Centro
Politécnico Superior
Universidad Zaragoza

Proyecto Fin de Carrera de Ingeniería en Informática

# Desarrollo de un Sistema de Información para la Red Española de Supercomputación


Víctor Civitani Monzón

Codirector: Guillermo Losilla Anadón
Codirector: Arturo Giner Gracia
Instituto de Biocomputación y Física de Sistemas Complejos
(BIFI)

Ponente: Eduardo Mena Nieto
Departamento de Informática e Ingeniería de Sistemas
Área de Lenguajes y Sistemas Informáticos


Abril 2011

Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Instituto Universitario de Investigación
Biocomputación y Física
de Sistemas Complejos
Universidad Zaragoza

"It always seems impossible,
until it is done."

Nelson Mandela

# Agradecimientos

A mis directores, Guillermo Losilla y Arturo Giner, que han logrado que me sienta integrado y a gusto en el BIFI desde el primer momento, por su atención y ayuda constantes, por esa actitud positiva que siempre me han transmitido.

A mi ponente Eduardo Mena, por su disposición y atención continua a todo tipo de consultas, por sus consejos e ideas que han simplificado tanto trabajo y que me han sacado de varios apuros.

A todos mis amigos que han estado ahí apoyándome y soportándome a lo largo de toda la carrera, Miguel, José Miguel, Carlos, Ruth, Esther, Álvaro, y todos los que me dejo en el tintero sin quererlo.

A todos mis compañeros y amigos del CPS, Eduardo, José Ignacio, Toni, Andy, Cynthia, Jose Antonio, Luis, Manu, José, Luis Javier, Luisa... tantos a quien recordar en tan poco espacio. No habría sido lo mismo sin ellos.

Por último, pero no por ello menos importante, a mis padres Santiago y Mª del Pilar, sin ellos todo ésto no habría sido posible, a mis hermanas Elisa y Raquel y a mis cuñados Jacobo y Javier, sin olvidarme de mis sobrinillas, todos ellos han estado siempre ahí, apoyándome, tanto en los buenos momentos como en los más difíciles y de tensión. A todos vosotros, gracias de corazón.

**Desarrollo de un sistema de información para la Red Española de Supercomputación**

## RESUMEN

El Proyecto Fin de Carrera que se detalla en esta memoria ha sido realizado en el Instituto de Biocomputación y Física de Sistemas Complejos (BIFI), dentro del área de supercomputación. Se ha iniciado en octubre del 2010 y su duración ha sido de unos seis meses. Para su realización se ha contado con la codirección de Guillermo Losilla y Arturo Giner, y con Eduardo Mena como ponente.

El objetivo final del proyecto ha sido diseñar e implementar un sistema de información que permita recopilar datos de los diferentes nodos de la Red Española de Supercomputación (RES) y mostrar dichos datos a los usuarios mediante un interfaz Web.

El sistema almacena la información que se recopila de los diferentes nodos y geoposiciona dicha información sobre un mapa. Posee una estructura modular y es interoperable, permitiendo ser usado en distintos sistemas operativos y navegadores Web, y provee un interfaz que permite un acceso Web seguro y con distintos niveles de autorización a la información almacenada en el sistema.

El proyecto ha constado de tres fases: la primera ha sido el estudio previo de las tecnologías, donde se ha invertido bastante tiempo, la segunda y tercera han sido diseño e implementación, que se han realizado casi a la par.

Durante la documentación se ha conseguido tener una idea global del funcionamiento de la RES, de forma que se han obtenido todos los detalles necesarios para la realización del proyecto. También se ha buscado información acerca de las posibles herramientas que se pueden utilizar durante el desarrollo. Se ha analizando la información obtenida, y se ha realizado una selección de las herramientas a usar, empezando así la fase de diseño, donde se ha decidido utilizar una base de datos de la cual los usuarios obtienen la información mediante un Applet de Java que usa World Wind para geolocalizar los datos, y los nodos son los encargados de actualizar la base de datos mediante el uso de scripts, de forma que es lo más automatizado posible.

Finalmente, el desarrollo ha consistido en hacer realidad el diseño previo mediante las herramientas elegidas, que se han procurado que fueran Software Libre. Por ello las herramientas utilizadas han sido la base de datos MySQL, la API de la NASA World Wind Java SDK, el lenguaje Java para realizar el applet principal de la interfaz, NetBeans, Subversion, PHPMyAdmin, Gimp, Dia, Ganttproject, OpenOffice y LaTeX.

Al finalizar el desarrollo del proyecto se han cumplido todos los objetivos iniciales, y se ha dejado el sistema en un estado estable que tiene grandes posibilidades con vistas al futuro.

# Índice general

# Índice de figuras

# Índice de tablas

# Parte I

# Memoria

# Capítulo 1

# Introducción

En este capítulo se realiza una introducción al sistema de información que se ha desarrollado. Se describe el contexto en el que se ha realizado el proyecto final de carrera, su alcance y objetivo, cómo se ha abordado el problema y finalmente, se concluye con una descripción del contenido del resto de secciones de la memoria.

## 1.1. Contexto del proyecto

Tras realizar en Suecia el último curso de la carrera, el autor de éste proyecto decidió buscar un PFC relacionado con las áreas que más le han gustado durante sus estudios. Mediante las listas de correo de la Universidad de Zaragoza (softlibre y púlsar concretamente) se puso en contacto con Guillermo Losilla y encontró un PFC bastante interesante, ya que le permitía realizar un proyecto de forma íntegra, donde tuvo la oportunidad de participar en todas las fases, desde el diseño inicial hasta la implementación final.

Por ello, el proyecto se ha realizado en las instalaciones del Instituto de Biocomputación y Física de Sistemas Complejos (BIFI) de la Universidad de Zaragoza, usando como base el nodo de la Red Española de Supercomputación (RES) "CAESARAUGUSTA", para posteriormente terminar de implementarse en el resto de los nodos.

La RES es una red de infraestructuras al servicio de la I+D en España en el área de la supercomputación, con núcleo en el supercomputador Marenostrum del BSC (Barcelona Supercomputing Center — Centro Nacional de Supercomputación). Fue creada por el MICINN (Ministerio de Ciencia e Innovación) en 2007, con el objetivo de dar soporte a los servicios que la comunidad científica demandaba en este campo.

Actualmente la RES está formada por 8 nodos, que están distribuidos por toda la geografía española. La Universidad de Zaragoza participa en la RES con el nodo "CAESARAUGUSTA", instalado en la Facultad de Ciencias y que es gestionado por el BIFI, un instituto de investigación universitario creado el 8 de Octubre de 2002, por decreto 311/2002 del Gobierno de Aragón, a instancia de la Universidad de Zaragoza.

Los nodos de la RES generan diariamente una cantidad ingente de información de diversa índole: usuarios y grupos asignados a cada nodo, disponibilidad y estado de los sistemas, monitorización de la red, *accounting*... Hasta hoy, dicha información se ha gestionado con varias herramientas de forma aislada, siendo parte de ella accesible sólo de forma local en cada nodo.

Dada esta situación, se ha decidido realizar un proyecto interno en la RES que consiste en recopilar toda esta información y hacerla accesible desde la Web, de forma segura y con distintos niveles de autorización a los datos. De forma que se pueda tener una visión general del estado de toda la RES de forma rápida y sencilla. Dicho proyecto interno ha sido asignado al BIFI, y ha sido el objetivo de este proyecto final de carrera.

## 1.2. Motivación

Los computadores están siendo de gran ayuda a la comunidad científica, gracias a ellos se pueden realizar simulaciones de los experimentos antes de realizaros, ahorrando tiempo y dinero. Además de los datos obtenidos en las simulaciones, también se genera una gran cantidad de información que ayuda a controlar el estado de las máquinas, especialmente importante cuando hablamos de cientos de procesadores trabajando conjuntamente en una misma tarea. Toda esta información es útil, pero si está dispersa se hace más complejo conocer el estado global de un sistema.

Además, si se dispone en España de una red de supercomputación, es interesante poder enseñar a los usuarios que el sistema funciona, para que vean las estadísticas y el entorno sobre el que trabajan, de forma que puedan disponer de una información actualizada y centralizada. También es interesante poder mostrar la RES a otra gente que no la usa pero que puede ser un usuario potencial, de manera que las personas puedan tener conocimiento de la existencia de la RES.

Pero hasta ahora no existía nada que permitiera lo anterior, toda la información estaba dispersa y no era posible dar a conocer la existencia de la RES de una forma visualmente atractiva. Dado que no se había realizado nada al respecto todavía, ha resultado ser un proyecto bastante estimulante, ya que ha permitido participar al autor por primera vez en todas las fases de un proyecto informático real.

## 1.3. Objetivos del proyecto

Dada la naturaleza del proyecto, los objetivos han estado marcados por los requerimientos del proyecto interno de la RES. Se requería un Sistema de Información específico para un entorno concreto perfectamente definido. Por lo tanto, los objetivos principales de este proyecto han sido los siguientes:

- Estudiar las tecnologías involucradas, así como las herramientas y frameworks de desarrollo ya existentes que puedan ser utilizados en el proyecto. En concreto, para la parte de recolección de datos de los nodos, analizar las herramientas ampliamente extendidas en entornos de supercomputación

como son Nagios, Ganglia o Munin. Estudiar la aplicabilidad de frameworks de desarrollo basados en mapas como Google Maps, OpenStreetMap y World Wind, con el objeto de posicionar la información geográficamente en la fase de desarrollo del interfaz Web.

- Analizar, diseñar e implementar el sistema que recopile y almacene toda la información distribuida geográficamente. Buscar un desarrollo lo más modular e interoperable posible, de manera que añadir un nuevo parámetro a monitorizar o desarrollar un nuevo interfaz no suponga modificaciones costosas en el sistema.

- Desarrollar el interfaz que provea un acceso Web seguro y con distintos niveles de autorización a la información almacenada en el sistema. Dadas las características de la RES como sistema de información distribuido, se hace énfasis en la geolocalización de la información.

Desde el principio se han tenido en cuenta los objetivos mencionados, y se han mantenido durante todo el desarrollo del proyecto sin ser alterados. Estos objetivos coinciden además con las tres fases en que se ha dividido el proyecto, estando cada fase destinada a cumplir uno de los objetivos.

## 1.4. Trabajo previo

Cuando se ha empezado a trabajar en el proyecto, lo primero que se ha hecho es conocer la situación en la que estaba la RES, conocer su estructura organizativa interna, comprender cómo funcionan los nodos y qué se hace en ellos. Se podía prever que el proyecto constaba de dos grandes fases, la recopilación de datos y la visualización de éstos, por lo que se ha podido realizar paralelamente un estudio previo de las tecnologías existentes, para ver cuales se podían utilizar y qué herramientas eran las más convenientes para realizar el proyecto.

