UNIVERSIDAD CARLOS III DE MADRID

Departamento de Ingeniería Telemática

PROYECTO FIN DE CARRERA

CITAS2.0

Un gestor de reuniones online basado en AJAX

Miguel Ángel Cabrera Bejarano

Tutor: Vicente Luque Centeno

Leganés, Abril 2009

Proyecto Fin de Carrera

UNIVERSIDAD CARLOS III DE MADRID

Departamento de Ingeniería Telemática

Título: Citas2.0: Un gestor de reuniones online basado en AJAX

Autor: Miguel Ángel Cabrera Bejarano

Director: Vicente Luque Centeno

EL TRIBUNAL

Presidente _____

Vocal _____

Secretario _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

SECRETARIO VOCAL PRESIDENTE

Agradecimientos

La realización de este proyecto no hubiera sido posible sin la paciencia y apoyo que he recibido de las personas que me rodean, dándome ánimos y soportando mis tan frecuentes cambios de humor. A mi futura mujer por su paciencia y aguante, a mis padres y hermanas por estar cuando les he necesitado, a mi familia política por apoyarme y animarme a acabarlo y a mis amigos por aceptar mis ausencias.

Para todos, Muchas Gracias.

Agradecimientos

Resumen

El objeto de este proyecto es el de realizar una aplicación web que nos permita organizar reuniones de tal manera que todas las personas convocadas pueden participar en la elección de la fecha del evento. La aplicación nos brinda la posibilidad de poder proponer actividades en unas fechas escogidas a un conjunto de contactos elegidos, y nos da la interfaz con la que todos los convocados a la actividad podrán realizar su elección.

Esta aplicación se ha implementado siguiendo técnicas modernas de enriquecimiento de interfaces mediante Ajax. La interfaz enriquecida nos permite realizar aplicaciones ágiles e intuitivas para el usuario a costa de recursos del cliente, los cual es preferible para descargar la transferencia de datos por parte del servidor. El hacer esta interfaz rica, en muchos casos, supone limitar su accesibilidad, por lo que se ha desarrollado otra versión accesible de la aplicación. Esta versión sigue los estándares de web tradicional con contenido dinámico dado por el servidor. Para poder agilizarla un poco se ha limitado con respecto a la versión de interfaz enriquecido.

Otra parte importante de este proyecto ha sido la realización de un sistema web genérico sobre el que desarrollar la aplicación. Ha sido interesante desarrollarlo de cara a la reutilización del código en futuras aplicaciones, lo cual nos va a permitir construir otras aplicaciones de forma más sencilla de lo habitual.

Las tecnologías utilizadas en este proyecto pasan por aquellas necesarias para un desarrollo web. En la parte de servidor se ha optado por php más MySQL. En la parte de cliente, en la versión accesible solamente el cliente se tiene que encargar de interpretar el código HTML y su hoja de estilo procedente del servidor, y en la versión enriquecida además hacemos uso de Javascript, XML y peticiones asíncronas al servidor (Ajax). Como formato de mensaje en estas peticiones asíncronas se utiliza JSON o XML indistintamente, aunque en la mayoría de los casos se utiliza JSON.

Índice General

Índice de tablas

Índice de figuras

Capítulo 1. Introducción

En los últimos años se ha producido un crecimiento muy grande en la usabilidad web. Disfrutamos día a día de aplicaciones web que se aproximan demasiado a aplicaciones de escritorio, llegando a extremos en los que no existe aplicación similar de escritorio ni tampoco la necesidad de tenerla. Usamos todos los días gestores de correo, lectores de noticias, aplicaciones ofimáticas, hasta incluso sistemas operativos remotos. Estas aplicaciones se denominan RIA , ***R**ich **I**ntenet **A**pplications* o *Aplicaciones Ricas de Internet*, la cuales combinan las ventajas de las aplicaciones web tradicionales y las aplicaciones de escritorio.

También todo este auge ha dado sentido a una nueva visión de la red surgiendo nuevos planteamientos e ideas. Ya no solo tenemos la necesidad de absorber la información que existe en el medio sino que queremos crearla y ser partícipes de otras. El usuario toma un papel activo en la red.

El proyecto fin de carrera que vamos a mostrar al lector, es el resultado de poner en práctica todos los conocimientos, metodologías de trabajos y pensamiento analítico adquiridos durante el ciclo formativo de la carrera, con la finalidad de desarrollar una aplicación web rica en usabilidad, mediante técnicas de programación no intrusivas, y con el objeto de hacer que, con esta aplicación RIA , el usuario pueda disponer de una aplicación que pueda suplir la necesidad de organizar eventos de forma colaborativa.

1.1 Motivación del proyecto

La idea de desarrollar esta aplicación surge de las siguientes partes:

- La necesidad del profesor tutor de disponer de una aplicación para la gestión de reuniones de código abierto y cumpliendo estándares de portabilidad de la información.
- La motivación del alumno de participar en el desarrollo de una aplicación web basándonos en las nuevas tecnologías que están utilizándose actualmente en la red.

1.1 Objetivos

Para poder llegar al objetivo final, que es el desarrollo de una aplicación web para la organización compartida de eventos llamada CITAS2,0, se definen los siguientes objetivos:

- **Disponer de un conjunto de librerías que realicen las tareas rutinarias de una**

aplicación web. Con la consecución de este objetivo se pretende disponer de un conjunto de librerías que nos aporten la funcionalidad básica necesaria para nuestra aplicación, que faciliten el desarrollo de la misma y las futuras modificaciones. Se busca que tengan un carácter genérico que permita reutilizarlas fácilmente para el desarrollo de otras aplicaciones. Con la inclusión de estas librerías en nuestra aplicación buscamos que se nos facilite el control de acceso de usuarios, la seguridad de acceso a los diversos programas de la aplicación, la posibilidad de utilizar plantillas de salida que nos van a facilitar la separación del código de aplicación frente a la presentación, y la posibilidad de acceder a datos enviados en peticiones HTTP de forma sencilla. Es decir, buscamos el separar lo que es propiamente la aplicación CITAS2.0 de lo que puede considerarse como común a otra aplicación web y por tanto reutilizable. Se pretende que estas librerías, junto con una definición de estructura de directorios, configuraciones y selector de aplicaciones, puedan considerarse como un servidor de aplicaciones simple, en una versión muy ligera y simplificada en comparación con otros sistemas actuales y que denominaremos como gestor de aplicaciones.

- **Construir una aplicación web para la organización compartida de invitaciones a eventos** que sea libre, de código abierto, fácil de usar, flexible a la hora de organizar los evento, accesible y que sea una RIA. No existe una alternativa libre para una aplicación de esta índole en la actualidad. Se pretende realizar su desarrollo siguiendo técnicas de enriquecimiento no intrusivas como Ajax.
- Darle a una aplicación de este tipo el **carácter de web 2.0** haciendo su contenido exportable mediante los estándares utilizados para este uso. Esto facilitará la exportación de la información para su uso en otros sistemas como puede ser Google Calendar y el formato Ical. Además esto supone una ventaja respecto a las aplicaciones similares existentes.

1.1 Contenido de la memoria

Esta memoria intenta reflejar todos los pasos que se han dado para la conseguir el objetivo final que es el de desarrollar una aplicación para gestionar eventos de forma compartida con los invitados. Se ha estructurado en varios capítulos siguiendo la lógica del desarrollo de la aplicación. Se verán, en los sitios pertinentes, las comparativas entre la elección tomada y el resto de opciones para tomar una decisión.

La estructura de la memoria podemos resumirla en los siguientes bloques que coinciden con los capítulos del libro:

1. **Estado del Arte.** En este bloque se tratan las tecnologías que se han utilizado en el desarrollo de este proyecto. Se muestra una visión de las nuevas técnicas y tecnologías, utilizadas en la programación web actual en la parte del interfaz de cliente, que permiten desarrollar aplicaciones RIA. Se trata de conseguir un punto intermedio entre el detalle de la explicación de cada técnica y tecnología, y el abanico de elementos disponibles, con la finalidad de no perder el sentido de esta memoria.
2. **Visión general de la aplicación y modelado de datos.** En este capítulo se muestra una visión global del diseño de aplicación definiendo sus partes. Se define el modelado de datos de la aplicación y se muestran las divisiones realizadas en la aplicación, como la parte de cliente, la parte de servidor referida a funcionalidad común y la parte de servidor referida a la que realiza la funcionalidad de la aplicación propiamente dicha.
3. **Descripción del gestor de aplicaciones** . Al conjunto de librerías desarrolladas, a la estructura de directorios, configuraciones y selector de aplicaciones, se le ha denominado como gestor de aplicaciones. Se hace también una comparativa con otros sistemas sobre los que se podría haber desarrollado la aplicación.
4. **Visión general de la aplicación CITAS2.0** . En este bloque se pasa a detallar como se ha creado la aplicación CITAS2.0. Se explica cada una de las aplicaciones que la componen así como las librerías y métodos asíncronos que la aplicación posee.
5. **Descripción de las aplicaciones de cliente de CITAS2.0** . En este capítulo se explica como son las aplicaciones creadas para los clientes, como funcionan , y las relaciones de ficheros y comunicaciones con el servidor que existen en la aplicación CITAS2.0.
6. **Ejemplos prácticos de CITAS2.0**. En este bloque se muestra un ejemplo práctico de la aplicación a modo de tutorial.

Capítulo 2. Estado del Arte

2.1 Introducción

Desde el nacimiento de la web hemos visto como se han usado diversas metodologías y tecnologías para dar vida a un medio cuyo contenido, en una primera instancia, es estático. La necesidad de dar un servicio instantáneo al usuario va siendo cada vez mayor, obligando a los desarrolladores a enriquecer la funcionalidad de sus aplicaciones y mejorar la interacción con el usuario.

De un visión inicial donde recargamos toda la página cuando una parte pequeña de información se ha modificado, queremos llegar a una visión donde una simple consulta al servidor, de forma asíncrona, nos de la información necesaria para actualizar una parcela de datos en la página, sin necesidad de recargarla entera, y consiguiendo una mejora substancial en la usabilidad de la página. Esto hace que la navegación sea mucho más rápida y amigable, permitiéndonos trasladar la mayor parte de la presentación web al cliente.

2.2 Breve historia

Desde los comienzos de internet muchas técnicas se han utilizado para incrementar la usabilidad web y darle un carácter dinámico a la información tratada. Entre ellas podemos destacar:

- ***Javascript:*** Nació de manos de Netscape con el nombre de LiveScript. Fue creado por Brendan Eich y fue introducido por primera vez en la versión 2.0 del navegador, lanzada en 1995. Por primera vez el desarrollador web podía interactuar con el usuario, y dotar de inteligencia a la página para realizar tareas simples, como validaciones de formularios, y así minimizar el número de veces que se realizaban las peticiones al servidor, en una época donde la limitación principal era la velocidad de acceso
- ***Marcos:*** Oficialmente se introdujeron en el HTML 4.0. La idea de poder mostrar varios documentos en una misma página. Si lo vemos como que cada marco representa distintas peticiones al servidor, y lo combinamos con el control que se podría hacer de los marcos con JavaScript, se abría un abanico de posibilidades
- ***La técnica del marco oculto:*** Esta técnica representa el primer modelo asíncrono de

petición/respuesta para aplicaciones web. Consiste en crear un marco con altura y anchura de cero pixels e introducir en él un formulario. Este formulario era tratado mediante funciones javascript las cuales rellenaban y enviaban los datos al servidor. La respuesta podría ser tratada por otra función JavaScript que mostrase los datos devueltos al solicitante.

- ***DHTML y DOM:*** Con la introducción de Dynamic HTML en el Internet Explorer 4.0 conseguimos que la información pudiera mostrarse en la página sin necesidad de recargarla entera, ya que con DHTML podemos alterar cualquier parte de la página cargada usando JavaScript. Si a esto le sumamos la técnica del marco oculto conseguimos poder refrescar cualquier parte de la página con información del servidor en cualquier momento. Con la influencia de Microsoft se introduce el Document Object Model (DOM) como pieza central del estándar. DOM nos da la estructura de toda la página web y no solo la posibilidad de modificar una etiqueta HTML como hace DHTML, sino que nos permite alterar la estructura de la página
- ***Iframes:*** La introducción de los iframes en el HTML 4.0 nos permite crear un frame en cualquier parte del documento. Mediante CSS podemos crear iframes ocultos para habilitar la comunicación cliente-servidor. Si le sumamos DOM podemos crear iframes bajo demanda mediante funciones javascript que creen lo creen, hagan la petición y recojan los datos del servidor.
- ***XMLHttp:*** Dado al éxito de las técnicas de marco oculto, Microsoft se propone dar cabida a esta interacción con el servidor en forma de objeto Activex, llamado XMLHttp e introducido en el 1999 en el Internet Explorer 5.0 como un control ActiveX. Este objeto se introduce una vez que Microsoft da cabida al manejo de XML mediante la librería MSXML. Con el objeto XMLHttp, además de acceder a los datos enviados por el servidor, podemos acceder a los códigos de estado y a las cabeceras HTTP. El formato de los datos enviados por el servidor son definidos por el desarrollador web y pueden ir estructurados en XML, pueden ser objetos Javascript serializados,etc. Otra ventaja del uso de este objeto frente a las técnicas anteriores es que se integra directamente en nuestro programa javascript sin necesidad de recurrir a cualquier función creada ni tener que esperar los ciclos de recarga de los frames. En contrapartida Mozilla saca su propio objeto para hacer peticiones al servidor llamado XMLHttpRequest.

Para referirnos indistintamente al objeto XMLHttp o al objeto XMLHttpRequest, lo

llamaremos XHR a lo largo del documento

2.1 Web 2.0

2.1.1 Introducción

El término Web 2.0 fue acuñado por Tim O'Reilly en 2004 para referirse a una segunda generación de Web basada en comunidades de usuarios y una gama especial de servicios, como las redes sociales, los blogs, los wikis o las folcsonomías, que fomentan la colaboración y el intercambio ágil de información entre los usuarios.

La Web 2.0 es una nueva filosofía de hacer las cosas por lo que no es de extrañar que se apoyen sobre los mismos estándares que se siempre han existido mucho antes de acuñar el término.

La Web 1.0, o tradicional, se apoyaba en contenido estático que no se actualizaba frecuentemente hasta que se empezó a dinamizar el contenido. En este paso, podríamos llamarlo Web 1.5, empezó a servirse contenido creado al vuelo mediante acceso a información actualizada encontrada en una base de datos. Lo que se perseguía como factores importantes en ambos casos era el número de visitas y la estética visual.

Debido a las facilidades y la gratuidad que actualmente da la red para que el usuario aporte contenidos facilita la iteración de usuario con la red y las redes sociales. Esto hace que los propulsores de la aproximación a la Web 2.0 crean que el uso de la web está orientado a la interacción y redes sociales, ya que el usuario no solo accede a la información sino que además aporta el contenido. Así, cuantas más personas accedan al servicio, mayor será el valor para el resto de los usuarios (efecto red) y, por tanto, más se fomentará el desarrollo de la inteligencia colectiva. . Es decir, los sitios Web 2.0 actúan más como puntos de encuentro, o webs dependientes de usuarios, que como webs tradicionales.

2.1.2 Definición

El concepto de 'Web 2.0' comenzó con una sesión de 'brainstorming' realizada entre O'Reilly y MediaLive International. Dale Dougherty, pionero de la web y vicepresidente de O'Reilly, observaron que lejos de 'estrellarse', la web era más importante que nunca, con apasionantes nuevas aplicaciones y con sitios web apareciendo con sorprendente regularidad.

Como hemos comentado anteriormente Web 2.0 es una nueva filosofía de hacer las cosas en las que el usuario es a la vez servidor de contenidos. En general, cuando mencionamos el término Web 2.0 nos referimos a una serie de aplicaciones y páginas de

Internet que utilizan la inteligencia colectiva para proporcionar servicios interactivos en red dando al usuario el control de sus datos.

