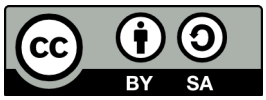

# ApplicationMap

Administración web y comercio electrónico en entornos de software libre

Autor: Alex Muller

Consultor: Samuel Lacarta

Fecha: 06/01/12



## Licencia

# Resumen del proyecto

Este proyecto tiene como objetivo desarrollar las herramientas necesarias para poder crear un mapa conceptual de las aplicaciones de una organización, representar gráficamente este mapa y controlar el estado de cada aplicación.

En concreto, se trata de desarrollar un formato XML que permita identificar y describir una aplicación, detallar con qué tecnología está desarrollada, qué componentes utiliza, especificar las interacciones o dependencias con otros sistemas, etc.

A partir de estos “carnets de identidad” de una aplicación, se podrá crear un inventario de aplicaciones, un mapa visual que facilitará la comprensión de la estructura IT de la organización y monitorizar el estado de las mismas.

Además facilitará la puesta en producción de nuevas aplicaciones ya que permitirá comprobar que el entorno donde se vaya a instalar la aplicación cubre sus necesidades en cuanto a tecnología instalada, conexiones con otros sistemas, etc.

Dado el entorno multi-plataforma de la organización donde se desarrolla el proyecto (sistemas operativos Microsoft Windows y GNU/Linux), la solución desarrollada deberá ser válida para ambas plataformas.

## Agradecimientos

Agradecer a Samuel Lacarta su tiempo, dedicación y aportaciones. A David Parra por su ayuda, ideas y propuestas de mejoras.

# Índice de contenido

# 1 Presentación del proyecto

## 1.1 Introducción

La empresa V. Airlines, situada en el sector del transporte, apuesta decididamente por las nuevas tecnologías, tanto para su gestión interna como para sus canales de venta. Tanto es así que mantiene su propio departamento de informática que se encarga de la administración de sistemas y del desarrollo y mantenimiento de un gran número de aplicaciones y servicios. Actualmente cuenta con varios sitios web, servicios web para al integración de terceros y de un elevado número de aplicaciones propias o de terceros; todo ellos hospedados en varias docenas de servidores repartidos en tres CPDs (Centros de Procesamiento de Datos).

Todo este sistema informático ha ido creciendo a lo largo de varios años, lo cual supone también un esfuerzo cada vez más grande a la hora de mantener, actualizar, instalar o simplemente monitorizar las aplicaciones o servicios.

En concreto, se han identificado ciertos problemas a la hora de localizar aplicaciones, es decir, determinar rápidamente en qué servidor se encuentra hospedado. También resulta muy difícil identificar su relación con el resto del sistema, es decir, determinar qué otros servicios acceden a la aplicación en cuestión, o al revés, qué servicios necesita una aplicación para funcionar correctamente. Estas cuestiones son de suma importancia a la hora de resolver una incidencia, actualizar componentes o cuando se tenga que migrar aplicaciones de un servidor a otra.

También se ha identificado la necesidad de homogeneizar la información o requisitos necesarios para poner en producción una aplicación. A día de hoy, los jefes de proyecto entregan el código binario de las aplicaciones al departamento de Explotación de Sistemas para que proceda a su instalación. Esta instalación resulta difícil cuando no se especifica claramente qué requisitos debe cumplir el servidor o entorno en el cual se va a instalar la aplicación.

Mediante este proyecto se pretende facilitar todas estas tareas.

## 1.2 Objetivos

Se ha detectado la necesidad de poder controlar dichas aplicaciones y servicios. En concreto, se necesita:

- poder localizar rápidamente una aplicación o servicio
- poder identificar la relación o dependencia de una aplicación o servicio con el resto de los componentes del sistema
- poder determinar los requisitos o necesidades de la aplicación o servicio necesarios para su correcto funcionamiento

Para cubrir estas necesidades, se plantea:

- estandarizar una especie de Carnet de Indentidad para las aplicaciones
- desarrollar una aplicación que, en base a ese carnet, pinte gráficamente el mapa de aplicaciones con sus detalles
- desplegar la aplicación en los entornos productivos de la organización

# 2 Estudio de viabilidad

## 2.1 Alcance del proyecto

Para alcanzar los objetivos mencionados anteriormente, se ha decidido impulsar el desarrollo de un sistema compuesto por:

### 2.1.1 “Carnet de identidad” de una aplicación

Este carnet de identidad, a partir de ahora *IOM,* deberá listar los parámetros o requisitos que condicionen la integración de la aplicación en el resto del sistema, por ejemplo, versiones específicas de determinadas librerías o *frameworks*, acceso a servicios externos (FTP, Webservices, colas, etc.) además de las propias características de la aplicación (descripción, versión, autor, fecha de puesta en producción, etc.).

Este carnet servirá para poder desarrollar una aplicación que permita acceder rápidamente a la información contenida en el mismo y crear un mapa visual de todas las aplicaciones instaladas en el servidor.

Además, deberá servir como requisito para poner en producción nuevas aplicaciones. La identificación de los requisitos de las nuevas aplicaciones evitará incidencias durante la puesta en producción originadas por incompatibilidades en la plataforma, problemas de acceso a servicios externos, falta de librerías, etc.

### 2.1.2 Desarrollar una aplicación multiplataforma

La información recogida en el IOM deberá poder ser consultada de manera fácil, segura y en remoto, por lo tanto, se propone el desarrollo de una aplicación web que satisfaga estos requisitos.

La aplicación deberá mostrar las aplicaciones instaladas en el servidor donde se ejecuta, las características de las mismas, las relaciones entre dichas aplicaciones y su estado. Para facilitar la comprensión de esta información, se propone mostrarla en forma de mapa.

Dado que la información mostrada en la aplicación puede ser sensible y confidencial, deberá integrarse en el sistema de autenticación de usuarios del a organización.

La aplicación deberá ser accesibles desde cualquier navegador e, idealmente, desde dispositivos móviles.

Es importante destacar que el conjunto de servidores de la empresa está formado tanto por servidores con el sistema operativo Microsoft Windows como con GNU/Linux. Esto implica que la aplicación a desarrollar deberá poder ejecutarse en ambas entornos, idealmente minimizando o eliminando la necesidad de adaptaciones entre los entornos.

## 2.2 Estudio de la situación actual

A día de hoy la empresa cuenta con:

- 16 servidores web con GNU/Linux para el portal de ventas online
- 4 servidores con *webservices* para el canal de venta alternativo
- 6 servidores para aplicaciones corporativas y servicios internos (Subversion, TFS, wikis, etc.)
- 3 servidores de bases de datos
- 2 servidores para procesos periódicos (“procesos nocturnos”)
- alrededor de 8 aplicaciones comerciales de nicho, desarrolladas por empresas externas y configuradas según las necesidades de la organización
- alrededor de 20 aplicaciones desarrolladas internamente, generalmente aplicaciones web (*Web-UI*)
- alrededor de 20 procesos periódicos encargados de reunir, calcular o tratar información (*data-mining*)
- alrededor de 10 bases de datos *MySQL* y 20 base de datos *Microsoft SQL Server*.

En cuanto al control o inventario de aplicaciones, se mantiene una página en la *wiki* del departamento con las características, funcionalidades, personas responsables, etc. de

cada aplicación. El problema que se ha detectado es que dicha página a veces contiene información desfasada o errónea.

El procedimiento de puesta en producción actual consiste en indicar las fechas, servidor destino, necesidades en cuanto a conexión, etc. en una hoja de cálculo. Esta hoja de cálculo la elabora el jefe de proyecto y se envía al responsable del departamento de Explotación de sistemas para planificar la puesta en producción.

## 2.3 Definición de los requisitos del sistema

A continuación se detallan los principales objetivos del proyecto y su correspondiente importancia.

| Tipo | Descripción | Importancia relativa |
|---|---|---|
| Arquitectura | | |
| Arquitectura | La aplicación deberá poder ejecutarse tanto en Microsoft Windows como GNU/Linux | 100 |
| Seguridad | La información mostrada en la aplicación deberá ser accesible únicamente a personal autorizada. | 100 |
| Seguridad | La aplicación no deberá afectar al funcionamiento de las otra aplicaciones o servicios. | 100 |
| Económicos | Evitar el uso de librerías o servicios de pago | 95 |
| Arquitectura | El fichero IOM deberá ser flexible para poder ampliar en el futuro sin necesidad de actualizar los ficheros ya existentes | 90 |
| Operativa | La información mostrada deberá ser obtenida de forma autónoma por la propia aplicación, es decir, no será necesario ningún tipo de mantenimiento | 90 |

| Tipo | Descripción | Importancia relativa |
|---|---|---|
| | ni introducción de información | |
| Operativa | Para el mapa de aplicaciones deberá evitarse el uso de Flash o Silverlight | 90 |
| Operativa | Incluir un buscador y filtro para facilitar la obtención de información (número elevado de aplicaciones) | 90 |
| Operativa | Elaborar un mecanismo de validación del fichero IOM (por ejemplo XSD) | 80 |

## 2.4 Estudio y valoración de alternativas

El estudio de viabilidad tiene como objetivo presentar la posibles opciones soluciones y elegir aquella que mejor cubra las necesidades planteadas.

A continuación se presenta las diferentes alternativas, la mayoría basadas en software libre o servicios gratuitos. Hay que tener en cuenta que un factor muy importante es la integración de la aplicación a desarrollar en el entorno ya existente. Uno de los aspectos más importantes es el conocimiento técnico de la plantilla de programadores, analistas, administradores y jefes de proyecto. El objetivo general es mantener todo el sistema informático de la empresa lo más homogéneo posible.

### 2.4.1 Fichero IOM

Para el formato del carnet de identidad se plantean varias soluciones:

#### 2.4.1.1 Formato XML

XML, siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML, por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades, de ahí que se le denomine metalenguaje.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.[1]

Las principales ventajas son sus posibilidades de extensión manteniendo compatibilidad, fácil lectura y amplia aceptación.

#### 2.4.1.2 Formato JSON

JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando el procedimiento eval(), lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.[2]

La principales ventaja de JSON es su facilidad de uso en la aplicación WebUI a desarrollar. Como inconveniente debemos mencionar que es muy difícil elaborar un documento JSON a mano.

1 http://es.wikipedia.org/wiki/Xml

2 http://es.wikipedia.org/wiki/Json

#### 2.4.1.3 CSV

Los ficheros CSV (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal: España, Francia, Italia...) y las filas por saltos de línea. Los campos que contengan una coma, un salto de línea o una comilla doble deben ser encerrados entre comillas dobles.

