

Máster Internacional en Software Libre
Administración de Web y de comercio electrónico en entornos de software libre
Memoria - Proyecto Web Fin de Máster
Enero 2011

Web de Control de Tráfico Marítimo (WCTM)

Memoria del Proyecto

Autor: José Lucas Nogales

Consultor: Francisco Javier Noguera Otero

La Haya 24 de Enero del 2011

Licencia

La licencia elegida para la documentación del proyecto es la de [Creative commons Reconocimiento-CompartirIgual](#)



Estas son las características de la licencia:

Usted es libre de:

- copiar, distribuir y comunicar públicamente la obra
- hacer obras derivadas

Bajo las condiciones siguientes:

- **Reconocimiento** — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).
- **Compartir bajo la misma licencia** — Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

Entendiendo que:

- **Renuncia** — Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor
- **Dominio Público** — Cuando la obra o alguno de sus elementos se halle en el dominio público según la ley vigente aplicable, esta situación no quedará afectada por la licencia.
- **Otros derechos** — Los derechos siguientes no quedan afectados por la licencia de ninguna manera:
 - Los derechos derivados de usos legítimos u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.
 - Los derechos morales del autor;
 - Derechos que pueden ostentar otras personas sobre la propia obra o su uso, como por ejemplo derechos de imagen o de privacidad.
- **Aviso** — Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.

Resumen del proyecto

El proyecto Web de Control de Tráfico Marítimo (WCTM) es un proyecto Web que va a gestionar información útil para el Control del Tráfico Marítimo de una determinada zona del mundo.

Hoy en día existen sofisticados sistemas de Control de Tráfico Aéreo. Sin embargo para el control marítimo, aun existen deficiencias. Cada país o zona tiene su propia información y gestiona su propia información, pero esa información no es compartida con otras zonas o países.

El sistema [AIS](#), es decir el Sistema de Identificación Automática, es ampliamente usado para la identificación, seguimiento y ayuda a la navegación marítima. Todos los barcos de pasajeros y aquellos de más de 300 toneladas están obligados a montarlo.

Por otra parte, la [OTAN](#) tiene un sistema llamado [NIRIS](#), que es capaz recibir los mensajes provenientes del sistema AIS y proporcionar a su vez esa información en forma de servicios Web ([Web Services](#)) a toda la Internet.

La red Internet nos proporciona unos medios inigualables a la hora de compartir información y recursos en cualquier parte del mundo.

El proyecto WCTM va emplear todos estos recursos en Internet para cubrir las carencias que existen en el Control de Tráfico Marítimo. El proyecto usa el concepto de arquitectura [SOA](#), una arquitectura que se basa en los servicios Web para intercambiar información entre sitios diferentes.

Nuestra aplicación proporciona en tiempo real información sobre los buques, apoyándose en los mapas de [Google](#) ([Google maps](#)) para posicionar cada uno de ellos en el lugar correspondiente del mapa.

Para tener la situación del Tráfico Marítimo en tiempo real, nuestra aplicación establecerá llamadas periódicas al servicio Web de NIRIS para mantener actualizada la posición de los buques en todo momento. Esta situación se actualizará con el intervalo que el usuario crea conveniente y usando el tipo de mapa que el usuario haya seleccionado.

Para conseguir los objetivos de actualización de las trazas en el mapa se usa la tecnología [AJAX](#). La cual permite, básicamente, actualizar una parte de la página Web sin cargar de nuevo la página completa.

Se pretende que aparte de la posición del buque, también proporcione otros datos importantes del mismo como identificación, rumbo, velocidad, etc.

Otro aspecto novedoso que nuestra aplicación ofrece al control del tráfico marítimo es la opción de registrar en una base de datos la posición de los buques en cada momento junto con otros detalles importantes.

La intención es la de poder analizar los datos a posteriori y determinar la posición de un determinado barco en una determinada fecha y hora.

Se va a realizar diseño del sistema usando una arquitectura multi-capa ([Multi-Tier](#)). Que consiste en separar cada uno de los componentes de la aplicación en una capa (Tier) separada del resto.

Para cada uno de los componentes, lenguajes y utilidades se emplearán estándares abiertos y software libre.

Índice

1 Presentación del proyecto.....	5
1.1 Introducción.....	5
1.2 Objetivos.....	6
1.3 Tecnologías y herramientas empleadas.....	7
2 Estudio de viabilidad.....	9
2.1 Establecimiento del alcance del sistema.....	9
2.2 Estudio de la situación actual.....	10
2.2.1 Sistema AIS.....	10
2.2.2 Sistema de Control de Tráfico Marítimo.....	12
2.2.3 NIRIS.....	13
2.2.4 Mapas de Google.....	13
2.2.5 Nuestra aplicación (WCTM).....	13
2.3 Definición de los requisitos del sistema.....	13
2.4 Estudio de las alternativas de solución.....	15
2.4.1 Propietaria.	15
2.4.2 Libre.	17
2.4.3 Virtualización.....	18
2.5 Valoración de las alternativas.....	20
2.6 Selección de la solución.....	22
3 Análisis del sistema.....	24
3.1 Definición del sistema.....	24
3.2 Establecimiento de requisitos.....	27
3.2.1 Presentación de trazas.....	27
3.2.2 Análisis de Históricos.....	28
3.3 Definición de interfaces de usuario.....	30
3.3.1 Página principal.....	31
3.3.2 Página de autenticación.....	32
3.3.3 Página de históricos.....	32
3.3.4 Página de ayuda.....	33
3.4 Análisis de riesgos.....	33
3.5 Planificación temporal.....	34
3.6 Presupuesto.....	36
3.7 Especificación del plan de pruebas.....	37
3.7.1 Pruebas Unitarias.	37
3.7.2 Pruebas de Integración.	38
3.7.3 Pruebas de Implantación.	39
3.7.4 Pruebas de Aceptación.	39
4 Diseño del sistema.....	40
4.1 Arquitectura.....	40
4.1.1 Identificación de subsistemas.....	41
4.1.2 Subsistema de Presentación de Tráfico.....	43
4.1.3 Subsistema de Análisis de Históricos.....	47
4.1.4 Diseño de la base de datos.....	52
4.1.5 Especificación de estándares, normas de diseño y construcción.....	53
4.2 Elección de alternativas de componentes y licencias más adecuadas.....	53
4.3 Especificaciones de desarrollo.....	54
4.3.1 Necesidades de equipos	54
4.3.2 Necesidades de software.....	55
4.4 Especificación de pruebas.....	56
4.4.1 Pruebas Unitarias.	56
4.4.2 Pruebas de Integración.	57
4.5 Requisitos de implantación.....	57
5 Desarrollo.....	58
5.1 Planificación de las actividades de desarrollo e integración de sistemas.....	58

5.2 Desarrollo.....	59
5.3 Documentación.....	61
6 Implantación.....	62
6.1 Formación.....	62
6.2 Implementación del sistema y pruebas.....	63
6.2.1 Instalación de servidores y aplicaciones.....	64
6.2.2 Pruebas de implantación.....	65
6.3 Nivel de servicios.....	66
6.4 Aceptación del sistema.....	66
7 Mantenimiento.....	66
8 Conclusiones.....	67
9 Referencias.....	69

1 Presentación del proyecto

1.1 Introducción

Este proyecto tiene su campo de actuación en la gestión de la información de tráfico marítimo. No se pretende reemplazar a los sistemas de Control de Tráfico Marítimo que existen en cada zona marítima, sino que nuestra Web sea un complemento que permite a los usuarios consultar de una forma ágil, fácil y rápida la citada información.

Para cumplir esos requisitos se va a emplear un recurso disponible a nivel mundial, la Web de Internet.

La página Web del sistema que proporciona esta información, estará disponible para todos los usuarios. Se podrá usar tanto en embarcaciones que tengan acceso a Internet como en estaciones de tierra, sin olvidar a cualquier usuario que lo necesite.

Cada buque hoy en día posee un sistema de posicionamiento global ([GPS](#)) que le permite saber en cada momento su posición con márgenes de error irrelevantes cuando se trata de una embarcación marítima, así como giroscópicas, correderas y otros aparatos que informan de su rumbo y velocidad.

Al mismo tiempo que esta información de posicionamiento es útil para la navegación del propio buque, también lo es para un sistema de control de tráfico ([VTS](#)), es decir una estación en tierra en la que gestiona toda la información en tiempo real recibida de cada uno de los buques de la zona.

Esta información marítima se intercambia a través del sistema [AIS](#) que usando una comunicación [VHF](#) con equipos transmisores y antenas, permite crear una red en la que todas las unidades que intervienen en el control del tráfico marítimo están conectadas.

El sistema [NIRIS](#) de la [OTAN](#) está conectado al sistema AIS para recibir los mensajes de información marítima. Estos mensajes son mensajes de texto codificado en sentencias [NMEA](#) de 64 bits de longitud. Los mensajes son interpretados por el [NIRIS](#) y puesto a la disposición de todos en Internet en forma de [Web Services](#).

El proyecto hará un uso extensivo de recursos existentes. Entre ellos el citado sistema NIRIS, así como los mapas de Google que serán utilizados para posicionar cada traza (buque) que manda el sistema NIRIS sobre el mapa adecuado.

Nuestra aplicación (WCTM) se encargará de usar toda esa información disponible para que cualquier usuario pueda consultar el estado del tráfico marítimo desde cualquier parte del mundo. También se guardará en una base de datos toda la información recibida, lo que permitirá realizar un análisis de esos datos registrados, pudiendo saber la historia (el recorrido que ha efectuado) de cada una de los buques participantes de la red AIS.

1.2 Objetivos

Este proyecto pretende crear una aplicación Web en la cual se presente de forma adecuada fácil de interpretar, información de buques y aeronaves que puede ser de mucha utilidad para la gestión del Control de Tráfico Marítimo de una zona.

Los sistemas de control marítimo actuales tienen ciertas limitaciones, principalmente el alcance y requieren una tecnología compleja y costosa. Debido al sistema de comunicaciones que usa ([VHF](#)) su alcance es limitado, máximo 80 Km. en condiciones meteorológicas favorables, a eso hay que sumar los costes tecnológicos que supone formar parte de la red [AIS](#). Cualquier embarcación que necesite implementar este sistema debe de instalar equipos transmisores, receptores y antenas. Una inversión económica importante.

Como se ha indicado anteriormente no es un objetivo de este proyecto reemplazar los actuales Sistemas de Control de Tráfico marítimo, se trata de un complemento a los mismos. Las capitanías de zonas seguirán contando con sus sistemas tradicionales de control de tráfico.

Pero la ventaja de este proyecto es la accesibilidad a esa información y la capacidad de grabarla para un posterior análisis. Existen situaciones en las cuales es necesario por ejemplo motivos judiciales o policiales demostrar donde se encontraba un determinado buque en una fecha dada. Esto es factible con nuestra aplicación. Ya que se puede recuperar de la base de datos o de las copias registradas de la misma, la información requerida.

A continuación se describen los objetivos del proyecto en detalle:

- **Proporcionar una aplicación Web accesible para todos.**
Se trata de una aplicación Web accesible desde cualquier navegador ya sea libre o propietario sin necesidad de contratación.
- **Presentar información del Tráfico marítimo de cualquier parte del mundo.**
La información se actualiza en el cliente usando los servicios Web de NIRIS. Si hay un NIRIS en cualquier parte del mundo recibiendo información marítima, esta información estará automáticamente disponible en Internet y accesible a nivel mundial.
En esta primera versión al disponer solo de un proveedor de servicios Web, se recibirá de la parte del norte de Europa, mar del Norte, canales Holandeses, Dinamarca, Finlandia y Suecia. Con la opción de visualizar trazas aéreas de [EUROCONTROL](#).
- **Proporcionar la información del Tráfico Marítimo en mapas de Google.**
Usando el API de Google Maps, que está disponible in Internet para todos, podemos añadir en nuestra aplicación Web sin coste adicional, cualquier de los mapas de Google. A partir de la versión 3 del API no es necesario solicitar una “clave” para acceder al servicio.
- **Posibilidad de seleccionar que información quiere presentar el usuario.**
La aplicación Web permitirá al usuario seleccionar que bloque de información quiere presentar en el mapa. Cada sistema del que recibe NIRIS, está encargado de una zona concreta, el usuario podrá seleccionar que zona quiere presentar.
- **Seleccionar que tipo de mapa quiere visualizar el usuario.**
Parte del API de Google map es seleccionar que visión del mapa queremos usar. Esas opciones estarán disponibles para el usuario así como seleccionar la escala o moverse a cualquier parte.

- Obtener detalles de cada uno de los buques presentados.**

Cuando el usuario seleccione un objeto (un buque) en el mapa, se desplegará una ventana con información básica sobre el buque. Adicionalmente se proporciona un enlace para acceder a todos los detalles sobre el buque disponible en la página propia del sistema original (NIRIS).

- Registrar las diferentes posiciones de los buques.**

El sistema guardará en una base de datos la identificación, posición, rumbo y velocidad de cada una de las trazas que entran en el sistema en una fecha dada.

- Analizar las trazas registradas en la base de datos.**

El sistema proporcionará mecanismos y utilidades para consultar y analizar la información guardada en la base de datos de cada una de las trazas.

- Liberarlo con licencia GPL V3.**

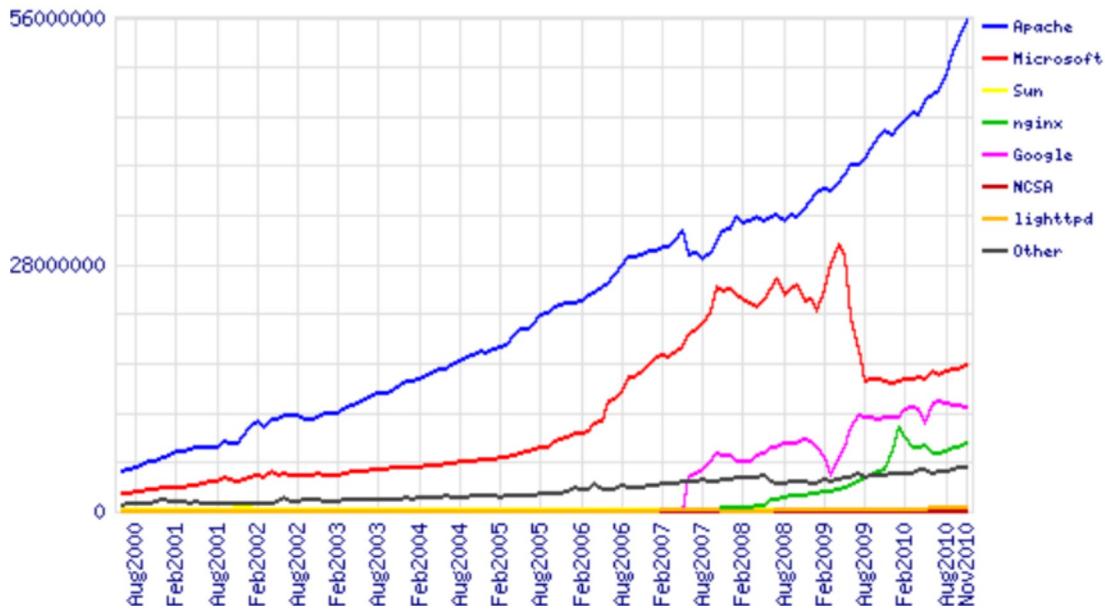
Esto permitirá que el proyecto se pueda mejorar por la comunidad de usuarios que estén interesados en él.

En un software licenciado con [GPL](#) el código fuente siempre estará disponible y todos los proyectos derivados de este, por ley deben ser también GPL.

1.3 Tecnologías y herramientas empleadas

Repasemos ahora qué herramientas software usaremos para este proyecto.

A la hora de realizar una aplicación Web es fácil escoger qué servidor Web sustentará nuestra aplicación, este es [Apache](#). Apache es con diferencia el servidor Web más usado de la actualidad. En una [encuesta](#) realizada en noviembre del 2010 el porcentaje de servidores usando Apache era casi el 60% frente al 16% de sistema de Microsoft. Con casi 56 millones de servidores usando Apache en todo el mundo.



Por lo tanto nuestra Web estará alojada en ese robusto y eficiente servidor.

Como hemos indicado anteriormente, otro componente importante es el mapa en el cual situaremos la posición marítima. Aquí también optamos por un sistema ya existente que ofrece sus mapas de forma libre, [Google maps](#).

Usaremos llamados al API de Google para presentar cada objeto en el mapa. Se ha decidido usar la versión 3 del API de Google ya es la recomendada por Google, es más rápida y no necesita una “key”, código de acceso, como su predecesora el API v2. Ambas usan [JavaScript](#).

En cuanto a los lenguajes de programación se ha optado por dos, una para realizar la parte que se ejecuta en el servidor, [PHP](#) y otro en la parte cliente, [JavaScript](#). El motivo para esta elección es por su sencillez y rapidez con la que se desarrollaran aplicaciones Web.

Para comunicarse con los servidores NIRIS y Google maps, se usa la tecnología [AJAX](#). Esta permite actualizar los contenidos de la página Web de forma asíncrona sin tener que recargar la página completa.

Como hemos indicado anteriormente el NIRIS ofrece su información en forma de [Web services](#). Un “servicio Web” es una aplicación a la que se accede por medio del protocolo HTTP y que ofrece sus funciones o métodos (servicios en una palabra) a través de la Web. Esto hace que sistemas heterogéneos con diferentes plataformas, distintos sistemas operativos y variados lenguajes de programación, se puedan comunicar y operar entre sí.

Típicamente existen dos modos de realizar llamadas al servicio Web.

Con mensajes [SOAP](#). Protocolo para intercambiar la información entre el servidor y el que consume el servicio. Está basado en XML y la información va empaquetada en una envoltura (envelope) común para cada mensaje.

O bien con llamadas [REST](#), que hace lo mismo pero no usa un mensaje predefinido para empaquetar la información, tan solo con hacer una llamada [HTTP](#) por ejemplo del tipo GET o POST recibimos la información. Este es el método favorito en los sistemas que comienzan a ofrecer sus servicios usando la Web.

WCTM usa llamadas del tipo [REST](#). Nuestra aplicación Web recibe el contenido en un formato XML. Interpreta ese XML para extraer la información necesaria.

Como base de datos se ha optado por MySQL, también libre y muy potente adecuada para almacenar nuestra información de las trazas.

Todos los componentes funcionaran con un sistema operativo GNU Linux, se ha elegido la distribución de Debian, por su robustez, escalabilidad y facilidad de actualización.

Con todo este tenemos un sistema conocido como “LAMP”: Linux, Apache, [MySQL](#) y PHP.

Para que sea un sistema de alto rendimiento se usara un load balancer. Un sistema que implementa esta tecnología permite desviar las llamadas por parte de los usuarios a dos o más servidores reales. Para el usuario es como si la aplicación estuviese en un solo servidor Web, es el load balancer el que de forma transparente se encarga de distribuir las diferentes peticiones entre todos y cada uno de los servidores Web que implemente el cluster. Para esta gestión se usa la herramienta [Ultra Monkey](#).

Para asegurar el sistema contra accesos no autorizados se empleará un firewall, que haga un filtrado de paquetes usando la herramienta “[iptables](#)” propia del sistema operativo GNU Linux.

En cuanto al diseño se empleará una arquitectura multi-capas ([Multi-tier](#)) en la que cada componente (tier) es independiente del resto.

Finalmente desarrollaremos todo el proyecto usando estándares abiertos, tanto el desarrollo como en la gestión y documentación del mismo.

Este es un resumen de los componentes empleados:

- Debian GNU **Linux** como sistema operativo.
- Servidor **Apache** para servidor Web.
- Base de datos **MySQL**.
- Lenguajes de programación **PHP**, JavaScript, **HTML**, **SQL**.
- Formato de intercambio de datos, estándar **XML**.
- **OpenOffice.org**, documentación, gestión del proyecto.
- **Dia**, **Gantt project**, para la creación de diagramas.
- **SoapUI**, herramienta para analizar Web services.

2 Estudio de viabilidad

Una vez realizada una presentación del proyecto, en este capítulo se realiza un estudio completo de viabilidad. Al final de este estudio tendremos una solución con la cual se debe decidir si con ella se pueden cumplir los requisitos del cliente con los recursos disponibles.

2.1 *Establecimiento del alcance del sistema*

Los sistemas de control marítimo actuales tienen ciertas limitaciones y requieren sistemas sofisticados y caros de comunicaciones. Los usuarios demandan una forma más ágil de obtener la información marítima sin tener que realizar costosas inversiones. Al mismo tiempo esa información se pide que esté accesible en cualquier lugar sin tener en cuenta restricciones de distancia con las estaciones de control de tráfico.

La proliferación de Internet y dispositivos que se conectan a ella ha abierto nuevas posibilidades de comunicación. El usuario quiere recibir la información marítima en tiempo real en su portátil, PDA, ipad o incluso en el móvil.

El proyecto WCTM va a proporcionar la información marítima en una forma fácilmente accesible a través de la Web.

Al utilizar sistemas y recursos existentes el coste del proyecto no será excesivamente elevado, lo cual lo hace bastante atractivo. Otro factor que cabe destacar es el uso de estándares y software abiertos, no restringido a ninguna firma o marca concreta. Esto facilita el desarrollo del proyecto así como su mantenimiento en el tiempo al no estar atado a ningún formato o compañía de software determinada.

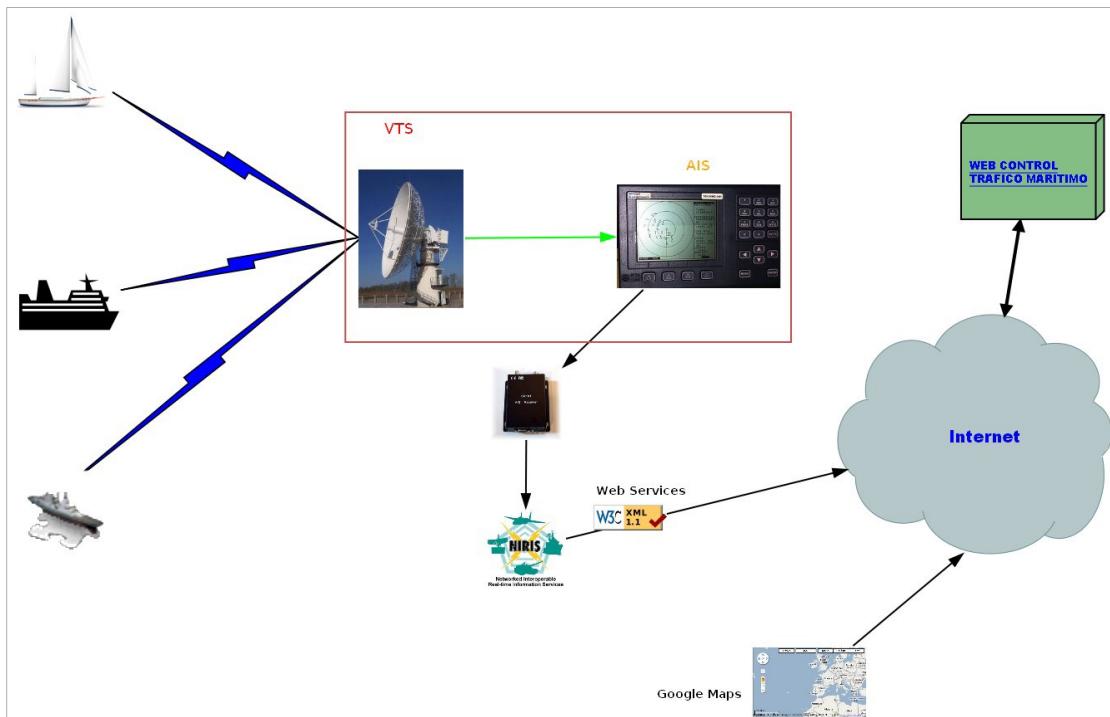
Por todo esto, los recursos técnicos exigidos se limitan principalmente a la parte de servidores en la que se albergará la aplicación Web. Para los clientes no existen requisitos especiales ya que cualquier cliente será capaz de recibir la situación marítima.