Hay tres principios que fomentan la Web 2.0:

- **Comunidad**: el usuario aporta contenidos, interactúa con otros usuarios, crea redes de conocimiento, etc.
- **Tecnología**: un mayor ancho de banda facilita la transferencia de información lo cual nos permite trasladar las aplicaciones a internet.
- **Arquitectura modular**: favorece la creación de aplicaciones complejas de forma más rápida y a un menor coste.

A continuación se muestra una tabla comparativa de aplicaciones entre Web 1.0 y Web 2.0:

Web 1.0	Web 2.0
Páginas Personales	Bitácoras
Especulación con nombres de dominio	Optimización en buscadores
Páginas vistas	Coste por clic
Informar	Participar, compartir
Sistemas de gestión de contenidos	Wikis
Directorios (taxonomía)	Etiquetas (folksonomía)
Fidelización	Sindicación
Sindicación de contenidos significa redifusión de contenidos	Publicidad contextual
Publicidad con banners y pop-ups	Publicidad contextual

Tabla 1: Comparativa entre Web 1.0 y Web 2.0

En la tabla siguiente se muestra una relación de aplicaciones en ambos ámbitos:

Web 1.0	Web 2.0
DoubleClick	Google AdSense
Ofoto	Flickr
Akamai	BitTorrent
mp3.com	Napster
Enciclopedia Británica	Wikipedia
webs personales	blogging
evite	upcoming.org y EVDB
screen scraping	servicios web
publicación	participación
sistema de gestión de contenidos	wiki
stickiness	sindicación

Tabla 2: Comparativa de aplicaciones entre Web 1.0 y Web 2.0

2.1.1 Tecnología

Una web que se caracterice por estar bajo la linea de Web 2.0 tiene que utilizar alguna de las siguientes técnicas:

- Técnicas:
  - CSS, marcado XHTML válido semánticamente y Microformatos
  - Técnicas de aplicaciones ricas no intrusivas (como AJAX)
  - Java Web Start
  - XUL
  - Sindicación/Agregación de datos en RSS/ATOM
  - URLs sencillas y con significado (SEM)
  - Soporte para postear en un blog
  - JCC y APIs REST o XML
  - JSON
  - Algunos aspectos de redes sociales
  - Mashup (aplicación web híbrida)
- General:
  - La información del sitio debe poderse extraer fácilmente
  - Los usuarios deberían controlar su propia información
  - Que desde el navegador se pueda utilizar completamente la aplicación

2.1.1 Redifusión de contenido

Uno de las premisas más importantes de la Web 2.0 es la de poder acceder a la información de forma fácil. Para ello se utilizan estándares definidos para poder normalizar la forma de acceder a la información y fomentar la sindicación a los sitios. Como usuario ponemos la información de nuestro sitio disponible a el resto del mundo mediante un estándar que facilita la extracción de la información. Como estándares tenemos los siguientes: RSS, RDF y Atom, todos ellos variedades del XML.

2.1.2 Servicios Web

La posibilidad de el envío bidireccional de mensajes con el servidor es otro de las pautas principales que definen la Web 2.0. Los dos tipos más importantes son los métodos RESTful y SOAP. REST indican un tipo de llamada a un servicio web donde el cliente transfiere el estado de todas las transacciones. SOAP y otros métodos similares dependen del servidor para retener la información de estado. En ambos casos, el servicio es llamado desde un API. Generalmente el lenguaje común de estos servicios web es el XML, si bien puede haber excepciones.

Otra forma de realizar comunicaciones con el servidor es la conocida como Ajax y que más tarde explicaremos en detalle. Mejora la experiencia la experiencia del usuario en las aplicaciones web basadas en el navegador. Mediante Ajax accedemos de forma asíncrona a métodos externos que se han programado en el servidor. Los mensajes que se envían no están normalizados sino que se hace necesaria la documentación de un API para poder utilizarlos. Su ventaja la facilidad de inclusión en una aplicación normal simplemente metiendo javascript.

2.2 Definición de Ajax

AJAX realmente no es más que una aproximación a la interacción web. Es más una técnica de programación que una tecnología, cuyo núcleo central es Javascript. Ni siquiera el corazón de cualquier aplicación Ajax, el objeto XHR, no es nada nuevo, ya que como hemos comentado previamente, se introdujo como control ActiveX en la versión 5.0 de Internet Explorer lanzada en 1999.

El 18 febrero del 2005 Jesse-James Garrett publico un artículo llamado “Ajax: A New Approach to Web Applications”. Es este artículo Garrett pone de manifiesto como se están acercando las aplicaciones web a las de escritorio. Define el término AJAX como etiqueta a la nueva generación de aplicaciones web, como un patrón de diseño de aplicaciones web

compuesto por muchas tecnologías e ideas.

Para hacer el sistema más usable básicamente agilizamos la transferencia de datos con el servidor descargando la cantidad de información precisa, teniendo la posibilidad de hacerlo en cualquier momento, y trasladando la carga de la presentación al navegador del cliente.

AJAX no es una tecnología sino que es un compendio de muchas para hacer realidad la mejora en la usabilidad:

- La presentación se basa en estándares basados en ***HTML/XHTML*** y estilos mediante ***CSS***
- Usamos el ***DOM***, Document Object Model, para actualizar dinámicamente el display de una página cargada
- El intercambio de datos mediante ***XML*** y la manipulación usando ***XSLT*** para convertirlo en XHTML
- La obtención de datos asíncronos mediante los objetos ***XMLHttpRequest*** ó ***XMLHttp***
- Con ***JavaScript*** desarrollamos la aplicación propiamente dicha

Por lo tanto AJAX, que significa “Asynchronous Javascript + XML” tendría que tener la siguiente definición: “Asynchronous Javascript + CSS + DOM + XMLHttpRequest”. De las tecnologías arriba citadas las tres imprescindibles son HTML/XHTML, DOM y JavaScript.

2.1 Modelo de comunicaciones cliente/servidor

El modelo de comunicaciones entre el cliente y el servidor se transforma cuando queremos aumentar la usabilidad de nuestras aplicaciones. Este modelo no cambia con la introducción de Ajax. Ya había cambiado con las técnicas de iframes ocultos donde se realizan también comunicaciones asíncronas con el servidor:

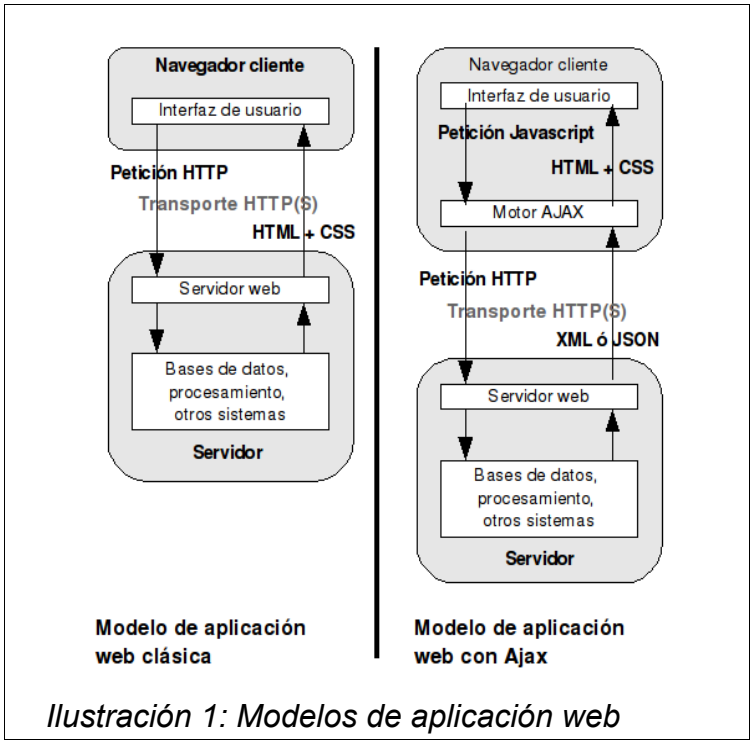

Ilustración 1: Modelos de aplicación web

- En un modelo web tradicional el usuario realiza las peticiones al servidor y espera la respuesta del mismo. Con esto conseguimos tener un patrón ***síncrono*** de acción/espera en la usabilidad de la aplicación.

- El modelo web con AJAX cambia bastante ya que tenemos un modelo ***asíncrono*** donde existe un intermediario que se encarga de realizar las peticiones al servidor.

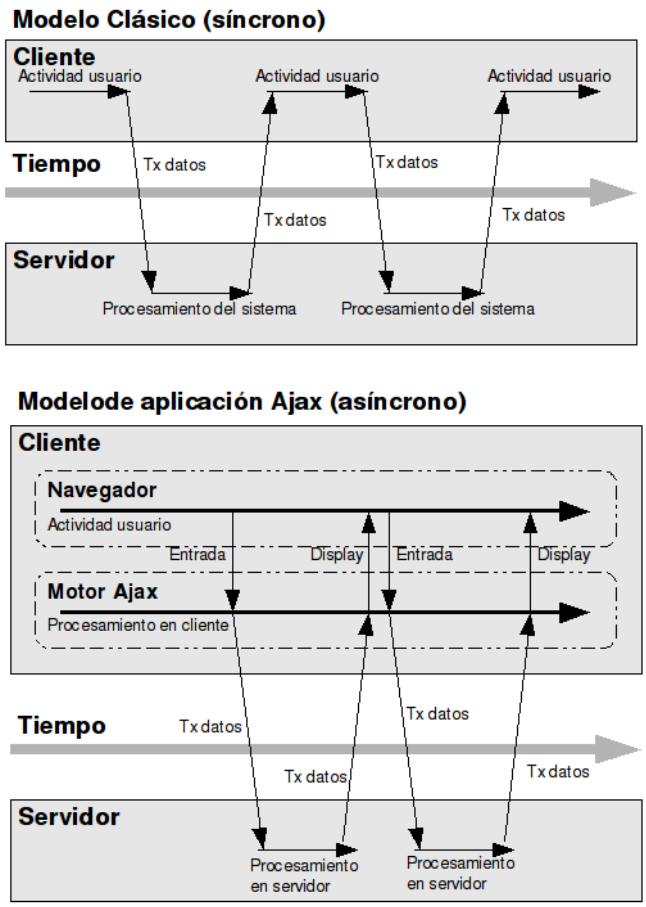

Ilustración 2: Esquema de tiempo de los dos tipos de diseño

Los ***ciclos en la comunicación*** con Ajax tienen el esquema siguiente:

- Inicialmente el navegador del usuario se conecta a la aplicación Ajax de la forma tradicional descargándose *todo el contenido.*
- El navegador carga el contenido inicial, el cual esta compuesto por el HTML a mostrar y que puede ser un plantilla, el CSS que establece el estilo y el *código JavaScript*, el cual es la aplicación propiamente dicha y se encarga de mantener la interacción de la aplicación
- Una vez que la aplicación se ha cargado en el navegador empieza la actividad de la misma, *controlada por eventos,* y en base a un intermediario mostrado en la figura como Ajax Engine.

- Un evento es cualquier acción que realice el usuario sobre la página. En el objeto HTML podemos definir funciones JavaScript que se ejecuten cuando se produce el evento deseado.
- Estas funciones pueden actualizar los datos de la página descargada, modificando el DOM, en base a datos actualizados del servidor obtenidos mediante una petición al mismo con el objeto Ajax Engine (objeto JavaScript XHR). En esta petición se configura el método a utilizar, la URL destino, sus parámetros y *se configura una función de vuelta para la respuesta del servidor*
- El servidor recibe la petición, la cual tiene el formato de petición GET o POST, y responde usando los estándares de HTTP. Desde el servidor *enviamos los datos estructurados en un formato definido* por el desarrollador y entendibles por la parte javascript del cliente. Este formato puede estar definido mediante XML o incluso puede ser una codificación de los datos directamente en forma de objeto Javascript.
- La función de vuelta configurada en el objeto Ajax Engine lee la respuesta del servidor y realiza las acciones que el programador haya definido, como mostrar la información en la página, modificar la estructura al disponer del DOM, actualizar variables del cliente, etc.

Como podemos ver, con una comunicación asíncrona, cambia la concepción de la navegación tradicional. Ahora tenemos que adaptarnos y estar abiertos a un nuevo tipo de aplicaciones web cuya navegación se basa en eventos y en la que cualquier parte de la página puede actualizarse sin necesidad de ser bajo demanda por parte del usuario. Esto que, a priori es una ventaja, puede convertirse en un problema debido a que el usuario esta acostumbrando a ver el refresco de toda la página, pudiendo pasar desapercibidas las actualizaciones de partes de la misma. Para tratar de ***centrar la atención del usuario*** podemos hacer uso de técnicas como Yellow Fade Technique, popularizada por 37signals en la aplicación Basecamp, la cual consiste en poner fondo amarillo a cualquier parte de la página que se haya actualizado, es decir, resaltar la zona actualizada de tal forma que llame la atención del usuario.

2.1 Características de una aplicación Ajax

Ajax como hemos dicho es un ***técnica de programación,*** una unión de tecnologías con una filosofía común, la de crear aplicaciones muy ricas y usables acercando el comportamiento de la misma a una versión de escritorio. Muchos son los desarrolladores que se han puesto manos a la obra para definir los patrones de una buena aplicación Ajax. Entre ellos esta

Michael Mahemoff, que es un desarrollador de software y experto en usabilidad que identifica varios principios clave para realizar buenas aplicaciones Ajax:

- ***Minimizar el tráfico:*** es importante minimizar las consultas y respuestas de la aplicación con el servidor evitando aquellas que sean innecesarias
- ***Sin sorpresas:*** siempre que se hace un interfaz con el usuario hay que pensar en el comportamiento que el usuario espera de una aplicación web. Con Ajax se han introducido otros muchos comportamientos como coger y arrastrar, doble click, etc que pueden sorprender al usuario
- ***Uso tradicional:*** es mejor utilizar patrones de interacción ya familiares por el usuario en aplicaciones web y de escritorio, que desarrollar nuevos modelos que no sean conocidos
- ***Evitar elementos innecesarios de distracción*** en la página como secciones que parpadeen, animaciones en bucle, etc
- ***Accesibilidad:*** cuando se desarrolle hay que pensar a que tipo de usuarios esta dirigida la aplicación y desde donde pueden acceder a ella. Hay que tratar de llegar al máximo número posible.
- ***Evitar carga de páginas enteras*** salvo en los momentos precisos
- Cuando se diseñe la aplicación nos tenemos que ***poner en el lugar del usuario*** y hacer fácil el uso de los casos más comunes

2.1 Ajax vs IFrame

Tras la aparición de Ajax como técnica de programación cuyo núcleo central es el objeto XHR, muchos de los programadores se preguntaban que qué les aportaba esta técnica ya que ellos ya realizaban comunicaciones asíncronas con el servidor de una manera exitosa mediante la técnica del Iframe oculto. La verdad es que ninguna de las dos es mejor ni peor solo tienen comportamientos diferentes para diferentes situaciones.

Aparte de que la técnica del Iframe oculto es un hack en el comportamiento del mismo, frente a un objeto JavaScript desarrollado para dar ese funcionamiento, las diferencias de comportamiento se pueden resumir en la siguiente tabla:

	XMLHttpRequest	Iframe
Multithread	si	no
Botón pagina anterior	no	si
Sitios cruzados	no	si
Estado respuesta	si	no

Tabla 3: Diferencias de comportamiento entre técnica XHR e Iframe

donde:

- ***Multithread***: XHR nos permite realizar peticiones simultaneas simplemente con instanciar un nuevo objeto XHR. Con iframes tendríamos que crear un conjunto de funciones que creasen iframes "al vuelo" basándonos en DOM, usarlos para la petición/respuesta y eliminarlos una vez usado.
- ***Botón de la página anterior:*** No podemos acceder a la página anterior porque con XHR tenemos una navegación basada en eventos y por lo tanto la página siempre es la misma. En muchos casos es deseable este comportamiento para no volver a chequear formularios por error.
- ***Sitios cruzados:*** debido a la naturaleza de XHR no se permite realizar consultas a dominios distintos del que me he bajado la aplicación por cuestiones de seguridad. Con Iframes no hay problemas en realizar peticiones a otros dominios.
- ***Estado de la respuesta:*** con Iframes no sabemos en que estado esta nuestra petición. Con XHR tenemos 5 estados: 0 (sin inicializar), 1 (cargando), 2 (cargado), 3 (interactivo) y 4 (completo). Esto es muy útil para darle una información más precisa al usuario.