El formato CSV es muy sencillo y no indica un juego de caracteres concreto, ni cómo van situados los bytes, ni el formato para el salto de línea. Estos puntos deben indicarse muchas veces al abrir el fichero, por ejemplo, con una hoja de cálculo.[3]

Los documentos CSV son muy fáciles de elaborar pero es más complicado de mantener y extender en el futuro sin afectar al formato anterior.

Los tres formatos pueden ser consumidos y generados fácilmente con cualquier tecnología, pero el formato XML es más comprensible y presenta menos complicaciones para el usuario humano a la hora de editar y leerlo. Como desventaja cabe mencionar que el tamaño resultante del fichero es más grande que un fichero JSON, pero no supondrá ningún problema ya que la información contenido será relativamente poca.

### 2.4.2 Aplicación WebUI

Para el desarrollo de la aplicación de consulta, se presentan varias alternativas teniendo en cuenta que el requisito más determinante es la necesidad de poder ejecutarse tanto en servidores Windows como GNU/Linux.

A continuación se presentan las tecnologías estudiadas y sus ventajas e inconvenientes relacionadas con el entorno de la empresa.

#### 2.4.2.1 JSP: Java Server Pages

Es una tecnología Java desarrollada por la compañía Sun Microsystems, que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

3 http://es.wikipedia.org/wiki/CSV

El funcionamiento general de la tecnología JSP es que el Servidor de Aplicaciones interpreta el código contenido en la página JSP para construir el código Java del servlet a generar. Este servlet será el que genere el documento (típicamente HTML) que se presentará en la pantalla del Navegador del usuario.

El **rendimiento** de una página JSP es el mismo que tendría el servlet equivalente, ya que el código es compilado como cualquier otra clase Java. A su vez, la máquina virtual compilará dinámicamente a código de máquina las partes de la aplicación que lo requieran. Esto hace que JSP tenga un buen desempeño y sea **más eficiente que** otras tecnologías web que ejecutan el código de una manera puramente interpretada como **PHP**.

**Ventajas**

- el lenguaje Java es un **lenguaje de propósito general** que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos de una manera prolija. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP.
- hereda la **portabilidad** de Java, y es posible ejecutar las aplicaciones en **múltiples plataformas** sin cambios. Es común incluso que los desarrolladores trabajen en una plataforma y que la aplicación termine siendo ejecutada en otra. [4]

**Desventajas**

- la organización ha optado por no utilizar esta tecnología de momento

#### 2.4.2.2 PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Se usa principalmente para la interpretación del lado del servidor (server-side scripting).

Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno gracias a su licencia libre. El lenguaje

4 http://es.wikipedia.org/wiki/JavaServer_Pages

PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como Unix (y de ese tipo, como Linux o Mac OS X) y Microsoft Windows, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.[5]

**Ventajas**

- Es un lenguaje multiplataforma.
- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- Permite desarrollar utilizando MVC
- Se trata de una tecnología utilizada y conocida por el personal de la empresa

5 http://es.wikipedia.org/wiki/Php

**Desventajas**

- Al tratarse de código interpretado, su rendimiento puede ser inferior a otras tecnologías que utilizan código compilado como JSP o .NET. Existe la posibilidad de compilar el código mediante herramientas con un coste y licencia no libre.

#### 2.4.2.3 ASP .NET

ASP.NET es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

**Ventajas**

- Código compilado lo que repercute positivamente en la velocidad de ejecución
- Potente IDE para desarrollo, ya presente en la organización
- Cuenta ya con un gran número de componentes
- Es un tecnología utilizada y conocida en la organización

**Desventajas**

- ASP.NET sólo funciona sobre el servidor de Microsoft IIS, lo que supone una desventaja respecto a otros lenguajes del lado de servidor, ejecutables sobre otros servidores más populares como Apache.

#### 2.4.2.4 Mono

Mono es el nombre de un proyecto de código abierto iniciado por Ximian y actualmente impulsado por Novell para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET según lo especificado por el ECMA.

Es un proyecto independiente de la plataforma. Actualmente Mono funciona en GNU/Linux, OpenBSD, FreeBSD, UNIX, Mac OS X, Solaris y plataformas Windows.

Presenta las principales ventajas de ASP .NET con al adyacente de poder ejecutarse en múltiples plataformas.

### 2.4.3 Mapa visual

El mapa de aplicaciones deberá ser muy visual, mostrando las relaciones entre aplicaciones y servicios. El cliente desea evitar el uso de Flash o Silverlight, proponiendo el empleo de javascript. Por lo tanto, se estudian varias tecnologías que permitan el desarrollo de dicho mapa:

#### 2.4.3.1 Raphaël—JavaScript Library

Raphaël es una pequeña librería JavaScript que simplifica el trabajo con gráficos vectoriales en la web. Permite crear gráficos e imágenes, rotar y ampliar vistas, entre otras cosas.

Raphaël utiliza la recomendación del W3C para SVG y VML como base para la creación de gráficos. Esto significa que cada objeto gráfico se crea también como un objeto DOM, por lo que se puede asociar controladores de eventos de JavaScript o modificar más adelante. El objetivo de Raphaël es proporcionar un adaptador para realizar el dibujo de gráficos vectoriales compatibles con múltiples navegadores de manera fácil.

Raphaël actualmente soporta Firefox 3.0 +, Safari 3.0 +, Chrome 5.0 +, Opera 9.5 + e Internet Explorer 6.0 o superior.[6]

Raphaël tiene una licencia MIT.

#### 2.4.3.2 Google Maps

Google Maps es el nombre de un servicio gratuito de Google. Es un servidor de aplicaciones de mapas en la Web. Ofrece imágenes de mapas desplazables, así como fotos satelitales del mundo entero e incluso la ruta entre diferentes ubicaciones o imágenes a pie de calle Street View.

En junio del 2005 Google lanzó su API de Google Maps, haciendo oficialmente

6 http://raphaeljs.com/

modificable casi cualquier aspecto de la interfaz original. Con la contraseña oficial de desarrollador, la API es libre de uso para cualquier sitio web.[7]

La API de Google Maps permite el uso de mapas propios, mediante un servidor de imágenes también conocido como *tile server*. Asimismo permite dibujar una serie de iconos y lineas encima del mapa.

Las ventajas que ofrece este servicio son una librería javascript completa y probada, con el desarrollo y mantenimiento futuro prácticamente asegurado. Como inconvenientes cabe destacar la imposibilidad de adaptar esa librería a las necesidades particulares del proyecto, un abanico de elementos gráficos (líneas, símbolos) muy limitado y la complejidad de implantación del servidor de imágenes.

#### 2.4.3.3 jsDraw2D : 2D Graphics Library for JavaScript

JsDraw2D es una biblioteca pura JavaScript para dibujar gráficos 2D en páginas web dentro del navegador web sin necesidad de utilizar SVG o VML. Los desarrolladores de JavaScript, los desarrolladores web y webmasters pueden tomar ventaja de la biblioteca para agregar funcionalidad de dibujo de gráficos en sus aplicaciones o sitios web utilizando la biblioteca. La biblioteca está completamente escrito en JavaScript y no necesita ningún software o plug-in adicional para ejecutarse. El código fuente JavaScript de la biblioteca es libre y gratuito bajo la licencia LGPL.[8]

La principal ventaja de esta librería es su flexibilidad y potencia. Como inconvenientes destacan su complejidad y una curva de aprendizaje medio elevada. Su potencia la hace adecuada para crear gráficas muy complejas.

## 2.5 Análisis de riesgos

Se han detectado tres riesgos que podrían afectar a la planificación temporal del proyecto:

1. **Funcionamiento del mapa visual:** dado el carácter de la aplicación a desarrollar

7 http://es.wikipedia.org/wiki/Google_Maps

8 http://jsdraw2d.jsfiction.com/

(WebUI) y el requisito de no utilizar Flash ni Silverlight, el desarrollo del mapa visual puede resultar técnicamente complicado. Es necesario identificar posibles alternativas a las tecnología anteriores y su posibilidad es de integración y compatibilidad con las plataformas que accederán a la aplicación, así como su coste de desarrollo.

2. **Implementación de la funcionalidad “Validar requisitos”:** esta funcionalidad puede resultar muy complicada técnicamente ya que implica acceder en un tiempo razonable a los distintos servicios a las que necesita acceder la aplicación a validar procurando no modificar ningún dato ni alterar el estado de dichos servicios. Debe elaborarse un protocolo para acceder correctamente a los servicios y con los permisos adecuados.
3. **Instalación y configuración del módulo mod_mono para Apache:** es necesario planificar con el departamento de Sistema la instalación de este módulo, realizar pruebas de funcionamiento. En cualquier caso está fuera del alcance del proyecto ya que se entiende que es un módulo bien soportado y el personal de la organización es quien debe realizar estas tareas.

## 2.6 Soluciones escogidas

Tras presentar las soluciones escogidas y estudiar sus respectivas ventajas e inconvenientes, se decide utilizar la tecnología Mono ya que satisface el requisito de poder ejecutarse tanto en los servidores GNU/Linux como en los Windows, además de ser compatible con la tecnología ASP .NET, apuesta de futuro de la organización, y conocida por los desarrolladores de la empresa.

Para el fichero IOM se optará por XML, dada su facilidad de edición humana y de procesado.

En cuanto al mapa de aplicaciones, se escoge la librería Raphaël ya que reúne todas las características exigidas y es relativamente fácil de utilizar.

Además se decide utilizar las siguientes tecnologías:

- **jQuery:** jQuery es una biblioteca de JavaScript, creada inicialmente por John

Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.[9]

- **jQuery UI:** es una biblioteca de componentes para el framework jQuery que le añaden un conjunto de plug-ins, widgets y efectos visuales para la creación de aplicaciones web. Cada componente o módulo se desarrolla de acuerdo a la filosofía de jQuery

El estudio de estas tecnologías no se ha incluido en este proyecto ya que han sido impuestas por parte del cliente, además de haberlas utilizado en otros proyectos.

### 2.6.1 Resumen

Las **herramientas escogidas** son las siguientes:

- Framework **Mono** para el desarrollo de la aplicación
- Formato **XML** para el IOM
- Librería **Raphaël** para el mapa de aplicaciones
- Framework **jQuery** y **jQuery UI** para la interfaz de usuario

## 2.7 Planificación temporal del proyecto

El proyecto está dividido en 8 fases:

| # | Tarea | Inicio | Fin | Duración |
|---|---|---|---|---|
| 1 | Estudio viabilidad | 17/10/11 | 18/10/11 | 2 días |
| 2 | Definición del sistema | 19/10/11 | 25/10/11 | 5 días |
| 3 | Diseño | 26/10/11 | 03/11/11 | 7 días |
| 4 | Desarrollo | 04/11/11 | 22/12/11 | 32 días |
| 5 | UAT | 20/12/11 | 20/12/11 | 1 día |
| 6 | Bug fixing | 21/12/11 | 21/12/11 | 1 día |
| 7 | Aceptación UAT | 22/12/11 | 22/12/11 | 1 día |
| 8 | Implantación | 23/12/11 | 23/12/11 | 1 día |

9 http://es.wikipedia.org/wiki/Jquery

La dedicación diaria se estima en 4 horas.