Dada la importancia de recibir la situación marítima actualizada y de forma ininterrumpida, el sistema garantiza la accesibilidad y operatividad 24/7.

2.2 Estudio de la situación actual

El siguiente diagrama resume el estado actual del Sistema de Control de Tráfico Marítimo.

También se muestra donde juega su papel nuestra aplicación WCTM.



Los barcos mandan su situación y otro tipo de datos como identificación, rumbo, velocidad, nombre, destino, etc. al [VTS](#) usando los equipos transmisores que instalan los barcos. El VTS procesa todos esos datos y los pone a disposición de todos los buques y sistemas conectados a la red [AIS](#). Uno de estos sistemas es el [NIRIS](#) que recoge esa información del AIS y la transforma a un formato XML para que cualquier sistema con acceso a Internet pueda recibir esa información en forma de "Web services".

La intención es que nuestra aplicación WCTM esté conectada a la red Internet y pueda aprovechar la información que vierte el NIRIS por una parte y usar los mapas de Google por otra.

Veamos ahora con más detalle cada componente.

2.2.1 Sistema AIS

El sistema [AIS](#) es un sistema de identificación automática ampliamente usado hoy en día para la identificación y seguimiento de buques. La "International Convention for the Safety of Life at Sea" de la "[IMO](#)", dicta que todos los barcos de pasajeros y aquellos de más de 300 toneladas están obligados a instalar el sistema AIS.

El AIS está concebido no solo para transmitir la posición de buques sino también puntos fijos como faros y boyas.

Cada buque instala un dispositivo transmisor (traspondedor) parte del sistema AIS, que transmite de forma periódica información sobre el buque. Esta información se envía en claro

y por [VHF](#) la cual tiene un alcance limitado, sobre 76 Km. (46 millas) dependiendo de las condiciones meteorológicas.

Esta información básicamente consiste en:

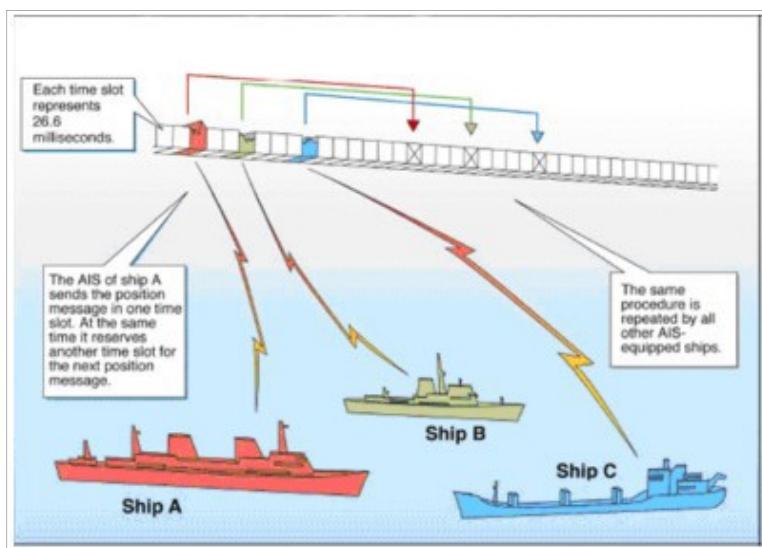
- **Información fija sobre el buque.** Nombre, pabellón y tipo de buque. Esta información se introduce manualmente cuando se instala el dispositivo AIS.
- **Posición (latitud, longitud), rumbo velocidad.** Esta información se obtiene de los sistemas GPS, giroscópicas y correderas que instalan los barcos.
- Así como una gran variedad de información adicional (calado, tipo de carga, destino, etc.)

En AIS se usan dos canales diferentes en la banda de VHF.

Canal A (161.975 Mhz) (87B) y canal B (162.025 Mhz) (88B). Como la red es compartida por muchas unidades transmisoras para que no haya colisiones, las señales se multiplexan usando un sistema derivado de la multiplexación por división de tiempo [STDMA](#).

Los canales se dividen en trozos, slots de tiempo. Y cada unidad puede usar uno a más de estos slots. A diferencia del clásico [TDMA](#), en la que es necesaria una unidad central que asigne los diferentes slots y sirva de sincronización de la red entera, el STDMA no necesita una unidad central y la sincronización en tiempo se hace usando [UTC](#).

Una franja de 60 segundos se divide en 2250 slots. Como vemos en la figura cada slot ocupa 26.6 milisegundos de tiempo. El slot es usado por cada barco para enviar sus datos.



Para hacer más eficiente el sistema los barcos que estén anclados o con velocidad baja, usarán pocos slots de tiempo, por lo contrario los que estén maniobrando rápidamente usaran más. Las actualizaciones están normalmente entre el rango de 3 minutos, para los primeros y de cada dos segundos para los más rápidos. Es cada estación AIS, instalada en las unidades, la que determina sus propias necesidades en slots de tiempo, según las maniobras que hagan y del trayecto que se prevean van a realizar. Según define la [IMO](#), la red AIS debe ser capaz de proporcionar como mínimo 2000 slots por minuto. El AIS normalmente proporciona 4500.

El formato AIS usa la codificación [NMEA](#) que son sentencias de texto de 64 bits de longitud, cada campo esta separado por comas.

Este es un ejemplo de mensaje AIS:

!AIVDM,1,1,,A,14eG;o@034o8sd<L9i:a;WF>062D,0*7D

En la siguiente tabla se explica el significado de cada campo

Campo	Significado
!AIVDM	El tipo de mensaje NMEA
1	Número de sentencias que componen el mensaje completo, algunos mensajes de información ampliada necesitan más de una.
1	Número de la sentencia dentro del mensaje, normalmente es 1 excepto cuando se trata de un mensaje con mucho información.
	Normalmente en blanco, excepto para mensajes multi sentencia en que este campo indica el Número de secuencia de esa sentencia en el mensaje
A	El canal AIS, puede ser A o B
14eG;o@034o8sd<L9i: a;WF>062D	Los datos AIS codificados
0*	Marca de final de datos
7D	NMEA Checksum

Los caracteres de texto van codificados en 6 bits cada uno a diferencia del ASCII normal que lleva 8. Para pasarlo a formato NMEA del AIS es necesario una conversión que consiste en restarle 48 (valor del carácter 0 en ASCII) y si este Número resultante es mayor de 40 se le resta 8 y se pasa a binario. Con eso nos aseguramos un campo de 6 bits.

2.2.2 Sistema de Control de Tráfico Marítimo

En cada zona marítima existe un sistema de Control de Tráfico Marítimo, [VTS](#), el cual recibe la información de los barcos de su área. Es un centro de control normalmente situado en la costa en el cual se recibe y procesa toda la información que puede ser importante para la gestión del tráfico marítimo.

Se podía comparar con la torre de control de encontramos en los aeropuertos. La información que recibe el VTS va desde las posiciones de buques, condiciones meteorológicas o cualquier tipo de alerta o peligro que pueda haber en la mar.



En cuanto a la información de tráfico, todos los datos recibidos en el VTS, son procesados adecuadamente por sistemas y operadores y estos se incorporan a la red AIS para que estén a disposición de todos los buques.

2.2.3 NIRIS

La OTAN tiene un sistema llamado [NIRIS](#), el cual es usado para la transmisión y diseminación del escenario de reconocimiento aéreo en las redes clasificadas (intranets) de la OTAN.

El NIRIS, entre otras opciones, se puede conectar al sistema AIS usando un conversor para recibir los mensajes de trazas marítimas vistos anteriormente.

Por otra parte este sistema esta dotado de otro servicio, [TIES](#) que lo que hace básicamente es proporcionar en forma de “Web services” y en formato [XML](#), libremente a toda la Internet estos tipos de información:

- Marítima proveniente del sistema AIS Holandés
- Aérea de EUROCONTROL
- Marítima de Dinamarca
- Marítima de Finlandia
- Marítima de Suecia

Este es un ejemplo de un bloque de información que envía el NIRIS.

```
<xltf:target xltf:atime="2010-11-03T14:34:21.521Z" xltf:mechanism="full" xltf:target-id="c98bbdf7-556c-491d-b6a1-128be3078dcf" xltf:type="track">
<xltf:identifier>
<xltf:track-id>c98bbdf7-556c-491d-b6a1-128be3078dcf</xltf:track-id>
<xltf:source-id>AIS</xltf:source-id>
<xltf:service-id>NC3A-NL AIS-Receiver</xltf:service-id>
</xltf:identifier>
<nvg:nvg xmlns:nvg="http://nc3a.nato.int/STANDARDS/TDL/XLTF/NVG/0.9d">
<point xmlns="http://nc3a.nato.int/STANDARDS/TDL/XLTF/NVG/0.9d" course="2.0"
href="http://niris.nc3a.nato.int:8080/tixo/monitor/TIXOTrack?rawTrackId=c98bbdf7-556c-491d-b6a1-128be3078dcf&amp;id=NC3A-NL AIS-Receiver" id="c98bbdf7-556c-491d-b6a1-128be3078dcf"
label="AIS:218143000 - Unknown" point-id="c98bbdf7-556c-491d-b6a1-128be3078dcf" speed="0.0"
symbol="app6a:SPSP-----C" x="4.421139026948156" y="51.91142581837485"/>
</nvg:nvg>
</xltf:target>
```

Cada traza equivale a un bloque “target” y los datos de la misma están en la parte “nvg”.

2.2.4 Mapas de Google

Como vemos en el diagrama los mapas de Google están disponibles en la Internet para que cualquier tipo de cliente los pueda utilizar. Ambas partes de nuestra aplicación se conectará a este servicio para mostrar el mapa correspondiente al usuario.

2.2.5 Nuestra aplicación (WCTM)

Finalmente se muestra donde encajaría nuestra aplicación en la situación actual del sistema de control marítimo.

Estaría conectada a Internet para poder usar los servicios de NIRIS y de Google maps y para que cualquier usuario pueda acceder a la misma.

2.3 Definición de los requisitos del sistema

En primer lugar se realizará una descripción de los requisitos teniendo en cuenta las necesidades del cliente y las restricciones temporales y de presupuesto.

Esta definición se realiza teniendo en cuenta diferentes requisitos por prioridad del 1 al 5 (5 la más alta).

- **Técnicos**

- (5) Se debe emplear hardware de alto rendimiento, que proporcione tolerancia a fallos y que permita realizar copias de seguridad.
- (5) Página Web accesible desde cualquier ordenador conectando a Internet.
- (5) Debe estar protegido contra intrusiones o ataques que puedan modificar el contenido de la página Web, así como bloquearla con algún tipo de ataque de denegación de servicios.
- (4) Se debe registrar la información marítima en una base de datos relacional.
- (4) Los servidores estarán en un cuarto adecuado y con acceso restringido.
- (3) Debe de emplear estándares abiertos tanto en el desarrollo como en la documentación del proyecto (Web services, XML, HTML, CSS, pdf, etc.)
- (3) Accesibilidad según los estándares del W3 Consortium

- **Operativos**

- (5) La aplicación debe proporcionar una información fiable y en tiempo real sobre el tráfico marítimo.
- (5) Debe estar disponible 24/7, es decir 24 horas al día todos los días del año.
- (5) El sistema asegurará la integridad de la información. El usuario no podrá modificar ninguna traza tanto en pantalla como de los datos grabados.
- (4) La aplicación debe ofrecer un tiempo de respuesta adecuado incluso con multitud de conexiones simultáneas.
- (4) La página Web del sistema se tiene que poder acceder desde cualquier tipo y marca de navegador.
- (3) Instalación, administración y mantenimiento debe ser fácil de realizar.
- (3) Se debe de proporcionar una página Web que permita analizar las trazas grabadas.

- **Legales**

- (5) La información presentada por el sistema no puede poner en riesgo la seguridad de buques (policiales, guerra, etc.) que necesiten ocultar sus detalles. Solo se tratará y presentará información no restringida.
- (4) Las licencias no pueden ser restrictivas, preferiblemente se deben usar licencias libres en todos los componentes.

- **Económicos.**

- (4) Los gastos originados por compra de licencias debe reducirse al máximo, usando software libre mayoritariamente.
- (3) Los gastos de desarrollo se deben de minimizar, intentando usar componentes ya existentes.

2.4 Estudio de las alternativas de solución

Se consideran varias alternativas teniendo en cuenta los objetivos anteriores. Se explorará tanto soluciones propietarias como libres así como una opción muy extendida actualmente, el uso de la virtualización.

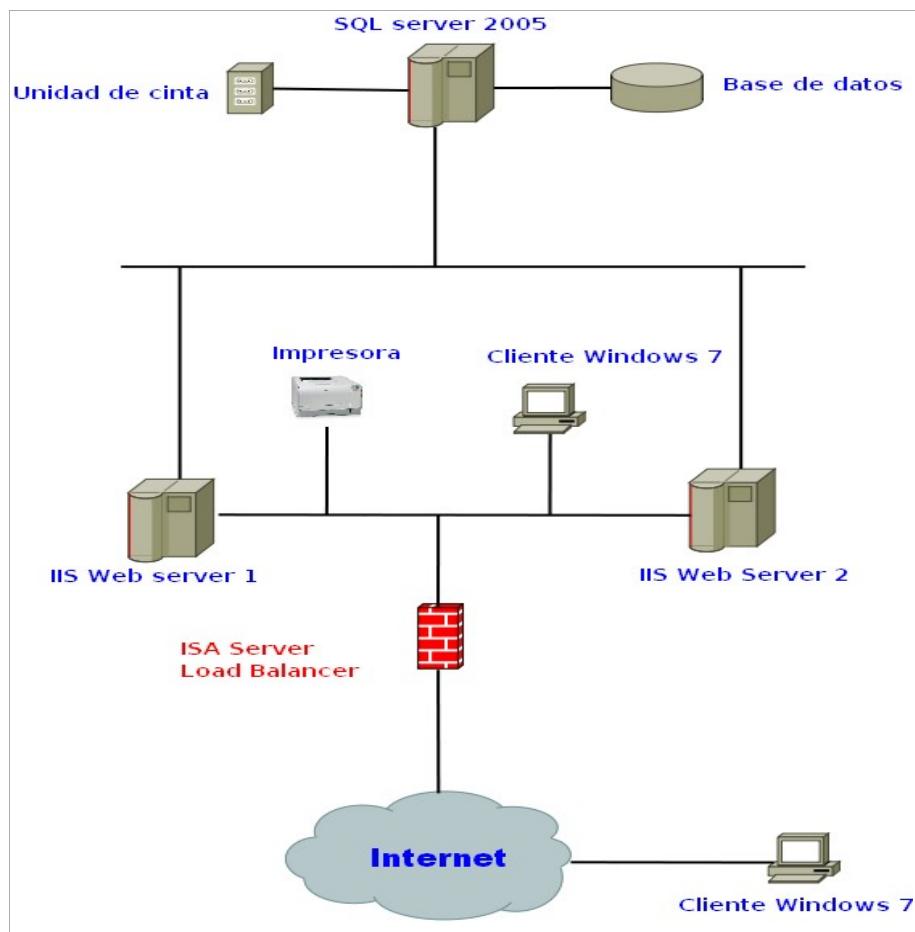
Por las características del proyecto y al no existir en el mercado una aplicación que cumpla con los requisitos, para todas las opciones se opta por una aplicación desarrollada a medida.

Se describe cada una de ellas:

2.4.1 Propietaria.

Plataforma (Sistema operativo, servidor Web, firewall) propietaria con aplicación Web también propietaria, desarrollada en la tecnología dotNet.

El siguiente diagrama explica esta solución.



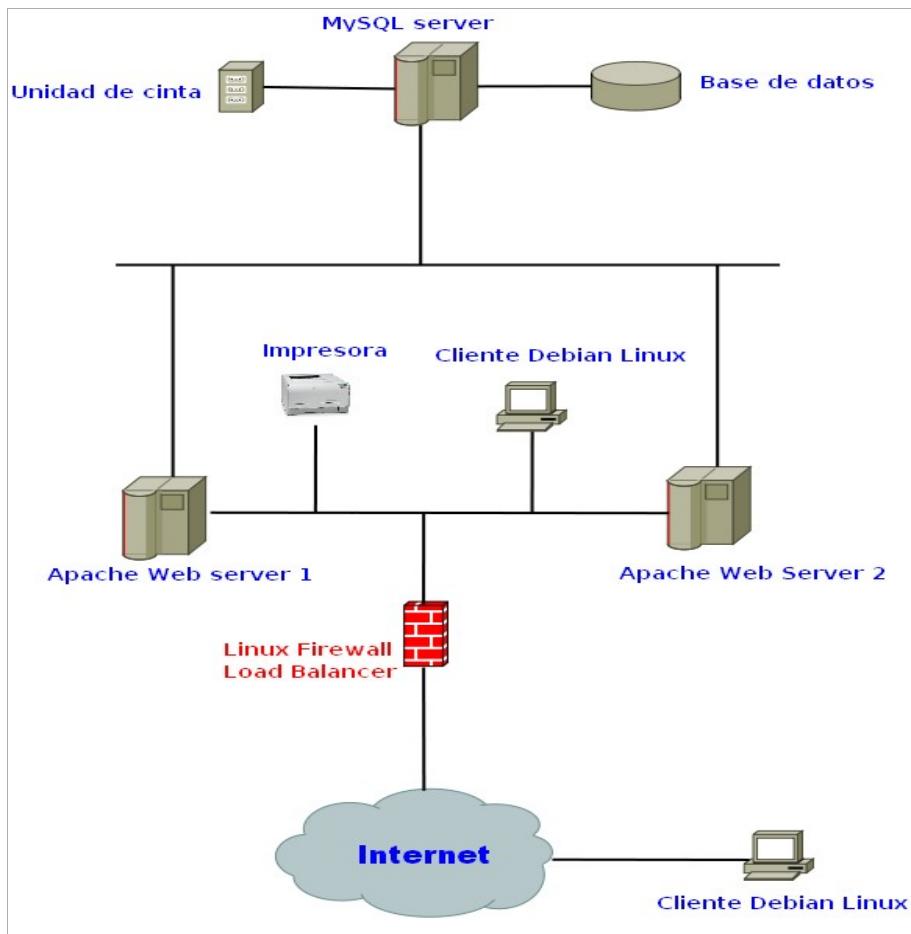
En esta tabla se describen los detalles del software necesario.

Aplicación	Notas
Aplicación Web	Desarrollada a medida. ASP (Active Server Pages, dotNet, ASP.net)
Microsoft Windows 2003 server EE	Sistema Operativo (SO) usado en servidores y firewall
IIS (Internet Information Services)	Servidor Web
Windows 7	SO de los clientes necesarios para gestión del proyecto, desarrollo, y pruebas.
ISA (Internet Security & Acceleration) server 2006	Servidor Proxy/Firewall implantando el “Load Balancer”. Balance de la carga de trabajo, entre dos servidores en este caso.
Microsoft Office 2007	Suite ofimática, usada para generar los diferentes documentos durante el la implantación del proyecto.
Microsoft Visual estudio	Entorno de desarrollo para dotNet
Microsoft SQL 2005 server	Sistema de Gestión de Base de datos
HP Quality Center	Control de versiones, seguimiento de errores

2.4.2 Libre.

Plataforma (Sistema operativo, servidor Web, firewall) libre con aplicación Web también libre.

Vemos primero un diagrama explicativo de esta solución libre:



A continuación se detalla los componentes software en la siguiente tabla:

Aplicación	Notas
Aplicación Web	Desarrollada a medida. PHP, JavaScript
Debian GNU Linux	SO Servidores, firewall y clientes
Apache 2	Servidor Web
Ultra Monkey	Para crear el "Load Balancer" entre los dos servidores Web
netfilter/iptables	Firewall
OpenOffice.org	Suite ofimática, para generar los documentos del proyecto.
MySQL	Sistema de Gestión de Base de datos
Subversion	Control de versiones
Trac	Control y seguimiento de errores

2.4.3 Virtualización

Esta consiste en emplear “maquinas virtuales” para cada uno de los componentes del sistema en lugar de usar servidores físicos. Esto permite que en un servidor físico se puedan crear varias maquinas virtuales que actúan como servidores o clientes separados.

Cada una de las soluciones anteriores (propietaria o libre) se puede virtualizar.

Hay que tener en cuenta que el software necesario, además del específico para la virtualización, es el mismo que para las soluciones con servidores físicos. El ahorro de esta solución no es en licencias sino en la compra del hardware.

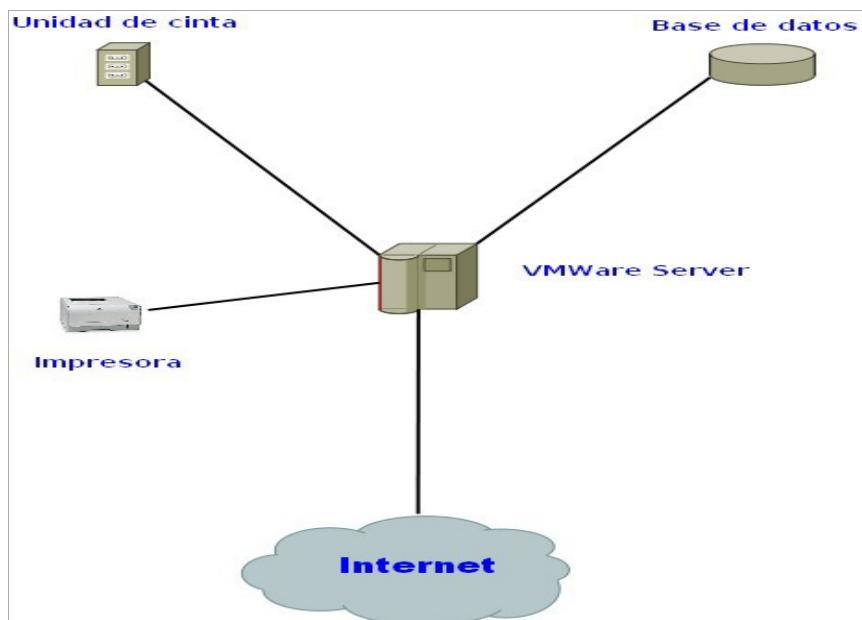
Se evalúan dos soluciones dentro de la virtualización:

1. Virtualización propietaria.

Combinación de software de virtualización propietario, [VMware](#) y usando a su vez la implantación con software propietario anterior.

Se crean maquinas virtuales para cada uno de los servidores y clientes que funcionan en Windows.

Diagrama de este tipo de virtualización.



A continuación se describe el software necesario, que será el software para realizar la virtualización + software de la implantación del proyecto.

