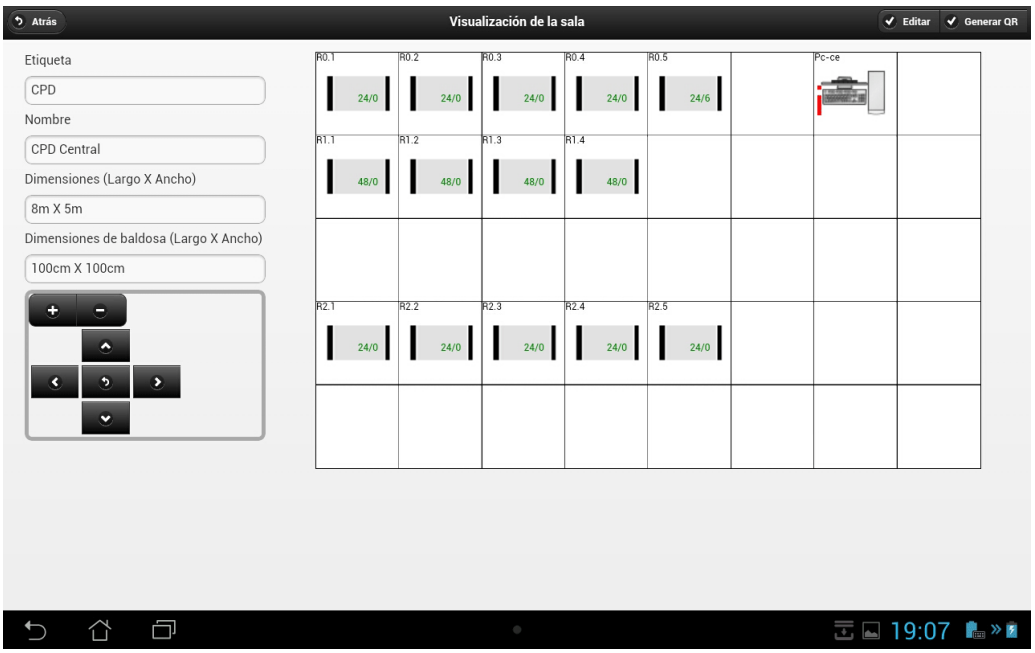


Figura 72. Interfaz de visualización de una sala de equipamientos

El acceso a un determinado espacio se puede realizar mediante la consulta de la lista de espacios disponibles. También se ha implementado una opción que permite acceder automáticamente a un espacio mediante la lectura de un código QR. Nos centramos en los códigos QR ya que se trata de un formato fácil de capturar por un dispositivo móvil provisto de cámara y que nos permite incluir todo tipo de información en formato texto dentro del código. Así, colocando un código QR en la entrada de cada sala del CPD el técnico autorizado podrá acceder de forma directa a la visualización de su representación gráfica.

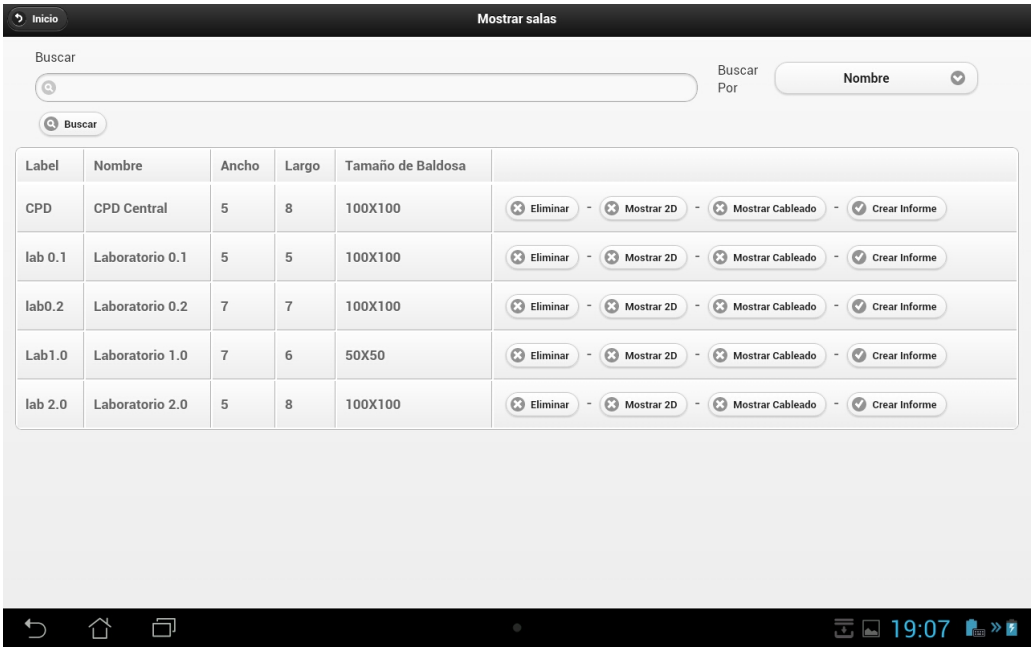


Figura 73. Visualización de la lista de salas disponibles en el modelador de infraestructuras

La aplicación desarrollada permite crear nuevos códigos QR que identifiquen un espacio, para su posterior impresión y colocación física, y leer los códigos QR existentes. Para conseguir estos objetivos se pueden utilizan múltiples librerías de desarrollo. Concretamente, en esta herramienta se han utilizado:

- jQuery.qrcode. Es un plugin para la librería jQuery que facilita la creación de forma dinámica de códigos QR [QRcode 2015]. Dispone de algunas opciones de personalización de los códigos QR generados como: el tamaño, el color, o el método de renderizado (*canvas*, *image*, *div*). Se utilizó para la creación de los nuevos códigos.
- BarcodeScanner. Es un plugin para PhoneGap que permite codificar y decodificar etiquetas con múltiples formatos de códigos de barras estándar, incluyendo los códigos QR [PhoneGap 2015]. Se utilizó para la lectura de los códigos creados.

Cuando accedemos a un espacio creado, lo podremos editar y además dispondremos de un conjunto de botones para realizar operaciones básicas como: eliminar el espacio, acceder en modo de visualización, diseñar el esquema del cableado, generar informes con el listado de los equipamientos disponibles. Para facilitar el manejo de la aplicación en interfaces no táctiles se han incluido botones para desplazamiento y zoom.

El proceso de visualización de un espacio y de sus equipamientos requiere de una serie de consultas a la base de datos de la aplicación. En primer lugar, es necesario obtener las propiedades del espacio para crear su representación en pantalla. A continuación se obtienen los datos de los elementos presentes en la sala. Entre los datos que identifican a un elemento figura su posición, tanto en el suelo técnico como en la unidad de un armario.