## 2.8 Coste económico

Se estima un coste medio por hora de 40 EUR, por lo tanto, el coste total del proyecto asciende a 8.000€ y se ajusta a los requisitos económicos del proyecto.

# 3 Análisis del sistema

## 3.1 Definición del sistema

El fichero IOM y la aplicación deberán cumplir los siguientes requisitos:

### 3.1.1 Requisitos técnicos

- La aplicación deberá ser compatible con los principales navegadores (Firefox, IE8+, Google Chrome).
- La aplicación deberá poder ejecutarse tanto en Microsoft Windows como GNU/Linux
- El fichero IOM deberá ser fácilmente comprensible (evitar CSV) y poder editarse a mano
- La información mostrada en la aplicación deberá ser accesible únicamente a personal autorizada.
- La aplicación no deberá afectar al funcionamiento de las otra aplicaciones o servicios.

### 3.1.2 Requisitos operativos

- La información mostrada deberá ser obtenida de forma autónoma por la propia aplicación, es decir, no será necesario ningún tipo de mantenimiento ni introducción de información
- Para el mapa de aplicaciones deberá evitarse el uso de Flash o Silverlight
- Incluir un buscador y filtro para facilitar la obtención de información (número elevado de aplicaciones)
- Incluir un buscador y filtro para facilitar la obtención de información (número elevado de aplicaciones)
- Elaborar un mecanismo de validación del fichero IOM (por ejemplo XSD)

### 3.1.3 Requisitos económicos

- Evitar el uso de librerías o servicios de pago
- El presupuesto total del proyecto es de 10.000€
- El plazo máximo de entrega del proyecto de 2 meses naturales

### 3.1.4 Entorno tecnológico del sistema

Las plataformas tecnológicas del proyecto estará formado por los siguientes sistemas y herramientas:

- Servidores Microsoft Windows
  - Sistema operativo Microsoft Windows Server 2008 R2
  - Framework .NET 4
  - Servidor Web IIS 7
  - Base de datos Microsoft SQL Server 2008 R2
- Servidor GNU/Linux
  - Sistema operativo GNU/Linux CentOS 6
  - Base de datos MySQL 5
  - Framework Mono 2.6.7
  - Servidor Web Apache 2

## 3.2 Establecimiento de requisitos

### 3.2.1 Fichero IOM

El principal requisito del fichero IOM es poder contener la información necesaria de las aplicaciones existentes, además de poder ser extendido en el futuro.
Se identifica la necesidad de contener al menos la siguiente información:

- Identificador único de la aplicación: Id de la aplicación

(V.NombreAplicacion.TipoAplicacion)

- Tipo: tipo de aplicación (Web Service, WebUI, etc.)
- Versión
- Descripción corta: breve descripción de la funcionalidad de la aplicación
- Aplicaciones relacionadas
- Descripción extendida: Se debe poner la mayor cantidad de información funcional posible sobre la aplicación
- Servidores donde se encuentra instalada
- Base de datos: base(s) de datos que usa esta aplicación
  - Identificador de la base de datos: Nombre de la BD
  - Lectura sí/no
  - Escritura sí/no
  - Propietario sí/no
  - Descripción / para qué necesita acceder a esta base de datos
  - Tablas
    - Identificador
    - Lectura sí/no
    - Escritura sí/no
    - Descripción/para qué se utiliza esta tabla
  - Procedimientos almacenados
    - Identificador
    - Descripción/para qué se utiliza este procedimiento almacenado
  - Funciones
    - Identificador
    - Descripción/para qué se utiliza esta función
- Servidores FTP
  - Identificador del servidor FTP
  - Descripción/para qué se utiliza este servicio FTP
- Colas

    - Identificador de la cola
    - Descripción/para qué se utiliza esta cola
- **Servicios Navitaire[10] a los que accede la aplicación**
    - **Reflection sí/no:** indica si la aplicación accede a Reflection
    - **Booking API:** indica si la aplicación accede a la BookingApi
    - **Checkin API:** indica si la aplicación accede a la CheckinApi
- Tareas automáticas que se deben crear al desplegar esta aplicación
    - Web Service Task:
        - Nombre: nombre del Job en el scheduler
        - **Id de la aplicación:** (V.Aplicacion.WebService)
        - Nombre del método que se desea ejecutar
        - Expresión cron que indica cuándo se debe ejecutar la aplicación
    - Stored Procedure Task
        - Nombre del job en el scheduler
        - Id de la base de datos donde se encuentra el procedimiento almacenado
        - Nombre del stored procedure
        - Expresión cron que indica cuándo se debe ejecutar el stored procedure

Un posible formato podría ser el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<component id="Id de la aplicación(V.NombreAplicacion.TipoAplicacion)" type="Tipo de aplicación(Web Service, WebUI, etc.)" version="versión" shortDescription="Breve descripción de la funcionalidad de la aplicación" svnRoot="Directorio raíz en el repositorio" relatedApplications="Aplicaciones relacionadas separadas por punto y coma">
  <description><![CDATA[Se debe poner la mayor cantidad de información funcional posible sobre la aplicación]]></description>

  <productionServers>
    <!--Servidor(es) de producción donde se desplegará esta aplicación-->
    <server name='Nombre del servidor de producción'/>
  </productionServers>

  <databases>
    <!--Base(s) de datos que usa esta aplicación-->
```

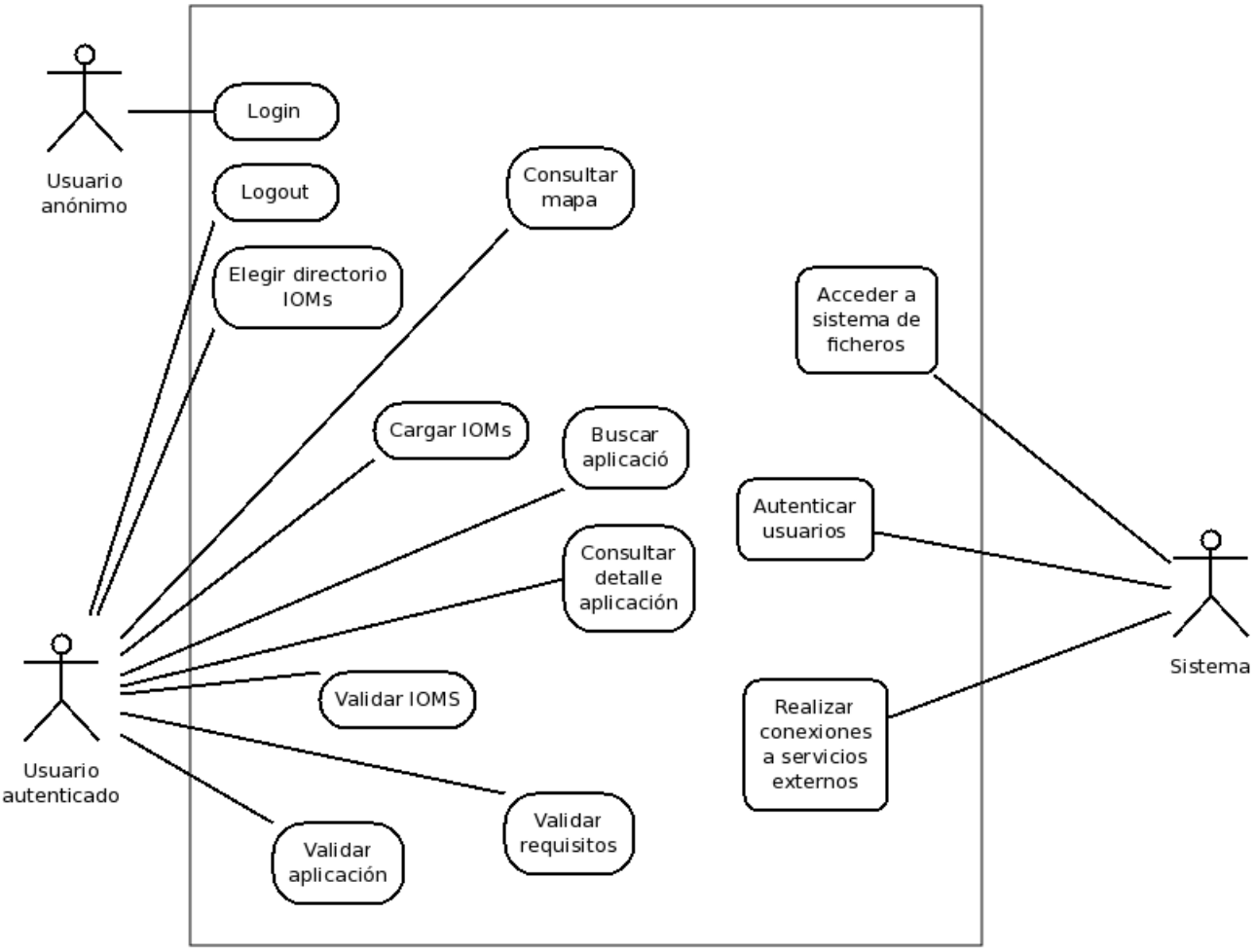
10 Proveedor externo

```
    <database id="Nombre de la BD" read="true" write="false" owner="false" description="Puede utilizarse el
atributo description o una etiqueta description para explicar para qué se utiliza esta base de datos en esta
aplicación.">

      <tables>
        <table id="TableName" read="true" write="false" description="Descripción de para qué se utiliza esta
tabla"/>
      </tables>

      <sps><!--Stored procedure(s) que utiliza esta aplicación>-->
        <sp id="SPROCName" description="Descripción de para qué se utiliza este procedimiento almacenado" />
      </sps>

      <functions><!--Funciones que utiliza esta aplicación>-->
        <function id="FunctionName" description="Descripción de para qué se utiliza esta función" />
      </functions>

    </database>
  </databases>

  <webServices>
    <webService id="Id de la aplicación(V.Aplicacion.WebService)">
      <description><![CDATA[Explicación de para qué se utiliza este servicio web en esta
aplicación]]></description>
    </webService>
  </webServices>

  <ftps>
    <ftp id="Nombre del Ftp">
      <description><![CDATA[Explicación de para qué se utiliza este FTP en esta aplicación]]></description>
    </ftp>
  </ftps>

  <queues>
    <queue id="Nombre de la cola">
      <description><![CDATA[Explicación de para qué se utiliza esta cola en esta aplicación]]></description>
    </queue>
  </queues>