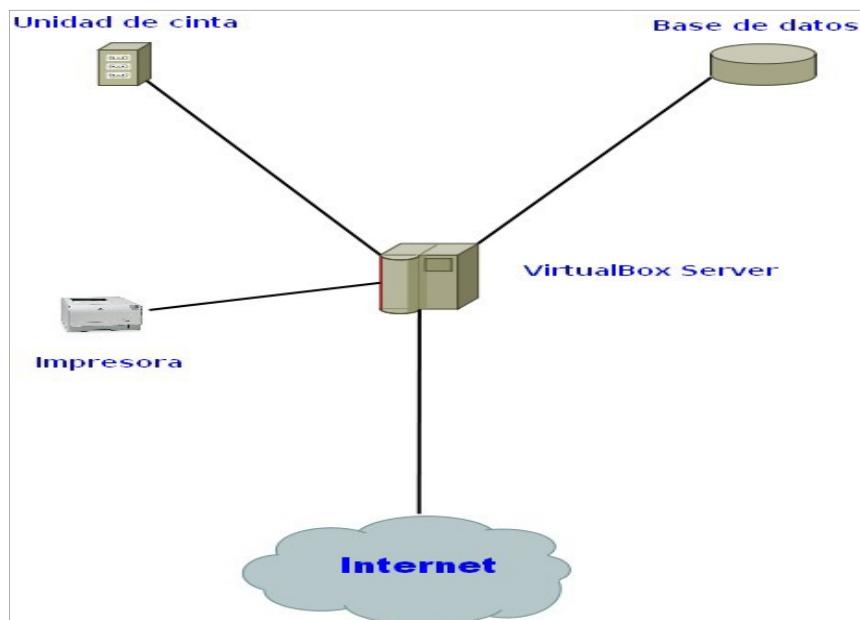
Aplicación	Notas
Aplicación Web	Desarrollada a medida. ASP (Active Server Pages), dotNet, ASP.net
VMware vSphere Enterprise	Software de virtualización
Microsoft Windows 2003 server EE	SO Servidores y firewall
IIS	Servidor Web
Windows 7	SO de los clientes necesarios para gestión del proyecto, desarrollo, y pruebas.
ISA (Internet Security & Acceleration) server 2006	Servidor Proxy/Firewall
Microsoft Office 2007	Suite ofimática, desarrollo del proyecto y documentación
Microsoft Visual estudio	Entorno de desarrollo para dotNet
Microsoft SQL 2005 server	Base de datos
HP Quality Center	Control de versiones, seguimiento de errores

2. Virtualización libre.

En este caso se usan soluciones libres tanto para el software de virtualización como para la implantación de los componentes del proyecto en sí.

En este caso se usa el software “[VirtualBox](#)” que es libre.

Aquí tenemos el diagrama de esta solución:



Como en el caso anterior el software necesario es el software propio de la virtualización más el necesario por el proyecto.

Aplicación	Notas
Aplicación Web	Desarrollada a medida. PHP, JavaScript
VirtualBox	Software de virtualización
Debian GNU/Linux	SO Servidores, firewall y clientes
Apache 2	Servidor Web
Ultra Monkey	Para crear el “Load Balancer”
netfilter/iptables	Firewall
OpenOffice.org	Suite ofimática, desarrollo del proyecto y documentación
MySQL	Base de datos
Subversion	Control de versiones
Trac	Control y seguimiento de errores

2.5 Valoración de las alternativas

Para cada una de las soluciones propuestas se analizaran tanto el coste como los riesgos y la forma de paliarlos.

Veamos en primer lugar los costes de software de cada una de las soluciones analizadas.

En el coste de la aplicación Web está incluido el coste de instalación y formación. Se estima que éste es igual para cada una de las opciones propietaria y libre. La dificultad de instalación y configuración de servidores se considera similar en las alternativas libres y propietarias, así como el aprendizaje.

Sin embargo para las opciones de virtualización se necesita más recursos, a los anteriores hay que sumarles la instalación configuración y aprendizaje del software de virtualización. Aunque aquí tenemos la ventaja de clonar servidores y clientes con las mismas funcionalidades.

En la tabla siguiente se compararan los costes de cada una de las alternativas estudiadas.

Tipo	Conceptos y Costes Unitarios	Coste total
Propietaria	Aplicación Web	33000 €
	Windows Server 2003 Enterprise para 25 CALs (OEM)=1,000 €	4*1000=4000 €
	Windows 7 profesional=309,00 €	2*309=618 €
	ISA (Internet Security & Acceleration) Server 2006 = 5300 € por procesador	5300 €
	Microsoft Office 2010=699,00 €	2*699= 1398 €
	Microsoft Visual estudio 2010=1.599,00 €	1599 €
	Microsoft SQL 2005 Server Enterprise = 19000 €	19000 €
	HP Quality Center	4000 €
	Costes de hardware y mantenimiento	44000 €
	Total Propietaria	112.915 €
Libre	Aplicación Web	33000 €
	Sistemas operativos y aplicaciones	0 €
	Costes de hardware y mantenimiento	44000 €
	Total Libre	77.000 €
Vir. Propietaria	Aplicación Web	35000 €
	VMware vSphere Enterprise=2330	2330 €
	Licencias de software	35915 €
	Costes de hardware y mantenimiento	20000 €
	Total Vir. Propietaria	94.245 €
Vir. Libre	Aplicación Web	35000 €
	VirtualBox	0
	Licencias de software	0
	Costes de hardware y mantenimiento	20000 €
	Total Vir. Libre	55.000 €

Se ha realizado un estudio exhaustivo de los posibles riesgos que pueden aparecer en cada una de las alternativas propuestas, así como las acciones que se pueden realizar para evitarlos o mitigarlo.

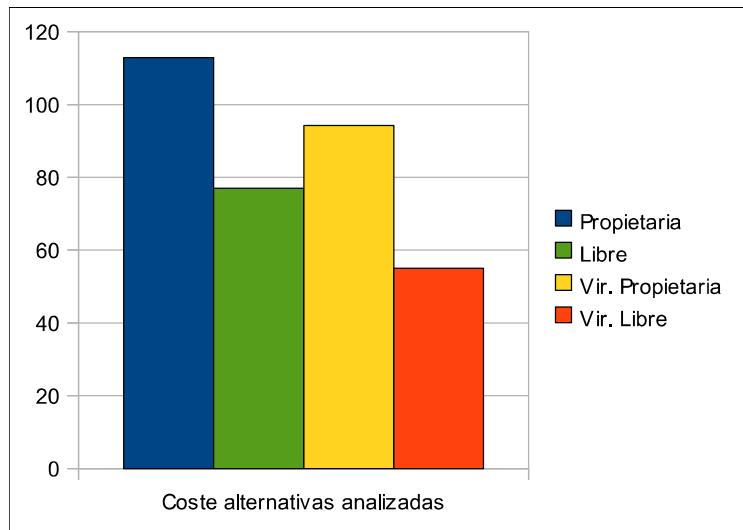
Esto se explica en la tabla siguiente para las cuatro soluciones.

Tipo	Riesgos	Acciones de prevención y remedio
Propietaria	Cambio frecuente de versiones y falta de soporte en versiones anteriores	Firma de contratos de mantenimiento, que cubra actualizaciones.
	Fallos de seguridad en el sistema operativo y el servidor Web IIS	Mantener actualizado el software con las últimas actualizaciones del fabricante.
	Falta de rendimiento del servidor Web para servir peticiones	Añadir más recursos hardware (servidores más potentes y con más memoria).
	Formatos no estándar en la documentación	Adquisición por separado software para la conversión a formatos estándares (pdf).
Libre	Finalización del proyecto encargado del mantenimiento de un determinado producto de Software Libre	Al disponerse del código fuente se pueden contratar servicios de mantenimiento.
	Mayor complejidad de administración	Proporcionar cursos de adiestramiento y documentación clara.
	Incompatibilidad entre licencias de las soluciones	Controlar en cada momento la licencia de cada aplicación que forma parte del producto final. Si hay alguna incompatibilidad seleccionar otra alternativa.
	Reticencia a adoptar soluciones libres por parte de empresas y usuarios.	Proporcionar una visión clara de las soluciones. Proporcionar suficiente formación y dar una solución preconfigurada, que requiera el mínimo mantenimiento.
Vir. Propietaria	Fiabilidad, Operatividad interrumpida	Poner más de un servidor en "cluster" por si falla uno que el servicio quede garantizado.
	Eficiencia para servir múltiples peticiones	Añadir más recursos, servidores más rápidos con más memoria.
	Actualizaciones y mantenimiento del software de virtualización	Contratar el servicio de mantenimiento anual que ofrece la empresa VMware.
Vir. Libre	Fiabilidad, Operatividad interrumpida	Poner más de un servidor en "cluster" por si falla uno que el servicio quede garantizado.
	Eficiencia para servir múltiples peticiones	Añadir más recursos, servidores más rápidos con más memoria.
	Falta de soporte para el software de virtualización	Contratar a una empresa especializada.

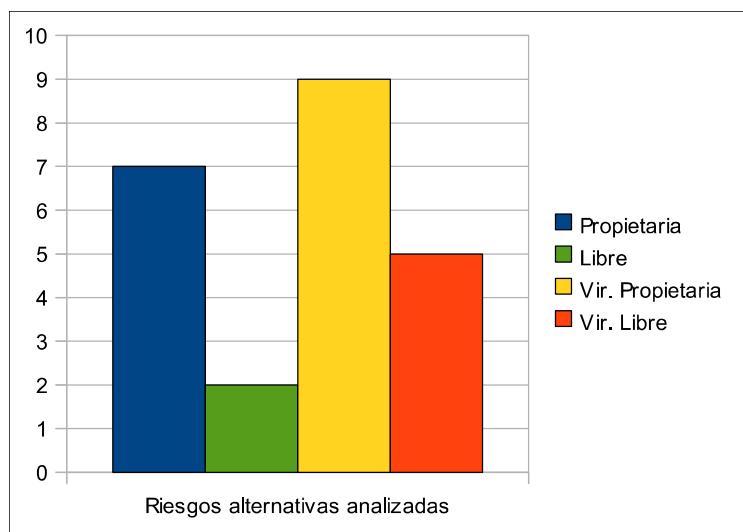
2.6 Selección de la solución

Para la selección de la solución que vamos a adoptar se tienen en cuenta los requisitos y las restricciones económicas. Los siguientes gráficos podemos apreciar una comparativa de las cuatro opciones estudiadas.

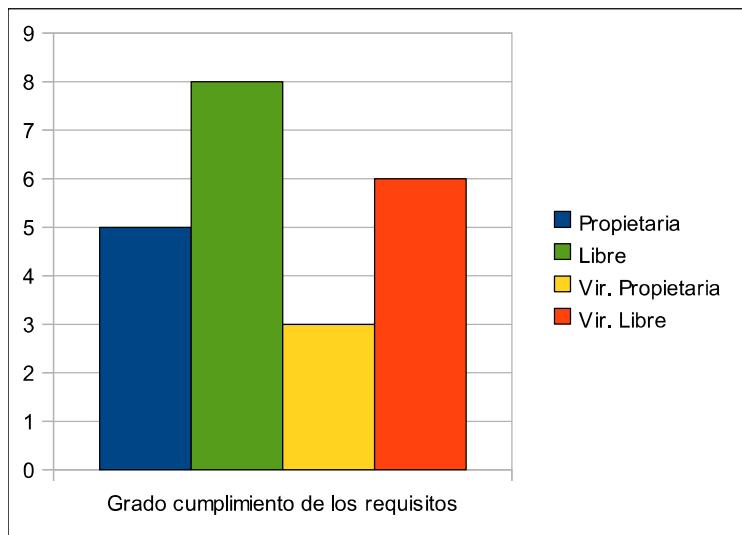
En el primero podemos ver la comparativa de costes en miles de euros de cada una de las soluciones. Vemos que la solución más económica es la virtualmente libre y la más costosa la opción propietaria.



En el siguiente diagrama tenemos una comparativa de los riesgos, cada uno de ellos se le ha dado una ponderación entre 0 (ningún riesgo) y 10 (máximo riesgo). Podemos apreciar que la opción que supone menos riesgos es la Libre y la que más es la Virtualización Propietaria.



En el último diagrama se compara en qué medida cada una de las soluciones cumple con los requisitos impuestos, teniendo en cuenta rendimiento, seguridad y soporte. Aquí al grado cumplimiento de los requisitos se le ha dado una valoración de 0, no lo cumple en absoluto hasta 10, lo cumple completamente. Podemos comprobar que la opción que cumple mejor con los requisitos es la Libre y la que menos la Virtualización Propietaria.



Analizado estos tres factores, finalmente se ha optado por una solución de **servidores físicos sin virtualización y una plataforma libre con aplicaciones también libres**.

El motivo de usar la solución libre es por una parte por el menor presupuesto, pero también porque los posibles riesgos son menores y también más fácilmente mitigables. Precisamente por el hecho tratarse de software libre, no se tiene las ataduras del software propietario lo que da más flexibilidad al buscar soluciones y empresas que se puedan encargar del mantenimiento.

Otro aspecto que se ha tenido en cuenta es la eficiencia y seguridad que ofrece tanto el sistema operativo Debian Linux como la combinación de Apache, MySQL y usando PHP como lenguaje de programación, en una palabra el conocido sistema “LAMP”.

En cuanto a la virtualización, supone un ahorro considerable en material hardware, pero en detrimento del rendimiento y fiabilidad de nuestro sistema, requisitos muy importantes en un proyecto de Control de Tráfico Marítimo.

3 Análisis del sistema

Una vez visto cual de las soluciones posibles es más viable, en esta fase realizamos un análisis detallado de la solución que hemos elegido.

3.1 Definición del sistema

La Web Control Trafico Marítimo es la encargada de presentar las trazas del sistema de la OTAN NIRIS en un mapa de Google. Para ello se conectara al servicio Web del NIRIS que a su vez recibe la información marítima del sistema AIS usando el protocolo específico de este último sistema.

Se desarrollará principalmente en PHP y JavaScript usando HTML para la construcción de las diferentes páginas Web. Para intercambio de datos se empleará XML y SQL como lenguaje de consulta a la base de datos MySQL.

El sistema NIRIS sirve la información usando diferentes modos de acceso entre los que destacamos los siguientes.

- Descargando ficheros [KML](#) que son actualizados por el NIRIS de forma periódica.
- Mensajes [SOAP](#). Protocolo para intercambiar la información entre el servidor y el que consume el servicio. Está basado en XML y la información va empaquetada en una envoltura (envelope) común para cada mensaje.
- Llamadas [REST](#), que hace lo mismo que el SOAP, pero no usa un mensaje predefinido para empaquetar la información, tan solo con hacer una llamada HTTP por ejemplo del tipo GET o POST recibimos la información. Esta es el método favorito en los sistemas que comienzan a ofrecer sus servicios usando la Web y es el usado en este proyecto.

La información de trazas esta disponible en un documento XML que genera el NIRIS con cada llamada al servicio. El sistema NIRIS mantiene actualizada la información sobre trazas manteniendo una conexión periódica con el sistema AIS. Nuestra aplicación decodifica la información contenida en el documento XML y presenta adecuadamente la información (posición, nombre, rumbo, velocidad, etc.) sobre un mapa de Google. Para ello se hace uso del API de Google maps. Se emplea el API versión 3 ya que por una parte, es la recomendada por Google, es más rápida y por otra no necesita una “key”, código de acceso, como su predecesora la API v2.

Una vez presentadas las trazas en el sistema estas se actualizaran automáticamente según el intervalo definido por el usuario. La aplicación Web se conectara con el servicio del NIRIS cada vez que haya que actualizar las posiciones.

La conexión se realiza usando la tecnología [AJAX](#) y usando un “proxy” es decir un módulo que realiza las peticiones en nombre de nuestra aplicación. Esto es necesario debido a las restricciones que han implementado algunos navegadores al no permitir realizar llamadas a un dominio diferente al de la aplicación. Es decir si un navegador (cliente) se conecta a nuestra página Web “www.wctm.com”, no puede realizar llamadas al servicio NIRIS en el dominio “niris.nc3a.nato.int”. Para la implementación de este “proxy” se va a reusar la clase existente, escrita en PHP, “class_http.php”.

Por otra parte para el Análisis de Históricos, se va a usar una base de datos relacional, usando MySQL como motor de datos. Un módulo será el encargado de registrar las trazas y posiciones que va teniendo cada una de ellas a lo largo del tiempo. Este módulo, en PHP, se ejecutará de una forma periódica según la frecuencia necesaria de actualización de trazas. La ejecución de este módulo se hará usando las herramientas del sistema operativo añadiendo una llamada al mismo y el intervalo de ejecución en la tabla del “cron” de Linux (/etc/crontab). Cada vez que se ejecute este programa, se conectara al servicio Web de NIRIS usando una llamada REST y registrara los datos de todas las trazas en la base de datos de la aplicación.

Una vez registradas las trazas, otra parte del Análisis de Históricos se encargara de servir una página Web con la información contenida en la base de datos. El analista puede seleccionar una de las trazas y presentar información de sus posiciones sucesivas tanto en una tabla como en un pequeño mapa de Google.

A continuación se definen los diferentes perfiles de usuarios que pueden interactuar con la aplicación WCTM:

- **Usuario.** Este accede a la parte de Presentación de Tráfico de la aplicación. Puede acceder desde cualquier navegador sin necesidad de autenticarse.
- **Analista.** Este rol es el encargado de entrar en la parte de Análisis de Históricos. Como en el caso anterior accede desde cualquier navegador, pero necesita autenticación.

- **Administrador.** Este perfil es el del encargado del mantenimiento del sistema. Esto incluye la ejecución de scripts, realizar actualizaciones, analizar los ficheros "logs" y hacer las copias de seguridad.

El administrador puede usar las interfaces propias del sistema operativo y de las aplicaciones como por ejemplo MySQL [Workbench](#) o "[Webmin](#)" para Apache.

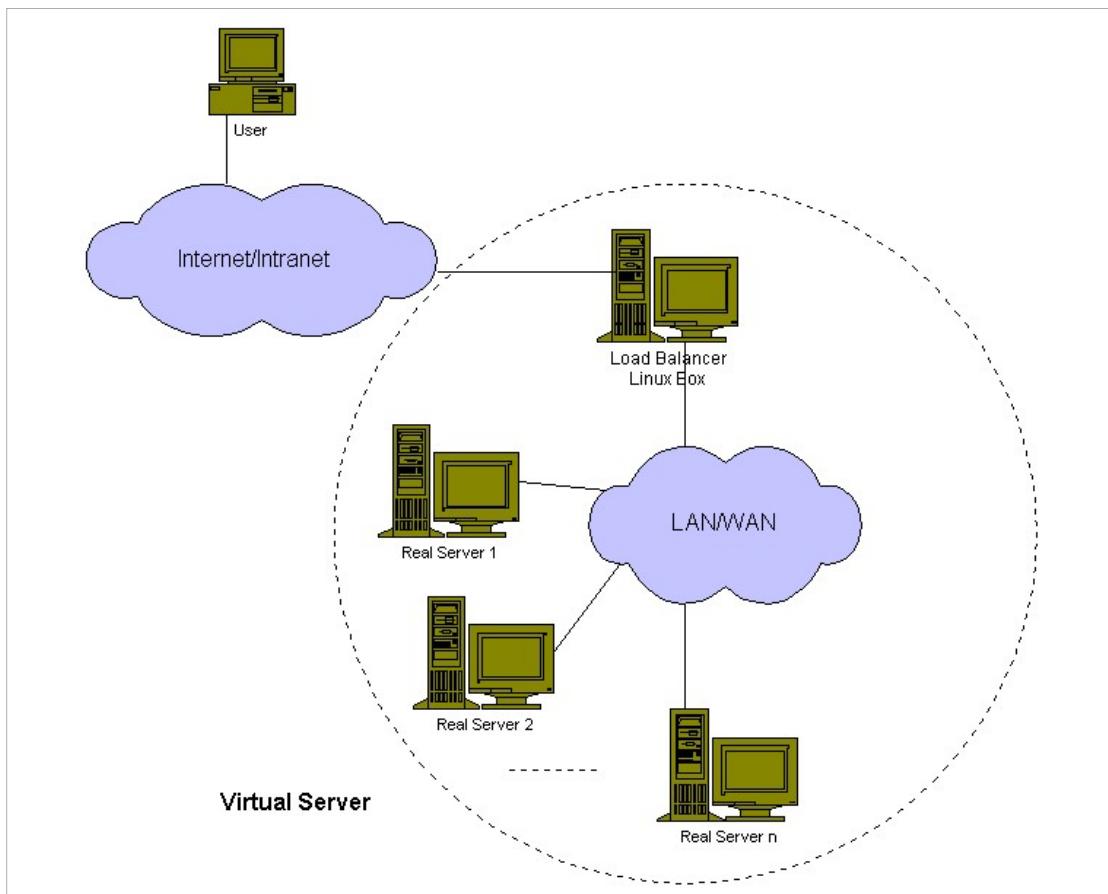
Para garantizar la disponibilidad y el tiempo de respuesta del sistema se va a usar balanceador de la carga "load balancer". Un [load balancer](#) es un sistema redundante en el que dos o más servidores distribuyen la carga de trabajo entre ellos. Aplicado a Internet es un servicio Web al que accede los usuarios como si se tratase de un único servidor pero que realmente está compuesto de varios servidores que reparten las peticiones a dicho servicio.

Para gestionar el reparto de esa carga es necesarios herramientas especiales.

En nuestra aplicación se usará una herramienta libre que funciona en GNU Linux, esta es [Ultra Monkey](#). Se trata de un proyecto encargado de proporcionar soluciones libres que permitir crear sistemas de alta disponibilidad y con balanceo de carga. Esto lo consigue gestionando varios servidores en "cluster". Para el usuario final este cluster es como se tratase de un servidor único.

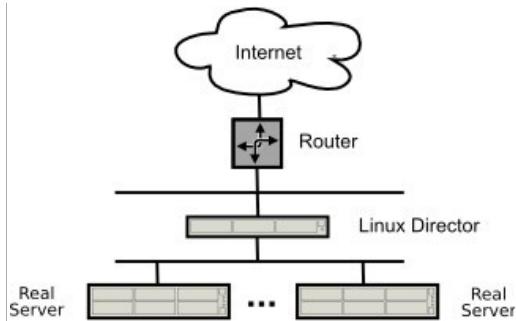
Estas herramientas son necesarias ya que por una parte, como hemos dicho para el usuario final solo debe existir un nombre de servidor o dirección IP y por la otra se pretende distribuir cada una de las peticiones recibidas. Ultra Monkey usa el concepto de Linux Virtual server ([LVS](#)). Un virtual server es un conjunto de servidores con las herramientas y la configuración necesaria de forma que el conjunto se comporta como si de un servidor único se tratase.

El siguiente diagrama de explica el concepto de virtual server.



Son diversas las opciones de configuración de Ultra Monkey según los recursos hardware y necesidades de alta disponibilidad y balanceo de carga.

En este proyecto se usa una solución intermedia que proporciona una solución de alta disponibilidad con balanceo de carga usando tres servidores de alto rendimiento y alta tolerancia a fallos.



Aquí vemos que se compone de un servidor para la función del “Linux Director” y dos servidores Web conectados al primero.

Linux Director, actúa de pasarela y de sistema de filtrado de paquetes. Cuando este servidor recibe una petición de conexión a la aplicación WCTM por parte de un cliente, primero hace el filtrado de paquetes según las reglas implementadas en el firewall. Si la conexión esta permitida, redirige la petición de conexión a uno de los dos servidores, según los criterios del algoritmo de balanceo de carga. Una vez conectado el cliente con uno de los dos servidores todos los paquetes de intercambio pertenecientes a esa conexión se realizaran con el mismo servidor. Esto asegura la integridad de la información (de los paquetes) que circulan entre cliente y servidor durante toda la sesión. En una situación real cada uno de los dos servidores Web tendrán establecidas conexiones con clientes diferentes pero eso si, sirviendo la misma información a todos ellos.

Para saber si los servidores Web funcionan correctamente es necesario instalar una herramienta ([Idirectord](#)) que se encargue de monitorizar cada uno de los dos servidores Web. Si un servidor Web deja de funcionar esta herramienta lo detecta y actualiza la tabla de servidores disponibles. Esta tabla es usada por “Ultra Monkey” para distribuir la carga de trabajo. Por lo tanto si un servidor Web cae, la aplicación no se verá afectada, el load balancer no escogerá ese servidor y los usuarios se conectarán todos al otro sin que se perciba la diferencia. Solo el rendimiento puede ser menor al contar con un solo servidor pero no la disponibilidad del servicio.