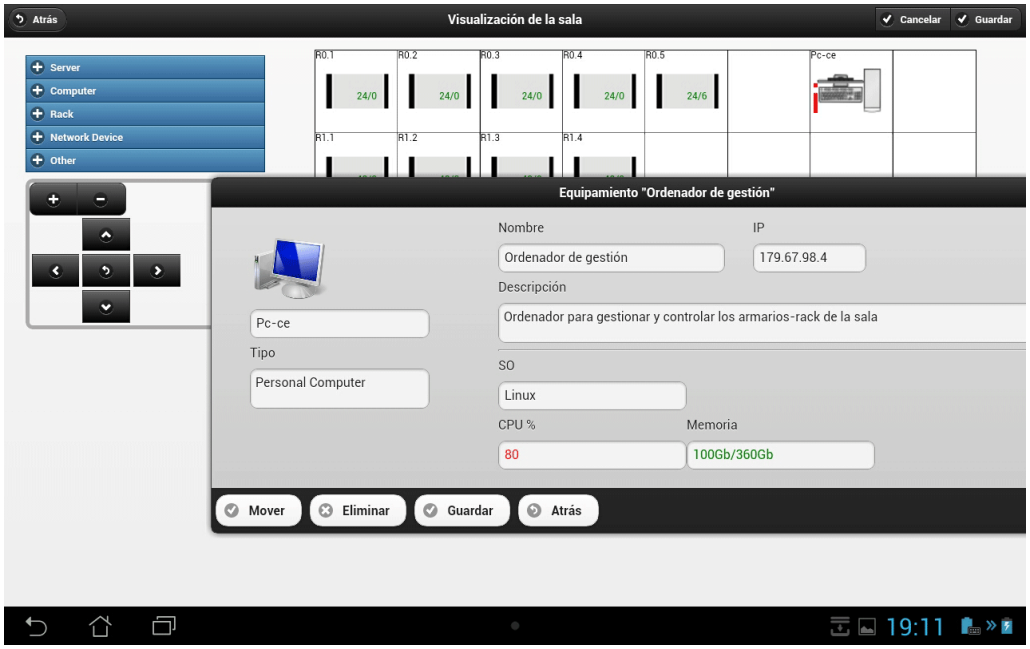


Figura 74. Detalle de la visualización de un equipamiento y su estado actual proporcionado por el sistema de monitorización

En la implementación de la interfaz gráfica se utiliza la librería KineticJS que es una librería JavaScript, para aplicaciones móviles y de escritorio, cuyo objetivo es extender las funcionalidades del contexto 2D del elemento "*canvas*" de HTML5 [KineticJS 2015]. Así, facilita tareas como: crear formas e imágenes, añadir eventos para interactuar con los objetos (mover, escalar, rotar, etc.), crear animaciones, crear transiciones, etc. Los elementos representan los activos del entorno real y podrán posicionarse libremente sobre la estancia. Para su manejo y representación se utilizan objetos *Kinetic.Image*, lo que permite disponer de eventos que simplifiquen tareas como la selección o el cambio de posición en el plano de la estancia.

Los elementos requieren de la introducción de una serie de parámetros básicos que permiten al técnico identificarlo y relacionarlo con el sistema de

telemonitorización central. Al igual que en la implantación de la aplicación descrita en el capítulo anterior, para las pruebas de esta aplicación se ha utilizado un sistema de monitorización Zabbix instalado en los servidores del CeCaFI. Entre los equipamientos bajo supervisión también se incluyeron 8 servidores de cálculo del grupo de investigación.

Los datos básicos que se incluirán para identificar a un elemento son: identificador, tipo de dispositivo, nombre, descripción, sistema operativo, IP, MAC, etc. El tipo de dispositivo permite realizar una clasificación de los activos existentes en una estancia y puede ser utilizado para consultas e informes. Por defecto se definen un conjunto de tipos de dispositivos (servidores, computadoras, armarios, dispositivos de red, otros) pero pueden crearse nuevas clasificaciones. Del mismo modo, dentro de cada categoría se incluirá una lista de modelos de dispositivos con su imagen representativa (simbólica o real). La aplicación está preparada para permitir modificaciones sobre los elementos existentes y para añadir nuevos modelos de elementos. Para facilitar la selección de un elemento se utilizan listas con conjuntos de imágenes y no sólo listas textuales. Esto mejora el manejo en interfaces gráficas táctiles y se ha implementado utilizando el plugin de la librería jQuery denominado *jQuery.imagepicker* [ImagePicker 2014].

Un caso especial dentro de los elementos disponibles son los armarios/*rack* ya que estos se caracterizan por agrupar un conjunto de dispositivos en su interior. Para ello, los armarios se dividen en unidades numeradas para permitir la colocación de los dispositivos. El número máximo de unidades ("U") de un armario servirá para identificar su altura y capacidad. Hay que tener en cuenta que hay elementos que ocupan una unidad pero también hay otros que ocupan más de una unidad en el armario.

Cuando se incluye un elemento de tipo armario en el plano de la estancia, por defecto se muestra su visión cenital indicando en una etiqueta el número máximo de unidades que permite y el número de unidades ocupadas. Al seleccionar el elemento se accederá a la pantalla de edición y visualización de los elementos que están incluidos dentro del armario. Al igual que en activos simples los activos de los armarios pueden ser de distintos tipos y por lo tanto tendremos la opción de seleccionar de una lista con imágenes los dispositivos que se precise. Estos

dispositivos podrán enlazarse a la posición de la unidad que ocupan en el armario. Finalmente, cada uno de estos elementos del armario también tendrá una serie de propiedades básicas para su identificación similares a las ya comentadas para elementos simples.

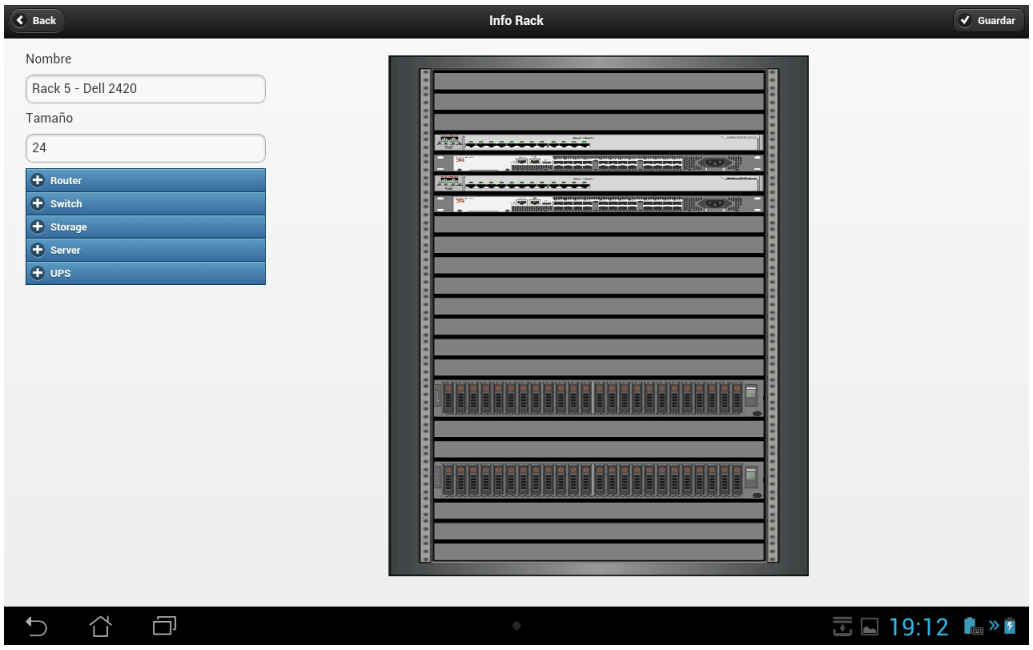


Figura 75. Visualización de un armario de equipamientos, los dispositivos que contiene y su posición física

Esta herramienta accede y almacena toda su información en una base datos local SQLite, facilitando de este modo el modelado de las estancias sin necesidad de conexión a la red de datos [SQLite 2015]. Para gestionar el acceso a los ficheros y a la base de datos local del dispositivo móvil se utilizan las funciones del API de PhoneGap. Estas funciones siguen las especificaciones *W3C Web SQL Database Specification* y *W3C Web Storage API Specification* [W3C 2015].