2.1 El objeto XHR

Como ya hemos comentado anteriormente el objeto XMLHttp fue introducido por primera vez por Internet Explorer 5.0 como objeto ActiveX y seguidamente por Mozilla 1.0 y Safari 1.2 como objeto XMLHttpRequest. Es importante subrayar esta distinción, ya que como veremos, tendremos que saber en que tipo de navegador estamos para crear el objeto JavaScript de una forma u otra.

El objeto XHR no es un estandar de la W3C por lo que su implementación puede variar de un navegador a otro siendo muy similares en comportamiento Firefox, Safari, Opera, Konqueror e Internet Explorer. El estandar de la W3C que hace cosas similares es "DOM Level 3: Save and Load Specifications"

Antes de poder usar un objeto XHR tendremos que crearlo mediante JavaScript. Como ya hemos comentado Internet Explorer lo implementa mediante un objeto ActiveX y otros como un objeto nativo de JavaScript. Estas diferencias se tendrán que contemplar en la ***función de creación del objeto*** que básicamente se reduce a chequear si nuestro navegador soporta objetos ActiveX. El código que vamos a utilizar para crear objetos es el siguiente donde se puede observar esa comprobación

```
var xmlHttp;

function createXMLHttpRequest() {
      if (window.ActiveXObject) {
            xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
      }
      else if (window.XMLHttpRequest) {
            xmlHttp = new XMLHttpRequest();
      }
}
```

Al ser un objeto tendremos definidas unas series de ***métodos y propiedades***. Están agrupadas en las siguientes tablas:

Métodos	Descripción
abort()	Detiene la petición actual
getAllResponseHeaders()	Devuelve todas las cabeceras como pares clave/valor
getresponseHeader(x)	Devuelve el valor de la cabecera x como un string
open('method','URL','a')	Especifica el método HTTP (por ejemplo, GET, POST o PUT), la URL objetivo, y si la petición debe ser manejada asíncronamente (Si, a='True' defecto; No, a='false'.)
send(content)	Envía la petición, opcionalmente con datos POST
setRequestHeader('x','y')	Configura un par parámetro y valor, x=y, y lo asigna a la cabecera para ser enviado con la petición

Tabla 4: Métodos del objeto XHR

Algunos de estos métodos tienen peculiaridades añadidas:

- `void open(string method, string url, boolean asynch, string username, string password):` Podemos introducir un usuario y una contraseña.
- `void send(content):` Si la petición es asíncrona el método sale inmediatamente. Si no lo es se queda esperando a la respuesta del servidor. Es en este método donde incluiremos los datos formateados de peticiones POST. En el caso de ser una

petición GET tendremos que invocarlo también pero enviando el valor “null” ya que en esta petición no hay datos en el cuerpo de la misma.

En suma a estos métodos tenemos las siguientes propiedades:

Propiedades	Descripción
onreadystatechange	Determina que gestor de evento será llamado cuando la propiedad readyState del objeto cambie
onreadyState	Número entero que indica el estatus de la petición: 0 (No iniciada), 1 (Cargando), 2 (Cargado), 3 (Interactivo), 4 (Completado)
responseText	Datos devueltos por el servidor en forma de string de texto
responseXML	Datos devueltos por el servidor expresados como un objeto documento
status	Código estatus HTTP devuelto por el servidor
statusText	“Phrase reason” HTTP devuelta por el servidor

Tabla 5: Propiedades del objeto XHR

Es importante notar las siguientes peculiaridades de las propiedades:

- `onreadystatechange`: aquí indicaremos que función javascript se va a encargar de procesar la salida. Esta función podrá acceder a la propiedad onreadyState viendo en que estado está la petición y realizando las operaciones que considere pertinentes. Se debe registrar la función antes de llamar al método open.
- `onreadyState`: esta propiedad nos ayudará en la función respuesta a saber en que estado esta la petición
- `responseText`: si utilizamos algún tipo de codificación de datos a javascript en el servidor, del estilo de JSON, esta es nuestra propiedad para sacar los datos respuesta. Luego tendremos que hacer un eval del string recibido para ejecutar el código javascript procedente del servidor

En este punto ya sabemos como inicializar una comunicación con el servidor. A modo de resumen los ***pasos para iniciar una comunicación*** serían los siguientes:

1. Crear un objeto XHR
2. Registrar la función de vuelta para procesar la salida

3. Invocar al método open del objeto con la URL destino con las variables codificadas si estamos hablando de una petición GET, el tipo de método deseado (GET, POST o PUT) y si vamos ha realizar una petición asíncrona.
4. Invocar el método send con las variables codificadas si es método POST o con el valor “null” si es método GET

Un esquema de este proceso es el siguiente:

```
var xmlHttp;

function createXMLHttpRequest() {
      if (window.ActiveXObject) {
            xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
      }
      else if (window.XMLHttpRequest) {
            xmlHttp = new XMLHttpRequest();
      }
}
function startRequest() {
      createXMLHttpRequest();
      xmlHttp.onreadystatechange = handleStateChange;
      xmlHttp.open("GET", "datosServidor.php", true);
      xmlHttp.send(null);
}
function handleStateChange() {
      if(xmlHttp.readyState == 4) {
            if(xmlHttp.status == 200) {
                  //Codigo   de   respuesta   donde   podremos
utilizar responseXML o responseText
            }
}
```

donde llamaríamos a la función `startRequest()` mediante el evento de algún objeto de la página web

Como ya hemos comentado, por **razones de seguridad**, el método open solo es aplicable a URLs destino que estén dentro del mismo dominio que aplicación original, evitando que se pueda modificar cualquier petición hacia servidores no confiables. Dependiendo de cada navegador obtenemos más o menos severidad si nos saltamos esta restricción: internet explorer simplemente nos muestra una alerta pero nos permite continuar, y Firefox por el contrario detiene la petición y nos muestra un error por la consola de errores JavaScript.

2.1 Comunicaciones con el servidor

En los apartados anteriores hemos visto qué elementos componen Ajax y cuales son los pasos básicos para establecer una comunicación con el servidor mediante el protocolo HTTP.

Las tecnologías a utilizar en el servidor pueden ser varias y utilizadas habitualmente para realizar aplicaciones web como pueden ser PHP, Java, Ruby, Perl, ASP, etc.

Según hemos visto la aplicación cliente hace un descarga inicial de todos los datos, y durante el trascurso de la sesión del usuario, va realizando consultas al servidor mediante JavaScript y el objeto XHR en base a eventos producidos en el navegador. Es en estas descargas intermedias, y en cómo tiene que ser el formato de la información, donde vamos a centrarnos ahora.

2.1.1 Recepción de la petición en el servidor

Nuestro script en el servidor debe leer los parámetros de entrada y realizar la acción para la que esta encomendado. Luego deberá mostrar la salida en el formato concretado. Un fragmento de código en PHP para leer las variables de entrada podría ser el siguiente:

```
<?php
// Fijamos las variables que vamos a recibir del formulario
$varsFormulario=array('parametro1','parametro2','parametro3'
,...,'parametroN');
// Fijamos las variables obligatorias que no pueden ser
nulas
$varsFormularioNoNulas=array('parametro1','parametro2');

// Hacemos globales las variables recogidas
foreach ($varsFormulario  as $var)
{
      if (!isset($_GET[“$var”]))
      {
            $$var=$_GET[“$var”];
      } else {
            $$var=””;
      }
}

// Verificamos las variables obligatorias
$error=0;
foreach ($varsFormularioNoNulas  as $var)
{
      if (!$$var)
      {
            $error=1;
      }
}
if ($error)
```

```
{
      die("Error");
}

// aquí realizamos la función deseada
...
...
...
?>
```

2.1.2 Formato de las respuestas

Las peticiones realizadas por los clientes son peticiones HTTP donde se invoca una acción en una URL remota en base a un método y a unas variables de entrada. La respuesta esperada del servidor debe tener un formato acordado previamente el cual puede ser de diferentes tipos:

- ***Texto***: El script servidor simplemente envía texto, el cual puede ser código HTML que el cliente escriba directamente. La variable de cabecera HTTP "Content-type" debe ser fijada a "text/plain"
- ***XML:*** Los datos respuesta van tipados en XML lo cual nos facilita el poder separar fácilmente cada uno de los datos al convertirse en un objeto JavaScript en el cliente. La variable de cabecera HTTP "Content-type" debe ser fijada a "text/xml"
- ***Conversiones directas a objetos JavaScript***, como el caso de JSON, el cual nos traduce directamente un objeto de un lenguaje origen a un string que representa un objeto en JavaScript. La variable de cabecera HTTP "Content-type" debe ser fijada a "text/plain"

La ***respuestas en modo texto*** son útiles cuando nuestra petición no requiere tratar la información recibida, sino realizar tareas simples donde la respuesta es directamente la información a tratar. Por ejemplo sería interesante en peticiones que nos alerten de mensajes de error, información para usuarios, etc.

Las ***respuestas en XML***, al igual que las conversiones directas a objetos JavaScript, nos permiten obtener respuestas en formato de objetos en el mundo JavaScript. Es útil ya que tendremos una representación única de los datos, lo cuales pueden ser más complejos, y pueden ser utilizables en diferentes lugares de la aplicación.

Las ***conversiones directas a JavaScript*** se utilizan como medida de ahorro en ancho

de banda al enviar la misma información que en formato XML pero si tanta carga de datos por el etiquetado XML. Directamente se envía la representación del objeto en JavaScript.

2.10.2.1 JSON

JSON nace como ***alternativa a XML,*** para reducir el tamaño de la respuesta quitando el etiquetado. Es una representación en modo texto de un objeto JavaScript que se realiza mediante una la librería existe para la mayoría de los lenguajes.

En el siguiente ejemplo podemos ver como quedaría una representación de un objeto en JSON y su pareja en XML

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

Y en XML:

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

La forma en la que la librería JSON realiza esta conversión dependiendo del tipo de dato origen se puede resumir en las gráficas siguientes:

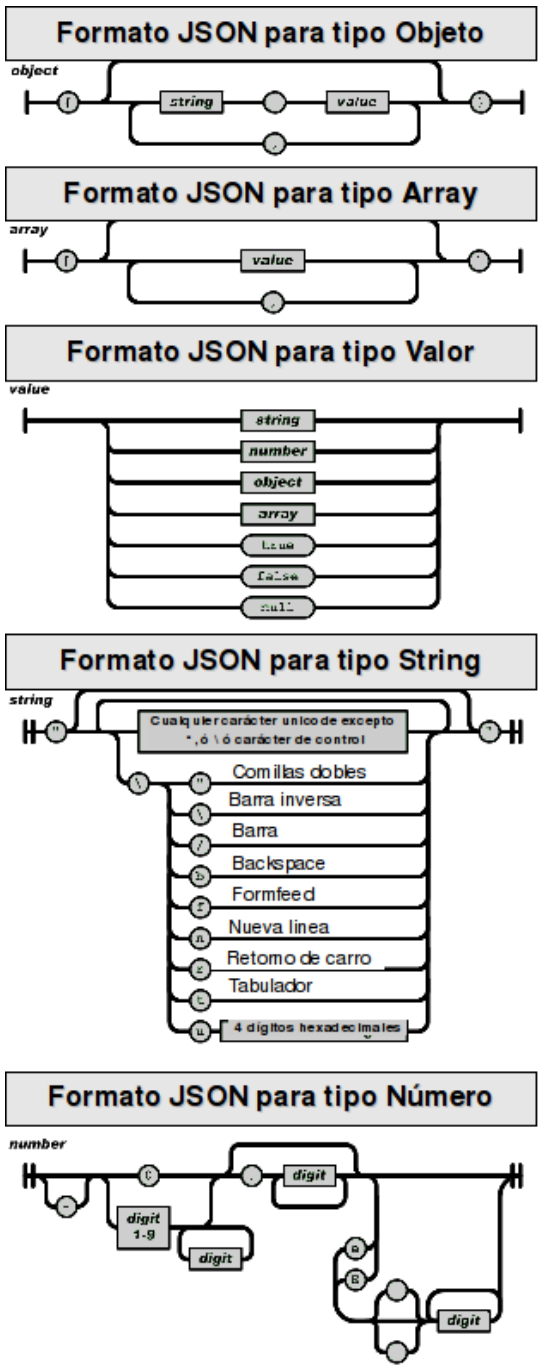

Ilustración 3: Resumen de conversiones de tipos a JSON

2.1.1 Envío de datos hacia el servidor

Dependiendo del tipo de petición HTTP que hagamos al servidor tendremos que realizar

lo siguiente:

- Tanto para **GET como POST** tendremos que crear un string siguiendo el formato de "var1=valor1&var2=valor2&..&varN=valorN"
- Para el caso de **GET** las variables irán anexados a la URL destino mediante el carácter "?", y para POST los datos irán en el cuerpo de la petición. En este último caso utilizaremos el método send del objeto XHR con el string creado a partir de las variables del formulario
- Para el caso de **POST** tendremos que añadir la variable de cabecera "Content-Type" como "application/x-www-form-urlencoded"

El siguiente ejemplo muestra como sería el código JavaScript de la petición

```
function createQueryString() {
      var var1 = document.getElementById("var1").value;
      var var2 = document.getElementById("var2").value;
      var queryString = "var1=" + var1 + "&var2=" + var2;
      return queryString;
}

function doRequestUsingGET() {
      createXMLHttpRequest();
      var queryString = "GetAndPostExample?";
      queryString = queryString + createQueryString()
      + "&timeStamp=" + new Date().getTime();
      xmlHttp.onreadystatechange = handleStateChange;
      xmlHttp.open("GET", queryString, true);
      xmlHttp.send(null);
}
function doRequestUsingPOST() {
      createXMLHttpRequest();
      var url = "GetAndPostExample?timeStamp=";
      url += new Date().getTime();
      var queryString = createQueryString();
      xmlHttp.open("POST", url, true);
      xmlHttp.onreadystatechange = handleStateChange;
      xmlHttp.setRequestHeader("Content-Type",
      "application/x-www-form-urlencoded;");
      xmlHttp.send(queryString);
}
```

Otra cosa a notar de las peticiones a realizar al servidor es que con la finalidad de *evitar que el navegador utilice la pagina cacheada y forzar a que haga la petición al servidor,* se añade a la URL destino una variable con un contenido que difiera en cada petición, como el timestamp multiplicado por un número aleatorio o similar.

2.1.1 Tratamiento de la respuesta en el navegador

Podemos leer la respuesta del servidor de varios modos diferentes en base a las dos propiedades del objeto XHR: responseText y responseXML

2.1.1.1 Procesamiento de las respuestas en modo texto

Al utilizar la propiedad del objeto XHR ***responseText***, convertimos la respuesta del servidor en un string en la aplicación JavaScript cliente.

Podemos utilizar esta técnica para enviar directamente mensajes del servidor y representarlos en la página del navegador. La problemática es que al ser datos en modo texto no disponemos de una estructura de datos que nos permita jugar con la información, e incluso, incluirlos de una manera lógica en la estructura DOM de la página. Si no queremos complicar demasiado la programación, este modo de respuesta nos será interesante solo cuando la información a mostrar ya venga maquetada por el servidor, pero no será útil si queremos mantener una estructura de datos en el cliente.

Es opción es muy utilizada en conjunción con la propiedad ***innerHTML*** de los elementos HTML la cual nos permite escribir directamente el código HTML dentro de una etiqueta, usualmente dentro de la etiqueta .