  <navitaireServices>
        <!--Servicios de Navitaire a los que accede esta aplicación-->
    <reflection>true</reflection><!--Booleano que indica si la aplicación accede a Reflection-->
    <bookingApi>false</bookingApi><!--Booleano que indica si la aplicación accede a la BookingApi-->
    <checkinApi />
  </navitaireServices>
```

```
  <automaticTasks>
    <!--Tareas automáticas que se deben crear al desplegar esta aplicación-->
    <webServiceTask id="Nombre del Job en el scheduler">
      <webServiceId>Id de la aplicación(V.Aplicacion.WebService)</webServiceId>
      <methodName>Nombre del método que se desea ejecutar</methodName>
      <cron-expression>Expresión cron que indica cuándo se debe ejecutar la aplicación</cron-expression>
    </webServiceTask>
    <storedProcedureTask id="Nombre del job en el scheduler">
      <databaseId>Id de la base de datos</databaseId>
      <storedProcedure>Nombre del stored procedure</storedProcedure>
      <cron-expression>Expresión cron que indica cuándo se debe ejecutar el stored procedure</cron-
expression>
    </storedProcedureTask>
  </automaticTasks>

</component>
```

### 3.2.2 Aplicacion WebUI

A continuación se detallan los casos de uso identificados:




### 3.2.2.1 Identificarse en el sistema (Login)

Descripción: el usuario indica su nombre de usuario y contraseña para ser autenticado en el sistema

Actores: Usuario anónimo, Sistema

Precondiciones: el usuario no está autenticado en el sistema y dispone de cuenta en el sistema

Flujo:

| Usuario anónimo | Sistema |
|---|---|
| **1.** Navega a la página de identificación de usuarios | |
| **2.** Indica su nombre de usuario y contraseña | **3.** Recoge los datos introducidos |
| | **4.** Autentifica el usuario contra ActiveDirectory o LDAP |
| | **5.** Redirecciona a la página principal de la aplicación |

Extensiones:

- **3a:** si uno de los campos está vacío, regresa a la página de login mostrando un mensaje de error pidiendo que se rellene los datos que falta
- **4a:** si los datos proporcionados son incorrectos, regresa a la página de login mostrando un mensaje de error indicando que el nombre de usuario o contraseña son incorrectos

#### 3.2.2.2 Cerrar la sesión en la aplicación (Logout)

Descripción: el usuario indica que quiere finalizar el uso de la aplicación

Actores: Usuario identificado, Sistema

Precondiciones: el usuario ha iniciado la sesión en la aplicación

Flujo:

| Usuario identificado | Sistema |
|---|---|
| **1.** Clica en el botón Logout | **2.** Recoge la petición |
| | **3.** Guarda los cambios realizados |
| | **4.** Destruye la sesión actual |
| | **5.** Redirecciona a la página de identificación de usuarios |

#### 3.2.2.3 Elegir directorio de los IOMs

Descripción: el usuario indica la ruta y nombre del directorio que contiene los ficheros IOM con los cuales desea trabajar.

Actores: Usuario identificado, Sistema

Precondiciones: el usuario ha iniciado la sesión en la aplicación

Flujo:

| Usuario identificado | Sistema |
|---|---|
| **1.** Clica en el botón “Directorio IOM” | **2.** Abre una ventana con una listado de los directorios accesibles por la aplicación |
| **3.** Elige el directorio que contiene los IOM | |
| **4.** Clica OK | **5.** Recoge la ruta y nombre del directorio elegida y los guarda en memoria |
| | **6.** Cierra la ventana abierta en el paso 2 |

#### 3.2.2.4 Cargar IOMs

Descripción: la aplicación carga la información contenida en los ficheros IOMs con los que desea trabajar el usuario

Actores: Usuario identificado, Sistema

Precondiciones: el usuario ha iniciado la sesión en la aplicación y ha indicado el directorio que contiene los IOMs

Flujo:

| Usuario identificado | Sistema |
|---|---|
| **1.** Clica en el botón “Cargar IOMs” | **2.** Recoge la petición |
| | **3.** Cargar el primero fichero IOM |
| | **4.** Valida la estructura del fichero |
| | **5.** Lee la información contenida, la carga en memoria |
| | **6.** Repite los pasos 3 a 5 para los ficheros restantes |
| | **7.** Actualiza la interfaz de la aplicación representando la información obtenida |

Extensiones:

- **3a:** si se produce un error accediendo al fichero, pasa a procesar el siguiente fichero. Cuando finalice el proceso, muestra un mensaje de error informando al

usuario del error producido.

- **4a:** si la estructura del fichero es incorrecta, pasa a procesar el siguiente fichero. Cuando finalice el proceso, muestra un mensaje de error informando al usuario del error producido.

#### 3.2.2.5 Validar IOMs

Descripción: la aplicación valida la estructura de los ficheros IOMs con los que desea trabajar el usuario

Actores: Usuario identificado, Sistema

Precondiciones: el usuario ha iniciado la sesión en la aplicación y ha indicado el directorio que contiene los IOMs

Flujo:

| Usuario identificado | Sistema |
|---|---|
| **1. Clica en el botón “Validar IOMs”** | **2.** Recoge la petición |
| | **3.** Cargar el primero fichero IOM |
| | **4.** Valida la estructura del fichero |
| | **5.** Repite los pasos 3 y 4 para los ficheros restantes |
| | **6.** Muestra en pantalla el resultado de la validación de cada fichero |

Extensiones:

- **3a:** si se produce un error accediendo al fichero, pasa a procesar el siguiente fichero. Cuando finalice el proceso, incluye un mensaje error en el informe global

#### 3.2.2.6 Buscar aplicación

Descripción: el usuario busca una o varias aplicaciones indicando ciertos criterios de búsqueda

Actores: Usuario identificado, Sistema

Precondiciones: el usuario ha iniciado la sesión en la aplicación; ha indicado el

directorio que contiene los IOMs y éstos están cargados

Flujo:

| Usuario identificado | Sistema |
|---|---|
| **1.** Introduce el criterio de búsqueda (texto) en la caja de texto del buscador | |
| **2.** Clica en el botón “Buscar” | **3.** Recoge la petición |
| | **4.** Realiza una búsqueda del texto introducido en la información relevante (nombres, descripciones) de los IOMs |
| | **5.** Refresca el listado de aplicaciones mostrando únicamente las aplicaciones afectadas por la búsqueda |
| | **6.** Refresca la interfaz de la aplicación indicando que se ha aplicado un filtro sobre el listado de aplicaciones |

Extensiones:

#### 3.2.2.7 Filtrar aplicaciones

Descripción: el usuario filtra el listado de aplicaciones indicando una o varias características predefinidas

Actores: Usuario identificado, Sistema

Precondiciones: el usuario ha iniciado la sesión en la aplicación; ha indicado el directorio que contiene los IOMs y éstos están cargados

Flujo:

| Usuario identificado | Sistema |
|---|---|
| **1.** elige los criterios que deben cumplir las aplicaciones de una lista de criterios definida a priori | **2.** Recoge la petición |
| | **3.** Calcula qué aplicaciones cumplen los criterios indicados |
| | **4.** Refresca el listado de aplicaciones mostrando únicamente las que cumplen los requisitos |

| Usuario identificado | Sistema |
|---|---|
| | **5.** Refresca la interfaz de la aplicación indicando que se ha aplicado un filtro sobre el listado de aplicaciones |

### 3.2.2.8 Consultar detalle aplicación

Descripción: el usuario consulta toda la información que se tiene de una aplicación

Actores: Usuario identificado, Sistema

Precondiciones: el usuario ha iniciado la sesión en la aplicación; ha indicado el directorio que contiene los IOMs y éstos están cargados

Flujo:

| Usuario identificado | Sistema |
|---|---|
| **1.** Clica en el nombre de la aplicación en el listado de aplicaciones | **2.** Recoge la petición |
| | **3.** Carga la información de la aplicación elegida |
| | **4.** Refresca la interfaz de la aplicación mostrando la información completa de la aplicación |