Cuando el dispositivo móvil establece la conexión de red, el usuario puede realizar el proceso de sincronizar y consolidar la información almacenada en la base de datos local en el servidor central definido. El servidor central será definido en el archivo de configuración de la aplicación mediante su dirección IP y las credenciales de acceso (nombre de usuario, contraseña). La comunicación y sincronización de datos se realizará utilizando el formato JSON (*JavaScript Object Notation*) [JSON 2015], que sigue la especificación ECMA-404 (*JSON Data Interchange Standard*) [ECMA 2015]. El servidor central dispondrá de una

funcionalidad, desarrollada en lenguaje PHP, que se encargará de recibir y procesar el envío de un dispositivo móvil y devolverá el resultado final de la operación. Del mismo modo, el servidor central también dispone de otra funcionalidad que permite realizar la actualización completa de la base de datos de un dispositivo móvil con los datos provenientes del servidor central.

La generación de informes permite al personal técnico llevar un control de los elementos disponibles en cada estancia o generar un registro con su estado. Estos informes se muestran en pantalla pero también pueden ser almacenados en formato PDF en el dispositivo móvil. Para la generación de estos documentos se ha utilizado la librería jsPDF que es una librería JavaScript para HTML5 para generar archivos en formato PDF [Parallax 2015]. El manejo de los archivos y directorios dentro del sistema de ficheros del dispositivo móvil se gestiona con las funciones del API de PhoneGap, estas funciones están basadas en el estándar *W3C File API* [W3C 2015].

Inicialmente se planteó la posibilidad de implementar una representación gráfica 3D interactiva de los espacios y elementos. Esto es posible utilizando las especificaciones HTML5 y WebGL. La especificación WebGL es un API JavaScript para el renderizado interactivo de gráficos 2D/3D en navegadores Web compatibles y sin necesidad de utilizar plugins. WebGL está basada en la especificación OpenGL ES y para su funcionamiento debe ser soportado por el dispositivo ya que utiliza aceleración gráfica por GPU. Debido a estos requisitos hardware y software, que limitaban los dispositivos móviles que se podrían utilizar para ejecutar la aplicación, y a que se consideró que la representación 3D no aportaba un valor añadido importante se descartó esta alternativa.

8.5. CONCLUSIONES

La aplicación desarrollada tiene como finalidad crear una herramienta de modelado de espacios físicos en espacios virtuales para su integración con un sistema de supervisión de sistemas y redes de datos. Mediante esta herramienta, orientada principalmente a dispositivos móviles, se ofrece al personal técnico la posibilidad de generar y utilizar planos informativos que reflejen las ubicaciones disponibles y los activos que contienen en cada una de ellas. La integración con el

sistema de supervisión central permitirá obtener la información básica sobre el estado de un equipamiento junto con su posición en la representación gráfica del espacio físico, para facilitar la localización real del activo y mejorar la respuesta ante incidencias.

Esta herramienta puede ser utilizada como visor o como editor, dependiendo de los permisos y necesidades del usuario. De este modo, es posible utilizar la aplicación como una interfaz alternativa para la gestión de los equipamientos disponibles desde una determinada estancia utilizando un dispositivo móvil.

Se ha contemplado la posibilidad de trabajar de forma conectada o desconectada de tal modo que se pueda modelar un espacio sin necesidad de conexión a la red de datos. Cuando la aplicación está conectada, la información almacenada en el dispositivo móvil se puede sincronizar con la existente en el servidor central, para consolidar la información registrada por un usuario. Finalmente, desde la aplicación instalada se podrá obtener la información consolidada en el servidor central para que pueda ser actualizada y utilizada por cualquiera de los dispositivos móviles del CPD destinados a esta labor.

Los sistema de supervisión existentes en el mercado disponen, en el mejor de los casos, de representaciones esquemáticas de los equipamientos existentes pero no proporcionan una visión espacial sobre la posición real que ocupan lo cual dificulta la localización en CPDs de gran tamaño o muy distribuidos. Con este desarrollo se ha experimentado con nuevas formas de representación gráfica para intentar mejorar los sistemas de monitorización existentes. Además, se han podido aprovechar las características y sensores que nos proporcionan los dispositivos móviles para dotar al personal técnico de una herramienta de trabajo portátil y fácil de utilizar (interfaz táctil, cámara y reconocimiento de códigos QR, etc.).

9 Conclusiones

9.1. Resultados

Los sistemas de telemonitorización en tiempo real, independientemente de su ámbito de aplicación, son herramientas básicas para la ayuda en el trabajo diario del personal especializado. Las tareas de monitorización y supervisión requieren de un elevado nivel de atención, y en muchos casos el volumen de información manejada hace que sea imprescindible gestionar e interpretar de forma adecuada el modo en que se visualiza.

Mediante el estudio exhaustivo de los parámetros bajo monitorización se consigue realizar una clasificación de la información, estableciendo cuales son los parámetros críticos que habrá que seguir bajo supervisión. Además, se definirán los umbrales de funcionamiento y valores posibles de cada uno de estos parámetros con el objetivo de detectar anomalías en estos valores. Finalmente, se consigue establecer las posibles relaciones entre los parámetros monitorizados.

El resultado del estudio de los valores recibidos por el sistema de supervisión permite definir mecanismos, como los sistemas basados en el conocimiento, para establecer reglas capaces de interpretar la información recibida y sugerir posibles acciones ante valores anómalos. De este modo, se proporciona al personal técnico un sistema de notificaciones en tiempo real para la ayuda a la detección y resolución de problemas.

Para conseguir el objetivo final de facilitar el trabajo y optimizar los tiempos de respuesta ante incidentes en un sistema bajo telemonitorización en tiempo real es

necesario conseguir que las interfaces de visualización sean capaces de fusionar y visualizar toda esta información de forma adecuada. Esto implica un esfuerzo importante para conocer el ámbito de aplicación concreto y requiere de la ayuda funcional del usuario para el diseño de la interfaz gráfica.

En muchos casos, las interfaces gráficas de monitorización se basan en la representación descriptiva o esquemática de la información. Si bien estos sistemas pueden ser suficientes, las tecnologías actuales nos proporcionan nuevas capacidades gráficas que pueden mejorar o complementar a los sistemas tradicionales. Así, hemos tratado de probar nuevas formas de visualización e interacción con el usuario para ayudarle en su operativa diaria. La inclusión de sistemas de visualización 3D interactivos trata de acercar al usuario al entorno real bajo supervisión, utilizando representaciones virtuales. El objetivo es conseguir una visualización más cercana a la realidad de los elementos monitorizados y facilitar también la detección de incidencias, mostrando la ubicación física del problema.