La unión de responseText e innerHTML nos simplifica mucho la capa de presentación en el cliente al trasladarla al servidor, pero estamos hablando de una propiedad HTML no estándar y que varia su implementación dependiendo del navegador. De todos modos perdemos un poco de la filosofía de Ajax al engordar las comunicaciones con código HTML generado en el servidor

Para hacer un poco más práctica esta propiedad del objeto XHR y con el fin de hacer más pequeñas las repuestas que el XML, utilizamos JSON. JSON se envía en formato texto desde el servidor y una vez leído en el cliente se hace uso de la función **eval** para interpretar el contenido de este string y crear el objeto. La forma de acceder al objeto creado es igual que cualquier objeto JavaScript donde cada propiedad es una representación de la propiedad del objeto original, es decir, no hay que navegar en la estructura de datos buscando etiquetas como en XML sino que accedemos directamente.

2.1.1.2 Procesamiento de las respuestas en modo XML

Cuando tenemos estructuras complejas hemos visto que podemos utilizar el modo texto con JSON, pero también podemos enviar la respuesta desde el servidor en formato XML. El

hacerlo en este formato nos garantiza que nuestras respuestas las pueda entender cualquier cliente, incluso los que no utilicen JavaScript.

En los navegadores modernos se maneja la respuesta XML en sintonía con W3C DOM. Esto significa que vamos a poder utilizar el API de funciones especificado por W3C DOM para navegar por el documento XML ampliando así su compatibilidad con diferentes navegadores.

Este API en JavaScript se resume en las siguientes propiedades y métodos DOM del objeto XML:

Nombre de la propiedad	Descripción
childNodes	Devuelve un array con los hijos del elemento actual
firstChild	Devuelve el primer hijo directo del objeto actual
lastChild	Devuelve el último hijo directo del objeto actual
nextSibling	Devuelve el elemento inmediatamente siguiente al actual
nodeValue	Especifica la propiedad de lectura/escritura que representa el valor del elemento
parentNode	Devuelve el nodo padre del actual
previousSibling	Devuelve el elemento inmediatamente precedente al actual

Tabla 6: Propiedades del objeto XML de Javascript

Nombre del metodo	Descripción
getElementById(Id)(document)	Obtiene el elemento en el documento que tiene el id único especificado
getElemenstByTagName(name)	Devuelve un array de con los hijos del elemento especificado por la etiqueta
hasChildNodes()	Devuelve un booleano indicando si el elemento tiene hijos
getAttribute(name)	Devuelve el valor del atributo del elemento especificado por name

Tabla 7: Métodos del objeto XML de Javascript

2.1.1.3 Edición dinámica del contenido de la página

En apartados anteriores hemos visto que podemos modificar el contenido de la página dinámicamente de una manera sencilla mediante el uso de la etiqueta innerHTML. También hemos comentado es complicado manejar estas etiquetas como si fueran datos de la aplicación y que lo interesante es que los datos tengan su representación en el DOM.

Para manipulación del DOM tenemos también un conjunto de funciones y métodos útiles

Nombre del propiedad/metodo	Descripción
document.createElement(tagName)	Crea el objeto identificado por la etiqueta. Por ejemplo si la etiqueta es un div crea un elemento
document.createTextNode(text)	Crea un elemento que contiene texto estático
<element>.appendChild(childNode)	Añade un elemento a la lista de elementos hijos del elemento actual
<element>.getAttribute(name)	Obtiene el valor de un atributo del elemento
<element>.setAttribute(name,value)	Fija el valor de un atributo
<element>.insertBefore(newNode, targetNode)	Inserta el nuevo objeto antes del objeto indicado el cual es hijo del elemento actual
<element>.removeAttribute(name)	Borra el nombre del atributo del elemento
<element>.removeChild(childNode)	Borra el elemento childNode del elemento
<element>.replaceChild(newNode, oldNode)	Cambia un nodo por otro
<element>.hasChildnodes()	Devuelve un booleano indicando si el elemento tiene hijos

Tabla 8: Métodos para la manipulación del DOM en Javascript

Para la búsqueda y situación de elementos dentro del DOM usamos las funciones definidas en el apartado anterior.

2.2 Librerias y frameworks de Ajax

Con la finalidad de facilitar la tarea de programación de aplicaciones Ajax han ido apareciendo bastantes librerías JavaScript y frameworks que ayudan a esta tarea. Las librerías se encargan de agrupar las funciones más comunes a realizar en el lado de cliente, consiguiendo agilizar y resolver de una manera muy eficiente estas tareas comunes con todas las posibles peculiaridades. Los frameworks por otro lado están orientados al lado del servidor facilitando la tarea de escritura del código JavaScript que se va a ejecutar en el cliente y que va a estar contenido en la página inicial de descarga.

Si atendemos a la clasificación que se hace en http://ajaxpatterns.org/Ajax_Frameworks podemos ver que tenemos frameworks clasificados de la siguiente forma:

2.2.1 Javascript

En esta sección tenemos los frameworks que facilitan las tareas del cliente JavaScript. Tenemos la siguiente subdivisión:

- ***Popular***:
  - Frameworks javascript de *propósito general:* agrupación de funciones más comunes, interfaces de propósito general, etc.
  - Frameworks javascritp *especializados en comunicaciones*
- **Especializados**:
  - Frameworks Javascript basados en *efectos visuales*
  - Frameworks Javascript basados en *integración con flash*
  - Frameworks Javascript basados en *logging* con la finalidad de ayudar al desarrollo
  - Frameworks Javascript basados en *XML*

Son destacables por su carácter fundamental para otros frameworks las librerías **prototype.js**, donde aparecen funciones para la ayuda a la programación javascript, y **Script.aculo.us**, para la creación de efectos visuales.

2.2.1 Desde el servidor e híbridos

Estos frameworks son los que ayudan la escritura de aplicaciones Ajax desde el servidor y están orientados a diferentes lenguajes de programación. Tenemos para C++, C#, Coldfusion, DoNet, Java, Lisp, Lotus Notes,Perl, PHP, Python, Ruby,Smalltalk.

2.3 Alternativas a Ajax

Ajax, como cualquier tecnología nueva, tiene sus alternativas. Vamos a hablar de las mismas en el sentido de las nuevas aplicaciones web, unidas al fenómeno web 2.0, y que tendrán más o menos ventajas/inconvenientes que Ajax como técnica.

Para el desarrollo de RIA, Rich Internet Applications, tenemos las siguientes alternativas a Ajax:

1. ***XUL***

   Pronunciado como “zool” es un lenguaje de etiquetado de alto rendimiento para la creacción de interfaces de usuario ricos en dinamismo y con posibilidad de trabajar con o sin conexión a internet. Es parte del navegador Mozilla por lo que las aplicaciones realizadas con XUL, widgets, solo estarán disponibles para sus él y sus derivados. Aunque no es un estándar, XUL esta basado en HTML 4.0, CSS, DOM, XML y ECMAScript.

Como ventajas tenemos su alto rendimiento, su rapidez, que trabaja con JavaScript y que esta basado en XML.

Como inconvenientes que esta basado en el motor Gecko, por lo que solo son aplicaciones accesibles por navegadores que soportan este motor como todos los derivados de Mozilla.

2. XAML

Pronunciado como "zammel", XALM es un lenguaje de etiquetado de alto rendimiento destinado a la creación de interfaces de usuarios ricos en dinamismo. Es propiedad de Microsoft y forma parte de la nueva generación de tecnología de interfaz soportadas en las futuras versiones de internet explorer. Es resultado de la convergencia de las RAD tools y del navegador como un plataforma de aplicaciones para empresa. Basado en la plataforma .NET, XAML esta compilado dentro de las clases. NET.

Las ventajas que tiene son su alto rendimiento, su robustez y que es altamente configurable.

Como limitaciones que es una tecnología que solo funciona en plataformas Microsoft y que esta disponible a partir de la versión Vista del sistema operativo.

3. Java Applets

Son programas escritos en Java que corren en el navegador del cliente. Los navegadores requieren el plugin de Sun para poder ejecutarlos.

Como ventajas que son muy rápidos, soportados en la mayoría de las plataformas, mediante plugin

Como desventajas que requieren el plugin de Sun el cual, por motivos de política de seguridad en una organización, puede estar deshabilitado en los navegadores.

4. Macromedia Flash y Shockwave

Permite la creación de interfaces con un alto nivel de configuración.

Como ventajas la compatibilidad de navegadores y plataformas, la rapidez y la flexibilidad y el incremento en la potencia de las herramientas de desarrollo.

Como desventajas que depende de tener instalado el plugin, el cual es propietario de Adobe, y la carga de la aplicación.

5. SVG (Scalable Vector Graphics)

Es un lenguaje de gráficos basado en texto que describe imágenes con formas vectoriales, texto y gráficos embebidos. Puede ser integrado con fuentes de datos XML o SMIL (Syncronized Multimedia Integration Language). Tiene buena interoperatibilidad con CSS y JavaScript.

Como ventajas la velocidad y la flexibilidad. Como desventajas que requiere de plugins propietarios y que los conocimientos para el desarrollo de estos componentes no son accesibles por muchos. Es un lenguaje que esta en desarrollo.

Capítulo 3. Visión general de la aplicación y modelado de datos

3.1 Descripción de la funcionalidad de la aplicación

CITAS2.0 es un sistema que nos permite organizar eventos de forma colaborativa entre lo participantes. Esta colaboración se realiza mediante las votaciones de unas fechas propuestas por el administrador, o creador de la invitación, por todos los participantes en el evento. La aplicación se encarga de dar al usuario la herramienta para controlar estas invitaciones y de dar la interfaz de votación a los participantes en el evento.

Una invitación va a estar compuesta por un **conjunto de fechas propuestas** elegidas, un **tipo de votación,** que va a definir como se ha de participar en la elección de la fecha, y un **conjunto de opciones** que van a permitir configurar la información que se ha de enviar a los participantes, la que es visible desde el interfaz de votación, la forma en la que se van a notificar las votaciones al administrador y el sistema de recordatorios sobre las votaciones ya elegidas.

Una vez que la invitación se constituye, se enviarán a los participantes las correspondientes invitaciones electrónicas al evento mediante un correo electrónico, y con los datos que el administrador ha decido que sean visibles por los participantes. En esta invitación, además de los datos del evento, se facilita una url con el acceso a una interfaz de cliente que permite poder realizar esta votación sin estar registrado.

El participante, además del correo con la invitación inicial, también puede recibir de la aplicación los mensajes que el administrador desee enviarle. Esto es porque también se facilita un aplicación de envío de mensajes para el administrador del evento. También el administrador recibirá por correo electrónico cada una de las votaciones realizadas por los participantes y un mensaje cuando la elección este realizada.

El administrador, como tal, tiene el control sobre todo el proceso de elección de la propuesta, pudiendo realizar las modificaciones que estime oportunas en las votaciones y en la definición de la invitación, como modificar el número de componentes, las fechas propuestas, los datos de la invitación. Todo esto se consigue desde la interfaz de control de las invitaciones

dispuesto a cada usuario registrado,

Si un usuario decide registrarse dispondrá de la posibilidad de crear nuevas citas así como controlar y almacenar las citas a las que ha sido invitado.

3.2 Descripción de la implementación de la aplicación

El sistema tiene dos implementaciones bien diferenciadas, una siguiendo pautas de web 2.0 en la que se antepone la usabilidad a la accesibilidad, y otra que esta orientada a la accesibilidad para aquellos dispositivos más sencillos.

La parte orientada más a la usabilidad esta basada en técnicas de programación asíncronas con el servidor siguiendo técnicas Ajax. De tal modo que, en una primera instancia nos bajamos del servidor una plantilla a rellenar y el programa javascript concreto para la aplicación solicitada. Mediante peticiones asíncronas el navegador del cliente va solicitando la información concreta necesaria y escribe sobre la plantilla los datos necesarios para mostrar al usuario la información requerida. Esto nos da rapidez en la ejecución de la aplicación, así como sensación de tiempo real al tener actualizaciones automáticas.

La parte más orientada a la accesibilidad esta basada en técnicas de programación más tradicionales donde el servidor envía toda la información en la página descargada inicialmente. Mediante el uso de programación en el servidor tenemos la posibilidad de construir contenido dinámico en la aplicación, pero siempre en cada petición, nos tendremos que descargar toda la información del sistema, tanto la que se actualiza como la que es estática. Para ajustar tiempos y evitar que la aplicación sea muy pesada, se han reducido algunas de las funcionalidades que se tienen en la parte 2.0.

3.3 Introducción a las partes de la aplicación CITAS2.0

Durante este apartado se van a describir las parte lógicas en las que se puede separar la aplicación. Se parte de la idea de separar las partes de la aplicación que son comunes a cualquier aplicación web, de las partes que dan la funcionalidad a la aplicación CITAS2.0. El objeto de esta separación es simplemente la de reutilizar todo lo común a cualquier aplicación web uniéndolo en un conjunto de librerías, una estructura de directorios y una lógica de acceso, que van a facilitar el desarrollo de la aplicación CITAS2.0. El considerar esta parte como un servidor de aplicaciones propiamente dicho es incorrecto, ya que principalmente son un conjunto de librerías que tratan de solventar el control de acceso y el manejo de la salida de las peticiones de los clientes. Un término adecuado para definirlo sería el de **gestor de aplicaciones.**

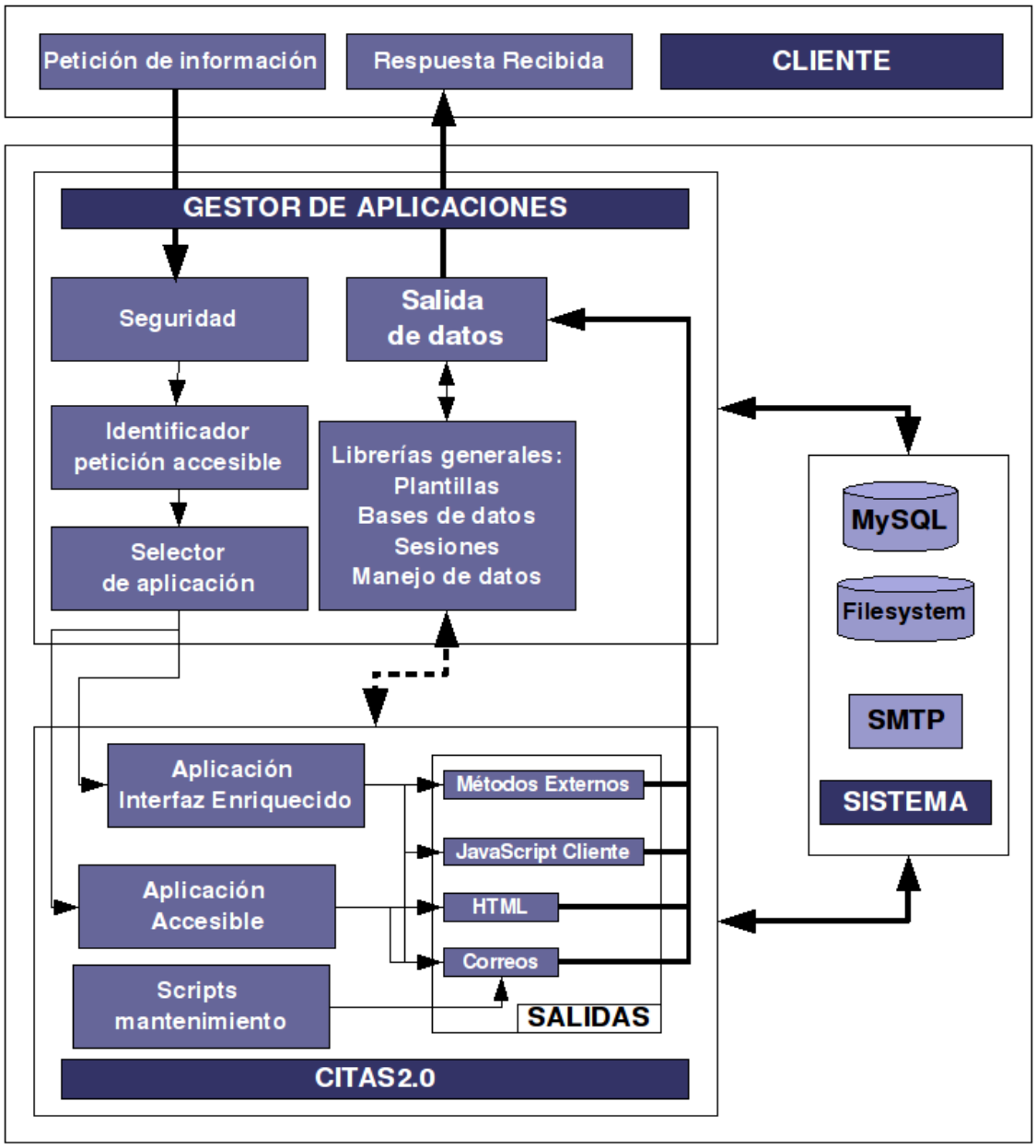

Ilustración 4: Diagrama de bloques del sistema

En la ilustración 4 se muestran los grandes bloques lógicos que definen la aplicación así como las interrelaciones que puede haber entre ellos. A lo largo de los puntos siguientes se van a ir explicando cada uno de ellos un poco más en detalle.