### 3.2.2.9 Validar requisitos

Descripción: la aplicación valida los requisitos de la aplicación, es decir, comprueba que puede realizar las conexiones a bases de datos, FTP, colas, etc.

Actores: Usuario identificado, Sistema

Precondiciones: el usuario ha iniciado la sesión en la aplicación; ha indicado el directorio que contiene los IOMs y éstos están cargados

Flujo:

| Usuario identificado | Sistema |
|---|---|
| **1.** Clica en el nombre de la aplicación en el listado de aplicaciones | |

| Usuario identificado | Sistema |
|---|---|
| **2.** Clica en el botón “Validar requisitos” | **3.** Recoge la petición |
| | **4.** Para cada requisito de la aplicación de tipo conexión a FTP, base de datos, webservice, etc., comprueba que dicha conexión puede realizarse correctamente |
| | **5.** Refresca la interfaz de la aplicación mostrando un informe detallado del proceso |

### 3.2.2.10 Consultar mapa

Descripción: muestra un mapa visual con todas la aplicaciones, sus conexiones entre ellas y hacia servicios externos.

Actores: Usuario identificado, Sistema

Precondiciones: el usuario ha iniciado la sesión en la aplicación; ha indicado el directorio que contiene los IOMs y éstos están cargados

Flujo:

| Usuario identificado | Sistema |
|---|---|
| **1.** Clica en el botón mostrar mapa | **2.** Recoge la petición |
| | **3.** Carga un mapa con los servicios externos y aplicaciones además de las relaciones entre los mismos |

### 3.2.2.11 Autenticar usuario

Descripción: el sistema deberá poder autenticar el usuario mediante un servicio externo

Actores: Sistema; Sistema externo

Precondiciones: ninguna

Flujo:

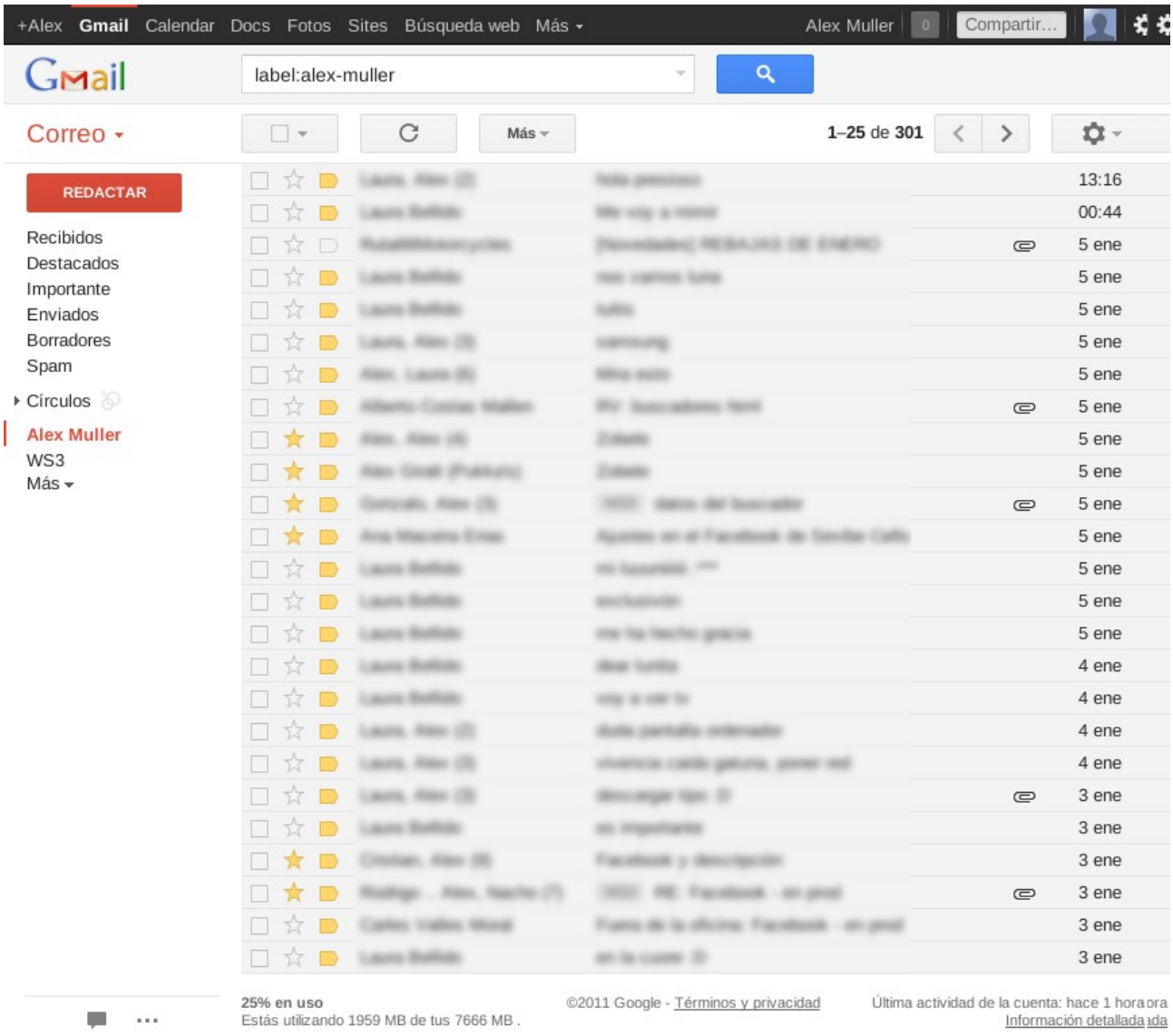
| Sistema | Sistema externo |
|---|---|
| **1.** Recoge los datos del usuario (nombre de usuario y contraseña) | |
| **2.** Se conecta al servicio externo pasándole los datos anterior | **3.** Recoge los datos |
| | 4. Valida la información contra su base de datos de usuario |
| 6. Recoge el resultado devuelto | 5. Indica al sistema la veracidad o no de la información de usuario y contraseña |

## 3.3 Definición de la interfaz de usuario

Una vez analizados los requisitos y prestaciones de la futura aplicación, procedemos a la definición de la interfaz. Dadas las posibles operaciones y prestaciones de la misma, se propone la utilización de una única interfaz, conocidas también por *Single-page applications*. Como en nuestro caso se trata además de una aplicación web, podemos definirla también como *Rich Internet Application (RIA)*.

En la fase de diseño se estudiarán los detalles técnicos de estos dos conceptos, por ahora es suficiente con tener en cuenta que la aplicación constará de una única pantalla.

Una ejemplo de Single-page Rich Internet Application puede ser Gmail de Google.




A partir de los casos, se determina que la interfaz deberá contener los siguientes elementos:

- Formulario de inicio y cierre de sesión
- Selector de directorio de trabajo
- Listado de aplicaciones
- Filtro y buscador de aplicaciones
- Detalle de una aplicación

- Resultados de las validaciones de ficheros IOM
- Mapa
- Botón cierre de sesión

Respetando la estructura clásico de aplicaciones web, se decide dividir la interfaz en cuatro secciones:

- Cabecera
  - Contendrá el selector de directorio y el botón para cerrar sesión
- Lateral izquierdo
  - Contendrá el listado de aplicaciones junto al buscador y filtro
- Sección principal, divida en dos subsecciones:
  - Mapa de aplicaciones
  - “Ventana” de mensajes
- Pie
  - Contendrá información de la aplicación y del sistema

La siguiente imagen muestra la estructura propuesta.

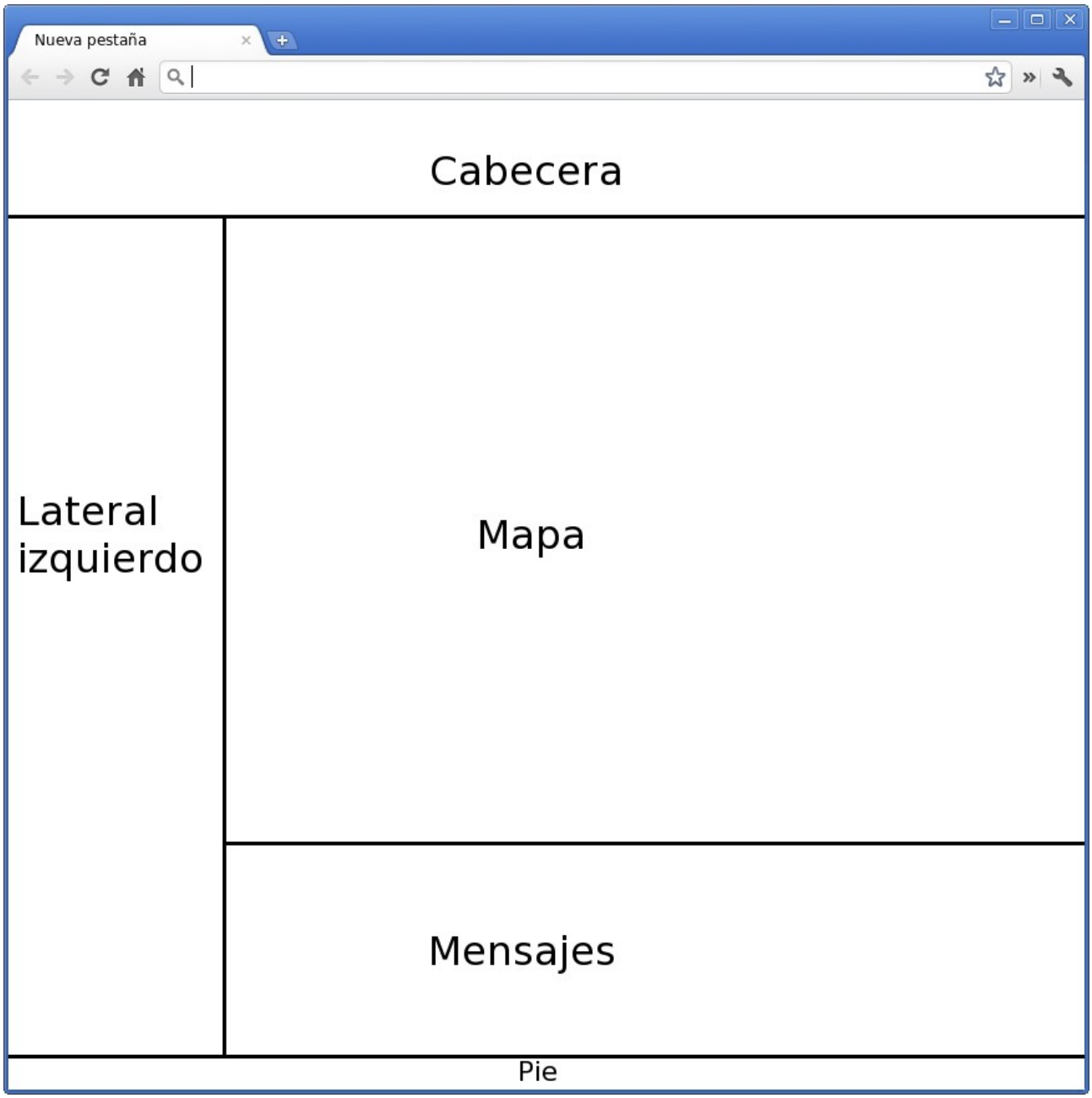



**Ilustración 1: Propuesto de estructura de la interfaz**

## 3.4 Especificación del plan de pruebas

A continuación se detalla el plan de pruebas, separado en tres apartados funcionales. Hay que recordar que estas pruebas se han realizado primero en el entorno de pruebas, antes de la primera UAT con usuarios finales. Una vez aprobada la UAT, se repetió en el

entorno de producción.

Al tratarse de una aplicación WebUI, cada prueba se realizó en los tres navegadores más usados: Firefox, Google Chrome e Internet Explorer 8 y superior.

#### 3.4.1.1 Pruebas de Interfaces y Contenidos

### 3.4.2 Elegir directorio de los IOMs

| | |
|---|---|
| Descripción | Indicar el directorio de trabajo |
| Caso(s) de uso relacionados: | Elegir directorio de los IOMs |
| Precondición | Sesión iniciada |
| Pasos | 1. Clicar en “Elegir directorio IOMs”<br>2. Clicar en uno de los directorios mostrados<br>3. Clica en Aceptar |
| Resultado esperado: | El paso 1 debe mostrar una ventana con un listado de los directorios accesibles desde la aplicación.<br>El paso 3 debe cerrar la ventana y mostrar en una caja de texto la ruta del directorio elegido |

### 3.4.3 Cargar IOMs

| | |
|---|---|
| Descripción | Cargar la configuración de los ficheros IOMs y mostrar las aplicaciones disponibles |
| Caso(s) de uso relacionados: | Cargar IOMs |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Clicar en “Cargar IOMs” |
| Resultado esperado: | La lista de aplicaciones debe refrescarse con las aplicaciones de los IOMs leídos. |

### 3.4.4 Validar IOMs

| | |
|---|---|
| Descripción | Validar la estructura de los ficheros IOMs |
| Caso(s) de uso relacionados: | Validar IOMs |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Clicar en “Validar IOMs” |
| Resultado esperado: | La aplicación muestra el resultado de validar contra un DTD los ficheros IOM |

### 3.4.5 Buscar aplicación

| | |
|---|---|
| Descripción | Buscar una aplicación por su nombre. |
| Caso(s) de uso relacionados: | Buscar aplicación |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Indicar el nombre de una aplicación en la caja de texto del apartado “Búsqueda”<br>2. Clicar en el botón “Buscar” |
| Resultado esperado: | El listado de aplicaciones se refresca mostrando únicamente los aplicaciones que satisfacen el criterio de la búsqueda |

### 3.4.6 Filtrar aplicaciones

| | |
|---|---|
| Descripción | Filtrar el listado de aplicaciones aplicando las distintas opciones del apartado “Filtro” |
| Caso(s) de uso relacionados: | Filtrar aplicaciones |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Hacer clic en uno o más checkbox del apartado filtro<br>2. Clicar en “Filtro listado” |
| Resultado esperado: | La aplicación refrescar el listado de aplicaciones con aquellos que satisfacen el criterio del filtro. En caso de no encontrarse ninguna, informando correctamente de ello. |

### 3.4.7 Consultar detalle aplicación

| | |
|---|---|
| Descripción | Mostrar el detalle de una aplicación. |
| Caso(s) de uso relacionados: | Consultar detalle aplicación |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Clicar en el nombre de una aplicación en el listado de aplicaciones<br>2. Clicar en el botón “Mostrar detalle” |
| Resultado esperado: | La aplicación debe refrescar la zona principal mostrando la información completa de la aplicación elegida |

### 3.4.8 Validar requisitos

| | |
|---|---|
| Descripción | Comprobar si se satisfacen los requisitos de una aplicación |
| Caso(s) de uso relacionados: | Validar requisitos |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Clicar en el nombre de una aplicación en el listado de aplicaciones |

| | |
|---|---|
| | 2. Clicar en el botón “Validar requisitos” |
| Resultado esperado: | La aplicación debe mostrar un mensaje con el resultado de la validación |

### 3.4.9 Pruebas de Funcionalidades y Operación

### 3.4.10 Elegir directorio de los IOMs

| | |
|---|---|