Las primeras semanas han sido de pura documentación, para desarrollar un sistema de información es necesario comprender bien el entorno sobre el que se va a trabajar y del que se quiere extraer la información. Las fases de análisis y diseño se basan en un buen conocimiento del sistema con el que se trabaja, por ello se ha dedicado bastante tiempo a entender la organización y funcionamiento de la RES.

## 1.5. Abordando el problema

Tras estudiar el sistema sobre el que se debía trabajar y entender su estructura, era necesario más información para poder diseñar un sistema de información acorde a los requisitos de la RES. Como se había previsto, el proyecto constaba de dos partes, recopilación de datos y visualización de éstos. En este momento ya se sabía al menos de qué datos se trataban, de dónde obtenerlos y de qué forma se debían mostrar. Se decidió dividir el proyecto en esas dos fases, de forma que la recopilación de datos fuera independiente a su visualización, empezando así el desarrollo modular del proyecto.

Era necesario, por lo tanto, buscar información para ver las opciones que había para almacenar los datos y para mostrarlos. Una vez que se tuvo la información suficiente como para poder tomar decisiones, se empezó a pensar en el diseño, tanto del almacenamiento como de la visualización. Se vió que era posible trabajar en paralelo y esa ha sido la opción elegida para desarrollar el proyecto, construir un sistema de almacenamiento de datos a la vez que mostraba dichos datos, de forma que se ha podido ir visualizando el resultado conforme el proyecto iba avanzando.

## 1.6. Métodos, técnicas y herramientas empleadas

El trabajo previo realizado ha servido para conocer el entorno, entender el problema que se estaba abordando y obtener la información necesaria para elegir las herramientas adecuadas. Desde el BIFI se apuesta por un uso preferente del software libre, por lo que se ha procurado potenciar su uso durante el desarrollo del proyecto, siendo usado como entorno principal de desarrollo el sistema operativo OpenSuse 11.3, con un entorno de escritorio KDE 4.4.4, instalados sobre una máquina de 64 bits que usa el kernel de Linux 2.6.34.7-0.5.

La comunicación entre los codirectores, ponente y proyectante ha sido fluida, utilizando reuniones periódicas donde se informaba del estado del proyecto y donde se daban las directrices oportunas para continuar con el desarrollo. También se realizaban críticas constructivas con intención de resolver los problemas y solucionar los fallos, se analizaban posibles mejoras y se resolvían dudas sobre la implementación. También se ha utilizado de forma asídua el correo corporativo del BIFI como herramienta de comunicación interna entre los codirectores y el proyectante.

Desde el comienzo se ha utilizado un repositorio SVN exclusivo para el proyecto, usando para ello un servidor del BIFI donde almacenan otros repositorios. En dicho repositorio se ha guardado todo documento relativo al proyecto, de forma que se ha tenido un control total sobre la versión de los documentos e información sobre la persona que los ha modificado, así como los comentarios aportados con cada actualización del repositorio. Se ha utilizado principalmente el cliente de KDE KDESvn, que permite una manipulación sencilla e intuitiva de los ficheros del repositorio, y muestra de una forma gráfica bastante clara las diferencias de los ficheros. Además, Netbeans ha sido configurado para usar dicho repositorio, por lo que se han guardado todos los cambios realizados en la programación del código fuente principal.

Se ha ido comprobando el correcto funcionamiento del sistema cada cierto tiempo en distintos navegadores, utilizándose principalmente los navegadores Firefox, Opera y Google Chrome desde OpenSuse, y ocasionalmente los navegadores IE, Firefox, Opera y Safari desde MS Windows. También se ha comprobado el funcionamiento en MAC OS X con Safari, para lograr la interoperabilidad que se especifica en los objetivos.

Finalmente, se ha utilizado OpenOffice, VIM, Kate y LaTeXcomo principales editores de texto, para la manipulación de imágenes se ha utilizado GIMP y DIA para los diagramas E/R de la base de datos y otros diagramas de la memoria.

## 1.7. Estructura de la memoria

El presente documento está dividido en dos bloques: el documento principal de la memoria y los anexos. El documento principal está estructurado de la siguiente manera:

1. Introducción: En este capítulo se realiza una breve introducción al proyecto, su contexto, su motivación, los objetivos a conseguir, un resumen de lo realizado y su entorno de realización.

2. Análisis del sistema de información: En este capítulo se realiza una descripción del sistema sobre el que se ha trabajado, se analizan los requisitos de la aplicación y se detalla y explica el resultado del estudio realizado de las tecnologías existentes.

3. Diseño del sistema: En este capítulo se detalla el diseño realizado del sistema de información.

4. Implementación y Despliegue: En este capítulo se realiza una descripción de la implementación, de los problemas encontrados y la solucion dada. También se explica el despliegue del servidor que almacena la base de datos y el interfaz Web, y de cómo se debe realizar el despliegue en cada nodo.

5. Conclusiones: En este capítulo se detallan las conclusiones, el cronograma del proyecto, el grado de cumplimiento de objetivos, mi opinión personal y posibilidades de mejora para el futuro.

Tras estos capítulos, se incluye la bibliografía utilizada para el desarrollo del proyecto, terminando así el documento principal de la memoria que es seguido de los diversos anexos donde se profundiza más en ciertos aspectos del proyecto.

# Capítulo 2

# Análisis del sistema de información

En este capítulo se realiza una descripción y análisis del sistema sobre el que se ha trabajado y se detallan sus requisitos. Se comentan las herramientas estudiadas que se pueden utilizar para el desarrollo del proyecto y se razona la elección realizada.

## 2.1. Descripción del sistema

En esta sección se realiza una descripción del sistema sobre el que se ha realizado el proyecto. Se detalla la estructura de los nodos de la RES y su organización. Finalmente se describen los requisitos que se han esperado de la aplicación final.

### 2.1.1. La Red Española de Supercomputación

Actualmente la RES está compuesta por 8 nodos situados en diferentes puntos de España, estos nodos y localidades son: Altamira en Cantabria, Atlante y La Palma en Canarias, Caesaraugusta en Zaragoza, Magerit en Madrid, Marenostrum en Barcelona, Picasso en Málaga y Tirant en Valencia.

Los 8 nodos de la RES usan una arquitectura similar, todos ellos usan PPC64, Marenostrum está formado por 10240 IBM PowerPC 970MP a 2.3 GHz con 20 TBytes de memoria, Magerit por 2408 IBM PowerPC 970FX a 2.2 GHz con 4.7 TBytes de memoria y el resto de nodos están formados por 512 IBM PowerPC 970FX a 2.2 GHz, con 1 TByte de memoria.

Es importante destacar que la potencia de cálculo por procesador es prácticamente la misma en todos los nodos, ya que esto es fundamental a la hora de distribuir la carga de trabajo en los nodos. Al tener la misma potencia de cálculo, podemos hablar en términos de "horas de cálculo", ya que una hora de cálculo en un nodo será igual de productiva que en cualquier otro nodo. Por ello, cuando alguien solicita un supercomputador de la RES, lo que realmente solicita y se le concede son horas de cálculo. De esta forma, es fácil organizar las tareas a ejecutar entre los distintos procesadores, y se pueden distribuir de forma equitativa las tareas entre los 8 nodos.

Así mismo, es también importante destacar que la estructura de cada nodo es similar, y que usan los mismos sistemas y *scripts* para recopilar la información, ya que ello nos permite diseñar aplicaciones sobre un nodo sabiendo que es portable a los demás. El esquema lógico de un nodo cualquiera de la RES es el que se puede observar en la Figura 2.1.

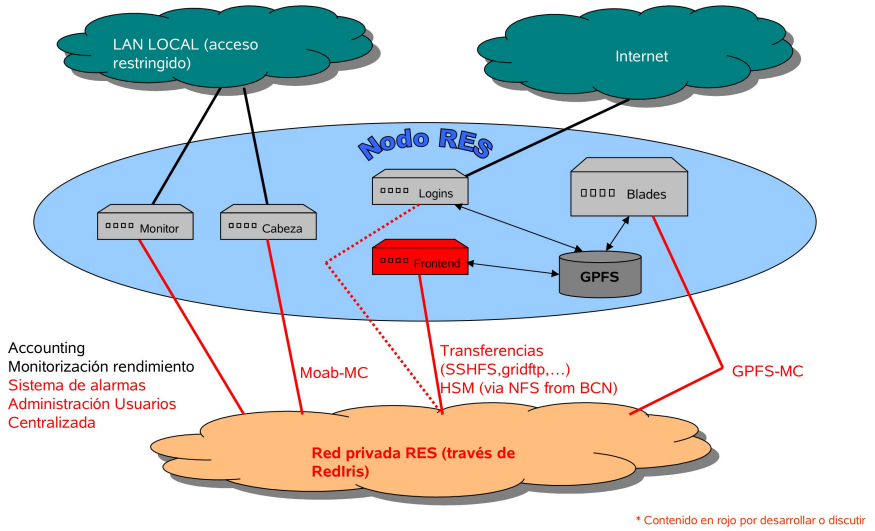


Figura 2.1: Esquema lógico de un nodo de la RES

Cada nodo puede usar el 20 % de las horas de cálculo para tareas propias (excepto Atlante, que posee un mayor porcentaje por ser propiedad del ITC), y el otro 80 % está gestionado por el comité de Acceso de la RES. Estas horas pueden ser de dos tipos, A y B, cuya diferencia es la prioridad. Aquellos trabajos que posean horas de cálculo de tipo A tendrán prioridad en las colas de trabajos sobre aquellos que sean de tipo B.

A la hora de solicitar horas de computación, un grupo de investigación debe rellenar un formulario y su solicitud es analizada por una comisión. De las solicitudes se obtienen datos como el nombre del grupo de investigación, un responsable y una forma de contacto. Estos datos se usan para identificar a los usuarios y para realizar estadísticas de uso.

Cada 4 meses se organizan las horas de cálculo de los nodos, y estos períodos son: Marzo-Junio, Julio-Octubre y Noviembre-Febrero. Los usuarios han de solicitar las horas de trabajo para cada período, y deben procurar respetar el número de horas asignado. Una vez realizadas todas las solicitudes, la comisión se encarga de gestionar los recursos y asigna un determinado número de horas en un nodo concreto a cada grupo según las peticiones.