3.2 Establecimiento de requisitos

A continuación se realiza un refinamiento de los requisitos definidos previamente durante el estudio de viabilidad del sistema. Para ello se divide la aplicación en dos subsistemas. Presentación de trazas y Análisis de históricos.

3.2.1 Presentación de trazas.

- Debe presentar las posiciones de las trazas sobre el mapa de una manera clara y fácil de identificar.
- Se debe presentar información completa de cada traza. Al seleccionar una de ellas debe aparecer una ventana con los detalles más importantes y un enlace al sistema NIRIS que presentará una página con todos los detalles registrados sobre esa traza.

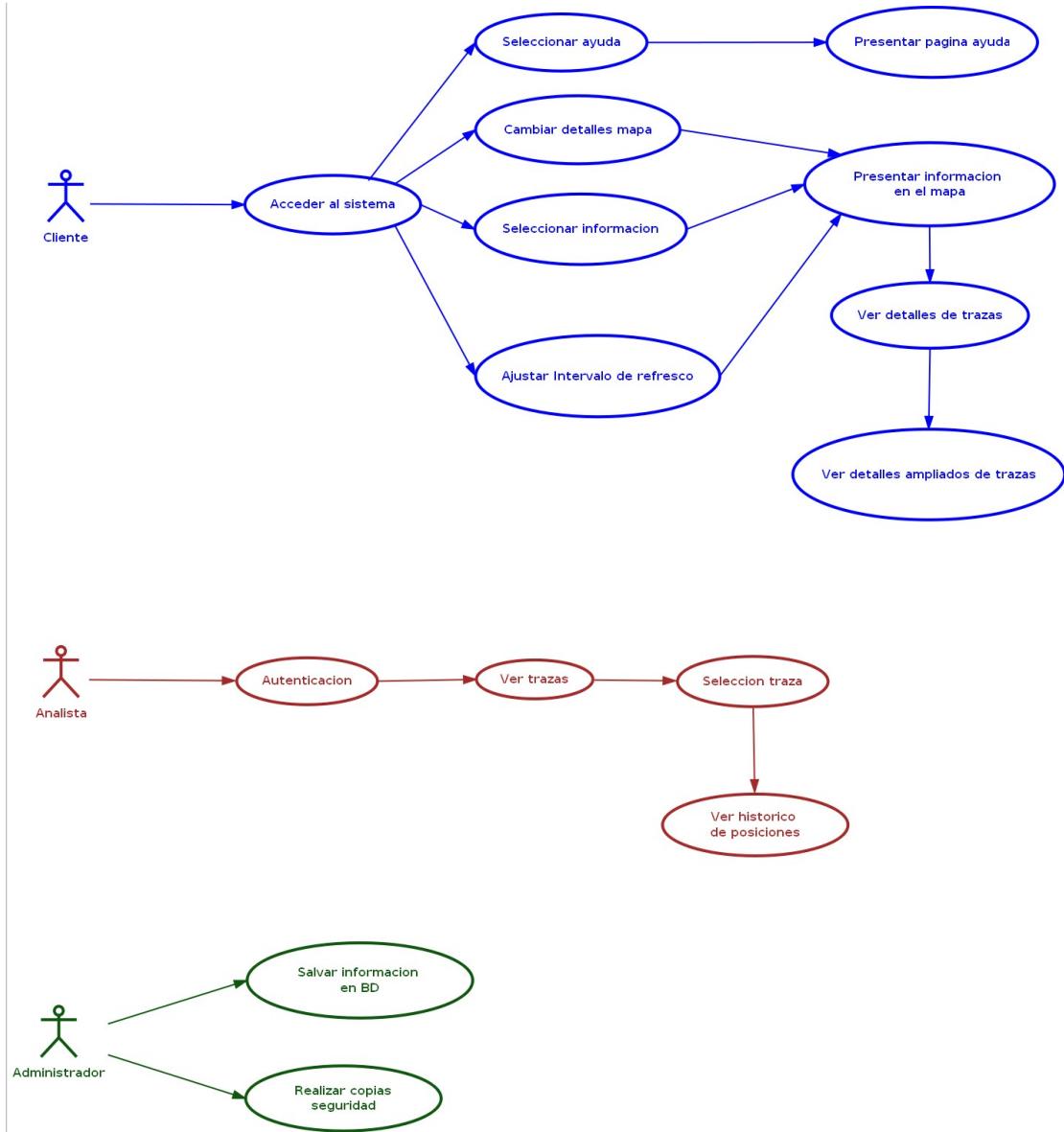
- Las posiciones de todas las trazas se actualizarán con el intervalo seleccionado por el usuario.
- El usuario puede elegir que tipos de trazas quiere presentar y actualizar de forma periódica.
- En cualquier momento el usuario puede parar las actualizaciones y volver al estado inicial de la aplicación Web.
- La aplicación se comunicará con otros sistemas usando servicios Web.
- La página Web debe ser fácil de usar, intuitiva y atractiva a los diferentes usuarios.
- La página Web debe estar organizada adecuadamente para permitir cambiar y añadir futuras funcionalidades o contenidos.
- La aplicación de Presentación de trazas será accesible por todos los usuarios de forma libre sin necesidad de autenticarse.
- El desarrollo se hará de acuerdo con las especificaciones dictadas por el [W3C](#), [HTML](#) 4.01, [XML](#) 1.0, [CSS](#) 2.0.

3.2.2 Análisis de Históricos.

- Debe registrar todas las trazas junto con sus posiciones sucesivas en una base de datos relacional. El sistema las registrara con el intervalo especificado por el administrador del sistema.
- Debe permitir el análisis del contenido de la base de datos. El analista tendrá la opción de seleccionar individualmente trazas para obtener información de las mismas en una tabla. Esta información será nombre de la traza, rumbo, velocidad, latitud y longitud para cada momento dado. Todas las posiciones pertenecientes a una traza se marcaran en un mapa de Google para facilitar la identificación de cada una de las posiciones. Al seleccionar cada una de estas posiciones debe aparecer una ventana con los detalles más importantes registrados sobre esa traza.
- La aplicación empleara SQL (SQL-92) tanto como lenguaje de consulta de la base de datos como de definición de la misma. Todas las operaciones sobre la base de datos se hará usando SQL.
- La base de datos debe de estar diseñada de forma que permita de una manera fácil modificar o añadir atributos.
- La aplicación recibirá la información de trazas usando servicios Web.
- La página Web debe ser fácil de usar, intuitiva y atractiva a los diferentes usuarios. Debe contar con un mapa que ayude al análisis.
- La aplicación de Históricos será solo accesible para usuarios autenticados, que tengan permiso para entrar en la aplicación. Todos los demás usuarios tendrán denegado el acceso.
- El desarrollo se hará de acuerdo con las especificaciones dictadas por el [W3C](#), [HTML](#) 4.01, [XML](#) 1.0, [CSS](#) 2.0.

- Cada uno de los accesos al sistema y a la base de datos tanto lícitos como ilícitos deben quedar registrados en el sistema, en un fichero log por ejemplo, para que los administradores puedan auditar que usuarios y desde donde acceden al sistema.

Para completar el establecimiento en detalle de los requisitos a continuación se describen en el siguiente diagrama los diferentes casos de uso.



Hay que tener en cuenta que los actores son “roles”, un cliente puede actuar como analista y viceversa.

A continuación se describen con más detalle cada uno de los casos de uso.

1. **Acceder al sistema.** Cuando el usuario accede a la Web del sistema recibirá una página Web con diferentes menús, opciones y con un mapa de Google.

2. **Seleccionar ayuda.** El usuario al seleccionar la opción de ayuda se abrirá una nueva página con información sobre del funcionamiento de la aplicación y de las diferentes opciones que tiene el usuario.
3. **Cambiar detalles mapa.** El usuario puede cambiar el tipo de mapa, cambiar la escala o moverse a cualquier parte del mundo.
4. **Seccionar información.** El usuario puede escoger uno o varias fuentes de información sobre trazas para presentar en el mapa.
5. **Ajustar intervalo de refresco.** El usuario selecciona el intervalo con el que desea refrescar la posición de las trazas en el mapa.
6. **Presentar información en el mapa.** Una vez seleccionado lo que el usuario quiere presentar y con qué intervalo, la información estará disponible en el mapa de Google.
7. **Ver detalles de trazas.** El usuario puede consultar información básica (nombre, posición, rumbo, velocidad) seleccionando un barco representado en el mapa.
8. **Ver detalles ampliados de trazas.** El usuario puede consultar toda la información referente a ese buque seleccionando un enlace que le redirige al sistema NIRIS para presentar una página con toda la información sobre esa traza.
9. **Autenticación.** El analista recibirá una pantalla para que pueda autenticarse en el sistema.
10. **Ver Trazas.** El analista, una vez autenticado, puede ver una lista de todas las trazas registradas en el sistema
11. **Selección de traza.** El analista al seleccionar una de las trazas obtendrá las posiciones de la misma para su análisis, tanto en una tabla como en un mapa.
12. **Salvar información en BD.** El administrador puede salvar y especificar el intervalo con el cual se grabará la información de trazas en la base de datos.
13. **Realizar copias de seguridad.** El administrador puede realizar las copias de seguridad de las bases de datos usando herramientas propias del sistema de gestión de la base de datos.

3.3 Definición de interfaces de usuario

Al ser una aplicación Web, todas las funciones las presta a través de páginas Web a las que se acceden por medio de un navegador.

El menú principal contendrá botones para realizar ciertas funciones.

La toma de datos por parte del usuario se hará con el uso de formularios. La información marítima se presentara en un mapa con marcadores adecuados para cada tipo de traza.

Las opciones de presentación se harán usando una “check box”.

Para los históricos, las trazas se listaran en una “select box” para permitir la selección de ellas. Por cada traza se mostrará los detalles en una tabla y se marcará cada una de sus posiciones de forma adecuada, junto con información, adicional en un pequeño mapa.

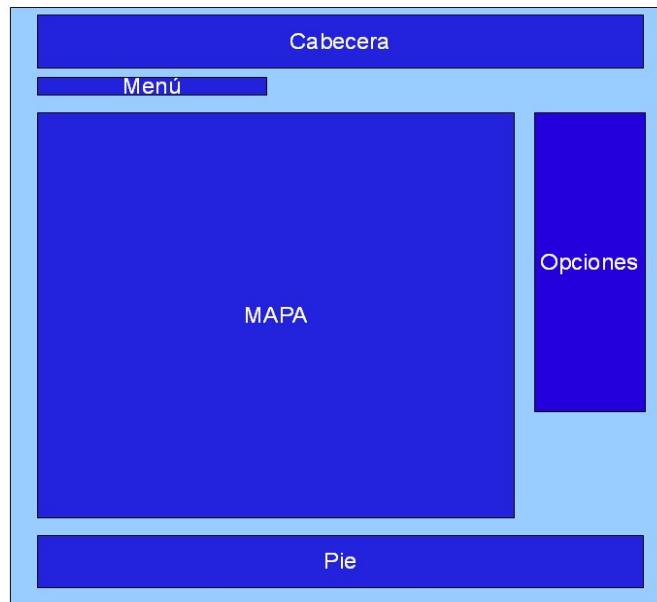
Los mensajes de error se mostrarán de forma adecuada por pantalla.

La ayuda mostrará información básica sobre las funcionalidades del sistema.

La aplicación se compone de cuatro páginas Web diferentes.

3.3.1 Página principal.

Es la pantalla por defecto del portal, mostrara principalmente la posición de los buques en el mapa. Desde esta página se podrá acceder a través de un link a la página de históricos. Este es el aspecto de la página.



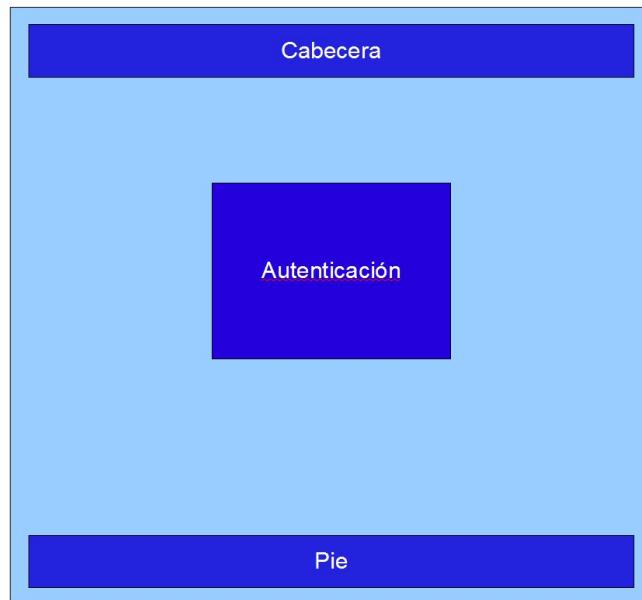
Consta de:

- Menú de opciones.
- Un mapa, que ocupa la parte principal de la página.
- Parte de opciones, en la que el usuario puede cambiar las selecciones deseadas.

Y como en todas las páginas del proyecto una cabecera y un pie.

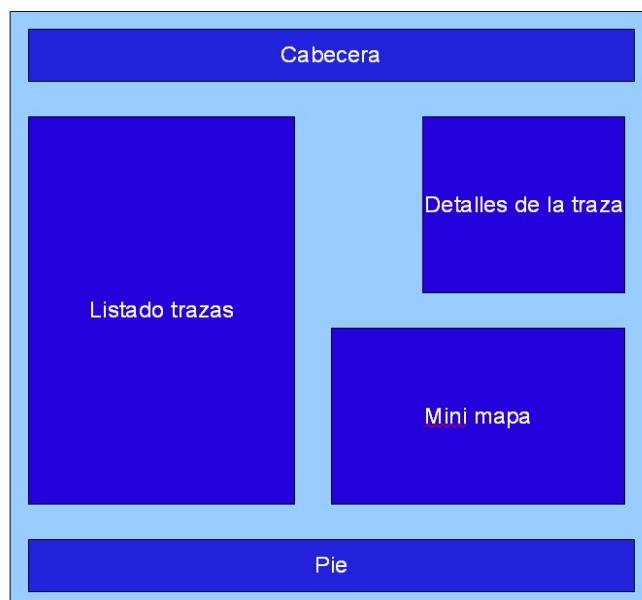
3.3.2 Página de autenticación.

Se accederá desde la página principal para pedir un nombre de usuario y password.



3.3.3 Página de históricos.

Mostrará el nombre de las trazas registradas, el usuario al seleccionar una tendrá los detalles de la misma.



Se compone de:

- Tabla de listado de trazas, donde aparecerán todas las trazas registradas en el sistema.
- Tabla de posiciones, donde aparecen todas las posiciones de la traza seleccionada.
- Mini mapa, donde se marcará cada posición de la traza.

3.3.4 Página de ayuda.

Mostrará una pantalla con texto de ayuda sobre las funcionalidades del sistema.



3.4 Análisis de riesgos

En todo proyecto, tanto si es uno que empieza de cero, como de uno existente en el que se quiere mejorar algún aspecto, existen riesgos.

Los riesgos se clasifican normalmente en dos tipos:

- **Externos**, es decir los originados por factores externos a la empresa. Aquí tenemos por ejemplo los factores económicos externos (la crisis internacional), como algún cambio de la legislación que deje a nuestro proyecto en un estado inestable.
- **Internos**, los que se crean en la propia empresa (técnicos, de gestión, de falta de infraestructura).

Además cuando el proyecto usa software libre hay que tener en cuenta los riesgos añadidos en este tipo de proyecto.

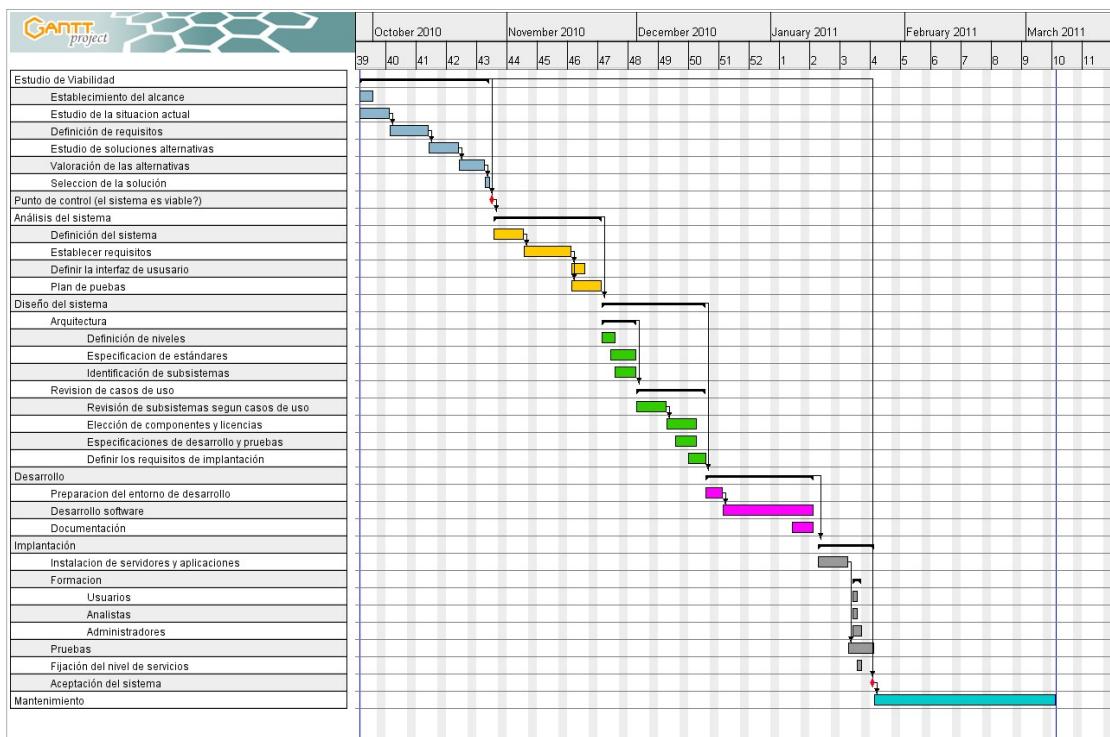
A continuación se describen los riesgos, su probabilidad de ocurrencia y la actuación necesaria para corregirlos o mitigarlos.

Tipo	Riesgos	Prob.	Acciones de prevención y remedio
Externo	Falta de capital	Alta	Minimizar los costes y buscar alternativas de financiación
	Aparición de soluciones de Control de Tráfico Marítimo	Media	Ser flexibles y mantener actualizado el sistema con las nuevas necesidades para mantenerse competitivos
	Cambio en la legislación que no permita la difusión libre de la información del tráfico	Baja	Estudiar alternativas para filtrar contenidos y restringir el acceso
	Que el proveedor de servicios Web (NIRIS) o de mapas (Google) deje de prestarlos	Media	Tener una solución alternativa para implementar ese servicio o usar otro similar.
Interno	Reticencias de la empresa de adoptar soluciones libres	Alta	Explicar claramente las ventajas e inconvenientes de usar software libre
	Complejidad de mantenimiento y configuración.	Baja	Implementar una solución no compleja. Proporcionar suficiente documentación y ofrecer cursos de formación al personal técnico.
	Bajo rendimiento al recibir múltiples peticiones simultáneas.	Media	Usar soluciones de software eficientes con configuraciones óptimas y hardware potente.
	Falta de soporte de la comunidad del software libre.	Baja	Escoger soluciones libres populares que garanticen su permanencia y ampliamente conocidas para, dado el caso, poder contratar el soporte.
	Problemas legales ocasionados por diferentes licencias.	Baja	Mantener en cada momento del proyecto un mapa con todas las licencias usadas y su compatibilidad entre sí.
	Que el proyecto no tenga popularidad y su página Web no sea visitado	Media	Realizar una aplicación Web fácil de usar y atractiva. Anunciarse en los distintos foros de la comunidad de software libre.

3.5 Planificación temporal

El coste temporal del proyecto es de 106 días por hombre. El proyecto debe finalizara el 24 de enero del 2011, para lo cual se proporcionar los recursos necesarios para cumplir los plazos de entrega.

El diagrama de Gantt muestra el reparto de tiempo entre las diferentes fases y tareas del proyecto.



Hay tareas que son independientes de otras y se pueden realizar de forma concurrente sin afectar al proyecto. Como por ejemplo vemos que el establecimiento del alcance y estudio de la situación actual se realizan a la vez. Lo mismo sucede en otras fases como por ejemplo la definición del plan de pruebas, la especificación de los estándares, la documentación o la formación.

En el diagrama se destacan dos puntos de control (hitos):

- **El sistema es viable.** Este punto de control se hace después de realizar el estudio de viabilidad. Si el proyecto es viable con los requisitos, objetivos y presupuesto se continúa con el mismo. Si por lo contrario el proyecto se considera no viable, hay que cambiarlo o desestimararlo.
- **La aceptación del sistema.** Al finalizar la fase de implantación se presenta el proyecto para su aprobación. Si el proyecto es aprobado entra en producción y comienza su mantenimiento.

El mantenimiento se considera por un periodo de un mes en concepto de garantía y luego se prolongara dependiendo del tipo de mantenimiento contratado.

Aunque en el diagrama de Gantt cada tarea tiene una fecha de comienzo y terminación concretas, esto no hace que pueda flexibilizar el comienzo/finalización de las mismas. Se puede por ejemplo comenzar con el desarrollo, una vez conozcamos requisitos, para tener un prototipo cuanto antes.

En esta tabla se especifica las fechas de comienzo, finalización y coste temporal de cada una de las fases del proyecto.

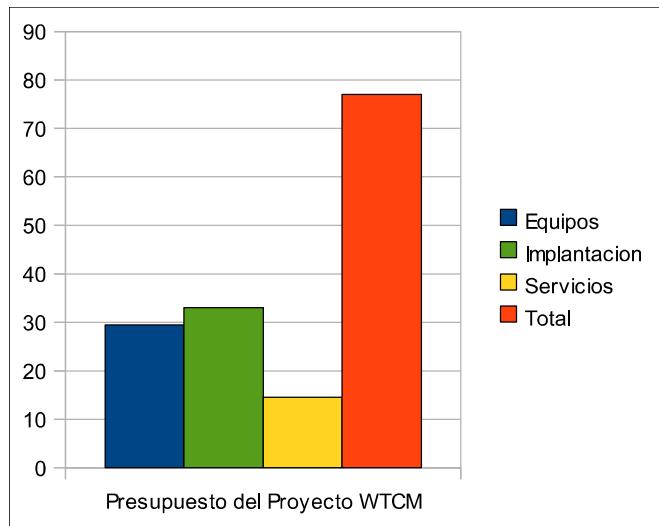
Fase	Comienzo	Final	Días/hombre
Estudio de viabilidad	28/09/10	28/10/10	25
Análisis	29/10/10	23/11/10	20
Diseño	23/11/10	17/12/10	27
Desarrollo	17/12/10	11/01/11	20
Implantación	12/01/11	24/01/11	14
Total	28/09/10	24/01/11	106

3.6 Presupuesto

El presupuesto disponible para este proyecto es de 77.000 Euros, los cuales se repartirán del siguiente modo.

- **Compra de equipos** (29.500 €)
Esto incluye la adquisición de servidores, estaciones de trabajo, discos duros, impresora, unidad de cinta magnética, switch de red, cables, etc.
- **Implantación del proyecto** (33.000 €)
Incluye el coste necesario para la realización de las diferentes fases del proyecto. Incluido un contrato de mantenimiento inicial de un año.
- **Servicios** (14.500 € anuales)
Aquí entran los gastos de alquiler del cuarto de servidores, los gastos de electricidad, conexión a Internet así como un seguro.