La utilización de dispositivos móviles en tareas de telemonitorización abre un nuevo campo de trabajo para el diseño de interfaces avanzadas. Aquí se han probado técnicas como la Realidad Aumentada que permite simplificar la comprobación del estado actual de un elemento bajo supervisión mediante la superposición en tiempo real de una capa informativa sobre su imagen física. También se ha experimentado con la posibilidad de registrar de forma dinámica y visual los elementos monitorizados, utilizando un dispositivo móvil, para generar un inventario completo e interactivo. Conectando este inventario con el sistema de telemonitorización existente también se consigue aportar una interfaz complementaria para la supervisión móvil de las infraestructuras informáticas.

En todo el proceso de diseño de estas herramientas es imprescindible una participación activa del personal especializado encargado de su utilización final. Así, se podrán plantear los sucesivos prototipos capaces de cumplir con las funcionalidades necesarias, adaptados a la operativa diaria de los usuarios del sistema, consiguiendo no sólo interfaces interactivas avanzadas sino también interfaces funcionales capaces de simplificar las tareas de supervisión en tiempo real. En las experimentaciones realizadas se han implantado herramientas capaces

de proporcionar al personal técnico nuevas interfaces centradas en la usabilidad y simplicidad de su interacción. La evaluación de sus funcionalidades por parte del personal técnico, que ha colaborado en su diseño y pruebas, ha demostrado su eficacia para la mejora de las tareas de monitorización en tiempo real.

9.2. CONCLUSIONES

Como conclusiones finales de este trabajo de tesis doctoral podemos destacar:

- Se ha desarrollado una herramienta, basada en agentes inteligentes, que trata de optimizar el uso del ancho de banda disponible en una red interna. De este modo, se consigue automatizar la distribución de tareas, teniendo en cuenta su prioridad, para no interferir con el tráfico correspondiente a las funciones propias de la monitorización.
- Se ha integrado la herramienta anteriormente citada en un sistema de Telemedicina en UCIs como un complemento a sus funcionalidades. Este sistema de Telemedicina incluye también un módulo de visualización basadas en objetos 3D interactivos para la representación de la información sobre el estado del paciente.
- Se han aplicado técnicas de visualización 3D inmersivas para conseguir una interfaz de supervisión de sistemas basadas en una representación virtual del entorno. Mediante este tipo de interfaces se ha conseguido mejorar la interacción con el usuario y la localización física de los equipamientos bajo supervisión.
- Se ha creado una aplicación para dispositivos móviles basada en técnicas de Realidad Aumentada para simplificar el acceso a la información de un activo bajo supervisión desde su ubicación física. La superposición entre imagen real e información virtual permite al usuario visualizar el estado actual de un activo sin necesidad de desplazarse a una sala de control. Esta aplicación se ha demostrado muy eficaz para representar la información de forma rápida y simple.
- Se ha implementado una herramienta para dispositivos móviles para la creación y gestión de una representación gráfica interactiva de los espacios e infraestructuras bajo supervisión. Así, se consigue crear un inventario completo de las infraestructuras mientras se recorre físicamente el entorno. Esta representación se realiza mediante objetos 3D interactivos capaces de conectarse con el sistema de telemonitorización para convertirse también

en una completa interfaz de supervisión. La utilización de esta herramienta se ha mostrado como una alternativa eficaz para la modelización y monitorización de activos.

9.3. PUBLICACIONES

A continuación se incluyen las referencias bibliográficas a las principales publicaciones científicas realizadas durante el desarrollo de esta tesis doctoral, y en las que ha participado el autor.

Artículos, libros y capítulos de libros:

- C. Dafonte, A. Gómez, D. Ordóñez, B. Arcay. "*Applications of videoconferencing technologies in bioengineering*". Videoconferencing: Technology, Impact and Applications. pp. 57-79. Nova Science Publishers. New York (EE.UU.). 2010.
- C. Dafonte, A. Castro, A. Gómez, B. Arcay. "*Intelligent agents technology applied to tasks scheduling and communications management in a critical care telemonitoring system*". Computers in Biology and Medicine, vol. 37, n. 6, pp. 760-773. Elsevier. 2007.
- C. Dafonte, A. Gómez, A. Castro, B. Arcay. "*Sistema de Supervisión Distribuido en Tiempo Real Intra e Inter-Hospitalario para Unidades Críticas*". El Futuro de Internet. Acceso y Teleservicios, pp. 89-107. Fundación Alfredo Brañas. Santiago de Compostela (España). 2002.
- A. Gómez, B. Arcay, C. Dafonte, A. Castro. "*Seguridad en Redes Hospitalarias*". Protección y Seguridad de la Información, pp. 185-208. Fundación Alfredo Brañas. Santiago de Compostela (España). 2002.
- B. Arcay, C. Dafonte, A. Gómez, A. Castro. "*Introducción a la Telemedicina y los Entornos Inmersivos*". Ingeniería del software en aplicaciones 3D. Entornos inmersivos e interactivos, pp. 39-70. Fundación Alfredo Brañas. Pontevedra (España). 1999.
- C. Dafonte, B. Arcay, A. Castro, A. Gómez, A. Rodríguez. "*Nuevas Tecnologías de las Telecomunicaciones en el ámbito de la Informática Biomédica*". Avances en Informática Biomédica, pp. 11-37. Universidade da Coruña-Servicio de publicaciones. A Coruña (España). 1999.

Comunicaciones en congresos internacionales:

- A. Gómez, C. Dafonte, B. Arcay. "*3D Visualization for System and Networks Monitoring Support*". 3rd International Conference on Human System Interaction (HSI10), pp. 145-148. Rzeszów (Polonia). 2010.
- A. Gómez, D. Cantorna, C. Dafonte, B. Arcay. "*FIPA-OS Agents Applied to Process Scheduling in Real Time Monitoring*". First International Conference on Informatics in Control, Automation and Robotics (ICINCO2004), pp. 70-75. Setúbal (Portugal). 2004.
- C. Dafonte, A. Gómez, A. Rodríguez, B. Arcay. "*Intelligent Data Communications and Tasks Management in a ICU Telesupervision System*". World Congress on Medical Physics and Biomedical Engineering (WC2003 IFMBE), pp. 1-4. Sidney (Australia). 2003.
- C. Dafonte, A. Gómez, A. Castro, B. Arcay. "*A proposal for improving ICU assistance through intelligent monitoring and supervision*". 7th Annual World Conference on the Internet and Medicine (MEDNET 2002), pp. 464-466. Technology and Health Care. Amsterdam (Paises Bajos). 2002.
- A. Gómez, C. Dafonte, B. Arcay, A. Castro, J. Pereira. "*Real-Time 3D Monitoring for Telemedicine in Critical Units*". 2nd European Medical & Biological Engineering Conference (EMBEC 2002), pp. 1352-1353. Viena (Austria). 2002.
- J.C. Dafonte, A. Gomez, A. Castro, B. Arcay. "*Intelligent Agents Technology applied to Tasks Control in ICU Telesupervision*". 24th Annual International Conference of the EMBS and Annual Fall Meeting of the BMES (EMBS 2002), pp. 1861-1862. Houston-Texas (EE.UU.). 2002.
- A. Gomez, C. Dafonte, B. Arcay, A. Castro. "*Intelligent Control and Visualization in Clinical Telesupervision*". 2nd IASTED International Conference, Artificial Intelligence and Applications (AIA 2002), pp. 87-92. Benalmádena-Málaga (España). 2002.