| Descripción | Indicar el directorio de trabajo |
| Caso(s) de uso relacionados: | Elegir directorio de los IOMs |
| Precondición | Sesión iniciada |
| Pasos | 1. Clicar en “Elegir directorio IOMs”<br>2. Clicar en uno de los directorios mostrados |
| Resultado esperado: | El paso 1 debe mostrar una ventana con un listado de los directorios accesibles desde la aplicación, en ningún caso un directorio no accesible o no configurado |

### 3.4.11 Cargar IOMs

| | |
|---|---|
| Descripción | Cargar la configuración de los ficheros IOMs y mostrar las aplicaciones disponibles |
| Caso(s) de uso relacionados: | Cargar IOMs |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Clicar en “Cargar IOMs” |
| Resultado esperado: | La aplicación lee correctamente los ficheros del directorio indicado, los proceso correctamente. En caso de producirse un error, muestra un mensaje informando del error. La aplicación debe seguir procesando el resto de ficheros. |

### 3.4.12 Validar IOMs

| | |
|---|---|
| Descripción | Validar la estructura de los ficheros IOMs |
| Caso(s) de uso relacionados: | Validar IOMs |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Clicar en “Validar IOMs” |
| Resultado esperado: | La aplicación valida correctamente los ficheros correctos. Realizar prueba también con ficheros incorrectos y comprobar el mensaje de error. |

### 3.4.13 Buscar aplicación

| | |
|---|---|
| Descripción | Buscar una aplicación por su nombre. |
| Caso(s) de uso relacionados: | Buscar aplicación |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Indicar el nombre de una aplicación en la caja de texto del apartado “Búsqueda”<br>2. Clicar en el botón “Buscar” |
| Resultado esperado: | La aplicación encuentra correctamente las aplicaciones que satisfacen el criterio de la búsqueda. Realizar prueba con símbolos, espacios, acentos, números y comprobar grado de acierto. |

### 3.4.14 Filtrar aplicaciones

| | |
|---|---|
| Descripción | Filtrar el listado de aplicaciones aplicando las distintas opciones del apartado “Filtro” |
| Caso(s) de uso relacionados: | Filtrar aplicaciones |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Hacer clic en uno o más checkbox del apartado filtro<br>2. Clicar en “Filtro listado” |
| Resultado esperado: | La aplicación encuentra correctamente las aplicaciones que satisfacen el criterio del filtro. Realizar prueba con varios filtros. |

### 3.4.15 Consultar detalle aplicación

| | |
|---|---|
| Descripción | Mostrar el detalle de una aplicación. |
| Caso(s) de uso relacionados: | Consultar detalle aplicación |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Clicar en el nombre de una aplicación en el listado de aplicaciones<br>2. Clicar en el botón “Mostrar detalle” |
| Resultado esperado: | La aplicación debe mostrar toda la aplicación contenida en el fichero IOM |

### 3.4.16 Validar requisitos

| | |
|---|---|
| Descripción | Comprobar si se satisfacen los requisitos de una aplicación |
| Caso(s) de uso relacionados: | Validar requisitos |
| Precondición | Sesión iniciada; directorio IOMs elegido |
| Pasos | 1. Clicar en el nombre de una aplicación en el listado de aplicaciones<br>2. Clicar en el botón “Validar requisitos” |

| | |
|---|---|
| Resultado esperado: | La aplicación debe obtener la lista de requisitos de la aplicación elegida (conexiiones con base de datos, FTP, webservices o colas) y comprobar una por una la posibilidad de realizar esta conexión, simulando un acceso al servicio, sin realizar ninguna modificación. |

### 3.4.17 Pruebas de Seguridad

### 3.4.18 Identificarse en el sistema

| | |
|---|---|
| Descripción | Iniciar sesión con un usuario válido |
| Caso(s) de uso relacionados: | Identificarse en el sistema, Autenticar usuario |
| Precondición | Sesión iniciada |
| Pasos | 1. Introducir nombre de usuario y contraseña<br>2. Clicar en “Iniciar sesión” |
| Resultado esperado: | Se muestra la pantalla principal de la aplicación |

### 3.4.19 Identificarse en el sistema con un usuario inválido

| | |
|---|---|
| Descripción | Iniciar sesión con un usuario no existente, desactivado o con contraseña incorrecta |
| Caso(s) de uso relacionados: | Identificarse en el sistema, Autenticar usuario |
| Precondición | Sesión no iniciada |
| Pasos | 1. Introducir nombre de usuario y contraseña de un usuario inexiste<br>2. Clicar en “Iniciar sesión” |
| Resultado esperado: | Se muestra la pantalla principal de login con un mensaje de error informando de un error en los datos introducidos |
| Observaciones | Repetir prueba con un usuario desactivado y con una contraseña incorrecta |

### 3.4.20 Cerrar la sesión en la aplicación

| | |
|---|---|
| Descripción | Cerrar la sesión de la aplicación |
| Caso(s) de uso relacionados | Cerrar la sesión en la aplicación |
| Precondición | Sesión iniciada |
| Pasos | 1. Clicar en el botón “Cerrar sesión” |
| Resultado esperado | Debe mostrarse la página de login |

### 3.4.21 Cerrar la sesión en la aplicación II

| | |
|---|---|
| Descripción | Cerrar la sesión de la aplicación e intentar acceder a la pantalla principal |
| Caso(s) de uso relacionados | Cerrar la sesión en la aplicación |
| Precondición | Sesión iniciada |
| Pasos | 1. Guardar la url de la página principal<br>2. Clicar en el botón “Cerrar sesión”<br>3. Acceder a la url guardad |
| Resultado esperado | Debe mostrarse la página de login o una página de error, en ningún caso la página principal |

# 4 Diseño

El objetivo de esta fase es, a partir del análisis realizado anteriormente, obtener los modelos y especificaciones del sistema. Se identificarán los diferentes componentes del sistema, sus licencias y las necesidades de desarrollos internos

## 4.1 Arquitectura general

Tal y como hemos mencionado en la fase de análisis, podemos considera nuestra aplicación una *Single-page application* además de una *Rich Internet Application*.

A continuación se explican ambos conceptos.

### 4.1.1 Single-page application

Una *single-page application* (SPA - aplicación de una sola página), también conocida como *single-page interface (SPI* - interfaz de una única interfaz), es una aplicación web o sitio web que cabe en una sola página web con el objetivo de proporcionar una experiencia de usuario más fluida, similar a una aplicación de escritorio.

En una SPA, o bien todo el código necesario (HTML, JavaScript y CSS) se obtienen con una carga de página, o bien se realizan modificaciones parciales cargando nuevo código desde el servidor web bajo demanda, por lo general debida a las acciones del usuario. A diferencia de las páginas web “tradicionales”, la página no se recarga con la interacciones del usuario, ni transfiere el control a otra página, sino que se refresca únicamente el contenido afectado. Las actualizaciones de la página pueden o no involucrar la interacción con un servidor.[11]

#### 4.1.1.1 Ventajas de una single-page application

Las ventajas principales de una single-page application radican en la experiencia de usuario. Esta arquitectura implica una menor interacción cliente-servidor ya que la información intercambiada entre ambos es mínima, básicamente se intercambia la

11 http://en.wikipedia.org/wiki/Single-page_application

información que necesita actualizarse o la necesaria para poner en marcha un proceso en el servidor, como actualizar un dato en una base de datos, recuperar un dato, etc. Esta reducción de tráfico se traduce en un tiempo de espera menor, lo cual facilita el trabajo con la herramienta.

Por otro lado, esta arquitectura reduce la carga en el servidor web lo que se traduce en una reducción de hardware necesario. También traslada la generación de la interfaz de usuario al navegador web, lo cual descarga aún más la CPU del servidor web.