Se observa que la información que hay que recopilar es bastante estática a excepción de las horas solicitadas y consumidas por los grupos. La información más dinámica corresponde a la información interna de cada nodo, como son los trabajos que se están ejecutando en el momento, las horas consumidas y

el estado de los nodos (CPU's activas, memoria consumida, espacio en disco disponible, etc).

### 2.1.2. Requisitos de la aplicación

La aplicación a desarrollar requería que la información recopilada fuera mostrada sobre un mapa, de forma que los datos se posicionaran sobre el mapa, donde geográficamente les correspondiera. Los datos requerían ser visibles única y exclusivamente por las personas autorizadas a ello, hay información que es pública e información que es privada, visible solo por determinados nodos o grupos de usuarios. Por lo tanto, requería una securización de los datos y un acceso por niveles a éstos.

Con vistas a posibles cambios en el futuro, se requería que la aplicación tuviera una estructura modular, de forma que se pueda modificar la parte visual sin afectar a la base de datos y *vice-versa*.

Finalmente, se pretendía que la parte visual pudiera ser accesible por el máximo número de usuarios posibles, por lo que era recomendable que pudiera ser accesible desde cualquier ordenador, lo cual implica que se adaptara a diferentes sistemas operativos y entornos de escritorio.

Visto todo esto, podemos afirmar que la aplicación tenía unos requisitos bien definidos desde el principio:

- Permitir posicionar la información geográficamente sobre un mapa.
- Obtener una estructura lo más modular posible, de manera que añadir un nuevo parámetro a monitorizar o desarrollar un nuevo interfaz no suponga modificaciones costosas en el sistema.
- Obtener una aplicación lo más interoperable posible, que permita ser usada en distintos sistemas operativos y desde diferentes navegadores Web.
- Desarrollo de un interfaz que provea un acceso Web seguro y con distintos niveles de autorización a la información almacenada en el sistema.

## 2.2. Estudio previo de las tecnologías

En esta sección se explica el estudio realizado sobre las tecnologías existentes que pudieran utilizarse durante el desarrollo del proyecto, y se explican las decisiones tomadas.

### 2.2.1. Tecnologías existentes

Durante el proceso de documentación, se realizó un estudio de las tecnologías existentes que se podrían utilizar para el desarrollo del proyecto, y se redactó un informe que se puede contemplar íntegramente en el Anexo A. Dicho informe tenía como función informar detalladamente de las posibilidades que había a los directores de proyecto. En esta sección se realiza un resumen de dicho documento.

En el estudio que se realizó se ha dividido en dos categorías las posibles herramientas a usar: por un lado, herramientas que sirvan para la recopilación

de datos y por otro, herramientas que ayuden a visualizar los datos de forma agradable por pantalla.

Para la recopilación de datos, nos centramos en dos herramientas conocidas, Munnin [18] y Nagios [17], pero como este estudio se hizo paraleramente al estudio de los nodos, vimos que en Caesaraugusta se utilizaba Ganglia [16], por lo que también se contempló su uso como una posibilidad. Tras analizar dichas tecnologías, se vió que las tres eran muy similares, no había grandes diferencias significativas entre ellas. En la tabla 2.1 se puede ver la comparación de las herramientas.

| **Nombre** | **Licencia** | **Documentación** | **Plugins** | **Dificultad creación de plugins** | **Monitorización distribuida** | **Método de almacenamiento de datos** |
|---|---|---|---|---|---|---|
| Ganglia | BSD | Muy completa | Sí | Media | Sí | RRDtool, memoria, texto plano |
| Munin | GPL | Muy completa | Sí | Media | Sí | RRDtool |
| Nagios | GPL | Completa | Sí | Media | Sí | SQL, texto plano |

Tabla 2.1: Comparación de los sistemas de monitorización

En dicha tabla, se ha destacado las propiedades que se creen que son más fundamentales para optar por una u otra herramienta, que son:

- *Licencia*: la preferencia es utilizar software libre.
- *Documentación*: se valoró más aquellas herramientas con mayor documentación y soporte.
- *Plugins*: se valoró más aquellas herramientas con posibilidad de usar y crear plugins para adaptar su comportamiento.
- *Dificultad de creación de plugins*: cuanto más sencillos y estándares fueran los procedimientos para crear los plugins, mejor.
- *Monitorización distribuida*: se requiere por definición del problema dado.
- *Método de almacenamiento de datos*: cuanto más estándar, mejor.

Para la visualización de datos, se optó por buscar herramientas que nos dieran la posibilidad de mostrar la información sobre un mapa de la Tierra. Tras buscar y ver varias opciones, se deció estudiar profundamente las siguientes: Google Maps y Google Earth API para JavaScript [19], OpenLayers [20] y WorldWind[21]. En la tabla 2.2 se puede ver la comparación de las herramientas

| **Nombre** | **Licencia** | **Tecnología** | **Documentación** | **Calidad del mapa** | **Calidad visual** | **Multiplataforma** |
|---|---|---|---|---|---|---|
| GoogleMaps JavaScript API | Gratuito, con restricciones | JavaScript | Muy completa | Muy buena | Media | Sí |
| GoogleMaps API for Flash | Gratuito, con restricciones | Flash | Completa | Muy buena | Buena | Sí (requiere flash) |
| GoogleEarth JavaScript API | Gratuito, con restricciones | JavaScript | Completa | Muy buena | Muy Buena | No (no funciona en GNU/linux) |
| OpenLayers | OpenSource | JavaScript | Muy completa | Depende del mapa usado | Media | Sí |
| World Wind | OpenSource | Java | Muy completa | Buena | Muy buena | Sí (requiere Java) |

Tabla 2.2: Comparación de visualización de mapas

Para estas herramientas, se buscaba algo que visualmente fuera atractivo, a ser posible que no fuera algo común, con cierta calidad tanto en los mapas como en los controles y especialmente, que permitiera ser utilizado con cualquier navegador Web en cualquier sistema operativo. Por lo tanto, las propiedades más destacables son:

- *Licencia*: la preferencia es utilizar software libre.
- *Tecnología*: cuanto más interoperable, mejor.
- *Documentación*: se valoró más aquellas herramientas con mayor documentación y soporte.
- *Calidad del mapa*: se requiere una calidad mínima, que permita hacer zoom sobre una región sin que se vea pixelada a una altitud aceptable.
- *Calidad visual*: que los controles que ofrezca sean visual y funcionalmente buenos.
- *Multiplataforma*: a cuantos más usuarios se llegue, mejor.

Como puede observarse, son muy similares a las propiedades elegidas para la recopilación de datos, en ambas listas de propiedades se da énfasis a las licencias y documentación, y en este caso también a la calidad visual.

### 2.2.2. Selección de tecnologías a utilizar

Una vez vistas las posibilidades de cada herramienta, se pudo realizar una valoración razonada de éstas, y se eligieron las herramientas adecuadas para cada parte del proyecto.

Para la parte de recopilación de datos, se vió que cualquiera de las 3 herramientas estudiadas podrían servir, las diferencias entre ellas son mínimas y tienen la misma funcionalidad. La que más destacaba era Ganglia, por ser una herramienta diseñada para tener una carga mínima para el sistema, pero era la única diferencia real con respecto a sus alternativas. Por ello, se decidió utilizar Ganglia para monitorizar y recopilar los datos, aunque se podría haber elegido cualquier otra herramienta sin problemas.

Para la parte visual, la herramienta que más destacaba visualmente era World Wind, pero por desgracia no tenía una API para JavaScript, solo para Java y Java Applets, por lo que la aplicacion a desarrollar se hace más pesada y dependiente del software instalado en las máquinas. Google Map sería la alternativa siguiente, pero la API de Google Maps tiene mayores limitaciones, visualmente es menos elegante, y actualmente es una aplicación muy vista ya por los usuarios de Internet.

Por todo ello, parecía que la opción más recomendable era World Wind, ya que pese a sus limitaciones tecnológicas, es visualmente superior, por lo que decidimos usar World Wind. Además, se vió que hay una API en desarrollo para JavaScript de World Wind, hecho que nos hizo reafirmar la decisión, ya que al ser un diseño modular, se podría modificar la parte visual en un futuro con la API para JavaScript de World Wind.

En la Figura 2.2 se puede observar un applet de World Wind, si lo comparamos con una aplicacion en JavaScript de Google Maps, como el de la Figura 2.3, podemos comprobar que la calidad visual es mejor en la API de World Wind, y no resulta tan familiar como Google Maps.


Figura 2.2: Ejemplo de un applet World Wind.


Figura 2.3: Ejemplo de Google Maps creado con Javascript.

Una de las ventajas que tiene World Wind es que permite ver el mapa de la tierra en 3D, por lo que tiene mayor impacto visual que Google Maps. Además, permite visualizar el mapa de forma limpia, sin logos a la vista, algo imposible con Google Maps, tal y como se puede apreciar en las Figuras 2.2 y 2.3.

# Capítulo 3

# Diseño del sistema

En este capítulo se presenta el diseño realizado del sistema de información, se detallan sus partes y se comentan las decisiones tomadas.

## 3.1. Arquitectura del sistema de información

El estudio del sistema y del problema ha demostrado que hay dos partes bien diferenciadas: por un lado, la recolección de datos y por otro, la visualización de éstos. Para mantener el requisito de modularidad requerido, se ha optado por dividir lo máximo posible estas dos capas. Por lo tanto, el desarrollo del proyecto se ha dividido en dos grandes fases o capas: el diseño para la recolección de los datos y el diseño para mostrar dichos datos.

Esta división en dos capas, permite que por un lado, se puedan realizar cambios en los datos recolectados, sin que afectara a la visualización existente de los datos actuales, y *vice-versa*, que se pudiera modificar la presentación de los datos sin tener que modificar los ya almacenados.

Por otro lado, también permite que en un futuro se puedan situar ambas capas en diferentes entornos. Actualmente están sobre la misma máquina, pero podrían separarse sin problemas.

Para la capa de recolección de datos, se ha optado por utilizar una base de datos MySQL [1, 2, 22], y utilizar diferentes scripts para obtener los datos.

Para la capa de visualización de datos, el estudio previo concluía que la mejor opción para obtener una visualización en diferentes plataformas era World Wind, por lo que se ha desarrollado un applet en Java para visualizar los datos. Además, se ha visto la necesidad de usar una herramienta que facilitara la administración de MySQL, por lo que se ha pensado en la instalación de alguna herramienta ya existente para administrar la BD.

Con este diseño se ha logrado dividir la forma de interactuar de las diferentes partes, los nodos que recopilan la información de la RES sólo interactúan con la BD, y los usuarios con la capa de presentación de los datos mediante el applet.