El siguiente gráfico representa la distribución de cada partida con respecto al total



En la siguiente tabla se detalla cada uno de los gastos y su valor.

Se considera que el coste de trabajo por día es de 300 euros de media. En este precio se incluye material de oficina, consumibles (papel, toner) y desplazamiento.

Concepto	Cantidad	Precio Unitario €	Total €
Equipos			
Aplicaciones: Servidores HP ProLiant serie DL380 G6	3	5833	17499
Firewall: HP ProLiant serie ML370 G6	1	2514	2514
Discos duros: Unid. disco duro HP Enterprise puerto doble 300GB 6G SAS 10.000 rpm SFF (2,5")	20	360	7200
Copias de seguridad: DLT8000 40/80GB LVD External DLT	1	250	250
Estaciones de trabajo: PC(Dell OptiPlex 380 Desktop) Monitor LCD 17"	2	600	1200
Impresora: HP Color LaserJet CP1215/12 ppm GDI	1	197	197
Switch: Cisco Catalyst 2960-24TT	1	240	340
Cableado y otros			300
Total Equipos			29500
Implantación del proyecto			
Estudio de viabilidad	25 h/día	300	7500
Análisis	20 h/día	300	6000
Diseño	27 h/día	300	8100
Desarrollo	20 h/día	300	6000
Implantación	14 h/día	300	4200
Mantenimiento	1 año	1200	1200
Total Implantación			33000
Servicios			
Alquiler de cuarto servidores	1 año	12000	12000
Teléfono, ADSL, Electricidad, agua	1 año	2000	2000
Seguro	1 año	500	500
Total Servicios			14500
Total			77000

3.7 Especificación del plan de pruebas

En general se realiza un plan de pruebas que permita la detección temprana de errores. Empezando por cada componente individual y progresivamente ir probando la integración con otros componentes. Para finalizar se planificarán las pruebas que deben hacer los usuarios para validar y aceptar el sistema.

3.7.1 Pruebas Unitarias.

Se realizará pruebas de los componentes individuales a nivel de módulo y de función dentro de cada uno.

- Para cada función se comprobara que con unos parámetros de entrada concretos devuelve el resultado esperado. Se deben probar parámetros validos, del tipo que espera la función, parámetros inválidos de tipo incorrecto, por ejemplo pasar un entero cuando se espera una cadena y viceversa. Usar parámetros con valores limites, un entero o real extremadamente grande o muy pequeño y cero para detectar posibles desbordamientos de división por cero.
- Para probar los módulos se creara un pequeño módulo de prueba que se comunique con cada una de los módulos y le envíe datos para ver si el resultado es el esperado. Por ejemplo el módulo que se conecta con el servicio Web remoto para recibir información de trazas, se puede crear una función que le mande un fichero XML con el contenido de algunas trazas.
- Para probar las páginas Web, se comprobará que muestra correctamente los contenidos para cada posible opción que el usuario pueda seleccionar. Aquí también podemos usar el módulo de pruebas, para por ejemplo enviar algunas posiciones a la página encargada de análisis de históricos para probar su comportamiento. En las páginas que contengan tablas y listas de opciones se debe probar como se comporta la página ante gran cantidad de datos para presentar.
- Para los Script del sistema se probará que la acción es la esperada usando diferentes parámetros de entrada y ejecutándolo por diferentes usuarios en directorios con diferentes permisos.
Si el script manipula cualquier estructura de la base de datos, se debe comprobar que las tablas y atributos, claves y tipos de datos son correctos.

3.7.2 Pruebas de Integración.

Se deben de “testear” cómo interactúan todos los componentes entre sí. Se partirá de los casos de uso previamente definidos para los roles de usuario, analista y administrador. Se debe de proporcionar datos en formato XML que contenga información de trazas, así como una colección variada de trazas, posiciones y usuarios en la base de datos para probar como los componentes funcionan conjuntamente.

- Se accederá al sistema como usuario normal, la Web de la aplicación debe presentar su página inicial sin necesidad de autenticación.
- El usuario al solicitar ayuda se debe presentar esta en una página, los contenidos deben de poder leerse correctamente.
- Cuando el usuario esta en la pantalla donde el mapa es visible este puede cambiar el tipo de mapa. La información se debe de visualizar independiente del tipo de mapa que el usuario seleccione, del zoom que elija y del punto del mapa al que se mueva.
- Cuando se seleccione información esta se debe de presentar en el mapa. Se comenzara seleccionando información que contenga pocos datos para comprobar que el mapa los presenta todos.
- Se introducirá un intervalo de refresco y se comprobará que el sistema actualiza la información en el mapa con el intervalo seleccionado
- Se probará que al seleccionar una traza concreta en el mapa se presentan los detalles de la misma en la ventana de información.

- Con los detalles de una traza presentados en pantalla se comprobara que se pueden ver los detalles ampliados de la misma al seleccionar el enlace.
- Se probará que cuando se pide entrar en la página de históricos, aparece una pantalla para autenticación y a continuación la página de históricos si la autenticación ha sido correcta. Si no, debe aparecer un mensaje de error y no permitir entrar para visualizar los históricos.
- Se probará con usuarios y password válida. Así como usuarios y password no válidas.
- Se probará que los datos mostrados por el análisis de históricos coincide con las registrados en la base de datos. Para eso se usara una base de datos con datos limitados fácilmente de comprobar.
- Se probará que entrando en el sistema como administrador, este puede ejecutar el módulo de salvar situaciones en la base de datos y que estos se registran correctamente en la misma
- Se probará, entrando como administrador, que se puede realizar una copia de seguridad de la base de datos.

3.7.3 Pruebas de Implantación.

Esta pruebas tendrán lugar una vez que cada uno de los servidores y estaciones de trabajo están instalados en su entorno de producción con la aplicación funcionando normalmente.

- Se realizarán múltiples accesos a la aplicación tanto desde clientes internos como externos, usando navegadores diferentes.
- Se pondrá una frecuencia de refresco mayor de lo normal para comprobar el rendimiento de los servidores Web.
- Se comprobara que las trazas se registran correctamente en la base de datos. Se analizará el rendimiento del sistema en los “picos” cuando el sistema esta registrando datos y al mismo tiempo sirviendo peticiones del usuario a la base de datos.
Será un test de estrés del sistema de gestión de base de datos ante una carga grande de trabajo.
- Se simulará el fallo de un servidor Web para comprobar que la Web sigue prestando sus servicios.
- Se probará el backup y la restauración de la base de datos.
- Se debe comprobar la seguridad, los accesos no permitidos. Se harán pruebas de intrusión en el sistema.
- Los ficheros logs ayudarán a analizar el tipo y número de peticiones.

3.7.4 Pruebas de Aceptación.

Estas pruebas se realizaran cuando el resultado de las pruebas de implantación haya sido satisfactorio. Esta pruebas a diferencia de todas las anteriores que son realizados por el equipo encargado de la implantaron del proyecto, serán realizadas por usuarios finales del nueva sistema. Es importante de que los usuarios encargados de las pruebas conozcan el

sistema, para ello los cursos de formación se deben de impartir antes que éstas pruebas de aceptación.

Por cada rol (cliente, analista y administrador) se realizaran las pruebas correspondientes siguiendo el manual del usuario del sistema para probar las distintas funcionalidades de la aplicación.

- Para usuarios normales (clientes) se comprobará el acceso al sistema. Así como las opciones de ayuda, cambiar detalles del mapa, seleccionar información y ajustar diferentes intervalos de refresco. Cada una de estas acciones debe de producir los resultados esperados según el manual de usuario.
- Se debe probar el rendimiento del sistema ante múltiples peticiones simultaneas
- Para usuarios analistas deben de probar las funcionalidades analizar posiciones pertenecientes a la historia de una traza seleccionada.
- Los administradores probaran el sistema de backups y de ejecutar scripts de acuerdo con el manual del administrador.

4 Diseño del sistema

Comenzamos con el diseño del sistema, aquí vamos a definir “como” nuestra aplicación va a cumplir los requisitos definidos previamente. Se van a describir los componentes necesarios para cumplir esos requisitos y la forma que tienen de interactuar (comunicarse) con otros componentes de forma que obtengamos un modelo de arquitectura.

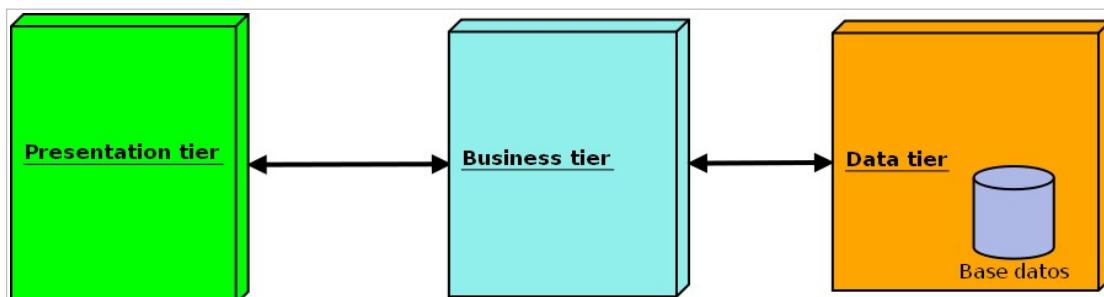
El conjunto de estos componentes deben de satisfacer los casos de uso definidos anteriormente.

4.1 Arquitectura

Se comenzara con la descripción del sistema en sus totalidad, es decir definiendo la “arquitectura conceptual”. Se define el sistema un bloques generales para ir poco a poco desglosándolos en subsistemas, componentes y módulos.

Para la arquitectura de este sistema se ha escogido un enfoque [multi-tier](#), es decir múltiples capas con diferentes cometidos. Esto tiene la ventaja que por una parte tenemos un sistema altamente modular que facilita el diseño y desarrollo y por otra parte obtenemos un sistema con acoplamiento débil. Cada parte es independiente del resto se puede modificar, actualizar a una versión superior sin impactar en el resto de las capas. Al mismo tiempo que se puede instalar cada componente en una servidor separado.

Este gráfico describe la arquitectura del sistema.



Vemos que el sistema tiene una arquitectura de tres capas (Three-tier), es la que cada una de las capas tiene una misión perfectamente delimitada e independiente de las demás.

- **Presentation tier**

Es la capa que interactúa con el usuario, en la cual puede introducir los datos y a su vez los datos del sistema son presentados en esta capa. En el caso de una aplicación Web, como es este caso, esta capa proporciona los contenidos de las páginas Web a la que el usuario accede. Tendrá la página de presentación de tráfico, análisis de históricos, autenticación y ayuda. Normalmente se ejecuta en la parte cliente.

- **Business o logic tier**

Es la capa encargada de realizar la computación, es decir los diferentes procesos en la aplicación. Aquí encontraremos las funciones que toman decisiones para que el usuario pueda visualizar los datos que necesita y sobre los que introduce en el sistema.

Sirve como puente entre las capas de presentación y datos para mover los datos entre ellas.

En nuestra aplicación, esta capa contendrá los procedimientos de llamada a los servicios Web y la decodificación de los datos devueltos por los mismos, así como servir de interfaz entre la base de datos y la presentación de trazas en las tablas presentadas al usuario.

- **Data tier**

Es la capa en la que residen y se manipulan los datos.

Todos los accesos a la base de datos se encuentran en esta capa.

4.1.1 Identificación de subsistemas

Una vez vista la arquitectura general del sistema pasamos a identificar cada una de las partes del mismo. Esto permitirá un diseño y por consiguiente un desarrollo, más fácil y compacto ya que no hay que abordar el conjunto del sistema como un “todo”

Para la división en subsistemas se consideran factores como:

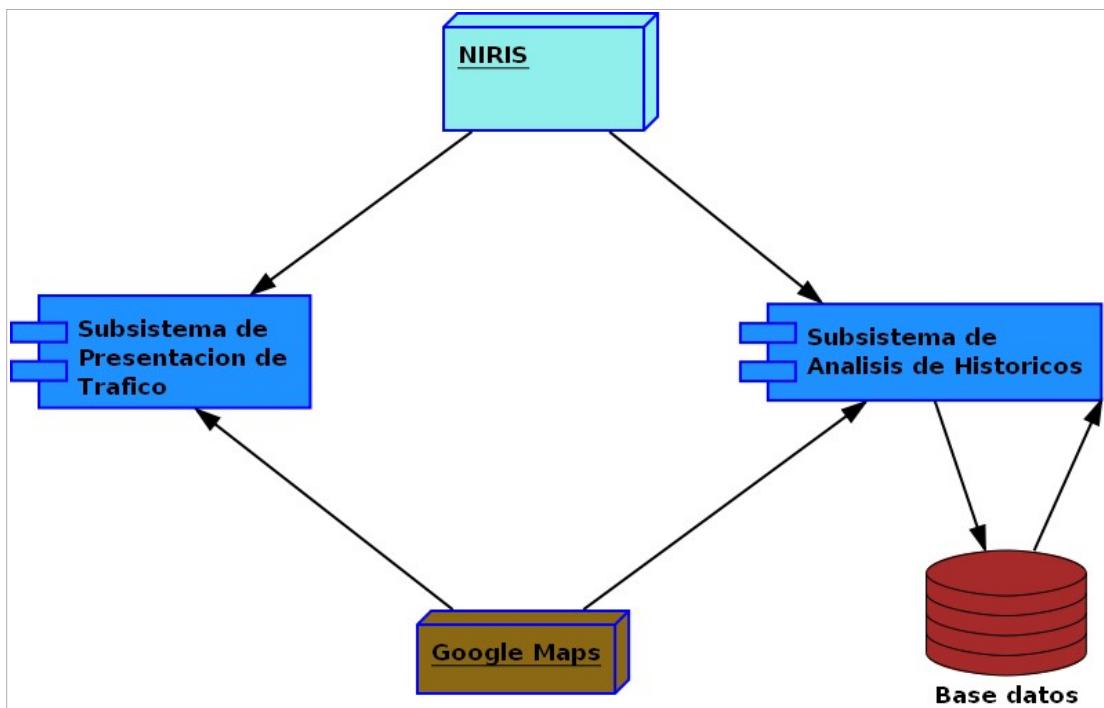
- Funcionalidad.
- Datos y tipos a los que acceden.
- Interfaz de usuario común.
- Optimización tanto en comunicación con otras partes del sistema local o remoto y de los recursos disponibles.

Teniendo en cuenta estos factores el sistema de Control Tráfico Marítimo se divide en estos subsistemas:

- Subsistema de Presentación de Tráfico
- Subsistema de Análisis de Históricos

En el siguiente diagrama de bloques se muestran estos y como se comunican con sistemas externos. Podemos apreciar que ambos reciben las trazas de NIRIS, uno para presentarlas en pantalla y el otro para grabarlas en la base de datos.

También se aprecia que ambos subsistemas hacen uso de Google maps para presentar sus datos en los mapas respectivos.

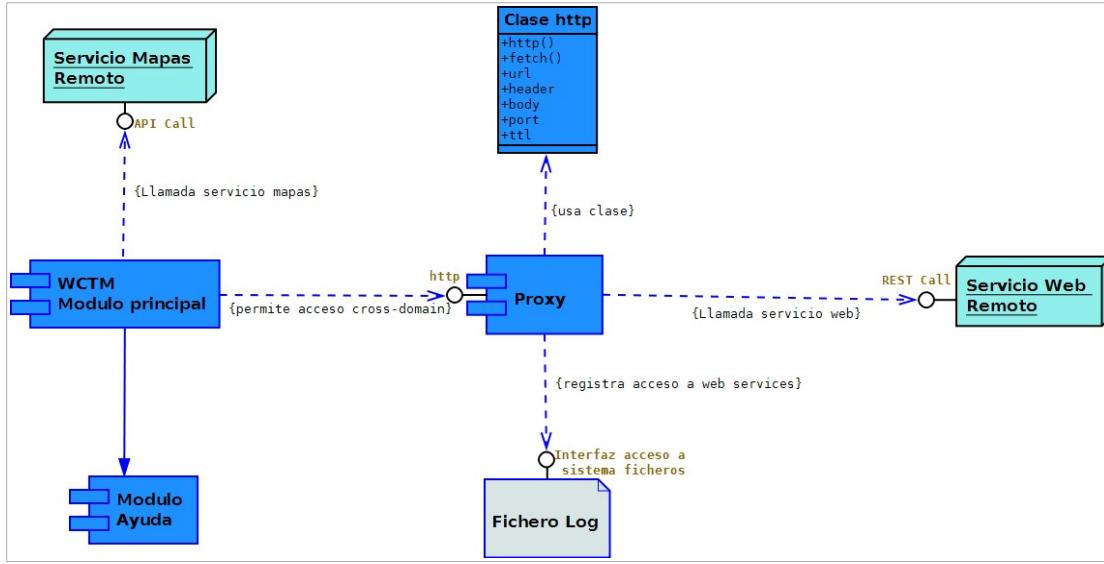


El subsistema de Presentación de Tráfico es el encargado de mantener actualizada la información del tráfico marítimo en la página principal de la aplicación. Para esto se conecta periódicamente al sistema NIRIS para recibir los detalles de las trazas. Para la presentación de las trazas se hace uso del mapa de Google, por lo tanto el subsistema de Presentación de Trafico necesita establecer una conexión con el servicio de mapa de Google.

El subsistema de Análisis de Históricos es el encargado de manejar las diferentes posiciones de las trazas. Por una parte se conectara periódicamente al servicio Web del NIRIS para registrar la información sobre ellas. Esta información será almacenada adecuadamente en la base de datos. Por otra parte el subsistema de Análisis de Históricos también es el encargado de tratar esa información grabada en la base de datos para que se presente de forma adecuada al usuario el cual realizará el análisis de históricos. Para eso accederá a la base de datos actualizando la aplicación Web con los datos registrados y por otra parte también se conectara al servicio de mapas de Google para marcar las posiciones en un mapa.

Una vez definido los subsistemas de la aplicación se pasa a definir los componentes de cada uno de ellos en mayor detalle.

4.1.2 Subsistema de Presentación de Tráfico



En este diagrama se puede apreciar los componentes o módulos que comprenden este subsistema. Se describe también la comunicación entre ellos y con los sistemas remotos así como el tipo de interfaz que se usa.

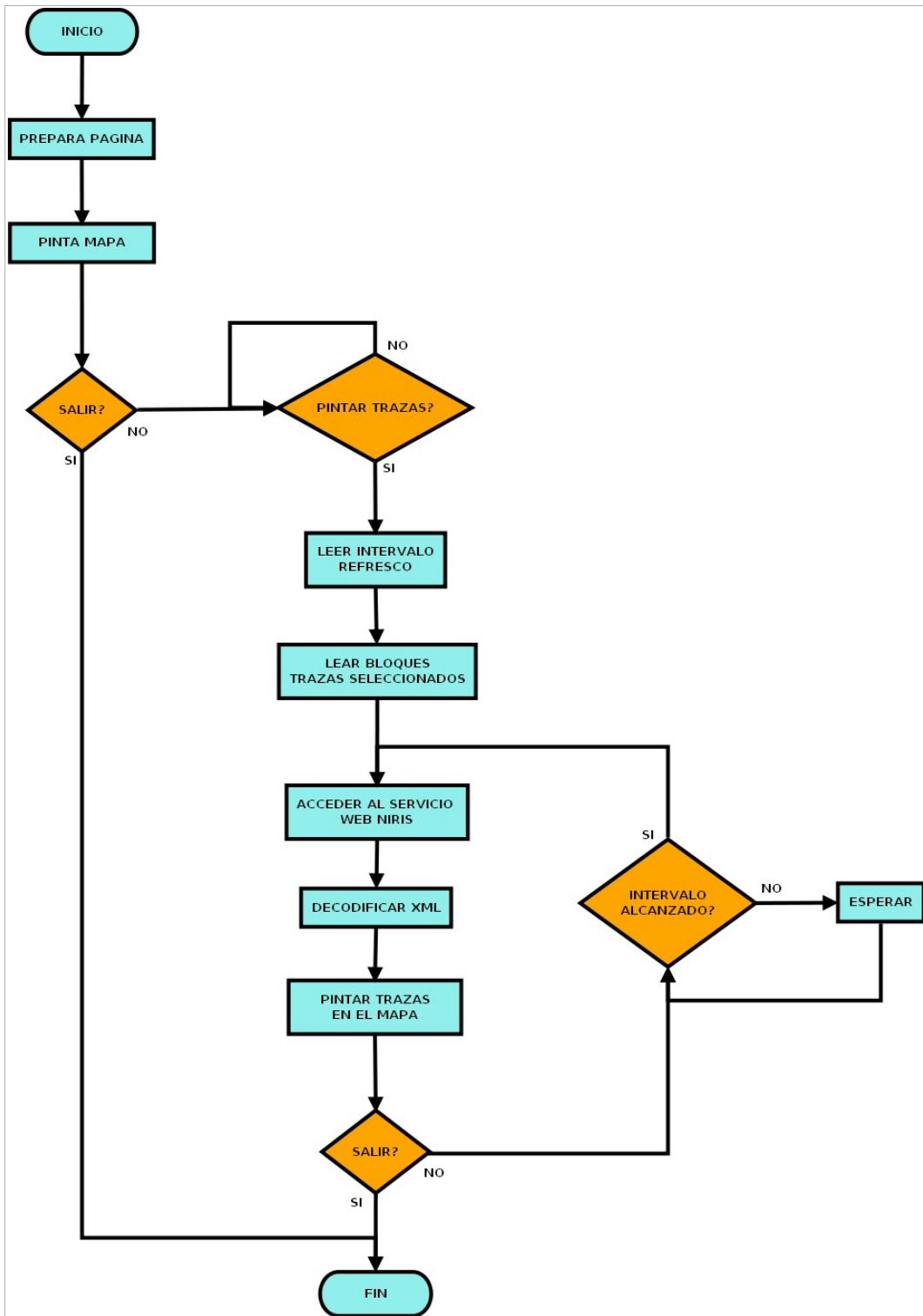
A continuación se detalla cada uno de los componentes.

- **Módulo Principal**

Es el que proporciona la información del tráfico marítimo el cliente que se conecte a la Web. Esta escrito en JavaScript y HTML. Es el encargado de actualizar cada una de las trazas que seleccione el usuario sobre el mapa de la aplicación para gestionar el control de tráfico marítimo de la zona.

Para ello realiza llamadas tanto al API de Google como a los servicios Web del sistema NIRIS a través de un proxy.

Aquí se representa el diagrama flujo de este módulo.



- **Módulo Proxy**

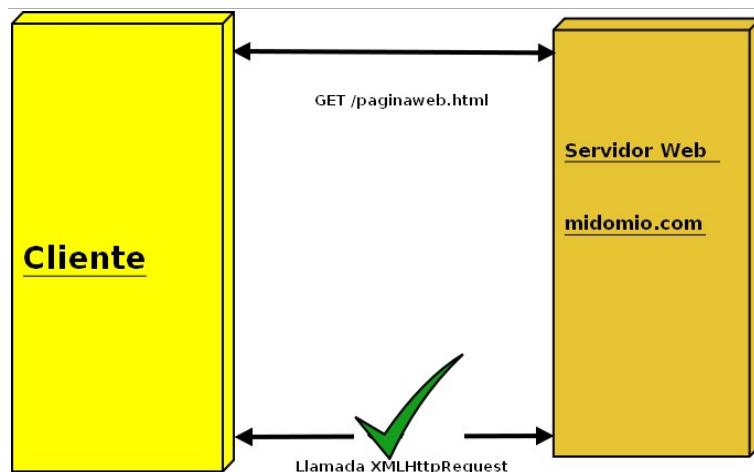
Es el encargado de realizar las llamadas http en nombre del módulo principal. Esta escrito PHP.