3.3.1 El cliente

El cliente es el que se encarga de realizar las peticiones a la aplicación web. Cuando un

cliente accede al sistema, se determina que tipo de terminal esta realizando la petición atendiendo a la variable UserAgent de la petición HTTP. Dependiendo de este valor, el gestor de aplicaciones decide entre una petición cliente que va a ser tratada por la parte de interfaz enriquecido o una petición cliente que va ser tratada la parte más centrada en la accesibilidad.

Las comunicaciones con el servidor van a depender de el tipo de cliente que usemos. Si el cliente es compatible con la interfaz enriquecida hará peticiones asíncronas al servidor durante toda la ejecución de la aplicación y en base a los eventos generador por el usuario. Este trasiego de información bajo demanda y específica hace que la aplicación tenga un uso más agradecido. Todo esto se consigue si el cliente sabe interpretar Javascript, lo cual es ya común en todos los navegadores de escritorio.

Si por el contrario el gestor de aplicaciones identifica que la petición proviene de un terminal accesible, el gestor de aplicaciones ejecutará la versión accesible de la aplicación, la cual solicita refresca la información nueva de la acción en cada descarga total de la página síncrona realizada por el usuario. El cliente aquí solo ha de saber interpretar HTML ya que la aplicación le da ya la salida procesada de la petición solicitada.

3.3.2 El gestor de aplicaciones

La necesidad de este bloque, como se ha explicado anteriormente, es la de la separación de funciones comunes a una aplicación web de las que son propias de la aplicación a implementar.

Las ventajas de realizar este bloque son principalmente la comodidad que aporta el tener un conjunto de recursos que se encargan de dar los elementos comunes de la aplicación, y que se han reutilizado en cada una de las subaplicaciones que componen CITAS2.0.

Como elementos comunes tenemos:

- El **procesamiento de plantillas** de salida de datos como pueden ser el código HTML enviado al cliente, el código JSON o XML que se da en cada petición asíncrona, etc. Esta función se encarga de dar un conjunto de métodos que tienen la finalidad de rellenar una plantilla indicada inicialmente, que puede contener algunos comodines de control, con los datos que el programador de la aplicación ha ido introduciendo en la clase. Se consigue formatear los datos introducidos en el objeto en una salida definida por la una plantilla.
- Se facilita también un objeto para acceder a **bases de datos**. Mediante este recurso se le da al programador de la aplicación un nivel más sencillo de acceder a bases de datos
- **Manejo de la seguridad y sesiones**. Se facilita una librería que implementa un conjunto de funciones y métodos para disponer de seguridad de acceso en nuestra aplicación, así como

el manejo de sesiones de acceso y variables asociadas a cada sesión para aumentar la rapidez de ejecución global.

- **Manejo de datos de entrada**. Igualmente se ha implementado una librería que ayuda en la extracción y comprobación de los datos que provienen de las peticiones realizadas sobre la aplicación. Se encarga también de introducirlas en el sistema

Además de estas librerías el gestor necesita un desarrollo que de la inteligencia de seguridad de acceso y selector de aplicaciones. Esto se ha implementado en el índice de la aplicación web limitando el acceso a cualquier aplicación directamente y forzando a que sea este índice el que importe cada una de las aplicaciones solicitadas.

También, y como medida centralizada de control del funcionamiento del gestor, se da la opción de controlar el gestor mediante parámetros contenidos en un fichero de configuración. Estos parámetros nos permiten definir que UserAgents van a ser accesibles, donde esta la base de datos, que tablas contienen la información de usuarios para la seguridad, que alias se ha definido para cada aplicación y cual es su path real, que aplicaciones no requieren seguridad, cual es el usuario de correo por defecto, etc. Se verán en más detalle cuando se expliquen los bloques de este gestor.

3.3.1 El sistema

La aplicación esta implementada en php y esta instalada en un servidor web Apache con el correspondiente módulo de php. También requiere de una base de datos MySQL montada sobre el mismo servidor.

A parte de las peticiones realizadas por los clientes soportadas por el servidor web, el sistema necesita disponer de salida SMTP para poder enviar los correos pertinentes a los participantes de cada aplicación así como los recordatorios sobre la proximidad de la fecha elegida.

También se requiere que el sistema tenga el interprete php en modo de comando ya que se la parte de recordatorios va implementada en un script que se ejecuta periódicamente gracias al cron de la máquina.

3.3.2 CITAS2.0

Como ya se ha visto en los apartados anteriores, la aplicación realizada tiene que tener la posibilidad de atender peticiones provenientes de terminales que pueden o no soportar Javascript. Este criterio es el que va a tomar el gestor de aplicaciones para ejecutar la aplicación en modo accesible, o en modo de interfaz enriquecido.

La decisión de realizar dos implementaciones distintas para un tipo de orígenes u otros es debido a la complejidad que tiene el interfaz y que es necesaria para hacerlo más usable. El código javascript no se utiliza simplemente como refuerzo de la aplicación en el modo de interfaz enriquecido, sino que es la base del interfaz.

Realizar una aplicación mixta basada en etiquetado <noscript> en el HTML de la página es una buena solución para realizar la integración con distintos dispositivos. No se ha implementado esta posibilidad en CITAS2.0 ya que no se ve una forma de aunar el código a ejecutar por el modo enriquecido y el modo accesible. No se puede añadir la misma complejidad de la aplicación enriquecida al modo accesible, ya que de hacerlo no se conseguiría tener una aplicación funcional y sin demasiado peso, es decir, disponer de una aplicación rápida, con una funcionalidad más limitada pero suficiente para realizar las funciones básicas y con unos tiempos de carga aceptables. Además el introducir la parte accesible entre etiquetas noscript ralentizaría la carga inicial de los datos para la parte de interfaz enriquecido en el sentido de tener que procesar también el esqueleto de la parte accesible, de tener que descargarse la parte accesible también, rellena con los datos desde el servidor y en un formato más largo, ya que iría acompañada del código HTML correspondiente. Por ello la forma elegida fue la de disponer de dos implementaciones diferentes para la parte de interfaz, ya que realmente no solo cambia la presentación sino que también se modifica la funcionalidad del interfaz. De todos modos se ha modificado algunas partes de la aplicación introduciendo la parte accesible entre estas etiquetas, y en todos los casos se introduce una nota informativa sobre la deshabilitación de javascript en el navegador. El núcleo de la aplicación se mantiene común para ambas implementaciones.

En la **implementación de interfaz enriquecido** la filosofía que se ha llevado a cabo es la de delegar en el cliente la parte de presentación. Así el cliente, en una petición inicial, se descarga el programa javascript asociado y una plantilla inicial, la cual sirve de esqueleto de la página que se irá rellenando con los diferentes datos que obtenga el programa de peticiones asíncronas al servidor. Recibirá los datos solicitados y hará las modificaciones necesarias en el interfaz acordes con la programación de la aplicación. Una vez que se realiza esta carga inicial

y las cargas asociadas al evento inicial de página cargada, el programa se quedará esperando a que el cliente realice los eventos que desencadenan tareas programadas en el código del programa javascript. Estos eventos pueden venir por el usuario o por temporizaciones programadas en el código.

Para dar este aspecto de aplicación en tiempo real, el cliente tiene que hablar con el servidor de alguna forma. Realmente no hay nada nuevo en este tipo de comunicaciones ya que son peticiones HTTP a una URL concreta. Lo único que se define, y dado que se busca solo enviar datos sin procesar y no el fragmento de HTML a actualizar, es el formato de datos que se envía. Se ha utilizado definiciones de objetos en formato JSON para comunicar datos del servidor a javascript, los cuales el cliente evalúa directamente e introduce como objetos nuevo en su código. Simplemente se define un formato de código Javascript (JSON), se rellena de forma adecuada con los datos del servidor, se envía al cliente y este hace un eval dentro de javascript, consiguiendo que los datos se introduzcan directamente en el programa. Con este formato de datos, JSON, se consigue que las comunicaciones asíncronas sean poco pesadas y por lo tanto rápidas.

Esta parte, además de la programación de cada código de control de interfaz para cada vista de la aplicación, requiere de la programación de la parte de servidor que se constituye en la parte que atiende las primeras peticiones, la definición del formato de datos a enviar al cliente para la información solicitada y la parte de métodos externos que son los que van a atender las peticiones asíncronas del cliente.

También, para aquellos interfaces que no sean compatibles con javascript, se ha implementado una interfaz en al que predomina la accesibilidad y que se ha denominado de **interfaz accesible**. El no poder contar con javascript para hacer el interfaz un poco más rico hace que la implementación de interfaz cambie e incluso no se puedan ofrecer todas las funciones que se dan desde la parte de interfaz enriquecida. Esto es debido a que implementarlas en una misma interfaz harían que la aplicación perdiera la posible usabilidad que le queda. Esta implementación se apoya en técnicas tradicionales de programación web apoyadas en HTML, CSS y contenido dinámico desde el servidor mediante php.

Debido a que cualquier actualización de la información lleva consigo el actualizar toda la página descargada hace necesario un cambio en las funciones e interfaz ofrecido al usuario con el fin de abaratar el coste de la descarga debido a su peso.

Otra parte importante de la aplicación CITAS2.0 son sus **scripts de mantenimiento** que se ejecutan el la máquina y que implementan la funcionalidad de recordatorios. Esto permite que el sistema pueda alertar a los participantes de un evento de la proximidad de la fecha

elegida desde unos días antes de la misma y en base a la configuración definida por el administrador de la cita.

3.4 Modelo entidad – relación de la base de datos

La estructura de la base de datos en la que se apoya CITAS2.0 muestra el siguiente modelo de entidad-relación permite obtener una visión general de la interrelación de los elementos del sistema.

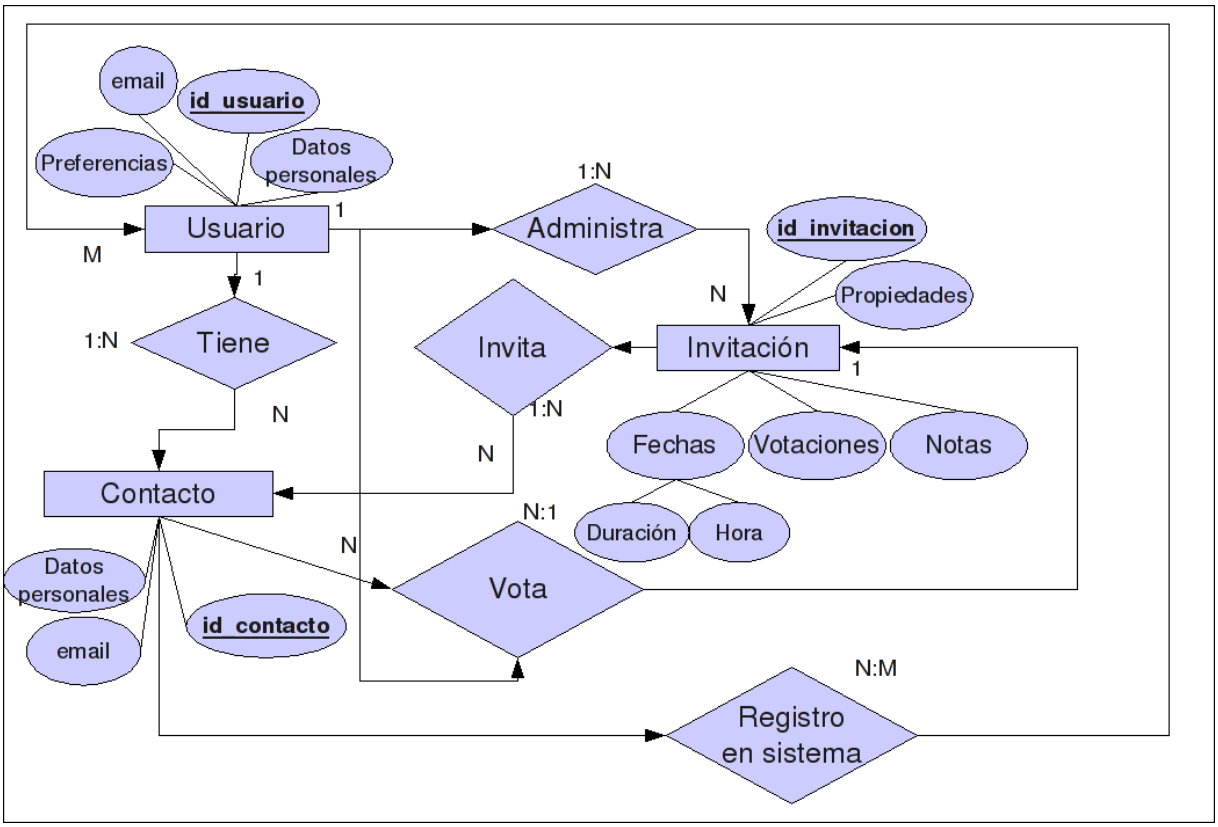

Ilustración 5: Modelo Entidad-Relación de CITAS2.0

Como se puede observar existen tres entidades definidas en el modelo de datos que se tienen varias relaciones entre ellas.

La **entidad usuario** representa al usuario registrado en el sistema. Este usuario tiene como atributos una serie de preferencias que conforman las invitaciones que envía, unos datos personales que serán los mostrados a los participantes al evento, un correo electrónico que será clave para buscar su pertenencia en los contactos de otros usuarios registrados y servirá como nombre de usuario en el sistema, y un atributo clave llamado id_usuario que será un identificador numérico y único del número de usuario en el sistema. Tanto el email como el atributo id_usuario se obligan a que sean únicos en sistema. El atributo id_usuario es el elegido como clave.

Otra entidad que vemos en el modelo es la **entidad invitación**. Sirve para representar cada una de las invitaciones que existen en el sistema. Como se observan se definen los atributos que pueden componer esta invitación como son las fechas propuestas, las votaciones realizadas por los invitados a las fechas propuestas, las notas que el administrador haya podido escribir, un identificador numérico de la invitación y un conjunto de propiedades que describen como va a ser la invitación y como se va a comportar el sistema para la misma. Se define como clave el atributo id_invitacion que va a ser único para cada invitación en el sistema.

La tercera entidad del modelo es la **entidad contacto**. Con esta entidad se representa a los contactos que el usuario registrado en el sistema puede contener en su lista de contactos. Un contacto puede tener varias cuentas de correos asociadas, unos datos personales como el nombre, apellidos y teléfonos, y un identificador clave de contacto llamado id_contacto el cual será único en el sistema.

Para poder unir estas entidades se han definido una serie de relaciones que permiten la comunicación de datos y da sentido a la aplicación.

Una entidad usuario puede **votar** o **administrar** una invitación. Si la relación que mantiene con la invitación es de **votación** el usuario ha sido invitado a una cita de otro usuario diferente y su papel en este caso es solo el de realizar las votaciones deseadas sobre las fechas propuestas. Si por el contrario el usuario quiere administrar una cita puede acceder al contenido de una invitación y modificar sus atributos o puede crear una nueva invitación. Con esta relación se permite realizar cualquier modificación sobre una invitación creada o introducir una nueva invitación.

Una entidad invitación tiene la posibilidad de **invitar** a entidades contactos, los cuales son los que van a participar en la invitación. Esta relación hace que determinados contactos sean partícipes de la invitación.