En la figura 3.1 podemos ver un esquema del diseño descrito, donde se observan las diferentes capas y la interacción con éstas de las diferentes partes implicadas, así como la ubicación de cada capa.

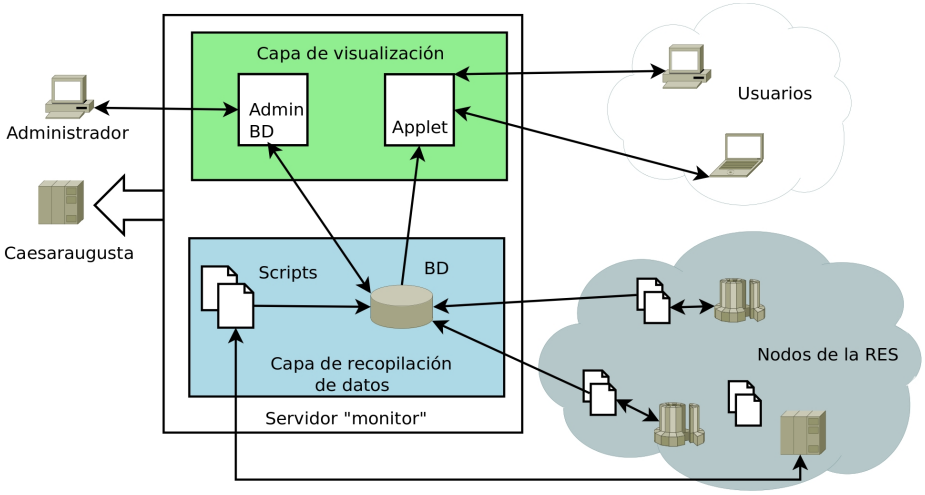


Figura 3.1: Esquema del diseño por capas.

## 3.2. Recolección de información

Para realizar un buen diseño es necesario conocer bien la información con la que se va a trabajar, por ello, una de las tareas que se ha realizado durante las primeras semanas de documentación ha sido estudiar detalladamente los datos que se querían almacenar. Se ha obtenido información sobre su origen y sus propiedades (el tipo de dato, su privacidad y cada cuánto tiempo se actualizaba).

En este apartado se va a explicar el tipo de datos a recopilar y su origen, información previa que es necesaria para realizar una base de datos acorde a los datos que se requieren almacenar.

### 3.2.1. Origen y tipo de datos a recolectar

Si tenemos en cuenta que existen diferentes grupos de investigación en la RES, que pertenecen a diferentes instituciones, y que ejecutan diferentes aplicaciones sobre los nodos, vemos que podemos recopilar información relativa a distintos objetos:

- *Nodos*: nos dan información sobre las aplicaciones que se están ejecutando en ellos, los grupos activos en el nodo, los trabajos que hay en ejecución y en espera, la carga total del nodo y en algunos casos el estado de la VPN que les conecta con otros nodos.
- *Usuarios*: tenemos información sobre su estado, el grupo y aplicación en los que trabajan, la institución a la que pertenece y el nodo sobre el que ejecutan sus procesos. Además de los usuarios de la RES, hay que tener en cuenta los administradores del sistema que trabajan en los nodos, puesto que necesitan tener la información necesaria para controlar su buen funcionamiento.
- *Grupos*: tenemos información sobre los distintos grupos que usan la RES, las aplicaciones que se diseñan en ellos y los usuarios que tienen.

- *Instituciones*: nos dan información de la localización de los grupos y de las aplicaciones que desarrollan.
- *Aplicaciones*: nos dan información sobre el líder de la aplicación, el grupo que la ha diseñado, la institución a la que corresponde el grupo, las horas y espacio en disco asignadas y consumidas, el nodo donde se han ejecutado previamente y más información variada sobre la propia aplicación.

La información generada por los nodos se recoge cada 24 horas por el BSC, y el resto de información se genera al inicio de cada período (cada 4 meses). La forma en que el BSC guarda la información es desconocida a fecha de la redacción de la presente memoria. Durante las últimas semanas del proyecto se han iniciado las conversaciones para integrar sus datos con la base de datos creada, ya que la idea original es que a largo plazo los datos recopilados se obtengan directamente de la fuente, del BSC.

Como no se podía contar con dichos datos, se ha tenido que recopilarlos de forma independiente. Los datos relativos a los nodos, grupos y usuarios se pueden obtener directamente y de forma automatizada de los propios nodos.

Parte de la información de las aplicaciones relativa a su ejecución (horas y espacio consumidos) se pueden obtener también de los nodos.

Sin embargo, la información de las instituciones (especialmente la geolocalización) y gran parte de los datos de las aplicaciones no se pueden automatizar, hay que introducirlas a mano en la base de datos por el momento.

Tras analizar detalladamente la situación, se ha decidido crear diferentes scripts que se ejecutan en cada nodo, que se encargan de recopilar la información y de almacenarla en la base de datos principal, como se puede observar en la Figura 3.1. Dichos scripts se comentan en el Anexo D, el la sección D.3.

Los scripts se encargan de actualizar los datos con mayor frecuencia que con la que se recogen desde el BSC, para procurar mostrar en la aplicación unos datos lo más actualizados posible.

La parte dinámica, como son los trabajos que se ejecutan, que están en cola y la carga de los nodos, se ha pensado que debía actualizarse cada 10 minutos. Son trabajos que se ejecutan durante largo tiempo por lo general, con lo que una actualización del orden de 10 minutos se ha considerado adecuada.

Los datos relativos a los usuarios y grupos se ha decidido actualizarla cada día, por la sencilla razón de que se introducen manualmente en los sistemas por los administradores cuando se les solicita permiso para ejecutar tareas en los nodos. Por ello, se ha considerado que una actualización diaria de estos datos es más que suficiente.

El diseño ha sido realizado de tal forma que los nodos son los encargados de subir la información a la base de datos (modelo *push*), mediante los scripts creados para recopilar y guardar los datos, de forma que la BD no tiene que preocuparse por solicitar la información.

### 3.2.2. Privacidad de los datos

Tras tener información específica de los datos a recopilar, se han identificado los datos que son privados, y quién tiene los permisos necesarios para verlos.

Se ha establecido que en la aplicación haya cuatro grupos diferenciados con sus respectivos permisos, que son los siguientes:

- *Guest*: usuario invitado de la aplicación. Puede ver cualquier dato público.
- *Resuser*: usuarios de la RES. Pueden ver cualquier dato público y cualquier dato privado relativo a su aplicación.
- *Sysadmin*: administradores de los nodos. Pueden ver cualquier dato público y los datos privados asociados al nodo que administren.
- *Boss*: un usuario con todos los permisos, el cual puede ver todos los datos, públicos y privados.

Tras conocer los datos, los permisos y los usuarios, se ha realizado una tabla en la que se reflejan todos estos datos de forma resumida, que puede observarse en la Tabla 3.1.

| TIPO DE USUARIO / INFORMACION | ANONIMO | USUARIO DE LA RES | ADMINISTRADOR DE NODO | ROOT |
|---|---|---|---|---|
| ESTATICA DE NODOS: descripción hardware + email de soporte + red RedIRIS | todo | todo | todo | todo |
| ESTATICA DE USUARIOS: applications + groups + users | applications geoposicionadas, horas y su lider (no gids ni resto de usuarios) | applications geoposicionadas, horas y su lider (no gids ni resto de usuarios). De su application, todo (gid y users) | applications geoposicionadas, horas y su lider (no gids ni resto de usuarios). De las applications de su nodo, todo (gid y users) | todo |
| VPN de la RES: topología estática + monitorización (ancho de banda, latencia, mrtg) | nada | nada | todo | todo |
| DINAMICA: % de jobs running sobre el total posible y gráficas de monitorización | running jobs (para la barra dinámica) y 3 gráficas como las públicas del BSC para MareNostrum | running jobs (para la barra dinámica) y 3 gráficas como las públicas del BSC para MareNostrum | además de lo anterior, una selección de las 5-10 gráficas más significativas de monitorización de su nodo | todo |
| ACCOUNTING | nada | accounting (horas y espacio consumido, % absoluto) de su application | accounting (horas y espacio consumido) de las application de su nodo | todo |

Tabla 3.1: Tabla de usuarios y permisos.

## 3.3. Base de datos

Para la base de datos se ha decidido utilizar MySQL, por su potencia y disponibilidad en el nodo en el que se iba a desplegar la base de datos. Por lo tanto, el diseño se ha realizado teniendo en cuenta las posibilidades de MySQL, por lo que se ha tratado de usar al máximo las propiedades del gestor.

Inicialmente, se iba a plantear una aplicación cliente-servidor, donde el servidor se conectara a la base de datos, pero tras hablar con Eduardo Mena, se ha visto que es posible y recomendable usar las vistas que ofrece MySQL de las tablas en vez de implementar una aplicación cliente-servidor.

El problema ha surgido con la privacidad de los datos, ya que ningún usuario debe ver más datos que los que le pertenecen dado su nivel de acceso. Pero como MySQL permite generar vistas de las tablas y restringir su uso de lectura y escritura a determinados usuarios, se ha podido resolver el problema del acceso a los datos por nivel de autorización. La solución ha sido sencilla, se han generado

unas vistas públicas, unas privadas y unas relativas a cada usuario particular, dando los permisos oportunos de lectura a las vistas a cada usuario.

Para realizar el esquema E/R, se ha tenido en cuenta los resultados del estudio previo realizado. Tras el análisis realizado en la sección 3.2.1 parece claro que las entidades principales serían: nodo, aplicación, usuario de la RES, usuario anónimo, administrador de nodo, usuario global, grupo de la RES e institución. Los atributos de dichas entidades son los datos que se pueden asociar directamente a cada una de ellas.

Al diseño inicial se han añadido dos entidades más, una independiente del resto con los datos de RedIRIS, para mostrar los datos de la red, y otra con los gráficos que se obtienen en los nodos que muestran el estado de las máquinas. Por problemas que se comentarán más adelante se ha decidido no incluir los atributos de las imágenes en la entidad Nodo y crear una entidad aparte con dichas imágenes.

En la Figura 3.2 se muestra el Modelo E/R de la base de datos simplificado. Debido al tamaño del diagrama sólo aparecen las entidades. Se puede observar el modelo E/R más detalladamente en el Anexo B.