- C. Dafonte, A. Gomez, B. Arcay, A. Castro, J. Pereira. "*3D Visualization Module in a Telemedicine Project*". 15th International Conference on Computer Based Medical Systems (CBMS'02), pp. 193-198. Maribor (Eslovenia). 2002.
- A. Gomez, C. Dafonte, B. Arcay, A. Rodriguez. "*Advanced Visualization in a Clinical Telemonitoring System*". 20th IASTED International Conference, Applied Informatics (AI'02), pp. 367-372. Innsbruck (Austria). 2002.
- A. Gómez, C. Dafonte, B. Arcay, A. Rodríguez, A. Castro. "*Knowledge-based agents for tasks Management in a Telesupervision System in Critical Care*". 6th World Congress on the Internet in Medicine (MEDNET 2001), pp. 485-487. Technology and Health Care. Udine (Italia). 2001.
- A. Gómez, C. Dafonte, B. Arcay, A. Castro. "*Intelligent Agents for the Control of Processes in a Telemedicine System*". IASTED International Conference, Artificial Intelligence and Applications (AIA 2001), pp. 382-387. Marbella (España). 2001.
- C. Dafonte, A. Gómez, B. Arcay, J.A. Taboada. "*Intelligent Management of Processes in a ICU Telemedicine System*". World Congress on Medical Physics and Biomedical Engineering (IEEE-EMBS), pp. 1-4. Chicago (EE.UU.). 2000.
- B. Arcay, C. Dafonte, A. Gómez, A. Rodríguez. "*Intelligent Scheduling of Processes and Data Transmission in a Telemedicine System*". IASTED International Conference, Signal Processing and Communications (SPC 2000), pp. 513-518. Marbella (España). 2000.
- A. Gomez, J.C. Dafonte, B. Arcay, A.. Santos. "*Intelligent System for the Acquisition and Telemonitoring in Intensive Care Units*". 18th IASTED International Conference, Applied Informatics (AI'00), pp. 397-401. Innsbruck (Austria). 2000.
- C. Dafonte, B. Arcay, A. Gómez, J. Flores. "*A prototype of intelligent telemedicine system in Intensive Care Units*". 17th IASTED International Conference, Applied Informatics (AI'99), pp. 330-332. Innsbruck (Austria). 1999.

9.4. TRABAJOS FUTUROS

En el momento de escribir este trabajo, nuestro grupo de investigación sigue realizando nuevos proyectos relacionados con el campo de la monitorización en tiempo real y la visualización avanzada. Estos desarrollos se están aplicando a diversos ámbitos como la gestión de infraestructuras informáticas o la investigación astrofísica.

Concretamente, podemos citar desarrollos encaminados a aprovechar la potencia y generalización de estándares Web como HTML5 y CSS3, unidos al lenguaje JavaScript, para crear interfaces avanzadas utilizando nuevas tecnologías gráficas. Este es el caso del uso de librerías JavaScript para la gestión de gráficos vectoriales (SVG) o motores 3D en JavaScript para el desarrollo de aplicaciones con WebGL.

Un ejemplo de estas nuevas aplicaciones lo constituye la implantación de un sistema de visualización y análisis de datos astrofísicos para el proyecto *Gaia*. La misión espacial *Gaia* tiene como objetivo principal la creación de un mapa extremadamente preciso de la Vía Láctea y por lo tanto implicará el procesamiento de una gran cantidad de información procedente de la observación de mil millones de objetos con sus correspondientes datos astrométricos, fotométricos, y espectroscópicos. Para ello, se utilizarán técnicas de aprendizaje no supervisado para la clasificación de objetos estelares, entre otras los mapas auto-organizados (SOM). La representación gráfica adecuada de estos mapas proporcionará al personal experto una mejor comprensión de los resultados obtenidos. En este caso las interfaces de visualización utilizarán gráficos vectoriales interactivos con representaciones cuadradas y hexagonales de los mapas, representaciones tridimensionales e interactivas, esquemas de colores parametrizables, ajustes de niveles visualizados, etc.

10
Bibliografía

[Akkermans 2004] Akkermans, H., Gustavsson, R., Ygge, F. (2004). *Structured engineering process for agent communication modelling*. Knowledge engineering and agent technology, pp.216-236. IOS Press.

[Allen 2008] Allen, C., Arnold, W., Balkan, A., Cannasse, N., Grden, J., Gunesch, M., Hughes, M., Jon MacDonald, R., Zupko, A. (2008). *The essential guide to open source Flash development*. Apress.

[Allen 2010] Allen, G., Owens, M. (2010). *The definitive guide to SQLite*. Apress.

[Alternativa3D 2012] AlternativaPlatform. (2012). *Alternativa3D*. Disponible en: <http://alternativaplatform.com/en/>.

[AMIA 1999]. AMIA - American Medical Informatics Association. (1999). *1999 annual conference*. AMIA.

[Amin 2015] Amin, D., Govilkar, S. (2015). *Comparative study of augmented reality SDK's*. International Journal on Computational Sciences & Applications (IJCSA), vol.5, num.1, 2015.

[Android 2015] Google Android. (2015). *Android platform*. Disponible en: <https://www.android.com>.

[Anyuru 2012] Anyuru, A. (2012). *Professional WebGL programming: Developing 3D graphics for the Web*. Wrox.

[AR-media 2015] AR-media. (2015). *ARmedia 3D SDK*. Disponible en: <http://dev.inglobetechnologies.com>.

[Arcay 1990] Arcay, B. (1990). *Sistema de supervisión dinámica y control en U.C.I. basado en técnicas de inteligencia artificial*. (Tesis doctoral). Dep. de Física Aplicada. Universidad de Santiago de Compostela.

[ARToolKitPlus 2006] ARToolKitPlus. (2006). *ARToolKitPlus*. Disponible en: <http://handheldar.icg.tugraz.at/artoolkitplus.php>.

[Aurasma 2015] Aurasma. (2015). *Aurasma*. Disponible en: <http://www.aurasma.com>.

[Away3D 2013] Away3D. (2013). *Away3D*. Disponible en: <http://away3d.com>.

[Azuma 1997] Azuma, R.T. (1997). *A survey of augmented reality*. Presence: Teleoperators and virtual environments, vol.6, num.4, pp.355-385. MIT Press.

[Azuma 2001] Azuma, R.T., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B. (2001). *Recent advances in augmented reality*. IEEE Computer Graphics and Applications, vol.21, num.6, pp.34-47. IEEE.

[Barth 2008] Barth,W. (2008). *Nagios: System and network monitoring*. No Starch Press.