Una entidad contacto o usuario podrá realizar modificaciones en los atributos votaciones de una invitación mediante la relación **vota**.

También se le brinda la posibilidad a un contacto de ser usuario mediante la entidad de **registro en el sistema**.

Capítulo 4. Descripción de la estructura del gestor de aplicaciones

4.1 Visión general del gestor de aplicaciones

4.1.1 Introducción

Como se ha ido comentando a lo largo de los capítulos precedentes se ha desarrollado un conjunto de librerías que junto a una estructura de directorios, configuraciones y un programa conforman un sistema que se ha denominado gestor de aplicaciones. Realmente se puede ver como una aplicación que controla el acceso, lanza las aplicaciones solicitadas y que ofrece el uso de objetos nuevos creados con la intención de facilitar la programación de las partes web más comunes de las aplicaciones que se ejecuten con este gestor.

Su principal objetivo ha sido el de ofrecer una ayuda a la programación de las aplicaciones que conforman CITAS2.0 y que faciliten las tareas comunes al desarrollo de una aplicación web. Se ha tratado de darle unos interfaces lo más genéricos posibles para facilitar su reutilización.

La implementación se ha realizado en php ya que es uno de los lenguajes de programación web más extendido y que consigue mejores prestaciones para una aplicación de este tipo. También se utiliza MySQL como motor de base de datos del sistema por ser una solución gratuita, eficiente y suficiente para las pretensiones actuales y futuras de este proyecto.

Como características básicas, reutilizables para otra aplicación, podríamos enumerar las siguientes:

- **Separación de programación y contenido.** Se trata que mediante el uso de plantillas se pueda separar el código que propiamente define la aplicación del que sería de presentación. El gestor aporta una librería que trata de ayudar a realizar esta separación mediante la inclusión de un objeto de plantillas en el código. Este permite definir que plantilla de salida se va a utilizar y que datos se van a mostrar en la salida mediante la modificación de una estructura de datos creada para almacenar la información.

- **Identificación de origen accesible o no.** Atendiendo a la variable UserAgent de una petición HTTP el gestor se encarga de discernir entre si la petición proviene de un elemento que puede interpretar javascript o no. Dependiendo de eso el gestor buscará la aplicación solicitada en la rama accesible o en la rama de interfaz enriquecido.
- **Seguridad.** Se define librería de seguridad para el uso de este sistema. Gestiona el acceso y mantiene las variables de sesión que permiten el continuo chequeo de la validez de la conexión. Son fácilmente extrapolables a otras aplicaciones,
- **Entrada de datos.** Se define librería de extracción de datos y comprobación de validez de la entrada de los mismos. Da rapidez y uniformidad a la hora de tratar formularios.
- **Acceso a base de datos**. Se define capa para el tratamiento de bases de datos MySQL. Aporta comodidad al uso de las funciones definidas en php.

Durante el trascurso del capítulo se van a ir explicando cada una de las partes que conforman la aplicación gestora de aplicaciones que son una estructura de directorios, una configuración, una aplicación que hace realmente la función de gestor y un conjunto de librerías que facilitan tareas comunes en una aplicación web.

4.1.1 Estructura de directorios

Para ayudar a la clasificación de las aplicaciones y darle un poco de orden al sistema, se ha definido una estructura de directorios para facilitar esta tarea. Esta estructura cuelga del root de CITAS2.0

Directorio	Descripción
./aplicaciones	Contenedor de los ficheros que implementan las aplicaciones del sistema
./aplicaciones/accesible	Contenedor de los ficheros que implementan las aplicaciones del sistema versión accesible
./exportar	Directorio accesible desde el exterior. Desde aquí cuelga la hoja de estilo y los programas javascript clientes.
./exportar/js	Programas javascript genericos
./exportar/images	Imágenes del sistema
./exportar/jsapp	Programas javascript de aplicación.
./exportar/ayudas	Ficheros de ayuda
./templates	Contenedor de las plantillas.
./templates/mensajes	Contenerdor de las plantilas para los mensajes

Directorio	Descripción
./templates/accesible	Contenedor de las plantillas versión accesible.
./includes	Contenedor de librerías
./includes/accesible	Contenedor de librerías versión accesiible
./scripts	Contenedor de scripts de mantenimiento del sistema.

Tabla 9: Estructura de directorios del portal

4.1.2 Plantillas

El uso de plantillas surge como idea para conseguir el objetivo de separar la presentación de la programación y escribir en cada parte el código correspondiente. En este apartado se trata de buscar cual sería el mejor motor de plantillas y como sería el formato de las mismas para cumplir este hito. Se evalúa también la necesidad de crear un motor con un formato de plantilla propio frente al uso de otras alternativas más estándar.

4.1.2.1 Evaluación de alternativas

El lenguaje de programación elegido ha sido php debido a que ya era conocido por el alumno y suponía un ahorro de tiempo el no tener que aprender otro lenguaje. Además es un lenguaje sencillo, que es de alta difusión, por lo que cuenta con muchos recursos , y suficiente para el desarrollo del proyecto. Cuenta con acceso a base de datos integrado en el lenguaje y facilita la escalabilidad al poder construir objetos y librerías para su reutilización.

Partiendo de esta elección del lenguaje se ha buscado alguna alternativa existente que desarrollase esta función de motor de plantillas.

Como características generales de los motores de plantillas podemos destacar que tratan de separar la lógica de la presentación, lo cual clarifica el código y facilita el reparto de tareas entre el diseñador y el programador. Cuentan con funciones en la sintaxis de la plantilla que maximizan su utilidad y abren un abanico de posibilidades como el manejo de hora, conexión con base de datos, lógica de control, manejo de cadenas, etc. Por contra suelen reducir el rendimiento de la página, aunque la mayoría de las librerías ofrecen solución en esta parte, y que es necesario en casi todos los casos aprender una nueva sintaxis. Precisamente por esto usar un motor de plantillas puede no ser la mejor opción para todas las aplicaciones a desarrollar, pero si es buena cuando se trabaja sobre una aplicación compleja.

En los siguientes apartados se muestran algunas de las opciones existentes en el mercado.

4.1.2.1.1 Smarty template engine

Smarty es el más popular el cual se denomina a si mismo como “template/presentation framework”. Compila las plantillas dentro de scripts PHP y los ejecuta para mostrar el resultado. Sus características son el mecanismo robusto de cacheo que posee y un buen soporte en los plug-ins y add-ons. Es de los más rápidos y flexibles.

4.1.2.1.2 Dwoo

Este sistema se presenta como una seria alternativa a Smarty, siendo además, compatible con el sistema de plantillas y plugins de Smarty.

Las principales características son que posee una herencia de plantillas, es flexible en la creación de plugins y soporta Unicode y UTF-8.

4.1.2.1.3 Savant3

La característica principal de este motor es que es muy ligero. No es necesario aprender un nuevo lenguaje de plantilla ya que usa PHP como lenguaje en las plantillas.

4.1.2.1.4 TinyButStrong

Este sistema es fácil de aprender y de implementar. Es un único fichero con solamente una clase PHP con seis métodos y cinco propiedades. Su fuerte es la simplicidad y cuenta con un sistema de cacheo.

4.1.2.2 Explicación de la elección del motor de plantillas

Viendo todas las posibles alternativas para usar un motor de plantillas, de las cuales solo se han mostrado unas pocas, no se ve la necesidad de crear un motor de plantillas propio.

Una buena elección hubiera sido Smarty ya que es de las mejores alternativas existentes al tener un rendimiento razonable y un juego muy amplio de funciones incluidas en la sintaxis de la plantilla.

El no elegir una opción existente y tomar la decisión de crear un motor propio no tiene más explicación que la motivación del alumno de realizarlo. Quizás no sea una buena decisión en un proyecto real donde se persigue obtener el mayor beneficio a menor coste, y realmente elegir Smarty hubiera sido la mejor opción, pero en este caso se ha conseguido realizar un motor muy simple, muy ligero y con las necesidades que han surgido en la realización de este proyecto. Por contra no cuenta con muchas de las funcionalidades que cuenta Smarty.

Para versiones futuras se puede considerar este punto a mejorar e integrar Smarty en el

diseño del gestor de aplicaciones.

En los siguientes apartados se va a realizar una explicación de la constitución del motor de plantillas diseñado empezando por la estructura de datos que almacena los datos de salida procesados así como la sintaxis utilizada y lógicas de control disponibles. Se han tomado ideas de diferentes sistemas CMS creados con php como phpBB, viendo como se gestionaban las plantillas, como se guarda la información y como es la sintaxis utilizada.

4.1.2.3 Estructura de datos

El paso intermedio entre el programa y la plantilla es una estructura de datos donde se va almacenando la información en un formato definido. Esta estructura va a estar definida de tal forma que va a permitir almacenar la información en forma de árbol y permitiendo definir un conjunto de elementos en una rama del árbol dando la idea de lista. Esto habilita la realización de repeticiones en la plantilla ya que se van definiendo niveles y subniveles.

La unidad principal de esta estructura es el BLOQUE, al cual se le asigna un nombre cualquiera. Dentro de este BLOQUE se irán encadenado los SUBBLOQUES que se deseen hasta terminar en una VARIABLE. Al nombre resultante del encadenamiento de BLOQUES y SUBLOQUES unidos por un punto se llama CAMINO. La cadena resultante de unir el CAMINO.VARIABLE se llama ETIQUETA. Como se puede observar ver es una estructura en árbol.

Para poder realizar repeticiones se define el concepto de ITERACIÓN. Esto permite introducir tantos valores como se elijan para una ETIQUETA.

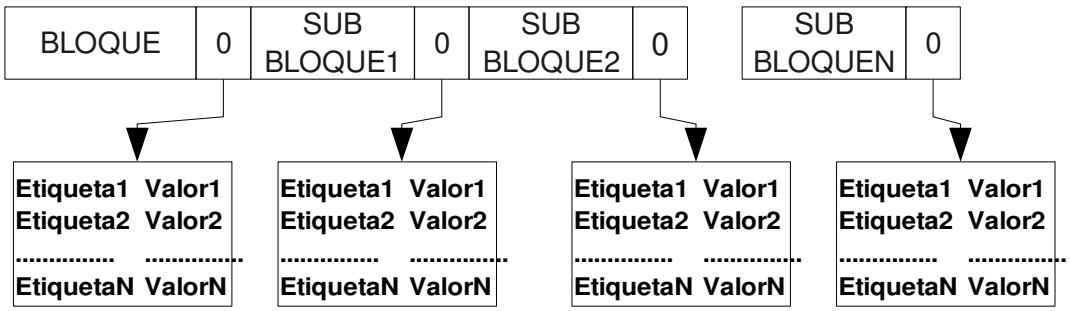

Ilustración 6: Descripción de un registro de la estructura de datos de las plantillas

Se podrán meter tantas iteraciones como se quieran en los puntos buscados, siempre y cuando se referencien correctamente con el camino correcto y la variable en cuestión. Si hay un error en la referencia, el sistema pensará que es otro camino diferente y se iniciará bajo la

iteración cero.

El valor de permitir iteraciones a la pareja compuesta por un camino y una o varias variables con sus correspondientes datos, da la idea de tabla. Va a permitir almacenar en una estructura de datos los valores repetitivos de una consulta que luego podremos mostrar en la plantilla simplemente poniendo el par (CAMINO,VARIABLES) y utilizando las estructuras de control creadas.

Existe un camino restringido definido como RAIZ, y mostrado literalmente como "." que permite meter variables en el raíz del documento, sin necesidad de que esté bajo ningún bloque concreto. Es útil para poder mostrar variables que tienen que estar presentes siempre en la plantilla.

4.1.2.4 Definición de comodines y estructuras de control

Para poder hacer rico el uso de las plantillas se han definido algunas estructuras de control que son de utilidad. Se podrían definir muchas más pero se dejan en la parte de aplicación.

La mejor forma de verlos es observando un trozo de una plantilla:

```
### Ejemplo de variables en la raíz de la plantilla y
### bajo un primer bloque

### Este es un ejemplo de comentario

### Estas 3 primeras etiquetas están bajo el bloque RAIZ
{CAMPO1}
{CAMPO2}
{CAMPO3}

### aquí empezamos el bloque PRIMERO e indicamos las
### variables que vemos bajo ese bloque
<!-- BLOQUE POSIBLE INICIO PRIMERO -->
Vamos a mostrar los datos que estan en la base de datos
<table>
<tr><td>CAMPO1</td><td>{PRIMERO.CAMPO1} </td></tr>
<tr><td>CAMPO2</td><td>{PRIMERO.CAMPO2} </td></tr>
</table>
<!-- BLOQUE POSIBLE FIN PRIMERO-->
```

Para este primer caso se pueden ver como se introducen comentarios en las plantillas, los cuales son filtrados cuando se procesa la misma. También se ve cual es la sintaxis para introducir una etiqueta, metiendo el CAMINO.VARIABLE entre llaves; y por último se ve como se introduce un bloque en la plantilla. Para ello hay que usar la expresión "BLOQUE POSIBLE INICIO CAMINO" y "BLOQUE POSIBLE FIN CAMINO" entre comentarios html y haciendo

referencia al CAMINO.

El que exista la palabra POSIBLE en la definición de bloque indica que solo se va a mostrar si en la estructura de datos se ha creado algún camino. Si alguna de la variables que se ponen en la plantilla no se encuentran en la estructura de datos, se elimina su valor en la plantilla cuando se procesa la misma. Realmente se eliminan todos los comentarios y todas las variables no mostradas

En el siguiente trozo de plantilla se ve otra etiqueta de control:

```
### Esta es una prueba introduciendo bloques repetivos
El alumno:
<ul>
<li>Nombre y apellidos: {CAMPO1}</li>
<li>Número de alumno:{CAMPO2}</li>
<li>Número de asignaturas aprobadas: {CAMPO3}</li>
</ul>
tiene aprobadas las siguientes asignaturas
<table>
<tr>
<td>CAMPO1</td>
<td>CAMPO2</td>
<td>CAMPO3</td>
<td>CAMPO4</td>
<td>CAMPO5</td>
</tr>
<!-- BLOQUE REPETITIVO INICIO PRIMERO -->
<tr>
<td>{PRIMERO.CAMPO1}</td>
<td>{PRIMERO.CAMPO2}</td>
<td>{PRIMERO.CAMPO3}</td>
<td>{PRIMERO.CAMPO4}</td>
<td>{PRIMERO.CAMPO5}</td>
</tr>
<!-- BLOQUE REPETITIVO FIN PRIMERO -->
</table>
```

donde se puede ver que se esta introduciendo la capacidad de realizar bucles. Para ello se inicia un bloque repetitivo con la directriz “BLOQUE REPETITIVO INICIO CAMINO” y se finaliza con “BLOQUE REPETITIVO FIN CAMINO”. Esto hará que se itere la estructura de datos mostrando toda la información guardada. Se pueden intercalar bucles repetitivos de diferentes niveles mostrando información de nivel superior en un bucle concreto. Para mostrar la de nivel inferior habría que crear otro bucle inferior.

Para mostrar u ocultar información en la plantilla existe la directriz POSIBLE, la cual muestra lo que hay en su fragmento de código si existe alguna variable dentro del camino de ese bloque. Otra forma de mostrar variables es simplemente la de poner la etiqueta de la variable, la cual solo se va a rellenar si existe en la estructura de datos. En caso de no existir se

eliminará al realizar el procesado de la plantilla.

A modo de resumen se muestra la siguiente tabla:

Sintaxis	Descripción
CAMINO	CAMINO es el literal de poner todos los bloques y subbloques unidos por puntos. Ej: BLOQUE.SUB1.SUB2
{CAMINO.VARIABLE} = ETIQUETA	Mediante este etiquetado se define una variable en la plantilla.
<!-- BLOQUE POSIBLE INICIO CAMINO --> **.....** **<!-- BLOQUE POSIBLE FIN CAMINO -->**	Todo lo que se incluya entre estas dos etiquetas solo se mostrará si existe alguna variable bajo ese CAMINO
<!-- BLOQUE REPETITIVO INICIO CAMINO --> **.....** **<!-- BLOQUE REPETITIVO FIN CAMINO -->**	Realiza tantas iteraciones de ese fragmento de plantilla como iteraciones tenga almacenada en la estructura de datos

Tabla 10: Resumen de sintaxis de las plantillas

El resto de sintaxis para el control de flujo que se pueden encontrar en otras aplicaciones similares no están implementados, ya que se delega a la aplicación propiamente, introduciendo o no valores dentro de la estructura de datos. Es un punto a mejorar en futuras implementaciones.