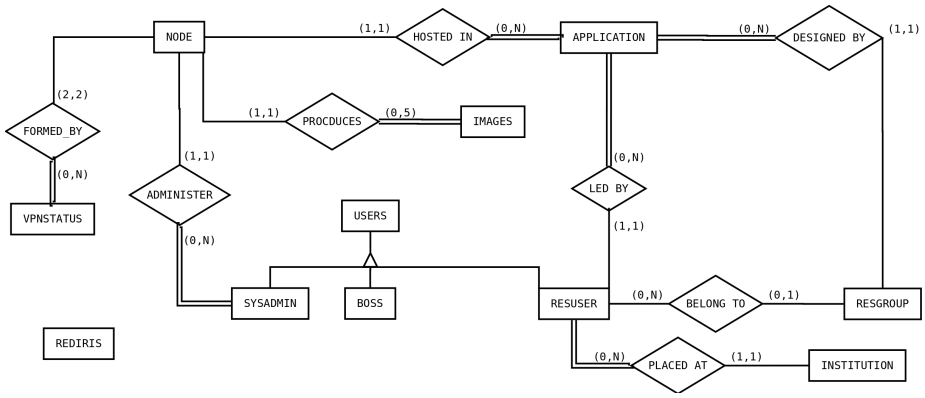


Figura 3.2: Modelo básico E/R de la base de datos

Para llegar al modelo final se ha usado principalmente el libro de Navathe [1], donde se detalla muy bien el diseño de las bases de datos y ha sido un libro de referencia frecuentemente utilizado durante el diseño de la base de datos.

## 3.4. Interfaz

A la hora de diseñar el interfaz, se ha tenido en cuenta en todo momento que iba a ser un interfaz para la Web. Esto ha implicado que el ratón sea para el usuario la principal forma de interacción con el interfaz, y como ayuda se ha pensado en el uso de ciertos atajos de teclado.

### 3.4.1. Diseño visual

El diseño debía ser algo sencillo, de forma que un usuario pudiera acceder de forma rápida a la información que busca. Además, debía de ser posible visualizar la máxima información posible a la vez que ésta se posicionaba sobre el mapa.

Debido a que se le ha dado más importancia a la geolocalización de la información, se ha decidido darle un mayor protagonismo al mapa sobre el que se situa la información, pero a la vez, no se ha querido perder la posibilidad de mostrar una información adicional y más específica.

Por ello, se ha decidido desde el principio que la aplicación debía mostrar el mapa con la información principal geolocalizada sobre él, y junto al mapa se debía mostrar información más específica cuando el usuario lo requisiera.

Tras analizar las posibilidades que ofrecía World Wind y Java, se ha diseñado una pantalla principal dividida verticalmente en dos sobre la que se mostrarían todos los datos, cuyo boceto inicial puede observarse en la Figura 3.3.

En dicha figura, podemos observar que la parte derecha ocupa la mayor parte, es donde se sitúa el mapa y la información principal. Cuando el usuario pasa el ratón sobre un objeto del mapa, aparece un globo relativo al objeto sobre el que aparece la información básica del objeto.

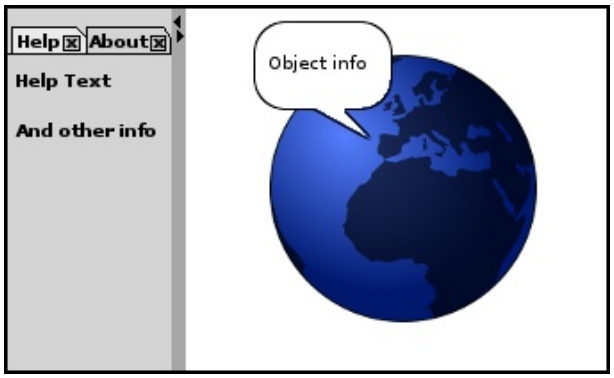


Figura 3.3: Boceto inicial de la ventana principal.

En la parte de la izquierda, se sitúa un panel donde se muestra información más detallada mediante pestañas. Cada vez que el usuario selecciona un objeto del mapa, se abre una pestaña con información específica del objeto. Hay diferentes tipos de pestañas, cada uno diseñado específicamente para mostrar un tipo de información. Las pestañas pueden mostrar información de nodos, instituciones, aplicaciones de la RES, capas del programa, y diferentes pestañas de información y ayuda sobre el applet.

La división de los paneles no está fijada a un tamaño concreto, el usuario puede jugar con su posición. Si lo desea, puede cerrar el panel de la izquierda y ver el mapa al máximo tamaño posible, o incluso puede hacer que el mapa no se vea y mostrar sólo el panel de la izquierda con la información detallada.

El diseño inicial también ha incluido una pantalla inicial donde el usuario debe identificarse para poder acceder a la información. Ocupa toda la pantalla y

desaparece tras iniciar la sesión. Al terminar la sesión, el usuario vuelve a dicha pantalla. Esta pantalla inicial permite al usuario que no tenga una cuenta de acceso poder iniciar sesión como invitado, desde la que accede a todos los datos públicos. El diseño de esta pantalla puede observarse en la Figura 3.4.

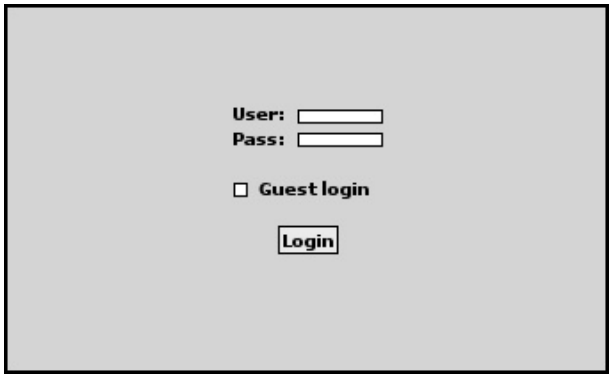


Figura 3.4: Boceto inicial de la ventana de login.

Tras finalizar el diseño inicial e iniciar la implementación, se ha visto que era conveniente realizar algunas modificaciones para mejorar el diseño. Las principales modificaciones han sido las siguientes:

- Mejora en el panel de información: los atajos de teclado son de gran ayuda, pero es necesario poder abrir todas las pestañas con el ratón. Por ello, se ha añadido un pequeño menu encima de las pestañas que permite al usuario abrir las principales pestañas de navegación y ayuda.
- Integración de la pantalla de acceso: para evitar confusiones al usuario y procurar que el aspecto visual sea similar, se ha integrado la pantalla inicial de acceso en el panel izquierdo, desactivando la visualización del mapa y la información hasta que no se haya acreditado el usuario.

Tras implementar las nuevas mejoras, se ha observado que el resultado es el deseado y su nuevo diseño es mejor que el inicial, logrando una aplicación más sencilla y óptima para el usuario.

### 3.4.2. Diseño de programación

Hasta ahora, se ha visto todo lo relativo al diseño gráfico de la interfaz. El diseño modular de su programación se ha realizado durante el principio de la fase de desarrollo. Inicialmente no se conocía bien la API de World Wind, y se ha ido aprendiendo sobre ella tras realizar diferentes pruebas al inicio del desarrollo.

El diseño inicial de los módulos a programar ha sido un boceto sencillo en el que se ha puesto de manifiesto la intención de separar la parte gráfica de la recopilación de datos, a la vez que se ha tratado de modularizar la programación.

La idea básica ha sido usar 4 módulos diferentes: uno para solicitar datos a la base de datos, otro para controlar las ventanas, el tercero para manejar los diferentes objetos con información y el cuarto para mostrar datos sobre el mapa, tal y como se puede observar en la Figura 3.5.

Dichos módulos serían lo suficientemente independientes entre sí como para poder realizar cambios en uno de ellos sin que afectaran al resto. La mayor carga de trabajo la realizaría el controlador de ventanas, quien se encargaría de centralizar toda la gestión y toda la información pasaría por él.

En el próximo capítulo se detallará más el diseño usado para programar la aplicación.

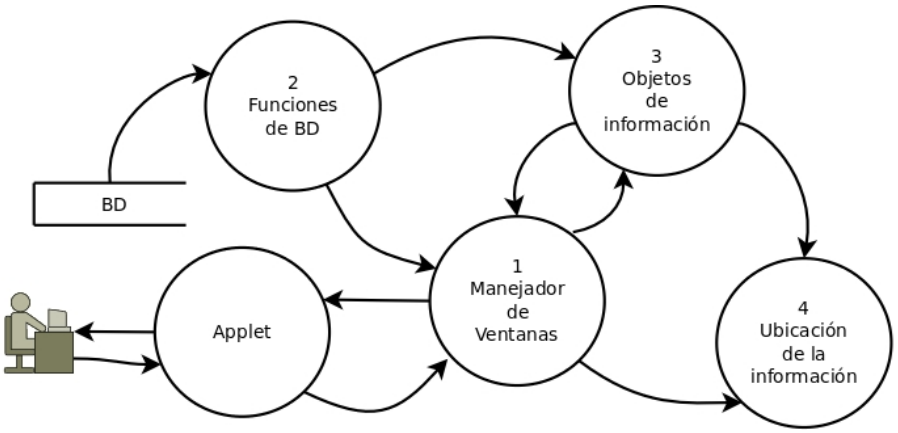


Figura 3.5: Módulos iniciales del interfaz. Las flechas indican la dirección del flujo de información.

# Capítulo 4

# Implementación y Despliegue

En este capítulo se va a comentar el proceso de implementación de la aplicación y su despliegue tanto en el nodo de Caesaraugusta como en el resto de los nodos. El proceso ha sido realizado a la par, ya que se ha querido comprobar su correcto funcionamiento multiplataforma desde el principio. Además, al ser un applet, se ha preferido comprobar su comportamiento al ser alojado en un servidor Web real, en vez de realizar pruebas locales.

También se comentan los problemas encontrados y las soluciones aportadas, ya que durante la fase de implementación se han encontrado algunos problemas que han provocado ligeros cambios en el diseño original.

## 4.1. Implementación y despliegue en Caesaraugusta

El proyecto se ha desarrollado usando solo el nodo de Caesaraugusta, y en esta sección se detalla el proceso. Debido a que todos los nodos poseen una estructura similar, como ya se ha visto en la Figura 2.1, y que utilizan los mismos programas de monitorización (al menos para obtener los datos que manejamos en la aplicación), se puede realizar la configuración sobre un nodo concreto, sabiendo que la configuración servirá para los demás.

Se observó que muchos de los scripts desarrollados por el BSC usados por Caesaraugusta estaban en inglés, al igual que casi toda la información que se envía a los administradores semanalmente sobre el *accounting*, por lo que se decidió realizar el desarrollo en inglés.

Nos ha parecido la opción más razonable, ya que muchos investigadores que usan la RES son extranjeros, y dado el carácter informativo de la aplicación se ha creido oportuno tratar de usar un lenguaje que es ampliamente extendido dentro del ámbito de investigación.