[Bimber 2005] Bimber, O., Raskar, R. (2005). *Spatial Augmented Reality: Merging real and virtual worlds*. CRC Press.

[Bird 1975] Bird, K.T. (1975). *Telemedicine: Concept and practice*. Telemedicine: Explorations in the use of telecommunication in health care. Springfield - Illinois (Estados Unidos).

[Bowman 2005] Bowman, D.A., Kruijff, E., LaViola, J.J., Poupyrev, I. (2005). *3D user interfaces: Theory and practice*. Addison-Wesley.

[Braunstein 2010] Braunstein, R. (2010). *ActionScript 3.0 bible*. Wiley Publishing.

[Burdea 2003] Burdea, G.C., Coiffet, P. (2003). *Virtual reality technology*. John Wiley & Sons, Inc.

[Cacti 2015] Cacti. (2015). *Cacti software*. Disponible en: <http://www.cacti.net>.

[Campi 2008] Campi, N., Bauer, K. (2008). *Automating Linux and Unix system administration*. Apress.

[Cawood 2008] Cawood, S., Fiala, M. (2008). *Augmented Reality: A practical guide*. Pragmatic Bookshelf.

[CECAT 1996] Committee on Evaluating Clinical Applications of Telemedicine. (1996). *Telemedicine: A guide to assessing telecommunications in health care*. National Academy Press. Washington D.C. (Estados Unidos).

[COLLADA 2012] COLLADA. (2012). *COLLADA digital asset schema specification for 3D visualization of industrial data (ISO/PAS 17506:2012)*. ISO-International Organization for Standardization. Disponible en: <http://www.iso.org>.

[ColladaMaya 2009] ColladaMaya. (2015). *ColladaMaya-COLLADA plug-ins for Maya and 3ds Max*. Disponible en: <http://colladamaya.sourceforge.net>.

[Conrath 1983] Conrath, D.W., Dunn, E.V., Higgins, C.A. (1983). *Evaluating telecommunications technology in medicine*. Artech House. Dedham - Massachusetts (Estados Unidos).

[Cortona3D 2015] Cortona3D. (2015). *Cortona3D viewer*. Disponible en: <http://www.cortona3d.com>.

[Craig 2013] Craig, A.B. (2013). *Understanding Augmented Reality: Concepts and applications*. Morgan Kaufmann.

[Crytek 2015] Crytek CryENGINE. (2015). *Crytek CryENGINE*. Disponible en: <http://www.crytek.com/cryengine>.

[Dafonte 2002] Dafonte, J.C. (2002). *Gestión inteligente de procesos, datos y comunicaciones en un sistema de telesupervisión en tiempo real*. (Tesis doctoral). Dep. de Tecnologías de la Información y las Comunicaciones. Universidade da Coruña.

[Dafonte 2007] Dafonte, J.C., Castro, A., Gómez, A., Arcay, B. (2007). *Intelligent agents technology applied to tasks scheduling and communications management in a critical care telemonitoring system*. Computers in Biology and Medicine, vol.37, num.6, pp.760-773. Elsevier.

[DAQRI 2015] DAQRI. (2015). *ARToolKit SDK*. Disponible en: <http://artoolkit.org>.

[David 2011] David, M. (2011). *Flash mobile: Developing Android and iOS applications*. Elsevier-Focal Press.

[DirectX 2015] Microsoft DirectX. (2015). *Microsoft DirectX*. Disponible en: <http://www.microsoft.com>.

[ECMA 2015] ECMA International. (2015). *ECMAScript standard*. Disponible en: <http://www.ecma-international.org>.

[Farley 2011] Farley, A. (2011). *PHP-Zabbix-API*. Disponible en: <https://github.com/AndrewFarley/PHP-Zabbix-API>.

[Ferber 1999] Ferber, J. (1999). *Multi-agent systems. An introduction to distributed artificial intelligence*. Addison Wesley.

[FIPA 2002] FIPA - IEEE Foundation for Intelligent Physical Agents. (2002). *FIPA specifications*. Disponible en: <http://www.fipa.org>.

[FIPA-OS 2003] FIPA-OS. (2003). *FIPA-OS software*. Disponible en: <http://fipa-os.sourceforge.net>.

[Firtman 2012] Firtman, M. (2012). *jQuery Mobile. Aplicaciones HTML5 para móviles*. O'Reilly.

[Flash 2015] Adobe Flash. (2015). *Adobe Flash platform*. Disponible en: <http://www.adobe.com>.

[Forgy 1982] Forgy, C.L. (1982). *Rete: A fast algorithm for the many pattern/many object pattern match problem*. Artificial Intelligence, vol.19, num.1, pp.17-37. Elsevier.

[Frain 2012] Frain, B. (2012). Responsive Web Design with HTML5 and CSS3. Packt Publishing.

[Franklin 1989] Franklin, D.F., Ostler, D.V. (1989). *The P1073 medical information bus*. IEEE Micro, pp.52-60. IEEE.

[Friedman-Hill 2003] Friedman-Hill, E. (2003). *Jess in action: Rule based systems in Java*. Manning Publications.

[Gailly 2013] Gailly, J.L., Adler, M. (2013) *ZLIB documentation and sources*. Disponible en: <http://www.zlib.net>.

[Gamma 1995] Gamma, E., Helm, R., Johnson, R., Vlissides, J.M. (1995). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.

[Gifford 2012] Gifford, M. (2012). *PhoneGap mobile application development cookbook*. Packt Publishing.

[Ginsburg 2014] Ginsburg, D., Purnomo, B., Shreiner, D., Munshi, A. (2014). *OpenGL ES 3.0 Programming Guide*. Addison Wesley.

[Gómez 1999] Gómez, A. (1999). *Sistema de información para telesupervisión colaborativa en unidades de cuidados intensivos*. (Tesis de licenciatura). Dep. de Computación. Universidade da Coruña.

[Grubert 2013] Grubert, J., Grasset, R. (2013). *Augmented Reality for Android application development*. Packt Publishing.

[HammerJS 2015] Hammer.JS. (2015). *Hammer.JS JavaScript library*. Disponible en: <http://hammerjs.github.io>.

[Handheldar 2015] Handheld AR. (2015). *Christian Doppler Laboratory for Handheld Augmented Reality*. Graz University of Technology. Disponible en: <http://handheldar.icg.tugraz.at>.

[Hay 2013] Hay, S. (2013). *Responsive design workflow*. New Raiders.

[IEEE 1993] IEEE. (1993). *IEEE Standard 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology*. IEEE.

[ImagePicker 2014] ImagePicker. (2014). *Image Picker jQuery plugin*. Disponible en: <http://rvera.github.io/image-picker/>.

[ISOICS 2015] ISO. (2015). *ICS 35.240.80: IT applications in health care technology*. ISO. Disponible en: <http://www.iso.org>.

[ISOIEC 2015] ISO. (2015). *ISO/IEC 27000: Information Security Management Systems*. Disponible en: <http://www.iso.org>.

[ITU-T 2005] ITU-T. (2005). *ITU-T Recommendation H.263. Video coding for low bit rate communication*. Disponible en: <http://www.itu.int/rec/T-REC-H.263>.

[Java3D 2008] Java3D. (2008). *Java3D API*. Disponible en: <https://java3d.java.net>.