4.1.2.5 Definición de métodos de la librería

Para poder darle realizar estas funciones se ha implementado una librería en php que se utiliza a modo de clase dentro de la aplicación. Se encuentra en el directorio root/include del servidor y se llama **libplantillas.php**. La nombre de clase es **plantilla**.

La forma de crear un objeto plantilla es la siguiente:

```
include ("includes/libplantillas.php");

$plantilla=new plantilla();
```

Se han definido los algunos métodos para poder realizar las funciones descritas en el apartado anterior y que se enumeran a continuación.

4.1.2.5.1 Resumen de métodos

Método	Breve descripción
asignar ($directorio,$nombre)	Asignar ficheros de plantillas
desasignar ()	Desasignar ficheros de plantillas
muestraPlantilla ()	Mostrar los ficheros de plantillas
introducirVariableEnRaiz ($etiqueta, $valor)	Introducir una VARIABLE en la raíz del documento.
introducirArrayEnRaiz ($datos)	Introducir un array de VARIABLE=>VALOR en la raíz del documento
introducirArrayRepetitivoEnBloque ($bloque,$etiqueta,$valores)	Introducir múltiples valores para una ETIQUETA dentro de un CAMINO. Realiza ITERACION.
introducirVariableEnBloque ($bloque, $etiqueta,$valor)	Introducir el valor de una ETIQUETA en un CAMINO. No realiza ITERACION.
nuevalteracion ($bloque)	Realización de ITERACION para un camino.
introducirArrayEnBloque ($bloque, $datos)	Introduce un array de ETIQUETA=>VALOR dentro de un camino. Realiza ITERACION.
volcar ()	Procesa la plantilla con la estructura de datos.

Tabla 11: Resumen de métodos de la librería de plantillas

4.1.2.6 Ejemplos de uso

En este apartado se muestra un ejemplo de como sería el procesamiento de una plantilla html. El ejemplo se basa en mostrar las asignaturas aprobadas por un alumno en la carrera.

Plantilla

```
### Esta es una prueba introduciendo bloques repetivos
El alumno:
<ul>
<li>Nombre y apellidos: {NOMBRE1}</li>
<li>Número de alumno:{NIA}</li>
<li>Número de asignaturas aprobadas: {APROBADAS}</li>
</ul>
Tiene aprobadas las siguientes asignaturas
<table>
<tr>
<td>Asignatura</td>
<td>Curso</td>
<td>Nota</td>
<td>Años presentados</td>
</tr>
```

```
<!-- BLOQUE REPETITIVO INICIO APROBADAS -->
<tr>
<td>{APROBADAS.ASIGNATURA}</td>
<td>{APROBADAS.CURSO}</td>
<td>{APROBADAS.NOTA}</td>
<td>
<!-- BLOQUE REPETITIVO INICIO APROBADAS.ANIO -->
{APROBADAS.ANIO.ANIO} 
<!-- BLOQUE REPETITIVO FIN APROBADAS.ANIO -->
</td>
</tr>
<!-- BLOQUE REPETITIVO FIN APROBADAS -->
</table>
```

Código

```
include ("includes/libplantillas.php");

$plan=new plantilla();
$plan->asignar(".",array("prueba1txt"));

$plan->introducirVariableEnRaiz("NOMBRE","Miguel    Angel    Cabrera
Bejarano");
$plan->introducirVariableEnRaiz("NIA","100035701");
$plan->introducirVariableEnRaiz("APROBADAS","5");
$plan->introducirArrayEnBloque("APROBADAS",                    array(
      "APROBADAS.ASIGNATURA"=>"Microondas",
            "APROBADAS.CURSO"=>"4",
            "APROBADAS.NOTA"=>"7"
            ));
$plan->introducirArrayRepetitivoEnBloque("APROBADAS.ANIO",
"APROBADAS.ANIO.ANIO",array("2001","2002","2003"));

$plan->introducirArrayEnBloque("APROBADAS",                    array(
      "APROBADAS.ASIGNATURA"=>"Transmision y propagacion",
            "APROBADAS.CURSO"=>"4",
            "APROBADAS.NOTA"=>"6"
            ));
$plan->introducirArrayRepetitivoEnBloque("APROBADAS.ANIO",
"APROBADAS.ANIO.ANIO",array("2002","2003"));

$plan->introducirArrayEnBloque("APROBADAS",                    array(
      "APROBADAS.ASIGNATURA"=>"Redes de ordenadores",
            "APROBADAS.CURSO"=>"4",
            "APROBADAS.NOTA"=>"8"
            ));
$plan->introducirArrayRepetitivoEnBloque("APROBADAS.ANIO",
"APROBADAS.ANIO.ANIO",array("2001"));

echo $plan->volcar();
```

Resultado

```
El alumno:
<ul>
```

```
<li>Nombre y apellidos: </li>
<li>Número de alumno:100035701</li>
<li>Número de asignaturas aprobadas: 5</li>
</ul>
Tiene aprobadas las siguientes asignaturas
<table>
<tr>
<td>Asignatura</td>
<td>Curso</td>
<td>Nota</td>
<td>Años presentados</td>
</tr>
<tr>
<td>Microondas</td>
<td>4</td>
<td>7</td>
<td>
2001 
2002 
2003 
</td>
</tr>
<tr>
<td>Transmision y propagacion</td>
<td>4</td>
<td>6</td>
<td>
2002 
2003 
</td>
</tr>
<tr>
<td>Redes de ordenadores</td>
<td>4</td>
<td>8</td>
<td>
2001 
</td>
</tr>
</table>
```

4.1.3 Seguridad

El gestor de aplicaciones cuenta con un módulo de autenticación de usuarios. Este módulo accede a la base de datos que se le indique en el método de creación y utiliza los campos que le digamos como login y password, así como el campo que se va a utilizar de identificador de usuario. Valida la autenticación y almacena en una sesión los datos válidos de la conexión.

Para darle más juego se permite que las aplicaciones puedan introducir valores en la sesión si es necesario para enriquecer su funcionamiento, por lo que esta librería hace las

veces de manejador de sesiones.

Este módulo solo realiza autenticación y registra la última hora del acceso. No hace autorización en el sentido de controlar el acceso a diferentes aplicaciones por usuario, ni accounting registrando los comandos que va dando el usuario. Esto se delega al siguiente nivel de aplicación, en el que si que se filtra la información de la aplicación por identificador de usuario.

Se ha implementado una clase de seguridad en una librería, y el flujo de control de seguridad se ha implementado en el index del portal.

4.1.3.1 Descripción de la librería

La librería se encuentra en el directorio root/include del sistema y se llama libseguridad.php. Implementa una clase que tiene los siguientes métodos y propiedades

4.1.3.1.1 Resumen de propiedades

Estas son las propiedades que dispone la librería

Propiedad	Breve descripción
db_host	Servidor que contiene la tabla a utilizar de usuarios
db_puerto	Puerto del servidor donde esta la base de datos
db_user	Usuario de acceso a la base de datos
db_pass	Password del usuario
db_bbdd	Nombre de la base de datos
db_tabla	Nombre de la tabla
db_campos	Array con los campos que vamos a utilizar en el chequeo del acceso
session_variables	Array que almacena los nombres de las variables de sesión

Tabla 12: Resumen de propiedades de la librería de seguridad

4.1.3.1.2 Resumen de métodos

Método	Breve descripción
chequeaLogin ($salida,$datos)	Comprueba el acceso con los datos dados y devuelve los datos indicados
datosConexionChequeo ($datos, $db_tabla,$db_campos)	Configuramos el acceso a la base de datos de autenticación
destruyeSesion ()	Elimina la sesión
eliminaArrayEnSesion ($variables)	Elimina un conjunto de variables de la sesión

Método	Breve descripción
eliminaVariableEnSesion ($variable)	Elimina una variable concreta da la sesión
extraeArrayDeSesion ($variables)	Introduce en el ámbito global las variables de sesión indicadas dentro del array
extraeVariableDeSesion ($variable_sesion,$variable_externa)	Introduce en el ámbito global la variable de sesión indicada.
finalizaSesion ()	Fuerza la escritura de los datos de sesión y la finaliza.
guardaArrayEnSesion ($variables)	Introduce en el ámbito de sesión las variables globales indicadas dentro del array
guardaVariableEnSesion ($variable_sesion,$variable_externa)	Introduce en el ámbito de sesión la variable global indicada.
inicioSesion ()	Inicia la sesión

Tabla 13: Resumen de métodos de la librería de seguridad

4.1.3.2 Flujo de control de acceso

Cualquier aplicación que sea lanzada por el gestor siempre pasa por el index del portal. Se ha elegido este punto como sitio donde implementar el flujo que va a controlar el acceso a las aplicaciones.

También, y con la finalidad de introducir más seguridad al sistema, la petición de aplicación se hace a un nombre ficticio, el cual se configura en el gestor y se le asigna la ubicación real de la aplicación a lanzar bajo un directorio raíz. Si la aplicación solicitada no esta registrada, la petición será fallida. Esto trata de evitar que consultas mal intencionadas puedan acceder a partes del sistema que no se desea que sean visibles.

El flujograma de que realiza el control de seguridad es el que se muestra en la figura siguiente.

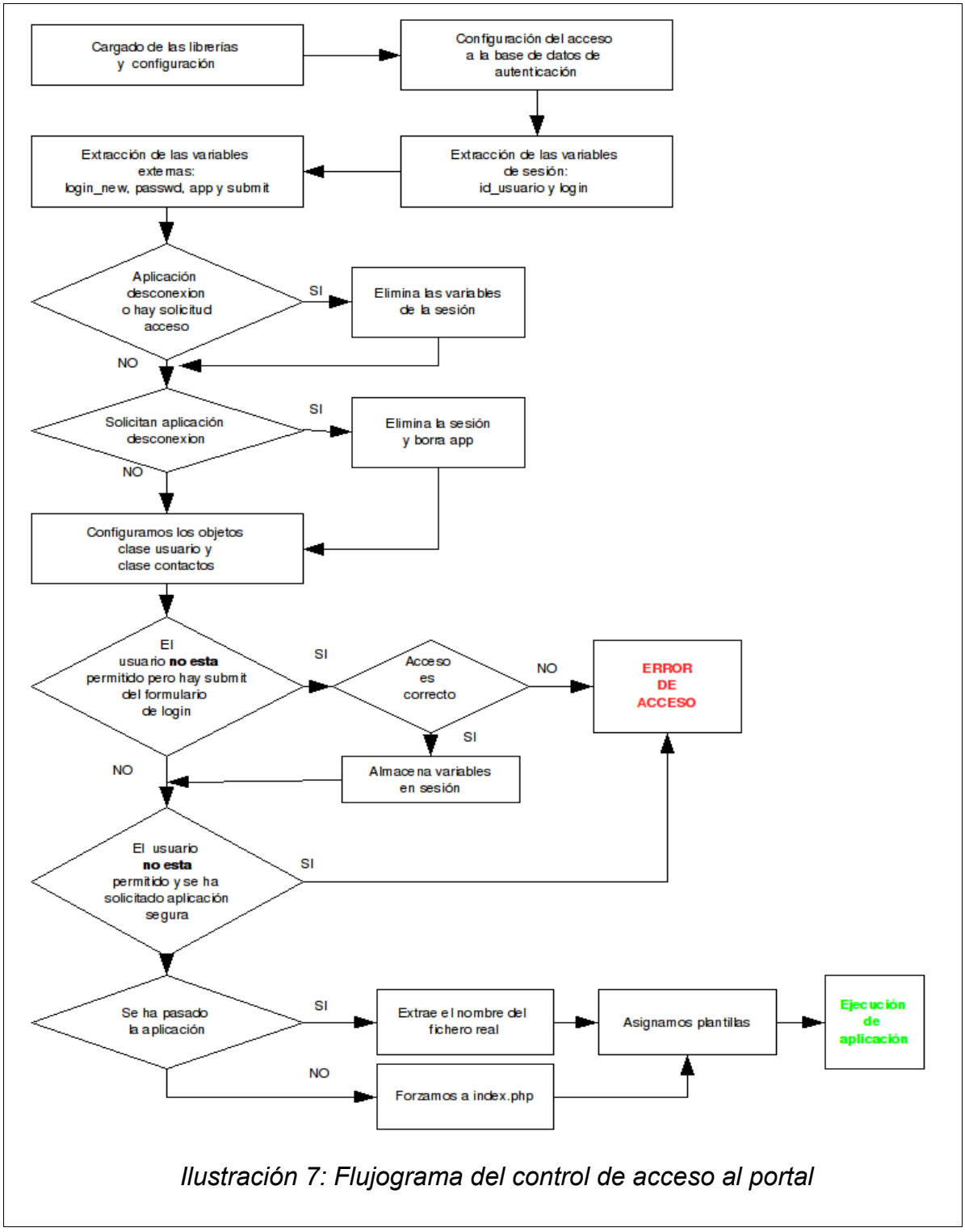

Ilustración 7: Flujograma del control de acceso al portal

4.1.4 Acceso a base de datos

Para facilitar el acceso y extracción de datos de bases de datos se ha implementado una librería que ofrece funciones de más alto nivel y controles que los propuestos por el lenguaje. De momento solo maneja base de datos MySQL pero sería fácil hacer que manejase diferentes tipos de bases de datos.

4.1.4.1 Descripción de la librería

La librería se encuentra en el directorio root/include del sistema y se llama libbbdd.php. Implementa una clase que tiene los siguientes métodos y propiedades.

4.1.4.1.1 Resumen de propiedades

Estas son las propiedades que dispone la librería

Propiedad	Breve descripción
db_host	Servidor que contiene la base de datos
db_puerto	Puerto del servidor donde esta la base de datos
db_user	Usuario de acceso a la base de datos
db_pass	Password del usuario
db_bbdd	Nombre de la base de datos
db_conn	Identificador de conexión
db_query	Cadena con la consulta SQL
db_result	Objeto resultado de la consulta.

Tabla 14: Resumen de propiedades de la librería de acceso a base de datos

4.1.4.1.2 Resumen de métodos

Método	Breve descripción
conexion ($datos)	Conexión con la base de datos
desconexion()	Desconexión
muestraUltimoIdentificador ()	Muestra el último identificador de un insert con autoincrement
lanzaConsulta ($consulta)	Realiza la consulta
extraeFila ($tipo)	Extrae la fila en el formato que le indiquemos
extraeCampo ($fila,$columna)	Extrae un campo concreto
numeroFilasAfectadas ()	Muestra el número de filas afectadas
numeroFilasResultado ()	Muestra el número de filas resultado

Método	Breve descripción
muestraConsulta()	Muestra la última consulta realizada.

Tabla 15: Resumen de métodos de la librería de manejo de base de datos

4.1.5 Manejo de datos

También para facilitar la extracción de los datos de las peticiones HTTP GET y POST se ha implementado una librería que se encarga de esta extracción así como del chequeo de los datos de entrada. Básicamente copia, al entorno de variables globales, las variables que vienen por peticiones GET o POST.

También se realiza un limpiado de los caracteres de control que pueden romper el código de la aplicación o que pueden ser malintencionados con fines de romper la seguridad del sistema.