La razón de usar el proxy es debido a la política que implementan ciertos lenguajes de programación, entre ellos JavaScript, de no permitir llamadas remotas. Es se

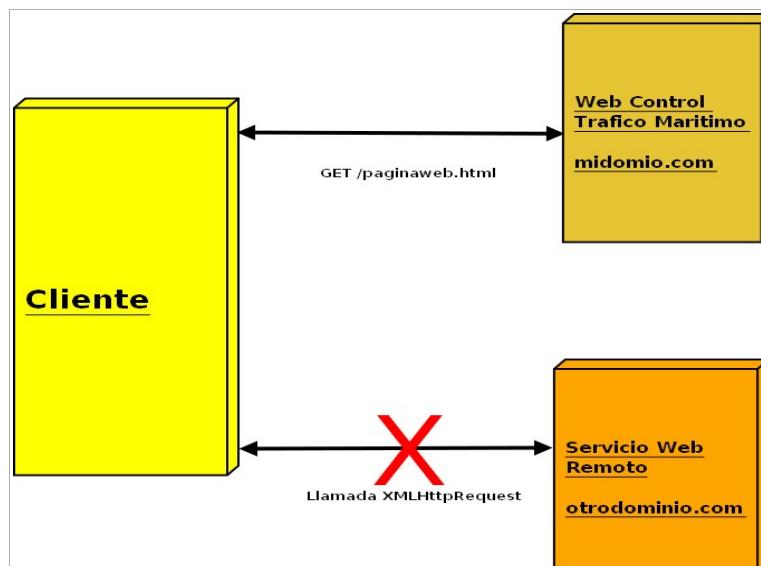
debe a razones de seguridad y se conoce como “same origin policy”, es decir una política que permite a un programa o script dentro de un sitio Web o dominio acceder a los métodos y propiedades de otros programas dentro del mismo dominio pero que por otra parte impide el acceso a esos mismos métodos o propiedades cuando el programa o script es de un dominio diferente (cross-domain).

Estas limitaciones solo existen en la parte cliente, es decir en el software que se ejecuta en la máquina de un usuario cuando este se conecta a nuestra Web. Este es el caso de nuestro módulo principal en el cual el código es ejecutado en el mismo cliente que realiza llamadas a los servicios Web de sistema remoto.

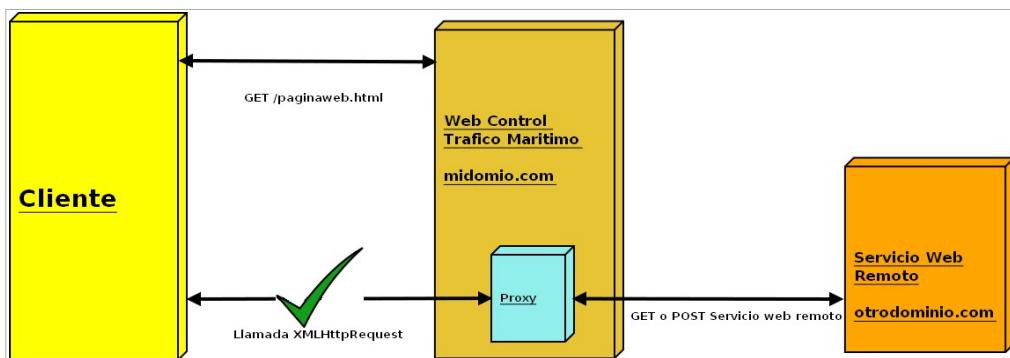
Como se puede apreciar en el siguientes diagrama, cuando la llamada http se realiza a un servidor Web en el mismo dominio el navegador no impone ninguna restricción de seguridad y la conexión se establece correctamente.



Si por lo contrario la llamada XMLHttpRequest se realiza a otro servidor, el navegador cliente no permite la conexión.



Por ese motivo tenemos que añadir un módulo “proxy” en nuestro servidor Web.



Aquí vemos que el cliente realiza la llamada al propio servidor Web, con lo cual no hay restricciones.

A parte de la opción de usar un proxy otras alternativas podrían ser:

- Usar los módulos de apache “mod_proxy” o “mod_rewrite” que permiten dirigir las peticiones de un servidor a otro servidor Web directamente. El cliente realiza la conexión con el sistema remoto como si del servidor Web local se tratase. Apache se encarga de realizar todas las trasformaciones para que sea “transparente” al navegador cliente.
- Emplear una firma digital en el script que accede al servicio remoto. En los navegadores estos scripts se consideran como de confianza y dejan realizar llamadas a dominios remotos sin restricciones.
- Usar [JSON](#) en lugar de AJAX (XML y llamadas XMLHttpRequest)

Para esta aplicación se ha elegido la opción de proxy.

El módulo proxy, escrito en PHP, se ejecuta en la parte servidor. Recoge las peticiones del cliente, escrito en JavaScript, y realiza la llamada http correspondiente al servicio Web remoto, en este caso el sistema NIRIS. Una vez recibida la respuesta es enviada al cliente, al módulo principal de la Web Control Tráfico Marítimo.

Aunque en algunos navegadores, como Internet Explorer, se puede activar la opción de acceso a otro dominio, Cross-domain Request ([XDR](#)). Se ha considerado implantar el proxy para que se asegure un correcto funcionamiento desde cualquier navegador independientemente de su configuración.

Este sería un ejemplo de llamada usando el módulo proxy (proxy.php):
http://www.domio_local.com/proxy.php?proxy_url=http://www.domio_remoto.com

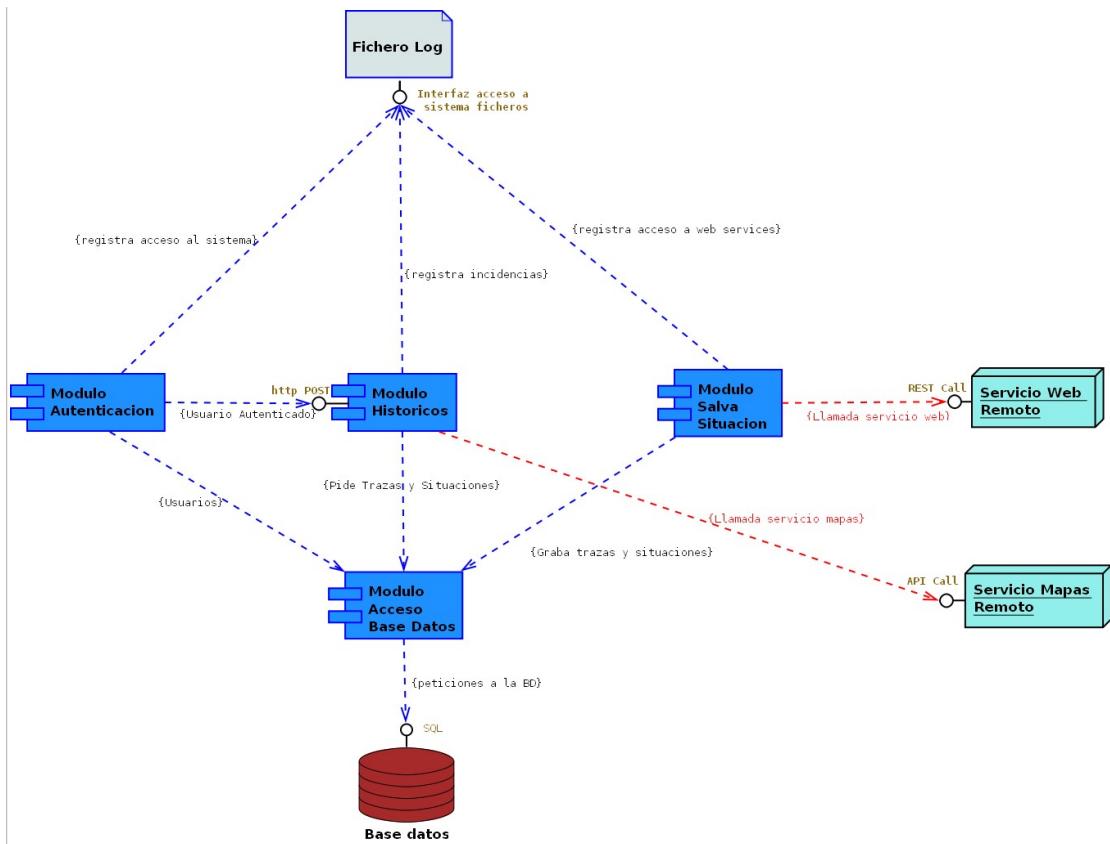
Este módulo proxy usa una clase (clase http) que contiene los métodos y propiedades necesarias para realizar las llamadas http y https a los servidores remotos.

El módulo proxy también accede al sistema de ficheros para recoger en un fichero 'log' las diferentes incidencias y las diferentes peticiones recibidas. Esto se empleará como forma de auditar las diferentes peticiones de los clientes por razones de seguridad en la aplicación y por otra para analizar el Número de peticiones en el tiempo para evaluar el rendimiento del sistema.

- **Módulo de Ayuda**

Es el encargado de presentar la información sobre el uso y funciones de la aplicación a los usuarios que lo soliciten. Esta escrito en [HTML](#).

4.1.3 Subsistema de Análisis de Históricos



- **Módulo Salva Situación**

Es el encargado de conectarse periódicamente al servicio Web del sistema NIRIS para grabar la información sobre trazas y sus posiciones en la base de datos. Esta escrito en PHP y se ejecuta de forma periódica a través del sistema de gestión de tareas “cron” del sistema operativo Linux. La frecuencia de ejecución la dicta el administrador del sistema, esta tiene que ser un compromiso entre una actualización adecuada para el mantenimiento del histórico de trazas marítimas y de la capacidad y rendimiento del sistema.

Cada vez que este módulo entre en ejecución establecerá una conexión al servicio Web para recibir toda la información sobre trazas. A diferencia del módulo principal, en el subsistema de Presentación de trazas, en el que el cliente tiene la opción de seleccionar que tipos de trazas gestiona el módulo, en este caso se registran solo las trazas marítimas del AIS.

Cabe destacar que en este caso, al tratarse de un módulo que opera en la parte servidor, no tiene las restricciones de acceso a otro dominio, la mencionada “same origin policy” y por lo tanto no necesita el uso de un proxy. Los accesos remotos los realiza directamente.

Por cada conexión al servicio remoto, el módulo extrae la información necesaria del mensaje en formato XML y la salva en la base de datos. Esta informatización consta del nombre de la traza la posición en latitud y longitud, el rumbo, velocidad y la hora actual.

En cada conexión se escribe el resultado de la misma en un fichero log.

- **Módulo de Históricos**

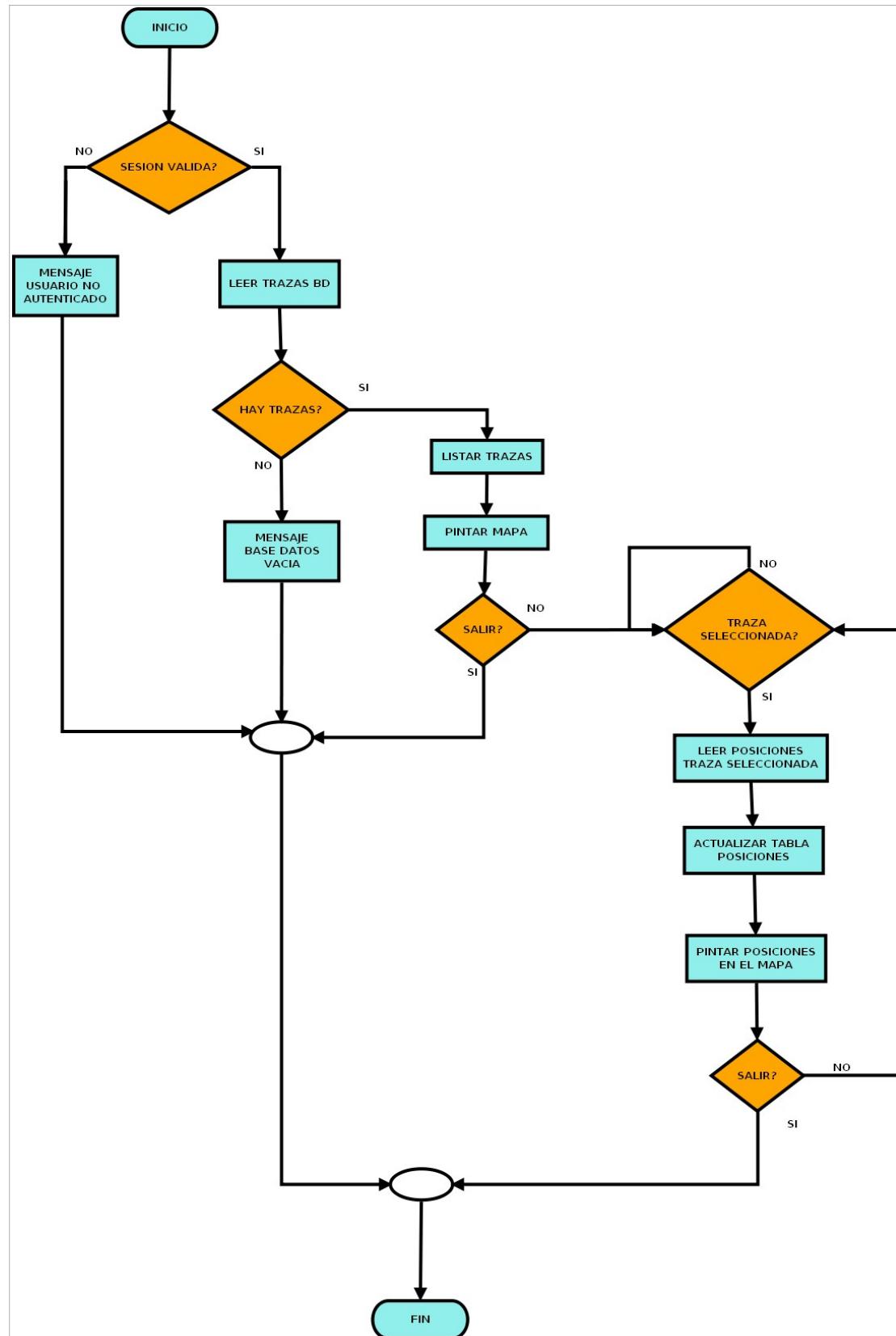
Permite la realización de un análisis de las trazas registradas en el sistema, en la base de datos. Esta escrito PHP, JavaScript y HTML. Realiza peticiones al módulo encargado del acceso a la base de datos para recoger las trazas y las posiciones que el usuario solicite para su análisis. Como hemos visto anteriormente, el módulo de salvar situaciones guarda información de todas las trazas y sus posiciones en la base de datos. Esta información es la que el módulo de históricos usa para ofrecer a los usuarios los detalles necesarios para el análisis.

Este módulo también realiza conexiones al servicio de mapas para presentar las posiciones sobre un mapa.

El acceso a este módulo solo es posible si un usuario se ha autenticado correctamente y se ha abierto una sesión. Este módulo comprueba que existe la sesión de autenticación y si el valor es el correcto. Este mecanismo impide que algún usuario pueda llamar directamente el módulo de históricos sin haber pasado previamente por la fase de autenticación.

Así como otros módulos, el de históricos escribe las incidencias en un fichero log, para su posterior auditoria por parte de los administradores del sistema.

A continuación se presenta el diagrama de flujo de este módulo.



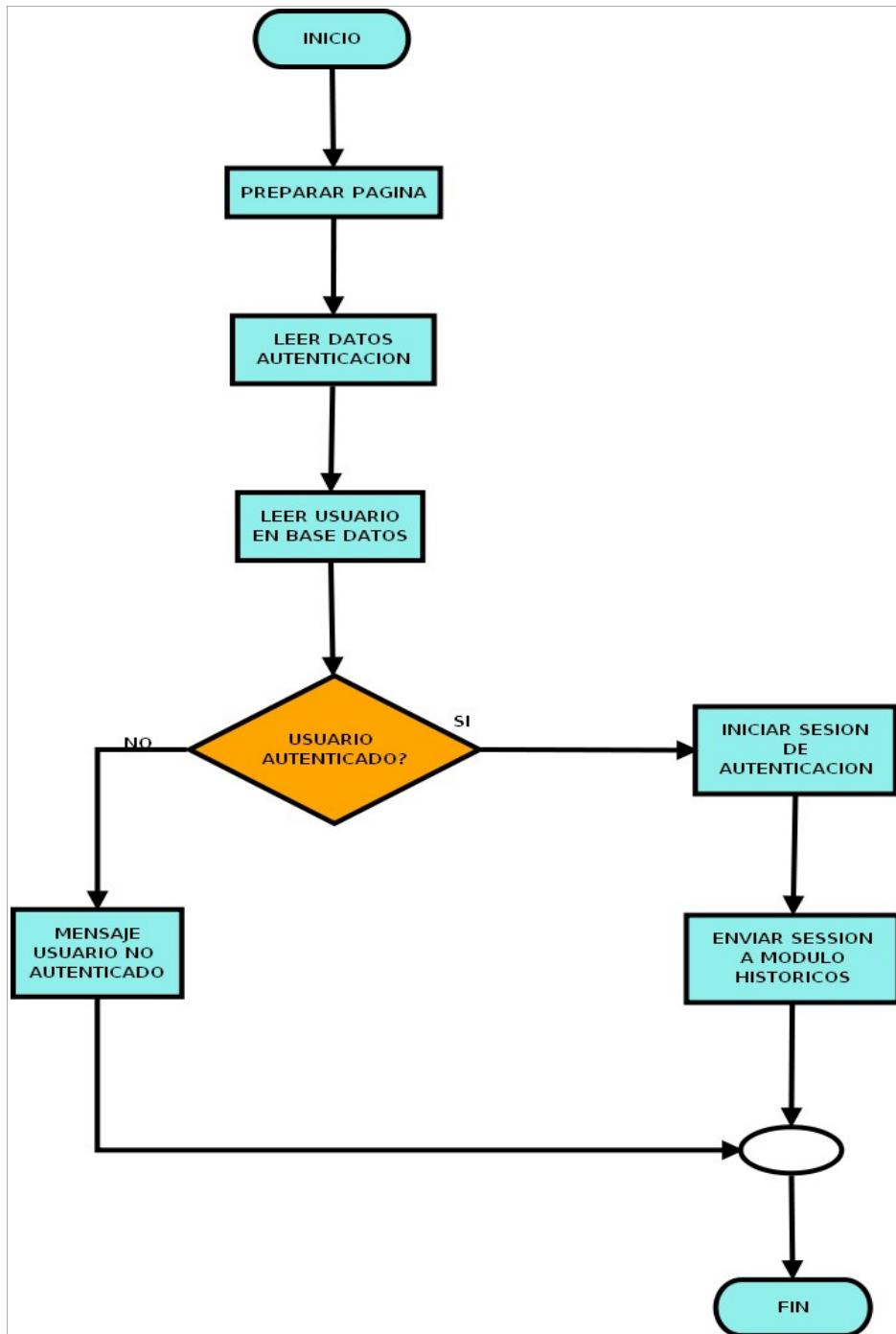
- **módulo de Autenticación**

Es el encargado de controlar el acceso al subsistema de históricos, solo los usuarios registrados en el sistema podrán acceder al mismo. En la base de datos se recogen todos los detalles de usuarios que pueden acceder al subsistema de históricos. Por cada petición de conexión por parte de un usuario, este módulo realiza una lectura de la base de datos para comprobar que el nombre de usuario y clave de acceso son correctos. Si la autenticación ha tenido éxito se crea una sesión y se envía el control al módulo de históricos.

Para guardar la clave del usuario en la base de datos se ha optado por usar encriptación. Esto garantiza de que nadie, ni siquiera los administradores de la base de datos puedan saber cual es esta. Se ha elegido un mecanismo de encriptación propio de PHP. La otra alternativa era usando las propias funciones de MySQL, esta opción tiene el inconveniente que durante el proceso de autenticación la comunicación entre el servidor Web e la base de datos la clave se envía en claro, y alguien la puede capturar con un analizador de tráfico de red.

Todos los detalles sobre los intentos de autenticación se registran en un fichero log.

Aquí se presenta el diagrama de flujos de este módulo.

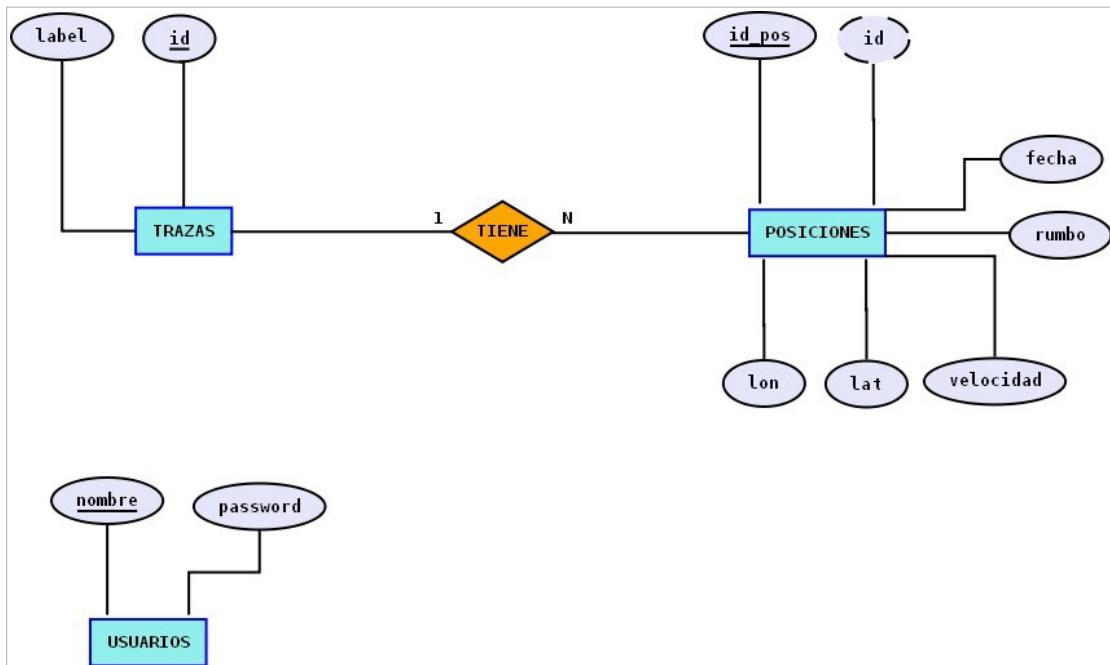


- **Módulo de Acceso a la Base de Datos**

En este módulo se centralizan todas las operaciones de apertura, cierre, lectura y escritura en la base de datos. Por consistencia ningún módulo puede acceder directamente a la base de datos, todos tienen que hacerlo a través de este módulo. Para cada una de la operaciones se usa la interfaz SQL. Como en el resto de los módulos se registra en un fichero log cualquier incidencia.

4.1.4 Diseño de la base de datos

Para la base de datos se ha optado por un [modelo relacional](#). Este es el diagrama [Entidad-Relación](#) de la misma.



En el diagrama E-R podemos apreciar que consta de tres entidades,

- **TRAZAS**
Mantiene la información de las trazas registradas en el sistema
- **POSICIONES**
Por cada traza guarda todos sus posiciones junto con el 'timestamp' de la misma
- **USUARIOS**
Mantiene la relación de usuarios registrados en el sistema, junto con su password encriptada.

Y una relación “**TIENE**”. Esta relaciona la entidad trazas con cada una de sus posiciones. Su cardinalidad es 1:N, lo que indica que una traza puede tener muchas posiciones, pero una posición solo pertenece a una traza concreta. El atributo que las relaciona es el “`id`” de la traza.

Este es el modelo relacional de la base de datos.