Como ya se ha visto al hablar de la arquitectura del sistema de información en la sección 3.1, la interfaz Web se conecta directamente a la base de datos y ofrece a los usuarios la información geoposicionada sobre un mapa. Para poder realizar las pruebas y comprobar que todo funcionaba correctamente, se ha decidido usar una misma máquina para alojar el applet y la base de datos, por

lo que se ha configurado el servidor *monitor* de Caesaraugusta con MySQL y Apache.

Lo primero que se ha empezado a implementar ha sido la base de datos, ya que se sabía que su estructura final iba a marcar el desarrollo del applet, debido a que las clases de java iban a ser muy parecidas a las entidades de la base de datos.

### 4.1.1. La Base de Datos

En Caesaraugusta se utiliza el sistema operativo SUSE Linux Enterprise (SLES) 10, por lo que usando el gestor de paquetes de Yast se ha podido instalar MySQL 5.0.26 sin problemas. Una vez instalado, se ha realizado la configuración inicial de MySQL, creando el usuario root y borrando las bases de datos y usuarios de pruebas que se generan inicialmente.

Se ha comprobado que estuviera activada la opción de control de transacciones, para evitar posibles pérdidas en los datos cuando sean guardados en la base de datos. También se ha comprobado que se pudiera utilizar el motor de almacenamiento InnoDB, que es el encargado de realizar el control de las transacciones, y se han realizado sencillas pruebas para comprobar su correcto funcionamiento.

Tras estas comprobaciones, ya estaba preparado MySQL para ser usado en la aplicación, por lo que se ha creado la base de datos "resdata" que contiene todas las tablas necesarias para este proyecto y el usuario "mysql", con permisos totales sobre dicha base de datos, para evitar el uso del usuario root, y así poder administrar sin problemas la base de datos durante la realización del proyecto.

La implementación final es como se ha mostrado en la Figura 3.2, y el diseño completo con sus atributos se puede observar en el Anexo B. Para rellenar la base de datos se espera que a largo plazo se use un método totalmente automatizado, que recoja los datos del BSC y los carge en la base de datos creada para este proyecto. Pero sabiendo que iba a ser un proceso largo, se han realizado sencillos scripts que recogen la misma información directamente de los nodos, y la almacenan en la base de datos. Los scripts recopilan la siguiente información:

- *Grupos y usarios de un nodo*: cada 24 horas se comprueba los cambios del sistema y se actualiza la base de datos. Se actualizan los campos de las entidades USERS, RESUSER y RESGROUP.

- *Estado del nodo*: cada 10 minutos se recopila la información sobre la carga del nodo, algunas gráficas del estado del nodo, los trabajos que están corriendo y los que están esperando en colas. También se recopila información de las *quotas* de los usuarios cada hora, de forma que se conoce el espacio consumido por los grupos. Con todos estos datos se actualiza la entidad NODE, IMAGES y APPLICATION.

Para recopilar la información, los scripts obtienen la información de Ganglia, MOAB[1] y otros scripts que se han generado en el BSC.

[1] Moab Workload Manager es un *metascheduler*, un sistema avanzado de administración y planificación de trabajos para clusters, grids y sistemas de computación bajo demanda.

Los atributos relativos a VPNSTATUS se pueden automatizar, y tanto la base de datos como la interfaz Web están listos para recoger y mostrar los datos. Pero debido a que en Caesaraugusta todavía no está activa la VPN, no se pueden obtener los valores para rellenar la base de datos.

Desafortunadamente, hay muchos atributos y entidades que no se pueden actualizar o crear de forma automatizada actualmente. Dichos datos son los siguientes:

- *Nuevas aplicaciones*: no se pueden crear de forma automatizada, ya que los datos obligatorios para crearlas (entre ellos la clave primaria) no se pueden obtener de los nodos, es una información que está centralizada en el BSC.
- *Instituciones*: actualmente no se almacena en ningún lugar las coordenadas de las distintas instituciones, por lo que hay que buscarlas e introducirlas manualmente.
- *Datos de RedIRIS*: no se almacenan tampoco en ningún lugar, pero como son datos estáticos, se ha creado un script inicial que inserta en la base de datos la información de RedIRIS al crearse las tablas.

Como se ha comentado en el diseño de la base de datos, se han creado vistas de las tablas, de forma que los usuarios de la RES y los invitados de la aplicación Web sólo tienen permisos de lectura sobre las vistas, y los usuarios creados para actualizar los datos desde los distintos nodos, tienen permiso de modificación sobre algunas vistas y permisos de insertar datos sobre otras.

Podemos distinguir los siguientes tipos de usuarios de MySQL (no de la aplicación Web):

- *Nodo*: Hay un usuario por cada nodo, se utiliza en los scripts para actualizar la base de datos.
- *Usuario de la RES*: Tiene permisos de lectura sobre los datos relativos a su grupo y de la aplicación que desarrolla.
- *Líder de una aplicación*: Tiene permisos de lectura sobre los datos relativos a su grupo y de la aplicación que lidera.
- *Administrador de nodo*: Tiene permisos de lectura sobre los datos relativos a su nodo.
- *Superusuario*: Tiene permisos de lectura de todo.
- *Invitado*: Tiene permisos de lectura sólo de los datos públicos.

Como vemos, son muy parecidos a los usuarios y permisos de la Tabla 3.1, y su diferencia como usuarios de MySQL estriba sólo en los permisos que tienen sobre las diferentes vistas. Mediante este sistema de vistas podemos garantizar una seguridad sobre el acceso a los datos, ya que podemos controlar qué usuarios acceden a qué datos. Podemos distinguir los siguientes tipos de vistas:

- *Públicas*: tienen acceso de lectura a todos los datos públicos.
- *Privadas de acceso total*: tienen acceso de lectura a todos los datos.

- *De grupo*: tienen acceso de lectura a todos los datos relativos al grupo y a la aplicación que desarrolla el grupo.

- *De líder de aplicación*: tiene acceso de lectura a los datos relativos a la aplicación que lidera.

- *De administrador*: tienen acceso de lectura a todos los datos relativos al nodo que administran.

- *De usuario*: tienen acceso de lectura a sus propios datos de usuario.

- *De nodo*: tienen permiso de inserción en determinadas tablas y de actualización sólo de sus datos en otras, de forma que no modifiquen datos que no les pertenecen.

En el Anexo B se puede contemplar una descripción completa de todas las vistas, usuarios, atributos y entidades del modelo E/R así como la transformación del modelo E/R a las tablas de MySQL.

### Problemas encontrados

Durante la implementación se comprobó que había un problema que no esperábamos para subir imágenes a la base de datos usando el tipo de dato BLOB, un objeto que contiene un binario de longitud variable.

El problema consistió en que a la hora de subir las imágenes a la base de datos, sólo funcionaba de forma local, no remotamente. Para añadir una imagen se utilizaba la función de MySQL "load_file()", la cual obtiene un fichero que se puede introducir como un dato binario en cualquier celda de tipo BLOB. Esta llamada, si se hace remotamente, no funciona, pues MySQL trata de obtener el fichero en la máquina local y no en la remota.

Para solucionarlo, se modificó un poco el diseño, y en vez de ser los nodos los que subieran las imágenes, sería un script situado en la máquina *monitor* de Caesaraugusta el que se encargara de recoger las imágenes de los diferentes nodos, guardarlas en un directorio local y entonces introducirlas en la base de datos.

Para evitar sobrecargar la entidad NODE, y poder realizar un borrado de las imágenes de forma sencilla, se creó una nueva entidad, llamada IMAGES, que es donde se almacenan todas las imágenes. De esta forma, el único dato que se obtiene mediante el método *pull* son las imágenes del estado de los nodos.

Además, realizando este cambio, se evita un problema de seguridad, ya que para que un usuario pueda guardar imágenes en MySQL necesita el permiso *File*, el cual es global para toda la base de datos y tiene problemas de seguridad. Si se usa indebidamente se pueden escribir ficheros en cualquier carpeta en la que el proceso mysqld tenga permisos. Como medida de seguridad no se permite sobreescribir ficheros, pero un usuario malintencionado podría crear ficheros indefinidamente y saturar el sistema de ficheros de la máquina.

### 4.1.2. El interfaz Web

Como se ha comentado en el apartado 2.2, para realizar la interfaz Web se ha utilizado la API World Wind Java SDK (versión 0.6.679.14208), y tras probar su funcionamiento en Eclipse y NetBeans, se ha decidido realizar el desarrollo en NetBeans 6.8 por su ligereza y mejor integración con la API, siendo Java 1.6.0_22 el lenguaje principal de programación usado.

Para usar la API World Wind en NetBeans, se ha empezado un nuevo proyecto en NetBeans, donde se han añadido todos los paquetes de la API y las librerías adicionales necesarias: *jogl* y *gluegen*, la primera es una librería gráfica y la segunda una librería que permite realizar llamadas a C desde Java.

Una vez que se ha logrado realizar un applet mínimo, y se ha entendido el funcionamiento de la API, se han ido añadiendo funcionalidades y módulos progresivamente para realizar la interfaz que se ha diseñado. A continuación se realiza un resumen de lo que se ha desarrolado en la interfaz Web, y en el AnexoB se puede contemplar una descripción más detallada.

Durante la implementación se ha utilizado javadoc para documentar todas las clases. Dado que el desarrollo del applet implica constantes llamadas a las funciones de la API de World Wind, se ha optado por adjuntar la documentación javadoc del paquete implementado junto a la documentación javadoc ya existente de la API, de forma que todo futuro desarrollador pueda entender fácilmente el código fuente. Dicha documentación se puede encontrar en el CD adjunto, en la carpeta /otrosAnexos/javadoc.

Se ha creado un nuevo paquete llamado "reswwja" (RES World Wind Java Applet), donde se ha incluido todo el código desarrollado para generar la nueva aplicación, de forma que se genera un fichero .jar que los navegadores pueden bajar para ejecutar el applet. La estructura del paquete es la siguiente:

- *reswwja*: contiene los ficheros html que contienen el applet así como la clase principal del applet.
- *reswwja.db*: contiene la clase con las funciones necesarias para obtener datos de la base de datos.
- *reswwja.images*: contiene todas las imágenes usadas por el applet.
- *reswwja.infoObjects*: contiene todas las clases de los objetos más usados por el applet, que coinciden con las entidades vistas en el esquema E/R de la base de datos.
- *reswwja.window*: contiene las clases necesarias para manipular las ventanas del applet, el mapa del mundo y todos los objetos que se sitúan sobre el mapa.