[JESS 2013] JESS. (2013). *JESS rule engine software*. Disponible en: <http://herzberg.ca.sandia.gov>.

[Josephsen 2007] Josephsen, D. (2007). *Building a monitoring infrastructure with Nagios*. Prentice Hall.

[Josephsen 2013] Josephsen, D. (2013). *Nagios: Building enterprise-grade monitoring infrastructures for systems and networks*. Prentice Hall.

[jQuery 2015] jQuery. (2015). *jQuery JavaScript library*. Disponible en: <https://jquery.com>.

[jQueryMobile 2015] jQueryMobile. (2015). *jQueryMobile JavaScript library*. Disponible en: <https://jquerymobile.com>.

[JSON 2015] JSON. (2015). *JSON (JavaScript Object Notation)*. Disponible en: <http://json.org>.

[Junaio 2015] Junaio. (2015). *Junaio AR Browser*. Disponible en: <http://www.junaio.com>.

[Kampmann 1989] Kampmann, J., Hernandez, C., Schwarzer, E. (1989). *Process control in intensive care - An integrated working place*. Proc. 11th Int. Conf. of the IEEE Engineering in Engineering in Medicine and Biology Society, pp.1215-1216, Seattle (Estados Unidos).

[KineticJS 2015] KineticJS. (2015). *KineticJS JavaScript library*. Disponible en: <http://kineticjs.com>.

[Kipper 2012] Kipper, G., Rampolla, J. (2012). *Augmented Reality: An emerging technologies guide to AR*. Syngress.

[Kocjan 2008] Kocjan, W. (2008). *Learning Nagios 3.0*. Packt publishing.

[Komatineni 2012] Komatineni, S., MacLean, D. (2012). *Pro Android 4*. Apress.

[Kretchmar 2003] Kretchmar, J.M. (2003). *Open source network administration*. Prentice Hall.

[LaGrone 2013] LaGrone, B. (2013). *HTML5 and CSS3 Responsive Web Design cookbook*. Packt Publishing.

[Layar 2015] Layar. (2015). *Layar*. Disponible en: <http://www.layar.com>.

[Lively 2010] Lively, M. (2010). *Professional Papervision3D*. Wrox.

[Lott 2007] Lott, J., Patterson, D. (2007). *ActionScript 3. Patrones de diseño*. Anaya Multimedia.

[Lunny 2011] Lunny, A. (2011). *PhoneGap beginner's guide*. Packt Publishing.

[Madden 2011] Madden, L. (2011). *Professional augmented reality browsers for smartphones: Programming for Junaio, Layar, and Wikitude*. Wrox.

[Maes 1994] Maes, P. (1994). *Social interface agents: Acquiring competence by learning from users and other agents*. AAAI Technical Report SS-94-03. AAAI Press.

[Marcotte 2010] Marcotte, E. (2010). *Responsive Web Design*. A List Apart Magazine.

[Marcotte 2011] Marcotte, E. (2011). *Responsive Web Design*. A Book Apart.

[Matsuda 2013] Matsuda, K., Lea, R. (2013). *WebGL programming guide: Interactive 3D graphics programming with WebGL*. Addison-Wesley.

[Mauro 2005] Mauro, D., Schmidt, K. (2005). *Essential SNMP*. O'Reilly.

[Metaio 2015] Metaio. (2015). *Metaio SDK*. Disponible en: <http://dev.metaio.com>.

[Moock 2007] Moock, C. (2007). *Essential ActionScript 3.0*. O'Reilly.

[MRTG 2015] MRTG. (2015). MRTG - Multi Router Traffic Grapher. Disponible en: <http://oss.oetiker.ch/mrtg/>.

[MySQL 2015] Oracle MySQL. (2015). *MySQL*. Disponible en: <https://www.mysql.com>.

[Nagios 2015] Nagios. (2015). *Nagios monitoring system*. Disponible en: <https://www.nagios.org>.

[Nagios-plugins 2014] Nagios-plugins. (2014). *Nagios plugin development guidelines*. Disponible en: <https://nagios-plugins.org/doc/guidelines.html>.

[Naur 1969] Naur, P., Randall, B. (1969). Software Engineering: Report on a conference sponsored by the NATO Science Committee. NATO.

[NINO 2015] NINO. (2015). *NINO network management solution*. Disponible en: <http://nino.sourceforge.net>.

[O3D 2009] O3D. (2009). *O3D*. Disponible en: <https://code.google.com/p/o3d/>.

[Octaga 2015] Octaga. (2015). *Octaga player*. Disponible en: <http://www.octagavs.com>.

[Olsson 2010] Olsson, R., Bateman, R. (2010). *The essential guide to 3D in Flash*. Apress.

[Olups 2010] Olups, R. (2010). *Zabbix 1.8 network monitoring*. Packt Publishing.

[OpenGL 2015] OpenGL. (2015). *OpenGL standard*. Khronos Group. Disponible en: <https://www.opengl.org>.

[OpenNMS 2015] OpenNMS. (2015). *OpenNMS project*. Disponible en: <http://www.opennms.org>.

[PandoraFMS 2015] PandoraFMS. (2015). *PandoraFMS*. Disponible en: <http://pandorafms.com>.

[Papervision3D 2009] Papervision3D. (2009). Papervision3D. Disponible en: <https://code.google.com/p/papervision3d/>.

[Parallax 2015] Parallax. (2015). *jsPDF JavaScript library*. Disponible en: <https://parall.ax/products/jspdf>.

[Parisi 2012] Parisi, T. (2012). *WebGL: Up and running*. O'Reilly.

[Parisi 2014] Parisi, T. (2014). *Programming 3D applications with HTML5 and WebGL: 3D animation and visualization for Web pages*. O'Reilly.

[Peterson 2014] Peterson, C. (2014). *Learning Responsive Web Design. A beginner's guide*. O'Reilly.

[PhoneGap 2015] PhoneGap. (2015). *PhoneGap framework*. Disponible en: <http://phonegap.com>.

[PolygonCruncher 2009] Polygon Cruncher. (2015). Mootools Polygon Cruncher. Disponible en: <http://www.mootools.com/plugins/us/polygoncruncher/>.

[Poslad 2000] Poslad, S., Buckle, P., Hadingham, R. (2000). *The FIPA-OS agent platform: Open source for open standards*. Proceedings of PAAM 2000. Manchester (Reino Unido).

[Poslad 2007] Poslad, S. *Specifying protocols for multi-agent system interaction*. (2007). ACM Transactions on Autonomous and Adaptive Systems, vol.2, num.4. ACM.

[Preim 2013] Preim, B., Botha, C.P. (2013). *Visual computing for medicine: Theory, algorithms, and applications*. Morgan Kaufmann.

[Pressman 2010] Pressman, R.S. (2010). *Ingeniería del software. Un enfoque práctico*. McGraw-Hill.

[QRcode 2015] jQuery.QRcode. (2015). *jQuery.QRcode plugin*. Disponible en: <https://larsjung.de/jquery-qrcode/>.

[Reinhardt 2009] Reinhardt, R., Dowd, S. (2009). *Flash CS4 professional bible*. Wiley Pub.