TRAZAS(id, label)

POSICIONES (id_pos, id, rumbo, velocidad, lat, lon, fecha)
donde {`id`} referencia a **TRAZAS**

USUARIOS(nombre, password)

4.1.5 Especificación de estándares, normas de diseño y construcción

Todos los documentos del proyecto se presentarán en forma digital (preferentemente en formato Open Document Format o PDF) en hojas DIN A4, numeradas correlativamente, a doble cara. La separación entre renglones y entre párrafos debe ser suficiente para permitir una lectura cómoda. El tipo de letra debe ser Arial 10.

La estructura de los documentos será el siguiente:

Portada: logotipo la Compañía, título del proyecto: Web Control Tráfico Marítimo, especialidad, nombre del autor, fecha y versión/revisión del documento.

Primera página: licencia de publicación del documento que debe ser [CC-BY-SA](#).

Tabla de contenidos, con una entrada por cada capítulo y anexo. Cada una de estas entradas puede tener subentradas haciendo referencia a secciones, subsecciones, etcétera, hasta un máximo de cuatro niveles.

A continuación se incluirán los capítulos necesarios que se consideren convenientes en el documento.

Los diagramas de diseño se realizaran usando la notación [UML](#). Para el diseño de la base de datos emplearemos los diagramas de entidad-Relación

El último capítulo incluirá las conclusiones.

Finalmente se incluirán las Referencias bibliográficas, debidamente indicadas en el resto del documento (con las citas oportunas).

4.2 Elección de alternativas de componentes y licencias más adecuadas

En este apartado se define el software que utilizamos para cada componente del sistema y su licencia.

Para cada componente se ha usado software libre con una licencia que no permita incompatibilidades entre ellas.

La siguiente tabla describe en detalle el software empleado en cada parte del sistema así como su licencia. Esta combinación nos permite que la aplicación final (Web Control Tráfico Marítimo) se licencie como tipo [GPL](#).

Componente	Software	Licencia
Aplicación Web	Pre procesador de Hipertexto: PHP 5.2.6	PHP Licencia (tipo BSD)
	Lenguajes: PHP, JavaScript, HTML, SQL, bash scripts	NO
	Estándares: XML, CSS	NO
	Tecnologías: AJAX, SOA	NO
Sistema operativo para servidores y firewall y clientes	Debian GNU Linux 5.0 (Lenny)	GPL
Servidor Web	Apache 2.2.9	APL versión 2.0
Load Balancer	Ultra Monkey 3	GPL
Firewall	netfilter/iptables	GPL
Suite ofimática	OpenOffice.org 3.2.1	GPL
Sistema de Gestión de Base de Datos	MySQL 5.0.51	GPL
Sistema de Control de versiones	Subversion 1.6	APL versión 2.0
Control y seguimiento errores	Trac 0.12.1	BSD
Herramienta de creación de gráficos	Dia 0.97.1	GPL
Herramienta de creación de diagramas	Gantt project 2.0.10	GPL
Herramienta para el análisis de los servicios Web.	SoapUI 3.6.1	LGPL

Los documentos, manuales y los materiales de formación se licenciaran como [Reconocimiento-CompartirIgual](#) de creative commons.

Para el código fuente se usara la licencia [GPL](#).

4.3 Especificaciones de desarrollo

Aquí se describe las necesidades tanto de equipos como de software necesarios para realizar el desarrollo.

4.3.1 Necesidades de equipos

Será necesario tanto hardware como un local y material de oficina. Estas son las necesidades.

- **(2) Estaciones de trabajo**
 Modelo: PC (Dell OptiPlex 380 Desktop)
 Procesador: AMD Athlon X4 2.60 GhZ
 Memoria: 3GB RAM
 Tarjeta gráfica ATI Radeon 256 MB
 Monitor LCD 17"
- **Switch Ethernet**
 Modelo:Cisco Catalyst 2960-24TT

- **Impresora**
Modelo: HP Color LaserJet CP1215/12 ppm GDI
- **Unidad de cinta**
Modelo: DLT8000 40/80GB LVD External DLT
- Local con mobiliario
- Material de oficina (papel, pizarra, lápices, etc.)
- Conexión a Internet. ADSL mínimo 20 Mb.

El local será de acceso limitado. Al repositorio de software solo podrán acceder los desarrolladores.

Se considera conveniente dos ordenadores tanto para el desarrollo en paralelo como para realizar pruebas de conexiones desde un maquina remota.

La unidad de cinta es necesaria para realizar los backups de las diferentes versiones y documentación del sistema. Cada día se creará una copia de seguridad.

El equipo usado durante el desarrollo pasará a formar parte de la implantación final del sistema.

4.3.2 Necesidades de software

Aquí incluiremos tanto las utilidades necesarias para el desarrollo en si, las aplicaciones que soportan nuestra sistema y las aplicaciones para generar la documentación.

Utilidades de desarrollo

- Sistema operativo Linux Debian 5.0
- Editor de texto
- Subversion
- Trac
- PHP 5.0
- SoapUI

Aplicaciones usadas en nuestro sistema

- Apache 2
- MySQL

Aplicaciones para la documentación

- OpenOffice.org
- Dia
- Gantt project

Como sistema de control de versiones se usara [subversion](#), se crearan varias ramas para las diferentes versiones que se separarán en dos grandes grupos “estables” e “inestables”.

Para el seguimiento de errores “bugs” se usa “[trac](#)” aquí separaremos los errores según por su estado (nuevo, asignado, resuelto, cerrado) como por su gravedad (crítico, mayor, menor, cosmético). Por cada error se anotará además de los detalles del mismo, quien lo detectó, en que versión del programa se detectó y en que módulo.

4.4 Especificación de pruebas

En la fase de análisis se ha definido un plan de pruebas que tiene que cumplir el sistema para que sea aceptado y entre en producción. En esta fase de diseño al tener definido cada una de los componentes y módulos de la aplicación se puede especificar las pruebas que se realizaran durante el desarrollo de cada módulo.

Aquí especificaremos las pruebas a dos niveles, las pruebas unitarias que se realizan sobre un módulo concreto, y las de integración las realizadas con los módulos trabajando juntos y comunicándose entre si.

4.4.1 Pruebas Unitarias.

Se realizará pruebas de los componentes individuales

- Módulo Principal.
Se probará que muestra adecuadamente todos los contenidos de la página con los estilos adecuados. Se usarán navegadores diferentes.
Se probará que todos los enlaces funcionan.
Las opciones de selección de trazas funcionan adecuadamente.
Los formularios aceptan los datos de forma correcta.
Las funciones del mapa actúan correctamente.
- Módulo de salvar la situación de las trazas.
Comprobar que las trazas y sus situaciones son salvadas correctamente en la base de datos y con el intervalo establecido.
- Módulo de autenticación.
Se probará que muestra adecuadamente todos los contenidos de la página con los estilos adecuados. Se usarán navegadores diferentes.
Se probará que acepta correctamente los datos de autenticación del usuario y los compara con los datos de usuarios almacenados en la base de datos.
Se probará tanto con usuarios presentes en la base de datos como no presentes así como usando password correcta y erróneas.
- Módulo históricos.
Se probará que muestra adecuadamente todos los contenidos de la página con los estilos adecuados. Se usarán navegadores diferentes.
Se comprueba que presenta todas las trazas que hay en la base de datos.
Al seleccionar una traza se presenta sus posiciones en la tabla y en el mapa.
- Script para crear la estructura de la base de datos.
Se comprobará que la base de datos se ha creado correctamente en MySQL con las tablas, atributos, tipos de datos, claves, claves externas y restricciones adecuadas para crear la base de datos relacional.
- Script para ejecutar la tarea de salvar posiciones periódicamente.
Se comprobará que el script crea las entradas adecuadas en el fichero /etc/crontab.
- Script de crear usuarios en la base de datos.
Se probará que el usuario es creado correctamente junto con sus password encriptada.
Se probará la creación de un usuario que ya existe para verificar el mensaje de error del script.

4.4.2 Pruebas de Integración.

Una vez probado que cada función y módulo individual funciona correctamente se procederá a las pruebas de integración del sistema. Como la aplicación esta dividida en dos subsistemas Presentación de tráfico y Análisis de históricos, se comenzada por las pruebas de integración de cada módulo en el subsistema correspondiente para luego pasar a la integración de los dos subsistemas.

- Se introducirá un intervalo de refresco y se seleccionara un grupo de trazas para ver si aparecen en el mapa y se refresca su posición con el intervalo establecido.
- Se probará si al seleccionar una traza se presentan los detalles sobre la misma y al seleccionar “Más detalles” se abre una página proporcionada por el sistema NIRIS con información adicional.
- Se probará que cuando se pide entrar en la página de históricos, aparece una pantalla para autenticación y a continuación la página de históricos si la autenticación ha sido correcta. Si no, debe aparecer un mensaje de error y no permitir entrar para visualizar los históricos.
- Después de que el módulo de “Salvar Situaciones” haya estado funcionando por un tiempo, se probará que los datos están disponibles en la historia.
- Al crear un usuario en la base de datos, se probará que este puede acceder al sistema de análisis de históricos.

4.5 Requisitos de implantación

Para la implantación del sistema es necesario de disponer de un cuarto de servidores. El cuarto de servidores debe ser un recinto cerrado y de acceso limitado. Solo los administradores del sistema y personal autorizado podrán acceder al mismo. Cada acceso quedara registrado en un hoja de control en la que se anotara la siguiente información:

- Fecha y hora de entrada
- Nombre y apellidos
- Cargo
- Motivo del acceso
- Firma
- Fecha y hora de salida

Este cuarto debe disponer de la capacidad suficiente en espacio y en potencia eléctrica para albergar en producción a cuatro servidores en una rack.

Además debe de disponer de espacio para montar un switch, unidad de cinta, impresora y dos pc's.

El cuarto debe disponer de aire acondicionado y [UPS](#) que permita a los servidos funcionar ininterrumpidamente 24 horas al día.

Estos son los requisitos hardware del sistema en producción.

- **Servidores Web (2 servidores, primario y secundario)**

Modelo: HP ProLiant serie DL380 G6 (2U)

Procesador: Intel® Xeon® E5530 (4 núcleos, 2,40 GHz, 8 MB L3, 80W)

Memoria: 12 GB PC3-10600R (RDIMM)

Controlador de almacenamiento: Smart Array P410i/256 MB

Disco duro: SCSI 2*300 GB 6G SAS 10.000 rpm SFF (2,5")
Adaptadores de red: Ethernet 1GB, dos adaptadores.

- **Servidor de base de datos (1 servidor)**
Modelo: HP ProLiant serie DL380 G6 (2U)
Procesador: Intel® Xeon® E5530 (4 núcleos, 2,40 GHz, 8 MB L3, 80W)
Memoria: 12 GB PC3-10600R (RDIMM)
Controlador de almacenamiento: Smart Array P410i/256 MB
Disco duro: SCSI 8*300GB 6G SAS 10.000 rpm SFF(2,5")
Adaptadores de red: Ethernet 1GB, dos adaptadores.
- **Firewall (1 servidor)**
Modelo: HP ProLiant serie ML370 G6
Procesador: Intel® Xeon® E5540 (4 núcleos, 2,53 GHz, 8 MB L3, 80W)
Memoria: 6 GB PC3-10600R (DDR3)
Controlador de almacenamiento: Smart Array P410i/256 MB
Disco duro: SCSI 2*300GB
Adaptadores de red: Ethernet 1GB, 4 adaptadores.
- **Estaciones de trabajo (2 PC's)**
Modelo: PC(Dell OptiPlex 380 Desktop)
Procesador: AMD Athlon X4 2.60 GhZ
Memoria: 3GB RAM
Tarjeta gráfica ATI Radeon 256 MB
Monitor LCD 17"
- **Unidad de cinta (1)**
Modelo: DLT8000 40/80GB LVD External DLT
- **Impresora (1)**
Modelo: HP Color LaserJet CP1215/12 ppm GDI
- **Switch Ethernet (1)**
Modelo: Cisco Catalyst 2960-24TT
- **Cableado y otros**

5 Desarrollo

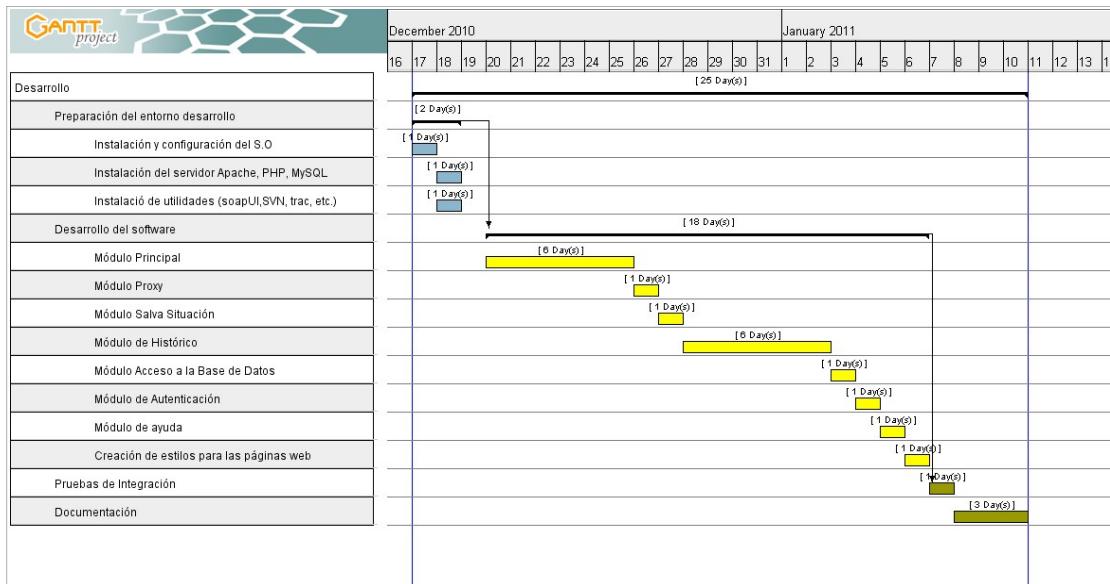
Esta es la fase en la que se codifica los diferentes módulos y clases del sistema.
En la fase de diseño ya hemos definido en detalle lo que hace cada módulo y las conexiones entre ellos. Aquí crearemos las funciones y procedimientos para llevar a cabo cada una de las operaciones que deben realizar cada uno.

Todos los módulos o componentes son de nueva creación, la única excepción es la clase que usa el módulo proxy que se usará una ya creada y con licencia libre.

La técnica de desarrollo será la de refinamientos sucesivos. Crear un prototipo lo antes posible e ir añadiendo funcionalidades al sistema paulatinamente.

5.1 Planificación de las actividades de desarrollo e integración de sistemas

En este diagrama de Gantt se describe la planificación de las diferentes tareas del desarrollo



Se comienza con la preparación del entorno de desarrollo en la que se instala los sistemas y aplicaciones para poder comenzar la codificación.

El “corazón” del sistema lo forman el sistema operativo [Debian GNU Linux](#), Servidor Web [Apache](#), Sistema de gestión de Base de datos [MySQL](#), e interprete de [PHP](#).

Para el control de versiones se usa [Subversion](#) y para el seguimiento y gestión de “bugs” se usa [trac](#). Ambas soluciones son necesarias para el control de la configuración del proyecto durante todo su ciclo de vida.

Con Subversion se consigue tener organizados los diferentes ficheros que forman el sistema tanto código fuente como documentación y binarios. Cada vez que se modifica un fichero se registra el cambio, esto permite seguir toda la historia de los cambios introducido. También gestiona el uso concurrente por diferentes usuarios de los ficheros de la aplicación. Permite crear diferentes ramas separadas, podemos tener por ejemplo, la ramas de desarrollo y las ramas de versiones finales.

Trac estará integrada en subversion, lleva un control eficiente de cada uno de los errores o “bugs” que aparecen en el sistema. Permite tener diferentes versiones e hitos “milestones” así como clasificar los bugs por módulos y sistemas.

Como editor de texto para escritura del código fuente se usa el propio editor del sistema operativo.

Para la generación de la documentación se usa principalmente la suite ofimática [openOffice.org](#)

5.2 Desarrollo

Esta es la fase en la que se realiza el desarrollo del sistema.

Cada fichero de código fuente debe de comenzar por un comentario que constara de:

- Proyecto
- Nombre del fichero
- Explicación del cometido del módulo
- Módulos a los que llama
- Módulos que lo utilizan

- Autor del módulo
- Licencia del mismo, que en todos ellos debe ser GPL

Cada función o método debe comenzar con un comentario que constara de:

- Explicación de lo que hace la función
- Parámetros de entrada si tiene alguno
- Valores devueltos, si devuelve alguno

Cada función o método debe finalizar con otro comentario que indicara la finalización de la función y nombre de la misma.

Todo el código estará justificado, separando cada bloque por una indentación de 2 tabulaciones.

A continuación se describe cada una de las tareas ha realizar en detalle.

1. Preparación del entorno de desarrollo.

- Instalación y configuración del sistema operativo GNU Debian Linux
- Instalación del servidor Apache, PHP y MySQL.
- Instalación de utilidades (soapUI, SVN, trac, etc.)
- El esfuerzo es de 2 días/ hombre.

2. Desarrollo del módulo principal, la página de inicio de la aplicación Web (index.html)

- Conexión con los servicios Web del NIRIS, uso de un proxy para establecer llamadas a un servidor en otro dominio.
- Interpretar el mensaje en XML para obtener la información relevante
- Conectarse a “Google maps” usando su API versión 3.
- Presentar las trazas (según su latitud y longitud) en el mapa.
- Añadir marcadores y datos de la traza en la ventana de información.
- Implementar la selección de tipos de bloques de trazas así como el refresco de las mismas de forma automática.
- Realización de pruebas unitarias del módulo.
- El esfuerzo es de 6 días/ hombre.

3. Desarrollo del módulo proxy (phoxy.php)

- Llamar la los métodos de la librería 'class_http.php' para establecer las conexiones. Esta librería no es necesario desarrollarla se usa una ya existente, aunque se han tenido que realizar algunas modificaciones para adaptarla a nuestro sistema.
- Implementar la función de generación de logs.
- El esfuerzo es de 1 días/ hombre.

4. Desarrollo del módulo de ayuda (ayuda.html)

- Crear la página con la documentación necesaria para el manejo de la aplicación.
- Pruebas de integración de los módulos del subsistema de presentación de tráfico.
- El esfuerzo es de 1 días/ hombre.

5. Desarrollo del módulo de salvar situaciones en la base de datos (salva_situacion.php).

- Crear las tablas correspondientes en la base de datos para salvar las trazas y usuarios registrados en el sistema.
- Crear un módulo php que se conecte al servidor Web de NIRIS y que de forma periódica grabe todas las posiciones de las trazas en la base de datos.
- Script para ejecutar tareas de forma periódica.
- Realización de pruebas unitarias del módulo.
- El esfuerzo es de 2 días/ hombre.

6. Desarrollo del módulo de histórico de trazas (historico.php).

- Crear un sistema de autenticación, una pantalla en la que el usuario pueda introducir nombre de usuario y password
- Implementar en la página de históricos las funciones para leer la información de la base de datos y que la presente en pantalla.
- Implementar la opción que el usuario seleccione una determinada traza y se presente en una tabla la fecha, situación, rumbo y velocidad de dicha traza.
- Presentan las diferentes posiciones de una traza en un mapa pequeño de Google.
- Implementar la función de generación de logs.
- Realización de pruebas unitarias.
- El esfuerzo es de 6 días/ hombre.

7. Desarrollo del módulo de acceso a la base de datos (control_db.php).

- Centralizar todas las funciones de acceso a la base de datos en este módulo.
- Implementar la función de generación de logs.
- Pruebas de integración de los módulos del subsistema de análisis de históricos.
- El esfuerzo es de 1 día/ hombre.

8. Creación de estilos para las páginas Web

- Cada una de las páginas Web de la aplicación tendrán un estilo CSS de acuerdo con la interfaz de usuario descrita anteriormente.
- El esfuerzo es de 1 día/ hombre.

9. Realización de pruebas de integración de los módulos anteriores.

- Una vez codificados cada módulo y realizadas las pruebas unitarias de los mismos se procesa a las pruebas en la que intervienen todos los módulos de la aplicación para trabajar en cooperación
- El esfuerzo es de 1 días/ hombre.

5.3 Documentación

Cada uno de los documentos de la aplicación se realizaran conforme a las normas y estilos de documentos redactada anteriormente en el apartado de “Especificación de estándares, normas de diseño y construcción”.

La documentación que se proporciona es:

Documentación del proyecto.

- Diferentes documentos que contienen la documentación, diagramas, esquemas, etc., que cubren las diferentes fases del ciclo de vida del proyecto.
- Diapositivas de presentación general del proyecto.

Documentación del usuario.

- Contiene la explicación de como usar la aplicación.
- Un video con las diferentes funcionalidades de la misma.

Documentación técnica.

- Proporciona a los administradores de las pautas para realizar la instalación y mantenimiento del sistema.

6 Implantación

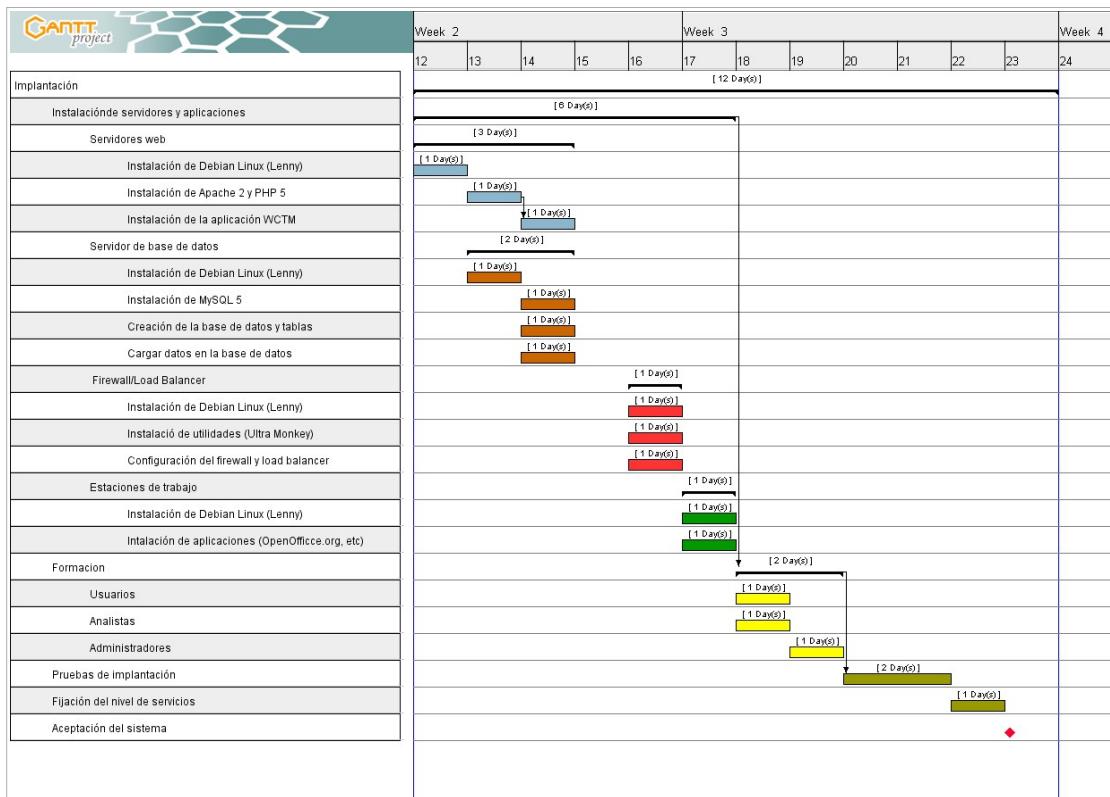
Una vez finalizado el desarrollo y las pruebas de los diferentes módulos que compone el sistema, la aplicación está lista para ser implantada en el que será su entorno de producción.