### 4.1.2 Rich Internet Application

Las rich Internet applications, o RIA (en español "aplicaciones de Internet enriquecidas"), son aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales. Estas aplicaciones utilizan un navegador web estandarizado para ejecutarse y por medio de complementos o mediante una máquina virtual se agregan las características adicionales.

Las RIA surgen como una combinación de las ventajas que ofrecen las aplicaciones web y las aplicaciones tradicionales. Buscan mejorar la experiencia del usuario.

Normalmente en las aplicaciones web, hay una recarga continua de páginas cada vez que el usuario pulsa sobre un enlace. De esta forma se produce un tráfico muy alto entre el cliente y el servidor, llegando muchas veces a recargar la misma página con un cambio mínimo.

En los entornos RIA, en cambio, no se producen recargas de página, ya que desde el principio se carga toda la aplicación, y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una base de datos o de otros ficheros externos.[12]

Existen varias librerías que facilitan este tipo de comunicación, comúnmente conocido como AJAX.

12 http://es.wikipedia.org/wiki/Rich_Internet_Applications

### 4.1.3 AJAX

Ajax, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).[13]

## 4.2 Definición de niveles de arquitectura

A continuación definirán la arquitectura conceptual y lógica del sistema.

El siguiente diagrama UML muestra los principales componentes del sistema.

13 http://es.wikipedia.org/wiki/AJAX

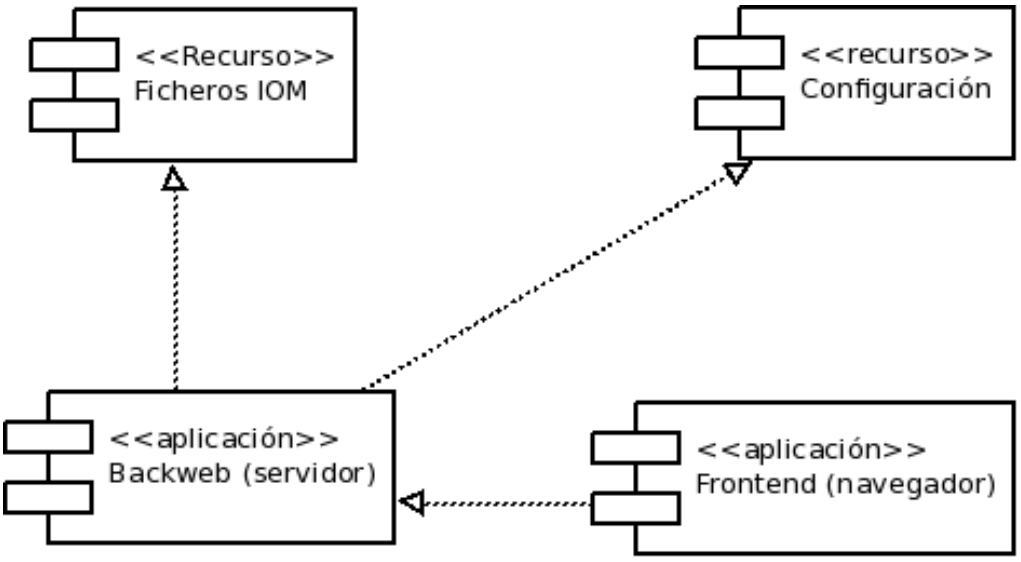



**Ilustración 2: Diagrama UML de componentes**

Las siguientes tarjetas CRC (clase responsabilidad colaborador) muestran, para cada componente, información sobre las responsabilidades y su relación con otros componentes.

| **Ficheros IOM** | |
|---|---|
| Contiene la información de una aplicación en formato XML en el formato desarrollado. | Backweb |

| **Configuración** | |
|---|---|
| Almacena la información de la propia aplicación (posibles directorios de trabajo, nombre servidor, etc.) | Backweb |

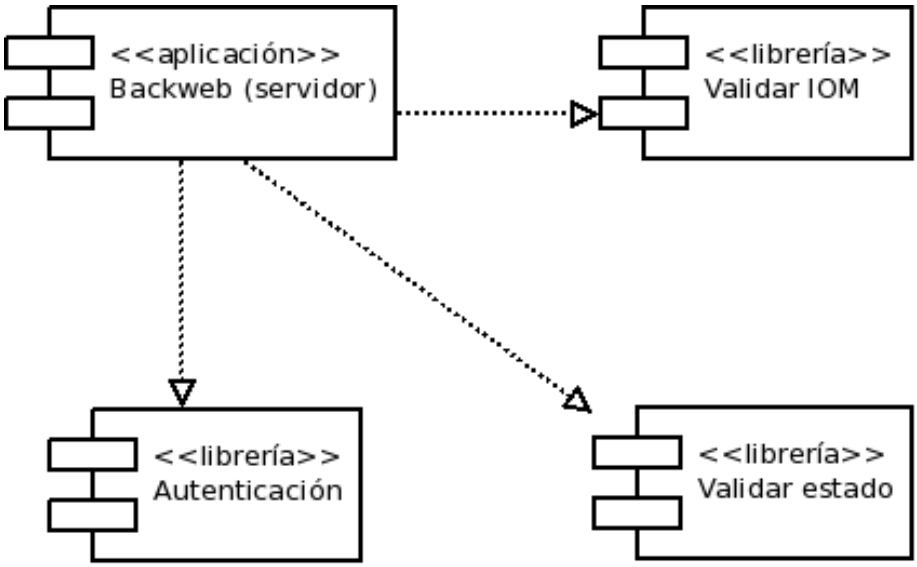
| **Backweb** | |
|---|---|
| Proporciona información al frontend para construir la interfaz.<br>Ejecuta los procesos de validación.<br>Ejecuta los procesos de comprobación del | Ficheros IOM<br>Backweb<br>Frontend |

| | |
|---|---|
| estado de las aplicaciones.<br>Recoge las peticiones de acceso a la propia aplicación. | |

| Frontend | |
|---|---|
| Muestra el mapa e información de las aplicaciones.<br>Permite eligir el directorio de trabajo.<br>Muestra el resultado de las validaciones de IOMs y estado de las aplicaciones.<br>Permite buscar y filtrar aplicaciones. | Backweb |

### 4.2.1 Identificación de subsistemas

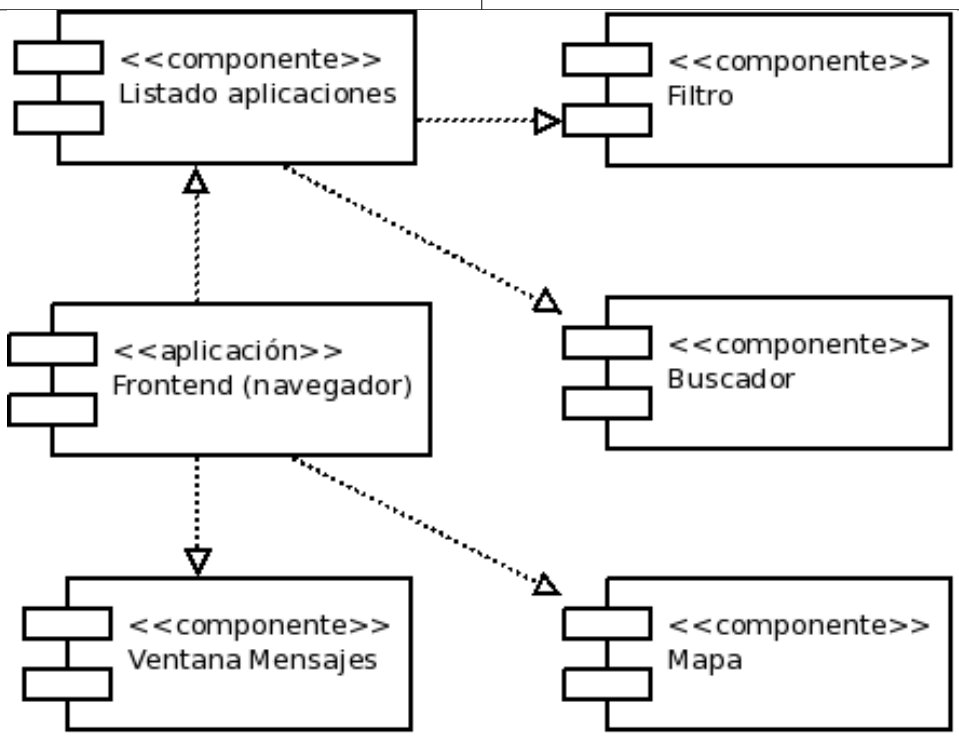
Los sistemas de Backend y Frontend se dividen a su vez en los siguientes subsistemas:




| Autenticación | |
|---|---|
| Autentica los usuarios | Backweb |

| Validar estado | |
|---|---|
| Realiza las pruebas necesarias para determinar el estado de una aplicación (pruebas de conexión, disponibilidad de recursos, etc.) | Backweb |

| Validar IOM | |
|---|---|
| Validar el formato de un fichero IOM | Backweb |




### 4.2.2

| Listado aplicaciones | |
|---|---|
| Muestra las aplicaciones disponibles | Frontend |

| Filtro | |
|---|---|
| Permite filtrar el listado de aplicaciones según determinados criterios preestablecidos | Listado de aplicaciones |

| Buscador | |
|---|---|
| Permite filtrar el listado de aplicaciones según un criterio de búsqueda libre (cadena de texto) | Listado de aplicaciones |

| Ventana de mensajes | |
|---|---|
| Muestra el resultado de las comprobaciones de estado y validaciones | Frontend |

| Mapa | |
|---|---|
| Muestra el mapa de aplicaciones | Frontend |

## 4.3 Revisión de casos de uso

En la fase de análisis se identificaron los siguientes casos de uso:

- Identificarse en el sistema (Login)
- Cerrar la sesión en la aplicación (Logout)
- Elegir directorio de los IOMs
- Cargar IOMs
- Validar IOMs
- Buscar aplicación
- Filtrar aplicaciones
- Consultar detalle aplicación
- Validar requisitos
- Consultar mapa

A continuación se revisarán cada uno de los casos de uso y se identificaron los subsistemas implicados y los mensajes que se intercambian.

### 4.3.1 Identificarse en el sistema (Login)

- Librería de Autenticación: permitirá comprobar la autenticidad de un usuario y si

tiene permisos para acceder a la aplicación.

- Frontend: recogerá las credenciales del usuario para pasárselas al componente de backweb. Recibirá la confirmación o denegación de acceso.
- Backweb: se encarga de recoger las credenciales recibidas del frontend y se las pasará a librería de autenticación

### 4.3.2 Cerrar la sesión en la aplicación (Logout)

Los componentes implicados y relacionados son idénticos al caso de uso anterior.

### 4.3.3 Elegir directorio de los IOMs

- Configuración: almacena los posibles directorios que contiene los ficheros IOM
- Frontend: mostrará los directorios disponibles
- Backweb: obtiene los posibles directorios y pasa un listado al frontend

### 4.3.4 Cargar IOMs

- Backweb: obtiene los ficheros IOM del directorio elegido, los proceso y manda la información correspondiente al componente Frontend
- Frontend: muestra los aplicaciones y su información obtenidas a partir de los fichero IOM

### 4.3.5 Validar IOMs

- Librería Validar IOM: se encarga de validar el formato del fichero XML
- Backend: llama a la librería recogiendo el resultado de la validación. Manda el resultado al frontend
- Frontend: recoge el resultado de la validación y la muestra en la ventana de mensajes

### 4.3.6 Buscar aplicación

- Buscador: aplica el criterio de búsqueda definido por el usuario y lo aplica al listado de aplicaciones
- Listado de aplicaciones: muestra únicamente las aplicaciones válidas según los criterios del buscador

### 4.3.7 Filtrar aplicaciones

- Filtro: aplica el criterio de búsqueda definido en el filtro y lo aplica al listado de aplicaciones
- Listado de aplicaciones: muestra únicamente las aplicaciones válidas según los criterios del filtro

### 4.3.8 Consultar detalle aplicación

- Frontend: muestra la información de la aplicación en cuestión, obtenida previamente durante la carga de los fichero IOM

### 4.3.9 Validar requisitos

- Librería Validar estado: validar los requisitos de la aplicación en cuestión realizando una serie de comprobaciones. Envía el resultado al componente de backweb
- Backweb: llama a la librería recogiendo el resultado de la validación. Envía el resultado al componente de Frontend.
- Frontend: recoge el resultado de la validación y lo muestra al usuario

### 4.3.10 Consultar mapa

- Mapa: muestra las aplicaciones en cuestión y su relación con otros sistemas.

## 4.4 Especificaciones de pruebas

### 4.4.1 Pruebas unitarias

El objetivo de estas pruebas es comprobar el funcionamiento de cada componente o módulo por separado, especialmente las librerías independientes.

- Componente Autenticación: se comprobará la correcta autenticación de un usuario existente, el rechazo de usuarios inexistente o contraseñas incorrectas así como la concesión y denegación de permisos
- Componente Validar estado: se comprobará el resultado de cada una de las posibles pruebas de conexión, acceso a recursos, etc.
- Componente Validar IOM: se comprobará el resultado de validar ficheros XML correctos e incorrectos
- Componente Buscador: se comprobará el resultado de aplicar varios criterios de búsqueda
- Componente Filtro: se comprobará el resultado de aplicar varios criterios de filtrado