La interacción entre los paquetes es la que se muestra en la Figura 4.1. En la figura podemos observar que la implementación final difiere un poco del diseño inicial. El módulo que se pensó que podía ser el que situaría los elementos sobre el mapa ha desaparecido para integrarse en el de las ventanas y se ha creado otro nuevo, que almacena todas las imágenes del applet.

Una vez terminada la programación, se obtiene un fichero .jar con todas las clases e imágenes empaquetadas, se firma el fichero digitalmente y se copia al directorio donde puede ser descargado por los navegadores junto al resto de ficheros html y librerías necesarias.

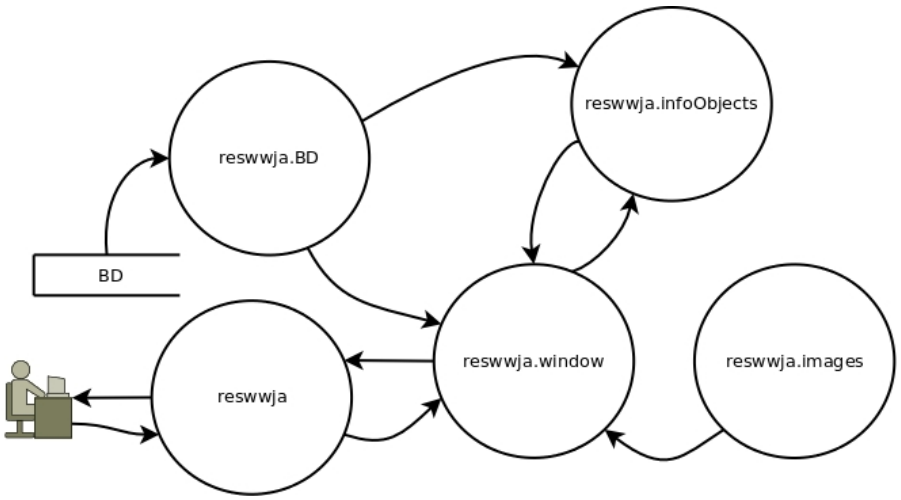


Figura 4.1: Módulos finales del interfaz. Las flechas indican la dirección del flujo de información.

Para poder tener en la máquina *monitor* de Caesaraugusta la aplicación Web, se ha instalado y configurado Apache 2.2.3, de forma que la página principal que contiene el applet es un virtual host, lo que permite que se puedan alojar más páginas en el servidor Apache. La página donde actualmente está alojada la aplicación es: `http://map.res.unizar.es`, un CNAME[2] solicitado por requerimiento del proyecto.

### Problemas encontrados

Uno de los problemas encontrados ha sido el comportamiento del applet en diferentes navegadores, ya que se esperaba que fuera idéntico y se vió que no era así. Vimos que dependía del navegador y del sistema operativo dicho comportamiento. Por ejemplo, el navegador Opera abre todos los enlaces a diferentes Webs sobre una misma pestaña y el resto de navegadores lo hace en diferentes pestañas. Otro comportamiento extraño era que un mismo navegador, Google Chrome, sobre Linux no respondía a las órdenes de teclado y sobre Windows sí.

Estos pequeños problemas se han solventado mediante reorganización de código y substitución de funciones por otras que sí parecen tener el mismo comportamiento en todos los navegadores y sistemas operativos.

Sin embargo, el mayor problema ha sido lograr que funcionara sobre el sistema operativo MacOS X y sobre Safari. Todos los demás navegadores sobre

[2]Un CNAME es un alias de un *hostname* DNS, de forma que se pueden alojar varios CNAMES en un mismo servidor

otros sistemas operativos funcionaban bien, pero por alguna razón MacOS no ejecutaba el applet.

Tras investigar el caso, se ha comprobado que era debido a la forma de ser lanzado un applet. Existen dos formas, la antigua, donde se especifican todos los parámetros del applet en un *tag* del fichero html que lo contiene, y la nueva, donde se lanza el applet usando Java Network Launch Protocol (JNLP). MacOs utiliza su propia versión de Java, y sigue utilizando la antigua forma de lanzar los applets, por lo que todo navegador que se ejecute sobre MacOS utiliza dicho método. Había un error de nomenclatura en el tag y faltaba un fichero que ayuda a lanzar el applet, pero como la consola de Java de MacOs no daba ninguna información más que la versión de Java que se estaba ejecutando, costó encontrar el problema.

Finalmente se han solucionado los errores, modificando el *tag* correctamente y añadiendo el fichero que hacía falta, comprobándose posteriormente el buen funcionamiento del applet.

Otro problema que ha surgido ha estado relacionado con NetBeans y SVN, resultando ser que la versión estable de NetBeans (versión 6.8) que se usa en OpenSuse 11.3 tiene un *bug* que no permite a un usuario normal configurar un repositorio SVN, por problemas de permisos. El problema se ha solucionado ejecutando NetBeans como usuario root, configurando el repositorio SVN que se utilizaba y copiando los ficheros de configuración de NetBeans al usuario que se usa por defecto, haciendo que los ficheros pertenezcan a dicho usuario.

Por último, otro problema encontrado ha sido el comportamiento del recolector de basura de Java. Al empezar a crear objetos nuevos para mostrar información sobre el mapa, especialmente objetos con imágenes, se ha observado un comportamiento extraño en el consumo de memoria RAM, ya que se estaban utilizando 500Mbs más de lo habitual al ejecutarse el applet. Y se comprobó que en otros sistemas y navegadores el comportamiento era similar, en cada uno se utilizaba una cantidad de memoria extra que era variable dependiendo del entorno.

Tras realizar varias pruebas, se ha comprobado que era debido a la creación de objetos nuevos en Java. El recolector de basura no actuaba hasta que se alcanzaba una cuota mínima de memoria.

Para solucionar el problema se ha reutilizado todo lo posible las variables de objetos, y para las imágenes que eran dinámicas, como la barra que indica la carga de los nodos, se ha utilizado un método que se emplea en CSS para diseño Web, que consiste en generar una imagen más grande de lo necesario y mostrar sólo una parte de ella, la que nos interesa en cada momento.

### 4.1.3. Administrador Web de la base de datos

A mitad del desarrollo del interfaz Web, se ha visto que empezaba a ser necesario contar con un gestor de MySQL, tanto para agilizar las pruebas que se realizaban como para dejar instalado un gestor desde el que los futuros administradores de la aplicación pudieran controlar el estado de la base de datos.

Se ha analizado la posibilidad de usar Joomla o PHPMyAdmin como posibles gestores de la BD. Tras ver las posibilidades, y dado que era para gestión interna y se prefería una aplicación ligera, se ha optado por PHPMyAdmin.

Lo primero que se ha intentado es utilizar el gestor de paquetes de Yast para instalar PHPMyAdmin, pero nos encontramos con que había problemas. Existían ciertas dependencias en los paquetes que Yast no sabía resolver, por lo que no era posible instalar el programa mediante el gestor de paquetes de SLES. Por lo tanto, hubo que mirar la documentación oficial de PHPMyAdmin [24] para ver qué paquetes eran necesarios y descargar los rpms[3] necesarios para realizar una instalación manual. Lo más destacable fue la instalación de la extensión *mcrypt*, ya que la instalación se realizaba sobre una arquietectura de 64 bits.

Tras instalar manualmente PHPMyAdmin y configurarse para que pida el usuario y contraseña de acceso en cada sesión, se ha configurado Apache para que el directorio donde se encuentra PHPMyAdmin fuera privado, de forma que se solicita un usuario y contraseña a nivel de Apache para poder acceder al administrador de la base de datos. Además se ha configurado también Apache para que a dicho directorio sólo se pueda acceder desde unas determinadas IP's concretas del BIFI. De esta forma, securizamos el gestor garantizando que pueda ser accedido únicamente desde unas máquinas determinadas y por las personas que conozcan la contraseña del directorio.

## 4.2. Pruebas del sistema

En las dos últimas semanas se han realizado las pruebas oportunas para comprobar el correcto funcionamiento del sistema desarrollado. Dichas pruebas se han realizado en dos fases, la primera usando datos reales de Caesaraugusta y la segunda usando datos ficticios. Los datos se han introducido manual y automáticamente en la BD, pero la comprobación de los resultados a sido totalmente manual, de forma que se ha podido garantizar que el comportamiento es el esperado.

Los datos reales han servido para demostrar que la información se obtiene de los nodos de forma correcta, garantizando que no hay fallos a la hora de mostrar los datos y que éstos están actualizados. Las pruebas han servido para saber que los scripts realizados funcionan bien, que la BD se está actualizando correctamente y finalmente, que la interfaz Web puede acceder a los datos sin problemas y que además los muestra como se espera.

Para la segunda fase se han usado diferentes baterías de pruebas, insertando los datos en la BD manualmente. Las distintas pruebas se han diseñado para abarcar todos los casos posibles, de forma que se ha comprobado el correcto funcionamiento del applet. Se ha podido demostrar que el acceso por nivel de autorización es correcto, ya que los datos mostrados se corresponden con los permisos que tiene el usuario. Se ha visto que la integridad de la BD se mantiene en todo momento, los datos no se guardan en las tablas si no cumplen las restricciones y se evitan problemas a la hora de mostrarlos en la interfaz Web. También ha servido para comprobar que toda la información se representa con el formato deseado y de forma correcta.

[3] RPM es el formato estándar de empaquetamiento de software utilizado por OpenSuse y SLES en este caso concreto

## 4.3. Despliegue en los demás nodos

Como se ha comentado, en Caesaraugusta se han desarrollado unos scripts que recopilan la información y la almacenan en la base de datos.

Dichos scripts se han diseñado para que puedan funcionar en cualquier otro nodo siempre y cuando tenga las mismas aplicaciones que en Caesaraugusta se utilizan para enviar los datos al BSC (Ganglia y Moab).

Lo único que han de hacer los nodos es instalar en un directorio todos los scripts y habilitar en los cortafuegos el acceso a la máquina de Caesaraugusta, donde se aloja el script que recopila las imágenes del estado de los demás nodos. Además, se ha de modificar el cron del usuario desde el que se instalen los scripts, de forma que se ejecuten automáticamente. Todo esto está detallado en el Anexo D, donde se encuentra el manual de despliegue para un nodo cualquiera.

# Capítulo 5

# Conclusiones

En éste capítulo se realiza una valoración crítica del conjunto del trabajo realizado. Se valora el cumplimiento de los objetivos y las posibilidades de continuación y mejoras. Finalmente, se concluye con una valoración personal del trabajo realizado.