Los errores detectados en las fases anteriores se han debido subsanar, solo se hará la implantación si estos errores son considerados como no críticos o importantes para el funcionamiento del sistema.

Al ser un sistema de nueva creación el impacto a los usuarios es bajo. Durante la implantación los usuarios realizarán las funciones habituales de forma normal.

No existe problemática de migración de sistemas existentes o datos. Los sistemas ya implantados, AIS, NIRIS y Google maps no se verán afectados en absoluto por el nuevo sistema. El WCTM es solamente un consumidor de los servicios que ellos ofrecen. Usando las interfaces (Web services, API) ya establecidas en los mismos sin cambiar ninguna interfaz o función que realiza el mismo. Esa es una de las ventajas de la Arquitectura Orientada en Servicios, en inglés [SOA](#).

Aquí en este diagrama de Gantt vemos como esta planificada el proceso de implantación



6.1 Formación

Se les ofrecerá cursos de adiestramiento a los diferentes usuarios del sistema.

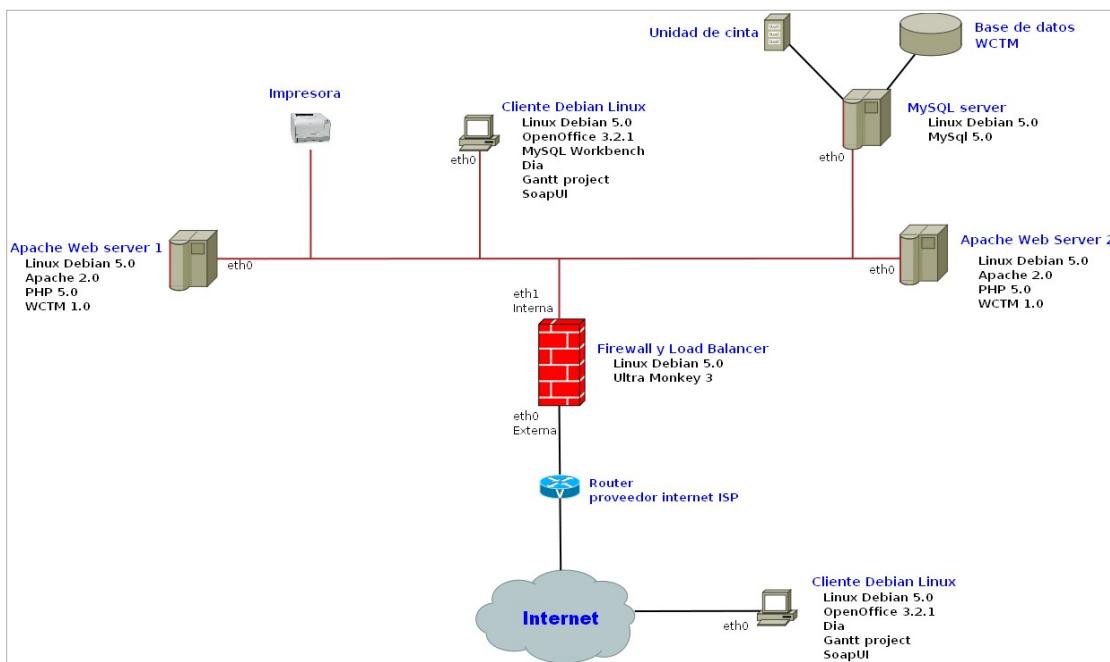
El contenido será diferente según la labor que ejerzan en el mismo.

- **Técnico.** A los administradores del sistema, se le instruirá en la administración de la aplicación. Gestión de la base de datos, copias de seguridad, auditorio de los ficheros logs.
Manejo de scripts, creación e eliminación de usuarios.
- **Nivel de Usuario.** A los diferentes usuarios (Control de Tráfico y Analistas) se les instruirá sobre el manejo eficiente de la aplicación.

6.2 *Implantación del sistema y pruebas*

En esta fase se instala en el entorno de producción los servidores y las aplicaciones que componen el sistema.

El siguiente diagrama describe la configuración final de cada una de los componentes del sistema en su entorno de producción.



En esta fase del proyecto ya se tienen identificados donde se van a colocar físicamente los servidores. Se hará en un rack dentro del cuarto de servidores. El sistema de cableado estructurado ya estará disponible. Los cables de red (UTP CAT-5) estarán marcados con una etiqueta con el puerto del SW en el que se conectan y nombre del servidor.

Como nuestra aplicación es accesible desde Internet es necesario que nuestro proveedor del servicio registre nuestro dominio en su DNS.

Antes de proceder con la instalación de los servidores, será necesario recoger información sobre la configuración de red que van a tener.

- Direcciones IP externas, las proporciona el proveedor de servicios de Internet
- Dirección IP interna de cada uno de los servicios y PC's
- Máscara de red tanto externa como interna.
- Pasarela por defecto
- DNS

Las cuentas de usuario que van a acceder a cada sistema.

Lista de la aplicaciones en cada uno y puerto de red usado por cada una.

Esta información nos servirá durante el proceso de instalación y para la configuración de los servidores y firewall.

6.2.1 Instalación de servidores y aplicaciones

Aquí procedemos a la instalación y configuración de las aplicaciones necesarias de los distintos servidores y estaciones de trabajo

Los detalles de instalación y configuración se pueden consultar en el manual del administrador.

Estos son las instalaciones que debemos realizar.

- **Servidores Web**

Se instalará dos servidores Web con la misma configuración en ambos.

Primero se realiza la instalación del sistema operativo Linux Debian 5.0, con las opciones por defecto sin instalar ninguna aplicación extra durante este proceso de instalación.

Una vez configurado con las opciones de red se irán instalando las aplicaciones que funcionan en estos servidores.

- ✓ PHP 5.0.
- ✓ Apache 2.0.
- ✓ Instalación de la aplicación WCTM 1.0.

- **Servidor de base de datos**

Primero se realiza la instalación del sistema operativo Linux Debian 5.0, también con las opciones por defecto. A continuación, después de la configuración, se instalan las aplicaciones propias de este servidor

- ✓ Instalación de MySQL 5.0
- ✓ Creación de la base de datos WCTM.
- ✓ Creación de usuarios en la base de datos.
- ✓ Carga de datos iniciales.

- **Firewall**

Instalamos el sistema operativo Linux Debian 5.0, en este caso se instalará el sistema operativo básico sin instalar ninguna aplicación que no sea estrictamente necesaria. Al ser este servidor el firewall, cualquier aplicación que se instale en él puede comprometer la seguridad del sistema entero. A continuación debemos configurar los parámetros de red para las dos interfaces.

- ✓ Instalación de Ultra Monkey 3.
- ✓ Configuración del Load balancer.
- ✓ Implementación de la reglas de filtrado de paquetes para solo permitir acceso http (puerto 80) a nuestra aplicación.

- **Estaciones de trabajo**

En cada una de las estaciones de trabajo se instalará el sistema operativo Linux Debian 5.0 y a continuación instalaremos las diferentes aplicaciones en cada una de ellos.

- ✓ OpenOffice.org.
- ✓ Dia.
- ✓ Gantt project.
- ✓ SoapUI.

En la estación de trabajo situada en nuestra LAN se recomienda la instalación del paquete MySQL [Workbench](#) para la administración del sistema de gestión de datos MySQL y de [Webmin](#) para la administración de apache.

La decisión de usar dos estaciones de trabajo es para tener una dentro de nuestra LAN con la que realizaremos labores de administración y otra fuera para realizar las pruebas de implantación y después facilitar el mantenimiento del sistema al disponer un ordenador en la parte de los clientes.

6.2.2 Pruebas de implantación

Con el sistema ya implantado en su entorno final de trabajo se procede a realizar las pruebas finales de implantación.

Estas son las últimas que realizamos en el sistema. Si las pruebas son satisfactorias y consideramos que los errores encontrados no son importantes se propondrá el sistema para su aceptación.

A continuación se enumeran cada una de estas pruebas:

- Acceso al sistema. Se probará el acceso tanto desde dentro de nuestra LAN como fuera, un PC conectado a Internet. Se comprobara que se resuelve el nombre de nuestro dominio para acceder a la aplicación.
- Se probará en la medida de lo posible con otros sistemas operativos que no sean Linux y con diferentes navegadores (Internet Explorer, Opera, Firefox).
- Aunque las funcionalidades ya se han probado en etapas anteriores, es conveniente de repetir esas pruebas aquí. Se hará un recorrido de las funcionalidades para comprobar que todo sigue funcionando adecuadamente en el entorno de producción.
- Para comprobar el comportamiento de refresco de trazas se seleccionaran todas y se establecerá un intervalo de refresco de 1 minuto. Durante el refresco se realizaran operaciones en el cliente para saber si funcionan correctamente y detectar posibles errores por falta de memoria o recursos.
- Durante la conexión de clientes se desconectara un servidor Web de la red para simular un fallo hardware. Se comprobará que el sistema sigue respondiendo adecuadamente.
- Con este servidor Web desconectado se iniciaran nuevas conexiones tanto a la página de presentación de tráfico como a la de análisis de históricos.
- Se configurará el módulo de salvar situaciones para que este registre las trazas en la base de datos con una frecuencia mucha más elevada que en situaciones normales. Será un test de estrés del sistema de gestión de base de datos ante una carga grande de trabajo.
- Se realizara un backup de la base de datos y se simulara un fallo en la base de datos para proceder a continuación al proceso de restauración de la misma.
- Se intentara acceder a la página de históricos sin realizar una sesión de autenticación previamente.
- Desde fuera del firewall se intentara acceder a otros servicios que no sean el http.

6.3 Nivel de servicios

Una vez implantado el sistema fijaremos un nivel de servicios mínimo.

- La aplicación debe estar accesible 24 horas. Para eso se ha implantado un sistema redundante con dos servidores Web
- El sistema está preparado para atender 2000 consultas concurrentes a la Web. Por ese motivo se ha implantado un sistema de “Load Balancer” y el hardware es de alto rendimiento.
- Con un nivel normal de peticiones, sin considerar los picos, las CPU's de cada servidor no pueden estar por encima del 25% de uso.
- Con un nivel normal de carga el consumo de memoria no puede ser mayor de 8 GB.
- Con un nivel normal de peticiones, el tráfico en la interfaz externa del Firewall no puede superar el 35% de utilización.
- El tamaño de la base de datos no puede nunca superar el 85% del espacio en disco del servidor.
- Ante un fallo en la base de datos el sistema se recuperara en menos de 4 horas.
- Cuando el mantenimiento entre en vigor, cualquier incidencia reportada se debe atender en menos de 6 horas.

6.4 Aceptación del sistema

Finalizada la implantación se presenta la documentación de la misma ante a los responsables de la aprobación del proyecto.

Se incluirá el resultado de las pruebas y el nivel de servicio acordado.

Si el proyecto es aprobado entra en producción y se comienza la prestación de servicios y el mantenimiento.

7 Mantenimiento

Inicialmente se ofrece un servicio de mantenimiento en concepto de garantía por un mes, el cual se podrá prolongar.

Este servicio incluye:

- Soporte técnico en línea a través de email para solucionar problemas en la aplicación y responder a consultas. En caso de que sea necesario el desplazamiento, este se facturara por separado.
- Actualizaciones, cuando haya alguna actualización de la aplicación, por mejora o error corregido se procederá a la actualizaciones.
- Gestión de errores reportados por los clientes, llevando una base de datos de los mismos usando “[trac](#)”.
- Información sobre cambios en las versiones de software usada por la aplicación. Linux, Apache, MySQL. Se le informara al cliente sobre la conveniencia de actualización de estas.

Para ofrecer estos servicios será necesario contratar a una persona a tiempo parcial durante el tiempo que dure el mantenimiento.

En el caso de encontrar errores o realizar nuevas implementaciones se contará en primer lugar con el equipo de desarrolladores del proyecto. Si estos no están disponibles se contactará alguna empresa del sector con conocimiento en aplicaciones Web en entornos libres.

8 Conclusiones

Durante todo el ciclo de vida de este proyecto se ha creado una aplicación Web y se ha demostrado que usando software libre ésta puede servir de ayuda al Control del Tráfico Marítimo.

Hemos estudiado varias alternativas, siendo la mas adecuado, según nuestro criterio, la elección de software libre sin usar la virtualización.

El campo de la virtualización está en expansión, cada vez se encuentran mas sistemas virtualizados. Desde mi punto de vista son muy útiles para sistemas compuestos por muchos servidores diferentes cuando es necesario crear un sistema de referencia, por ejemplo para un laboratorio de pruebas o cuando hay que impartir enseñanza sobre el sistema. Pero en producción, en sistemas que demandan alto rendimiento como sería nuestra Web de control de tráfico que tiene que procesar información en tiempo real, no los considero adecuados. Para obtener resultados de alto rendimiento con la virtualización el coste sería similar a optar por servidores físicos.

Para los costes del proyecto se han tenido en cuenta precisos reales, se ha consultado las páginas de las proveedores tanto de hardware como software. Esto se ha realizado para resaltar con datos reales las ventajas de usar componentes libres en un proyecto.

Desde el primer momento la idea era usar los componentes existentes que pudieran ser de utilidad para nuestro proyecto, el principio de “no reinventar la rueda” es un concepto cada vez mas empleado en la realización de proyectos. Normalmente la parte mas costosa de cualquier proyecto, no solo de software, es la mano de obra. Con el uso de componentes ya desarrollados se abarata considerablemente cualquier proyecto. Recuerdo cuando empecé con el mundo de la informática sobre la era de los 90, que para crear cualquier aplicación era necesario crearla desde cero. El coste en tiempo y personal era tremendo. Hoy en día crear aplicaciones es mucho mas rápido. Esto es especialmente verídico cuando usamos software libre.

Durante este proyecto se ha usado al máximo los recursos disponibles, se ha empleado un sistema de la OTAN, que es de los pocos que ofrecen información de forma libre a los usuarios de Internet. La elección de este sistema, el NIRIS, ha sido por introducir un aspecto novedoso en el proyecto y por el deseo de emplear una tecnología basada en servicios Web.

En cada momento se han pretendido dos objetivos primordiales, crear una aplicación Web interesante, al presentar información del tráfico marítimo y por el otro usar en este proyecto de fin de máster la máxima cantidad de conceptos tecnológicos relacionados con un proyecto Web.

Por eso se ha utilizado una arquitectura orientada a servicios, aunque seamos meros consumidores de servicios Web se han tratado los mismos.

Se han manejado datos en formato XML, este es el estándar que nos encontramos en la mayoría de las aplicaciones cuando necesitan intercambiar datos. Hemos visto que existen diferentes formas de llamar a un servicio Web, siendo la favorita hoy en día el método REST.

Otra aspecto interesante es la incorporación de mapas de Google en nuestra Web, realizando llamadas a su servicio a través de su API para conseguir diferentes operaciones en el mapa.

Hoy en día muchas páginas Web presentan algún tipo de mapa, por lo que esta inclusión en nuestra Web se ha considerado muy útil, algo que le da un valor añadido a la misma.

Hemos usado como lenguajes de programación, creo que los dos más usados en la actualidad para la realización de aplicaciones Web, JavaScript y PHP. No hemos enfrentado con la problemática de pasar datos de la parte en PHP, la que se ejecuta en el servidor con la parte JavaScript la que se ejecuta en el cliente.

Hemos visto como usando AJAX es una forma más eficiente de actualizar las páginas Web, es el sistema que usa los servicios de los mapas de Google.

También se ha afrontado la problemática de realizar llamadas desde un cliente de una página Web a servicios que están en otro servidor Web, la conocida como “same origin policy”, esto se ha resuelto después de mucha investigación gracias al uso de un “proxy”.

Para el diseño también se ha pretendido realizar algo que vaya más allá de un trabajo de fin de máster, se ha introducido la arquitectura “multi-tier”, de forma que en nuestra aplicación se separa lo que es la parte de acceso a la base de datos, la lógica de programación y de la parte de presentación que son las páginas Web que ve el usuario.

Como en un proyecto no suele faltar una base de datos se ha introducido la opción de salvar trazas en ella para poder recuperarlas posteriormente. Con esta inclusión se ha conseguido por una parte introducir en el trabajo de fin de máster el sistema de gestión de base de datos MySQL tan potente y popular junto con el modelo relacional y el lenguaje SQL. Pero por otra se le ha dotado al proyecto de una funcionalidad muy interesante la posibilidad de analizar, de saber, donde estaba un determinado barco en el pasado.

Junto con la parte de analizar los datos han surgido otro componente importante en las aplicaciones Web que es la autenticación de usuarios, se ha optado por una autenticación basada no en el servidor Web, lo que complicaría la configuración del servidor, sino en PHP usando sesiones para solo permitir acceder a una página al usuario que previamente ha establecido la sesión en otra (la página de autenticación). Para guardar los usuarios se ha optado de hacerlos en la propia base de datos de la aplicación, pero se ha establecido un sistema para que la contraseña esté encriptada en la base de datos.

Se ha explicado también como crear un sistema redundante y de alto rendimiento con la inclusión de dos servidores Web y un balanceador de carga.

Nos hemos apoyado en las posibilidades del sistema operativo, Debian Linux en este caso, para ejecutar tareas de forma periódica y en su facilidad para manejar scripts.

No hay que olvidar el aspecto que se le ha dado a las páginas Web, es una labor muchas veces artística, ya que muchas páginas Web parecen obras de arte, pero aquí se ha pretendido por lo menos usar estilos en las páginas y darle un aspecto similar a todas ellas.

Por supuesto que este proyecto se puede mejorar. Se podría por ejemplo implementar un sistema similar al actual NIRIS que se conectase a cada una de las redes AIS que existen en el mundo para proporcionar un control del tráfico marítimo a nivel mundial. Con ello se abordaría un factor que no está presente en este proyecto y es el de crear un servidor que proporcione servicios Web como puede ser usando, por ejemplo Java Servlets y Tomcat.

También se pueden implementar funcionalidades de búsqueda y filtro, para que por ejemplo se pueda localizar un barco por su nombre o filtrar las trazas que se encontraban en una

zona entre unas fechas dadas. O diferenciar los barcos que estén parados de aquellos que superen una velocidad dada usando diferentes iconos en el mapa.

Otro factor a mejorar sería un aprovechamiento de las posibilidades del servidor Web, Apache para control de acceso y para usar protocolo seguro (HTTPS) para aquellas páginas críticas de nuestro proyecto.

Con todo lo relatado anteriormente creo que se ha desarrollado un proyecto muy interesante, que combina muchas de las tecnologías actuales y que porque no, puede ser útil para el control del Tráfico Marítimo ya que lo hace universalmente accesible e incorpora la novedad de poder registrar la información de buques en la base de datos.

9 Referencias

Creative commons (<http://creativecommons.org/licenses/by-sa/2.5/es/>)

Debian GNU Linux: (<http://www.debian.org/>)

SOA: Service Oriented architecture (http://www.service-architecture.com/Web-services/articles/service-oriented_architecture_soa_definition.html)

Google maps: (<http://code.Google.com/apis/maps/documentation/javascript/>)

AIS: Automatic Identification System
(http://en.wikipedia.org/wiki/Automatic_Identification_System)

International Maritime Organization (<http://www.imo.org/Pages/home.aspx>)

NMEA: National Marine Electronics Association (<http://www.nmea.org/>)

NIRIS: Networked Interoperable Real-time Information Services,
(<http://www.npc.nato.int/htm/niris.htm>)

Web Services: (http://www.service-architecture.com/Web-services/articles/Web_services_definition.html)

AJAX: Asynchronous JavaScript and XML (http://www.w3schools.com/Ajax/ajax_intro.asp)

Multi-Tier: (http://en.wikipedia.org/wiki/Multitier_architecture)

GPS: Global Positioning System (http://en.wikipedia.org/wiki/Global_Positioning_System)

VHF: Very high frequency (http://en.wikipedia.org/wiki/Very_high_frequency)

TIES: TITO Information Exchange Services (<http://niris.nc3a.nato.int/>).

VTS: Vessel Traffic Service
(http://www.puertos.es/ayudas_navegacion/VTS_Vessel_Traffic_Service.html)

OTAN: Organización del Tratado del Atlántico Norte
(<http://www.nato.int/cps/en/natolive/index.htm>)

EUROCONTROL: (http://www.eurocontrol.int/corporate/public/subsite_homepage/index.html)

VirtualBox: (<http://www.virtualbox.org>).

W3C: World Wide Web Consortium (<http://www.w3.org/>)

Apache: (<http://www.apache.org/>)

HTTP: Hypertext Transfer Protocol (<http://www.w3.org/Protocols/>)

HTML: HyperText Markup Language (<http://www.w3.org/TR/html401/>)

XML: Extensible Markup Language (<http://www.w3.org/TR/REC-xml/>)

CSS: Cascading Style Sheets (<http://www.w3.org/TR/CSS2/>).

VMware (<http://www.vmware.com>)

STDMA: Self-Organized Time Division Multiple Access
(<http://www.icao.int/anb/panels/acp/meetings/amcp4/item-5d.pdf>)

TDMA: Time division multiple access
(http://en.wikipedia.org/wiki/Time_division_multiple_access)

UTC: Coordinated Universal Time (http://en.wikipedia.org/wiki/Coordinated_Universal_Time)

GPL: GNU General Public License (<http://www.gnu.org/copyleft/gpl.html>)

Encuesta comparativa de uso de servidores Web:
(<http://news.netcraft.com/archives/2010/11/05/november-2010-Web-server-survey.html>)

PHP: PHP: Hypertext Preprocessor (<http://www.php.net/>)

JavaScript: (<http://en.wikipedia.org/wiki/JavaScript>)

SOAP: Simple Object Access Protocol (<http://en.wikipedia.org/wiki/SOAP>)

REST: Representational State Transfer
(http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)

Ultra Monkey: (<http://www.ultramonkey.org/3/lvs.html>)

Load Balancer: Balanceo de carga (http://kb.linuxvirtualserver.org/wiki/Load_balancer)

LVS: Linux Virtual Server (<http://www.linuxvirtualserver.org/Documents.html>)

Idirectord: (<http://www.ultramonkey.org/3/Idirectord.html>)

MySQL (<http://dev.mysql.com/doc/>)

iptables: (<http://es.wikipedia.org/wiki/Netfilter/iptables>)

Multi-tier: Multi capas (http://en.wikipedia.org/wiki/Multitier_architecture)

KML: Keyhole Markup Language (http://en.wikipedia.org/wiki/Keyhole_Markup_Language)

JSON: JavaScript Object Notation (<http://www.json.org/>)

XDR: Cross-domain Request (<http://msdn.microsoft.com/en-us/library/dd573303%28v=vs.85%29.aspx>)

Modelo Relacional: (http://es.wikipedia.org/wiki/Modelo_relacional)

Modelo Entidad-Relación (http://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n)

UML: Unified Modeling Language (<http://www.omg.org/uml/>).

subversion: Control de versiones (<http://subversion.tigris.org/>)

trac: seguimiento de errores (<http://trac.edgewall.org/>)

UPS: Uninterruptible power supply (http://en.wikipedia.org/wiki/Uninterruptible_power_supply)

openOffice.org: Suite ofimática (<http://www.openoffice.org/>)

Gantt project (<http://www.ganttproject.biz/>)

Dia: (<http://live.gnome.org/Dia>)

soapUI: (<http://www.soapui.org/>)

MySQL Workbench: (<http://wb.mysql.com/>)

webmin: (<http://www.webmin.com/>)

Apuntes de Proyecto Web de la UOC (<http://www.uoc.edu>)

Puertos del estado (<http://www.puertos.es/>)

Wikipedia (<http://es.wikipedia.org>)