### 4.4.2 Pruebas de integración

Se comprobará el funcionamiento de todos los componentes actuando juntos:

- el acceso al sistema
- lectura y aplicación de a configuración de la aplicación
- obtención y listado de aplicaciones
- mostrar información de las aplicaciones
- funcionamiento del buscador y filtro
- comprobación del funcionamiento del mapa y exactitud de la información mostrada
- resultados de los procesos de validación de IOMs y comprobación de estado

### 4.4.3 Pruebas de implantación

Se ejecutarán una vez instalado el sistema en su entorno definitivo. Se repetirán las pruebas de integración.

### 4.4.4 Pruebas de aceptación

Estas pruebas se realizarán en el entorno de producción y se comprobará el grado de cumplimiento de los objetivos planteados. El objetivo de estas pruebas es la aceptación del proyecto por parte de los usuarios finales.

## 4.5 Requisitos de implantación

Los requisitos de implantación del proyecto son los siguientes:

- Sistema operativos Microsoft Windows o GNU/Linux
- Framework Mono 2
- Servidor Apache

# 5 Desarrollo

## 5.1 Planificación de las actividades de desarrollo e integración del sistema

El esfuerzo estimado de desarrollo e implantación es de 136 horas, repartidas en las siguientes tareas:

| # | Tarea | Esfuerzo (horas) |
|---|---|---|
| 1 | Implantación entorno de desarrollo | 4 |
| 2 | XSD validación fichero IOM | 4 |
| 3 | Diseño HTML/CSS | 16 |
| 4 | Listado aplicaciones | 8 |
| 5 | Obtención de configuración | 4 |
| 6 | Lectura directorios IOM | 4 |
| 7 | Procesado IOMs | 16 |
| 8 | Buscador | 8 |
| 9 | Mapa | 28 |
| 10 | Filtro | 4 |
| 11 | Login/Logout usuario | 8 |
| 12 | Validación IOMs | 8 |
| 13 | Validar estado | 16 |
| 14 | Bug fixing | 4 |
| 15 | Implantación | 4 |

A continuación se muestra la planificación temporal:

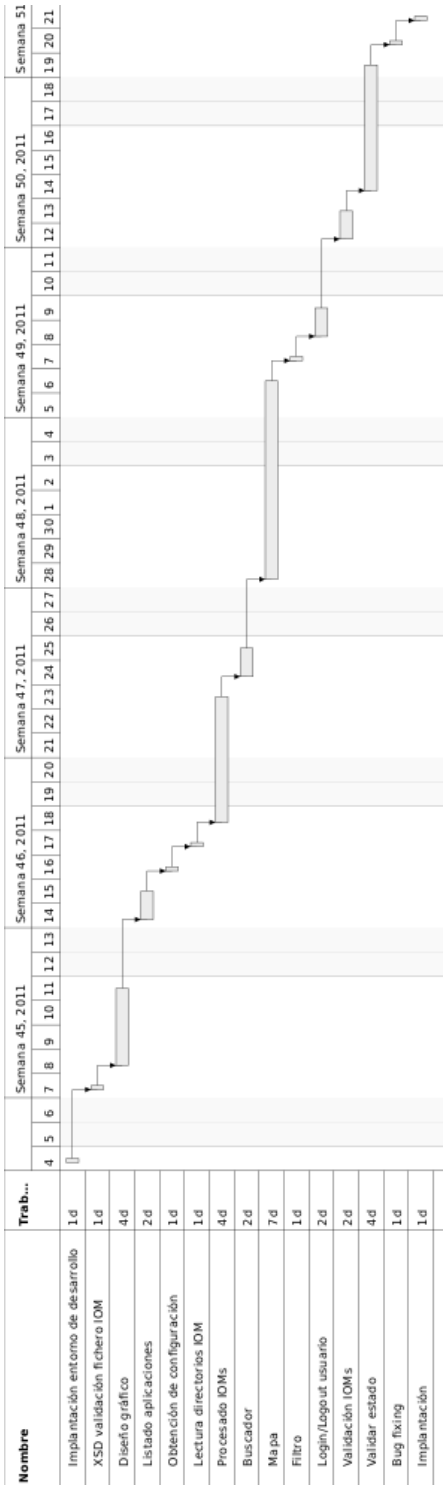



**Ilustración 4: Planificación temporal fase desarrollo**

## 5.2 Desarrollo

Las tareas realizadas durante el desarrollo son las siguientes:

- Implantación entorno de desarrollo
  - Se instala Monodevelop
  - Se instala el módulo mod_mono para Apache
  - Se obtienen las librerías jquery, jquery, räphael y la librería de autenticación de usuarios (ya desarollada por la empresa)
- XSD validación fichero IOM
  - A partir del XML propuesto se desarrolla en XSD de validación
- Diseño HTML/CSS
  - Se implementa la estructura propuesto en la fase de Análisis
- Listado aplicaciones
  - Se desarrolla la funcionalidad que genera un Xml con el listado de IOMs, el cual se consume con javascript en la parte del Frontend y se genera el listado de aplicaciones
- Obtención de configuración
  - Se desarrolla una librería que obtiene los valores de configuración de la aplicación a partir de un fichero XML
- Lectura directorios IOM
  - Se desarrolla una clase que busca los ficheros IOM de manera recursiva dentro de un directorio dado
- Procesado IOMs
  - Se desarrolla un clase que procesa la información de un fichero IOM para enviar dicha información a la parte del Frontend
- Buscador
  - Se desarrolla un buscador capaz de buscar la cadena de texto introducida por el usuario entre la información disponible de las aplicaciones
- Mapa
  - Con ayuda de la librería Raphäel, se desarrolla un mapa que muestra las aplicaciones y servicios en forma de icono

    - Los iconos se conectan mediante líneas se están relacionados entre sí
    - Se desarrolla la opción de mover los iconos para mejorar la visualización del mapa
    - Se desarrolla una leyenda con información de la aplicación seleccionada
- Filtro
    - Se desarrolla un filtro capaz de obtener las aplicaciones que cumplen el criterio del filtro configurado por el usuario
- Login/Logout usuario
    - Se desarrolla el inicio y cierre de sesión, autenticando el usuario con ayuda de la librería proporcianda
- Validación IOMs
    - Se desarrolla la funcionalidad que valida los ficheros IOMs contenidos en un directorio
- Bug fixing
    - Se solucionan los error detectados en las pruebas de integración en el entorno de pruebas
- Implantación
    - Se instala la aplicación en un entorno Windows y GNU/Linux para realizar las pruebas de integración
- Validar estado
    - Por cuestiones técnicas y de tiempo no se desarrolla esta funcionalidad

# 6 Conclusiones

El objetivo del proyecto consistía en obtener una aplicación capaz de representar gráficamente el complejo sistema informático de la empresa, además de desarrollar una especie de carnet de identidad para las aplicaciones.

Podemos afirmar que se han cumplido los Objetivos funcionales expuestos en el apartado 1.2 y los requisitos técnicos y económicos expuestos en los apartados 3.1.1 a 3.1.3

Un objetivo no mencionado durante el desarrollo de esta memoria ha sido la prueba del framework Mono, alternativa interesante al framework propietario de .NET. Podemos afirmar que esta primera aproximación a este framework ha sido satisfactoria, dejando la puerta abierta para su posible uso en el futuro.

Debemos destacar que esta primera versión de la aplicación es muy básica, ya que muchos aspectos se pueden mejorar en el futuro, como explico más adelante.

El único objetivo no conseguido en el proyecto es la monitorización del estado de la aplicación. Durante el desarrollo se detectó la complejidad de esta tarea, dada la cantidad de aplicaciones diferentes y sus distintos requisitos de funcionamiento.

En particular se vio que la comprobación de acceso a servicios FTP, base de datos o servicios web podría afectar al funcionamiento de dichos servicios, cosa que se quiso evitar desde el principio. También se presentó un problema a la hora de gestionar estos accesos debido a que las credenciales de acceso a estos servicios se gestionan de manera estricta, en nuestro caso la aplicación necesitaría todas las credenciales de todas las aplicaciones, circunstancia que podría suponer un riesgo de seguridad. Este problema se podría mitigar creando usuarios de prueba que no tuvieran acceso de escritura, sino únicamente de lectura a los servicios mencionados.

Para facilitar la elaboración de los IOMs, se podría implementar la edición y generación de los mismos a través de la aplicación. De esta manera, una persona sin conocimientos de XML podría mantener la información contenida en los IOMS.

En este proyecto se ha diseñado la aplicación para que ofrezca la información de los

IOMs de un único servidor, sin embargo, la posibilidad de poder consultar la información de varios servidores a través de una misma aplicación facilitaría las tareas de mantenimiento.

# 7 Glosario

- **XML:** siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).
- **JSON:** acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.
- **CSV:** Los ficheros CSV (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma) y las filas por saltos de línea
- **JSP:** es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.
- **PHP:** es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Se usa principalmente para la interpretación del lado del servidor (server-side scripting).
- **Jquery:** es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.
- **Single Page Application:** Una aplicación de una sola página (SPA), es una aplicación web o sitio web que cabe en una sola página web con el objetivo de proporcionar una experiencia de usuario más fluida similar a una aplicación de escritorio.
- **Rich Internet Application:** es una aplicación web que tiene muchas de las características del software de aplicaciones de escritorio. Normalmente se ejecuta a través de un navegador o de un browser plug-in y hace un uso intensivo de

JavaScript o máquinas virtuales