## 5.1. Objetivos cumplidos y resultados obtenidos

En el desarrollo de este proyecto se han cumplido los objetivos inicialmente propuestos. En líneas generales estamos satisfechos con los resultados obtenidos.

Se realizó un estudio previo del sistema y se propusieron diferentes alternativas, recomendando las que mejor se adaptaban a los requisitos.

Se diseñó un sistema que permitiese recopilar los datos y mostrarlos a los usuarios, dando especial énfasis a la geolocalización. Además se muestran los datos de acuerdo a unos niveles de autorización establecidos, mediante un sistema de contraseñas, que además añade seguridad a la aplicación.

Respecto a los resultados logrados, estamos contentos de ver que la idea inicial ha podido ser implementada. La interfaz Web final que permite la visualización de los datos nos ha sorprendido especialmente, gracias a World Wind hemos obtenido una aplicación que muestra los datos geoposicionados de una forma visual bastante impactante, mejor de lo que nos esperábamos.

La recopilación de los datos se realiza de una forma automatizada en gran parte gracias a los scripts realizados, pero no hay una recopilación totalmente automatizada, actualmente no es posible con los recursos con los que contamos, pero esperamos que en un futuro pueda ser factible.

Las pruebas realizadas demuestran que es posible el uso de este sistema de información para recopilar y visualizar los datos de la RES, el estado actual del proyecto así lo permite, y estamos convencidos de que las posibilidades que tiene este proyecto de cara al futuro son muy amplias.

## 5.2. Desarrollo del proyecto

El proyecto ha estado marcado en todo momento por una fase contínua de documentación. Conforme se iban realizando las diferentes tareas, siempre he tenido que investigar y documentarme sobre la tarea que estaba realizando en ese momento.

Por ello, exceptuando la primera fase de análisis previo del sistema, el diseño e implementación se han solapado, como puede observarse en la Figura 5.1. Mientras se desarrollaba el diseño inicial, se observaban las posibles mejoras y problemas del diseño, por lo que se modificaba conforme a los nuevos datos obtenidos en la documentación u observados durante la implementación.

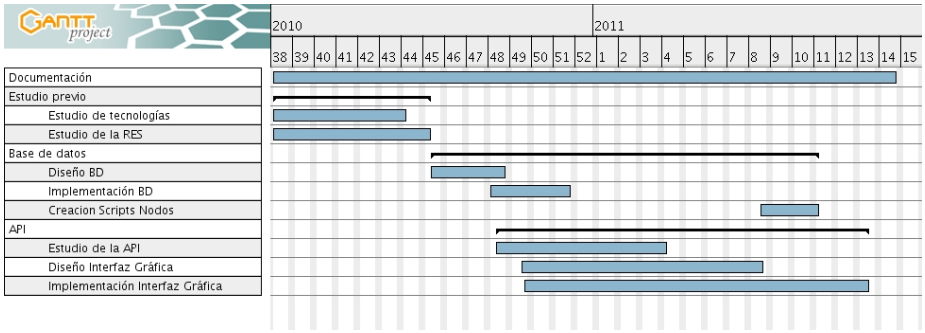


Figura 5.1: Diagrama Gantt del avance del proyecto.

Otro hecho importante es la documentación propia del proyecto. Desde el principio el proyecto se fue documentando, de forma que los codirectores tenían acceso al estado actual del proyecto y a toda la información relativa a él.

Esto es significativo, dado que a la hora de finalizar el proyecto tras seis meses de trabajo, la documentación estaba prácticamente realizada, sólo ha sido necesario reorganizarla y presentarla adecuadamente. Toda persona que quiera continuar con el desarrollo de este proyecto podrá encontrar la información adecuada para poder realizar dicha tarea.

La dedicación del proyecto ha sido completa desde octubre hasta comienzos de abril, realizando una pequeña pausa de dos semanas en febrero para realizar los últimos exámenes que me quedaban. Durante este tiempo he dedicado unas 7 horas diarias a la realización del proyecto.

## 5.3. Posibles mejoras y vías de desarrollo

El estado actual del proyecto tiene bastantes aspectos que pueden ser mejorados, y las posibilidades del desarrollo futuro son innumerables, aquí se detallan algunas de ellas:

- Mejorar la automatización de la actualización de la base de datos. Actualmente se actualiza con los scripts desarrollados para tal efecto que se

ejecutan en cada nodo, pero a largo plazo se espera que la base de datos esté sincronizada con la del BSC. Durante las últimas semanas del proyecto se iniciaron las conversaciones con los encargados de las bases de datos del BSC, por lo que es una mejora que podría realizarse antes de lo esperado.

- Automatizar la creación de los usuarios de MySQL. Debido a que hay ciertos datos que no se pueden actualizar y añadir en la base de datos de forma automática, la creación de los usuarios de MySQL conlleva una pequeña intervención humana. Es de esperar que en un futuro no sea necesaria dicha intervención, ya que todos los datos estarán disponibles en la base de datos. Por ejemplo, algunos de los datos que no se pueden obtener automáticamente son las coordenadas de las instituciones o a qué institución pertenece un usuario.

- Desarrollar una segunda vista, que en vez de mostrar los datos relativos a un nodo, muestre los datos relativos a un grupo. Actualmente se muestran los datos desde el punto de vista de un nodo, es decir, al selecionar un determinado nodo se muestran los datos asociados a él. Si se implementa una segunda vista, podría obtenerse la información desde otro punto de vista, más interesante para un usuario de la RES, ya que el punto de vista actual es más interesante para un administrador de un nodo o para una persona ajena a la RES.

- Mejorar el sistema de seguridad mediante un cifrado SSL. En los últimos días del desarrollo se recibió un certificado de TERENA que permite tanto cifrar como firmar, que podría ser utilizado para cifrar las comunicaciones con la base de datos y firmar de paso los ficheros .jar que el navegador ha de descargar.

- Desarrollar un método por el cual los datos de la VPN que conecta los nodos se actualice en la base de datos. Actualmente, tanto la base de datos como el interfaz Web están preparados para mostrar dichos datos, pero al carecer el nodo Caesaraugusta de conexion VPN no ha podido ser posible estudiar la forma en que los datos puedan ser introducidos en la base de datos.

- Añadir nuevas capas, objetos o datos a monitorizar en la base de datos. Según se utilice esta aplicación, podrán observarse nuevas formas de uso o de parámetros a monitorizar, que gracias al desarrollo modular se podrán implementar con facilidad.

Estas son algunas de las posibles mejoras y vías de desarrollo que se podrían realizar, y nos permiten darnos cuenta de las grandes posibilidades que tiene este proyecto.

Visto todo esto, podemos ver que el proyecto, pese a estar en una fase estable, está en los comienzos de su vida, si continúa su desarrollo podrá mejorar y cambiar de una forma sorprendente para todos aquellos que lo hemos visto en su actual estado.

## 5.4. Opinión personal

Durante los seis meses que he estado en el BIFI desarrollando este Proyecto Fin de Carrera, tengo que destacar que he estado muy a gusto trabajando con las personas que me han dirigido y ayudado a realizar el proyecto. El entorno laboral ha sido inmejorable, algo imprescindible para poder trabajar en un equipo.

A nivel personal, estoy satisfecho con el trabajo realizado, me ha permitido tomar parte de todas las fases de un proyecto: análisis, diseño, implementación y puesta en producción. He podido aprender y trabajar en diferentes áreas que es lo que buscaba, e integrar todo ello en un único proyecto.

Por otro lado, he de agradecer el grado de libertad que me han otorgado mis codirectores, su contínuo apoyo y paciencia. Creo que gracias al buen ambiente que generaban he podido terminar este proyecto satisfactoriamente, no habría sido lo mismo sin ellos. Así mismo, debo agradecer a mi ponente toda la ayuda y los consejos proporcionados, gracias a los cuales he podido simplificar muchas tareas.

# Bibliografía


[1] Ramez Elmasri, Shamkant B. Navathe: Fundamentos de sistemas de bases de datos, Pearson Addison Wesley (2007)

[2] B. Shuwartz: MySQL Avanzado, ANAYA Multimedia (2009)

[3] K. Mukhar, T. Lauinger, J. Carnell: Fundamentos de bases de datos con java: JDBC, SQL, J2EE, EJB, JSP, XML, ANAYA Multimedia (2002)

[4] J. Gerner: LAMP: desarrollo web con Linux, Apache, MySQL y PHP 5, ANAYA Multimedia (2006)

[5] A. Ford: Apache 2 Pocket Reference, O'Reilly Media (2008)

[6] R. Bowen K. Coar: Apache Cookbook, second edition, O'Reilly Media (2007)

[7] F.J. Ceballos Sierra: Java 2: Interfaces gráficas y aplicaciones para internet, RA-MA (2008)

[8] J.R. Garcia-Bermejo Giner: Java 2, Pearson educación (2001)

[9] C. Deffaix Remy: Programación shell en Unix/Linux SH (BOURNE), KSH, BASH, ENI (2010)

[10] C. Newham: Learning the BASH shell, O'Reilly & associates (2005)

[11] P. Katcheroff: El gran libro de Linux, MP Ediciones (2006)

[12] F. Mittelbach, M. Goossens, J. Braams, D. Carlisle, C. Rowley: The LaTeX Companion, Second Edition, Addison-Wesley Professional (2004)

[13] H. Kopka, P. W. Daly: Guide to LaTeX, Fourth Edition, Addison-Wesley Professional (2003)

[14] K. Goelker: GIMP 2 for Photographers, Rocky Nook (2006)

[15] GanttProject:
`http://www.ganttproject.biz/`
Último acceso: abril 2011.

[16] Documentación de Ganglia:
`http://sourceforge.net/apps/trac/ganglia/wiki`
Último acceso: enero 2011.

[17] Nagios:
http://wiki.nagios.org/index.php/Main_Page
Último acceso: enero 2011.

[18] Documentación de Munin:
http://munin-monitoring.org/wiki/Documentation
Último acceso: enero 2011.

[19] Google Maps API:
http://code.google.com/apis/maps/index.html
Último acceso: enero 2011.

[20] Openlayers:
http://openlayers.org/
Último acceso: enero 2011.

[21] World Wind API:
http://worldwind.arc.nasa.gov/java/
Último acceso: enero 2011.

[22] Documentación de MySQL:
http://dev.mysql.com/doc/
Último acceso: enero 2011.

[23] Documentación de Java:
http://download.oracle.com/javase/1.4.2/docs/api/index.html
Último acceso: abril 2011.

[24] Documentación de PHPMyAdmin:
http://www.phpmyadmin.net/documentation/
Último acceso: abril 2011.