[RRDtool 2015] RRDtool. (2015). *RRDtool software*. Disponible en: <http://oss.oetiker.ch/rrdtool/>.

[Russell 2010] Russell, S., Norvig, P. (2010). *Artificial Intelligence: A modern approach*. Prentice-Hall.

[Schiesser 2010] Schiesser, R. (2010). *IT Systems Management*. Pearson Education Inc.

[Schreiber 1999] Schreiber, G., Akkermans, H., Anjewierden, A., De Hoog, R., Shadbolt, N.R., Van de Velde, W., Wielinga, B.J. (1999). *Knowledge engineering and management: The CommonKADS methodology*. MIT Press.

[Schubert 2008] Schubert, M., Bennett, D., Gines, J., Hay, A., Strand, J. (2008). *Nagios 3 enterprise network monitoring. Including plug-ins and hardware devices*. Syngress.

[Schwarz 2013] Schwarz, R., Dutson, P., Steele, J., To, N. (2013). *The Android developer's cookbook: Building applications with the Android SDK*. Addison-Wesley.

[Selman 2002] Selman, D. (2002). *Java 3D programming*. Manning Publications Co.

[Shortliffe 1984] Shortliffe, T. (1984). *The science of biomedical computing*. Medical informatics, vol.9, num.3/4, pp.185-193. North-Holland.

[Shotts 2014] Shotts, K. (2014). *PhoneGap 3.x mobile application development*. Packt Publishing.

[Shreiner 2013] Shreiner, D., Sellers, G., Kessenich, J.M., Licea-Kane, B.M. (2013). *OpenGL programming guide: The official guide to learning OpenGL, version 4.3*. Addison-Wesley Professional.

[Silverlight 2015] Microsoft Silverlight. (2015). *Microsoft Silverlight*. Disponible en: <http://www.microsoft.com>.

[Smithwick 2012] Smithwick, M., Verma, M. (2012). *Pro OpenGL ES for Android*. Apress.

[Sommerville 2012] Sommerville, I. (2012). *Ingeniería de software*. Pearson Educación.

[Sood 2012] Sood, R. (2012). *Pro Android Augmented Reality*. Apress.

[SQLite 2015] SQLite. (2015). *SQLite software*. Disponible en: <https://www.sqlite.org>.

[Sriparasa 2013] Sriparasa, S.S. (2013). *JavaScript and JSON essentials*. Packt Publishing.

[Studierstube 2014] Studierstube Tracker. (2014). *Studierstube Tracker*. Disponible en: <http://handheldar.icg.tugraz.at/stbtracker.php>.

[Tiwari 2010] Tiwari, S., Elrom, E., Schulze, C. (2010). *AdvancED Flex 4*. Apress.

[Tondeur 2009] Tondeur, P., Winder, J. (2009). *Papervision3D essentials*. Packt Publishing.

[Turnbull 2006] Turnbull, J. (2006). *Pro Nagios 2.0*. Apress.

[Unity 2015] Unity. (2015). Unity game engine. Disponible en: <https://unity3d.com>.

[Unreal 2015] Unreal. (2015). Unreal Engine. Disponible en: <https://www.unrealengine.com>.

[Vacche 2013] Vacche, A.D., Lee, S.K. (2013). *Mastering Zabbix*. Packt Publishing.

[Vacche 2015] Vacche, A.D., Lee, S.K. (2015). *Zabbix network monitoring essentials*. Packt Publishing.

[Ver Hague 2006] Ver Hague, J., Jackson, C. (2006). *Flash 3D: Animation, interactivity, and games*. Elsevier-Focal Press.

[VRML 1997] VRML. (1997). *VRML standard*. Disponible en: <http://www.web3d.org>.

[W3C 2015] W3C. (2015). *World Wide Web Consortium - W3C specifications*. Disponible en: <http://www.w3.org>.

[Wargo 2012] Wargo, J.M. (2012). *PhoneGap essentials: Building cross-platform mobile apps*. Pearson.

[WebGL 2015] WebGL. (2015). *WebGL standard*. Khronos Group. Disponible en: <https://www.khronos.org/webgl/>.

[Wei 2012] Wei, J. (2012). *Android Database Programming*. Packt Publishing.

[Wikitude 2015] Wikitude. (2015). *Wikitude*. Disponible en: <https://www.wikitude.com>.

[Wooldridge 2002] Wooldridge, M. (2002). *An introduction to multiagent systems*. John Wiley & Sons Ltd.

[X3D 2015] X3D. (2015). *X3D standard*. Disponible en: <http://www.web3d.org>.

[XML 2015] W3C. (2015). *XML standard*. Disponible en: <http://www.w3.org>.

[Zabbix 2015] Zabbix. (2015). *Zabbix monitoring platform*. Disponible en: <http://www.zabbix.com>.

[Zenoss 2015] Zenoss. (2015). *Zenoss monitoring solution*. Disponible en: <http://www.zenoss.com>.

Índice de figuras

Acrónimos

AJAX: JavaScript Asíncrono y XML / *Asynchronous JavaScript and XML*

API: Interfaz de Programación de Aplicaciones / *Application Programming Interface*

ARML: Lenguaje de Marcas de Realidad Aumentada / *Augmented Reality Markup Language*

CPD: Centros de Procesamiento de Datos / *Data Center (DC)*

DMI: Interfaz de Gestión de Escritorio / *Desktop Management Interface*

GPS: Sistema de Posicionamiento Global / *Global Positioning System*

HAL: Capa de Abstracción de Hardware / *Hardware Abstraction Layer*

IA: Inteligencia Artificial / *Artificial Intelligence (AI)*

IDE: Entorno de Desarrollo Integrado / *Integrated Development Environment*

IGS: Cirugía Guiada por la Imagen / *Image Guided Surgery*

KML: Lenguaje de Marcas Keyhole / *Keyhole Markup Language*

LAN: Red de Área Local / *Local Area Network*

MIB: Bus de Información Médica / *Medical Information Bus*

MRI: Imagen por Resonancia Magnética / *Magnetic Resonance Imaging*

MVC: Modelo-Vista-Controlador / *Model-View-Controller*

PET: Tomografía por Emisión de Positrones / *Positron Emission Tomography*

POI: Puntos de Interés / *Points Of Interest*

QR: Respuesta Rápida / *Quick Response*

RA: Realidad Aumentada / *Augmented Reality (AR)*

RV: Realidad Virtual / *Virtual Reality (VR)*

RWD: Diseño Web Adaptativo / *Responsive Web Design*

SDK: Kit de Desarrollo de Software / *Software Development Kit*

SIH: Sistema de Información Hospitalario / *Hospital Information System (HIS)*

SNMP: Protocolo Simple de Administración de Red / *Simple Network Management Protocol*

TAC: Tomografía Axial Computerizada / *Computerized Axial Tomography (CAT)*

TCE: Traumatismo CraneoEncefálico / *Traumatic Brain Injury (TBI)*

UCI: Unidad de Cuidados Intensivos / *Intensive Care Unit (ICU)*

WMI: Instrumental de Administración de Windows / *Windows Management Instrumentation*

XML: Lenguaje de Marcas Extensible / *eXtensible Markup Language*