4.1.5.1 Descripción de la librería

La librería se encuentra en el directorio root/include del sistema y se llama libDatosEntrada.php. Implementa una clase que tiene los siguientes métodos y propiedades

4.1.5.1.1 Resumen de propiedades

Estas son las propiedades que dispone la librería

Propiedad	Breve descripción
variables_entorno	Array con todos los nombres de las variables que se solicitan que se extraigan
variables_obligatorias	Array con todos los nombres de las variables que se consideran obligatorias para luego proceder a su chequeo

Tabla 16: Resumen de propiedades de la librería de manejo de datos

4.1.5.1.2 Resumen de métodos

Método	Breve descripción
introduceArrayObligatorias ($variables)	Copia el array de nombres a la propiedad $variables_obligatorias
introduceArrayVariables ($variables)	Copia el array de nombres a la propiedad $variables_entorno y recarga los datos globales
recargaDatosGlobales ()	Extrae los datos de los datos GET y POST. Realiza el limpiado de los mismos
chequeaVariablesObligatorias ()	Comprueba que no hay ninguna variable vacia.

Tabla 17: Resumen de métodos de la librería de manejo de datos

4.1.6 Configuración del gestor

Para hacer el sistema un poco más parametrizable, se dispone de un fichero de configuración donde se pueden configurar diferentes aspectos del comportamiento tanto de la aplicación como del propio lenguaje php.

Se puede dividir en los siguientes bloques:

- Configuración de **campos de la cabecera** de cada página html
- Configuración de **conexión de base de datos** y definición de los datos de autorización.
- Configuración de los **directorios de aplicación y de plantillas**.
- Configuración de los **UserAgents** que son considerados accesibles.
- Configuración de datos de los **envíos de mensajes** por parte del sistema.
- Configuración de las **aplicaciones** que tiene el sistema, el alias que se le asigna y su fichero de aplicación
- Configuración de las **aplicaciones sin seguridad**.

En apartados sucesivos se detallan las opciones posibles en este fichero de configuración.

4.1.6.1 Accesibilidad

Dependiendo del UserAgent de la petición cliente, se decida si la petición es para un dispositivo accesible o no. Esta decisión afecta principalmente a la elección de la ubicación de las plantillas y programas ya que cambia dependiendo de si es un dispositivo accesible o no. Como se explico en capítulos previos se tomó la decisión de tener dos implementaciones distintas para cada tipo de petición ya que la funcionalidad iba a ser diferente en un caso y en

otro para maximizar la usabilidad en los dos casos sin tener una carga pesada de datos.

4.1.6.2 Listado de parámetros de configuración

Cada uno de los parámetros se introducen en diferentes arrays en el formato NOMBRE => VALOR. En los siguientes apartados se describen un listado de los nombres de los parámetros.

4.1.6.2.1 Parámetros de conexión a la base de datos

Estos parámetros se almacenan en el array **$conexion_bbdd**.

Parámetro	Descripción
DB_HOST	Servidor donde está almacena la base de datos
DB_USER	Usuario de acceso
DB_PASS	Contraseña de acceso
DB_BBDD	Nombre de la base de datos
DB_PUERTO	Puerto del servidor remoto donde esta la base de datos

Tabla 18: Parámetros de configuración de acceso a la base de datos

La implementación del portal solo esta implementada para base de datos MySQL. Como mejora se podría desarrollar la implementación para otras bases de datos.

4.1.6.2.2 Parámetros de configuración generales

Estos parámetros se almacenan en el array **$configuracion**.

Parámetro	Descripción
APP_TITULO	Título que se le pone al portal.
APP_DESCRIP	Descripción de la funcionalidad del portal
APP_SLOGAN	Más información a modo de eslogan
MAIL_ADMIN	Correo del webmaster
SEGURIDAD_TABLA	Tabla que contiene los datos de usuarios para realizar la autorización
SEGURIDAD_CAMPOS	Campos que vamos a utilizar para chequear el acceso
SEGURIDAD_CLAVE	Campo de SEGURIDAD_TABLA que es el identificador de usuario
SEGURIDAD_DATOS_USER	Datos de usuario a extraer cuando el acceso es correcto.
DIRECTORIO	Directorio raíz donde está instalado el portal.

Parámetro	Descripción
DPNORMAL	Directorio de plantillas interfaz enriquecido para la aplicación. Relativo al raíz DIRECTORIO
DPMENSAJES	Directorio de plantillas para los mensajes. Relativo al raíz DIRECTORIO
DPACCESIBLE	Directorio de plantillas accesible para la aplicación. Relativo al raíz DIRECTORIO
DINORMAL	Directorio bajo el que se encuentra la aplicación interfaz enriquecido. Relativo al DIRECTORIO/aplicaciones
DIACCESIBLE	Directorio bajo el que se encuentra la aplicación accesible. Relativo al DIRECTORIO/aplicaciones
APPDEFECTO	Aplicación por defecto que el sistema muestra cuando no se introduce ninguna y el usuario esta logado.
URLSISTEMA	URL donde se encuentra ubicada la aplicación
URLVOTACIONES	URL que se envía por correo y es donde se indica donde está la aplicación de votaciones de cliente.
URLFORZAR	URL que se envía al administrador donde se indica donde está la aplicación para forzar una votación.
SMTP_HOST	Servidor de correo
SMTP_REMITENTE	Remitente que se pone por defecto en las notificaciones

Tabla 19: Parámetros de configuración general

4.1.6.3 Listado de aplicaciones configuradas

Las aplicaciones que se han creado para que funcionen en el portal se listan en el array $**aplicaciones**. El directorio real de la aplicación será el calculado cuando se chequea la accesibilidad. Los ficheros se llamarán igual pero en diferentes directorios.

Aplicación	Fichero	Descripción
registro	registro.php	Se encarga del registro de usuarios en el sistema
contactos	contactos.php	Gestión de contactos
invitaciones	invitaciones.php	Gestión de las invitaciones
invajax	invitaciones_ajax.php	Gestión de las invitaciones mediante interfaz asíncrono y salida JSON o XML
contactosajax	contactos_ajax.php	Gestión de los contactos mediante interfaz asíncrono y salida JSON o XML
votaciones	votaciones.php	Interfaz de cliente para las votaciones
usuajax	usuario_ajax.php	Gestión de los datos del usuario

Aplicación	Fichero	Descripción
opassword	password.php	Para realizar el cambio de password cuando se ha perdido
ayudainv	ayudainv.php	Ayuda de la aplicación
modifusu	usuario.php	Modificación de datos personales del registro en modo accesible.

Tabla 20: Listado de aplicaciones configuradas

De las aplicaciones listadas se definen mediante el array **$app_sin_seguridad** las que no tienen que pasar por el control de acceso ya que son aplicaciones públicas. Las que no requieren control de acceso son : **registro**, **votaciones** y **opassword**.

4.1.7 Conclusiones

La necesidad de realizar esta división en la aplicación global CITAS2.0 es la de simplificar y reutilizar el código común como se ha repetido en varias ocasiones.

Se podría haber usado algún gestor de contenidos y haber desarrollado sobre él la aplicación, pero no se ha visto la necesidad ya que era usar algo demasiado complejo para una aplicación muy específica y que bajaría el rendimiento de la aplicación.

Como se ha comentado en puntos anteriores quizás un fallo en el diseño fue el crear un motor de plantillas existiendo motores ya desarrollados que aportan muchas funcionalidades a las plantillas. Por contra se pudo desarrollar algo muy sencillo, que cumplía con las necesidades de esta aplicación y que es rápido en comparación con estas soluciones.

Sobre la elección de montar la aplicación sobre un sistema tradicional web creado por apache más mysql y php fue por el conocimiento del alumno de estas tecnologías.

Capítulo 5. Visión general de la aplicación CITAS2.0

5.1 Descripción de la aplicación

CITAS2.0 es un sistema de organización de eventos y de elección de los mismos mediante la votación de los participantes de la invitación. Al usuario registrado se le ofrece una plataforma con la que puede organizar eventos siguiendo unos simples pasos:

- Tener **definidas las fechas** y horas en las el usuario tiene disponibilidad para realizar el evento
- Tener definidos unos **contactos** a los que se invitar al evento. Pueden pertenecer a la agenda de contactos o pueden introducirse directamente en la creación de la cita.
- Elegir un **tipo de votación** que va a definir como tienen que elegir las fechas los invitados.
- Elegir como se **notifican el trascurso de las votaciones**.

Cuando el usuario, o administrador de la cita, crea el evento, el sistema envía un correo a cada uno de los invitados con un resumen del evento al que se le ha invitado a participar. En la invitación aparece un enlace al interfaz de votación de usuarios, donde el invitado podrá realizar su contribución al evento. Si el administrador así lo ha querido, el invitado también podrá ver la lista de invitados y las votaciones de los mismos. Si además el usuario esta registrado en el sistema, también le aparecerá la cita en su buzón de invitaciones en las que participa.

Una vez que el administrador ha creado la cita se empezará a realizar las votaciones de los participantes. Esta votación se realiza mediante el interfaz facilitada en el correo de invitación.

Dependiendo de como el administrador haya elegido la forma de notificación del trascurso de las votaciones, irá recibiendo notificaciones de cada una de las votaciones o un resumen final con todas. Si algún usuario no ha podido votar, o si al final se produce duplicidad en la elección de la fecha, el administrador tiene la posibilidad de modificar cada una de las

votaciones o incluso forzar una fecha como la propuesta elegida finalmente.

También se posibilita la realización de anotaciones en la cita, las cuales pueden tener carácter público y ser notificadas a cada uno de los invitados. Complementado a esta función se pueden enviar mensajes en el momento que se desee a los invitados seleccionados.

Si el administrador decide en un momento determinado anular la invitación o ampliarla en número de fechas e invitados, se le da la posibilidad de hacerlo.

También el administrador dispone de un sistema de recordatorio que puede configurar para enviar mensajes a cada uno de los participantes cuando la fecha ya ha sido elegida.

Se irá entrando un poco más en detalle en los conceptos del sistema en los siguientes apartados donde se va a ir desarrollando los bloques que componen la parte de aplicación del sistema y que implementan la aplicación CITAS2.0.

Es importante notar que se va a hablar sólo de la parte del servidor sin entrar en el desarrollo del cliente, que para la parte de interfaz enriquecido es bastante importante, y que se hablará en capítulos siguientes.

5.1 Gestión de usuarios del sistema

5.1.1 Descripción general

Una parte importante de la aplicación CITAS2.0 es su gestión de usuarios. Es necesaria para dar poder dar la visión de organizador de eventos a todo aquel que esté registrado en el sistema. Para ello se ofrecen las siguientes características:

- Una aplicación de **registro de usuarios**, donde los usuarios podrán registrarse para formar parte del sistema de CITAS2.0.
- Método de **recordatorio de contraseña olvidada**, donde el sistema reenviará por correo una nueva contraseña al login del usuario, que es una cuenta de correo por definición.
- Un **control de acceso al sistema** que se realiza con el bloque de seguridad del sistema genérico.
- Aplicación de **modificar datos personales** del registro, donde se puede cambiar los datos personales que se componen de : el login, la contraseña, el nombre y los apellidos, la firma de los mensajes que envíe el sistema en nombre del usuario, y algunos comportamientos globales de la aplicación.

5.1.1 Estructura de datos utilizados

La tabla de datos donde se almacenan los usuarios esta estructurada con los siguientes campos (los tipos son están en formato MySQL):

Campo	Tipo	Descripción
id_usuario	int(5)	Identificador numérico del usuario en el sistema
login	varchar(255)	Login de acceso. En el caso de esta aplicación es un correo electrónico
passwd	varchar(128)	Password del usuario
Nombre	varchar(255)	Nombre del mismo
Apellidos	varchar(255)	Apellidos del usuario
titulo	varchar(255)	Descripción del usuario.
tslogin	timestamp	Timestamp del último acceso
correorem	varchar(5)	Para que el usuario aparezca como remitente en los correos que se envían desde la aplicación
copiacorreos	text	Correos que se ponen en el campo CC del correo enviado
firma	text	Firma que se anexa a los correos enviados desde el sistema
clave	varchar(128)	Clave de acceso directo al sistema que hace las veces de acceso por usuario y contraseña

Tabla 21: Resumen de campos de la tabla de usuarios

Las aplicaciones y librerías se encargarán de modificar debidamente estos campos.

5.1.2 Aplicaciones creadas

Las aplicaciones creadas para dar la funcionalidad indicada en la introducción, las podemos resumir en las siguientes tablas, tanto para la parte accesible como la interfaz enriquecido. La localización de los ficheros depende de las variables de configuración globales comentadas en apartados anteriores, por lo que las aplicaciones interfaz enriquecidos cuelgan del directorio “aplicaciones/DINORMAL” y las accesibles de “aplicaciones/DIACCESIBLE”.

Aplicación	Descripción
registro.php	Aplicación que controla el registro de usuarios
password.php	Control de cambio de contraseña olvidada
usuario_ajax.php	Interfaz para realizar modificaciones de datos asíncronamente

Tabla 22: Aplicaciones de gestión de usuarios interfaz enriquecido

Aplicación	Descripción
registro.php	Aplicación que controla el registro de usuarios
password.php	Control de cambio de contraseña olvidad
usuario.php	Modificación de los datos personales del usuario

Tabla 23: Aplicaciones de gestión de usuarios accesible

También tenemos las plantilla de cada aplicación, que colgaran de los directorios "templates/DPNORMAL" y las accesibles de "templates/DPACCESIBLE". En las siguientes tablas se resume las plantillas utilizadas para cada aplicación y el orden en el que se muestran en la salida.

Aplicación	Plantillas
registro.php	cabecera.tpl, cuerpo_registro.tpl, pies.tpl
password.php	cabecera.tpl, cuerpo_opassword.tpl, pies.tpl
usuario_ajax.php	usuarioDatosPersonales_json.xml

Tabla 24: Plantillas de las aplicaciones de gestión de usuarios interfaz enriquecido

Aplicación	Plantillas
registro.php	cabecera.tpl, cuerpo_registro.tpl, pies.tpl
password.php	cabecera.tpl, cuerpo_opassword.tpl, pies.tpl
usuario.php	cabecera.tpl, cuerpo_invitaciones_modifDatos.tpl, pies.tpl

Tabla 25: Plantillas de las aplicaciones de gestión de usuarios accesible

5.1.3 Librerías creadas

Para poder realizar un manejo adecuado de los valores del usuario se ha creado una clase de usuario que facilita acciones a realizar sobre las propiedades del usuario. Esta clase se implementa en la librería **libusuario.php**.

5.1.3.1 Resumen de propiedades

Estas son las propiedades que dispone la librería.

Propiedad	Breve descripción
$id_usuario	Identificador del usuario
$datos_usuario	Array con todos datos del usuario en el formato NOMBRE => VALOR

Tabla 26: Resumen de propiedades de la clase usuario

5.1.3.2 Resumen de métodos

Método	Breve descripción
asignaDatos ($datos_conexion, $datos_usuario)	Asigna los datos de usuario iniciales y los de conexión
usuarioIntroduceDatos ($datos)	Mete los nuevos datos de usuario
usuarioActualizaTSLogin ()	Actualiza el tiempo de entrada
usuarioIntroduceUnDato ($campo, $dato)	Introduce un único dato
usuarioActualizaDatos ()	Aplica los cambios realizados
usuarioMuestraId ()	Muestra el identificador de usuario
usuarioPermitido ()	Indica si el usuario es correcto
usuarioMuestraUnDato ($campo)	Muestra solo el valor de un campo
usuarioMuestraTodosDatos ()	Extrae el array de todos los datos de usuario
usuarioModificaDatos ($datos)	Sobreescribe el array de datos de usuario
usuarioModificaUnDato ($campo, $valor)	Modifica solo un campo del array
usuarioActualizaClave ()	Crea la clave del usuario
usuarioActualizaClaveSiNoEsta ()	Comprueba si esta la clave y si no la crea

Tabla 27: Resumen de las propiedades de la clase usuario

5.1.4 Métodos externos para peticiones asíncronas

Una de las características de este proyecto es intentar estar en un modelo de desarrollo web 2.0. Para ello es necesario que se abran diversos métodos externos que habiliten las peticiones asíncronas que el cliente realice. Estos interfaces están abiertos para la aplicación en modo interfaz enriquecido.

Para realizar esta comunicación se definen un conjunto de mensajes, para cada aplicación, que están en formato JSON persiguiendo ahorrar tiempo de proceso en el cliente.

A continuación se describen los formatos de los mensajes y cada uno de los métodos externos.

5.1.4.1 Formato de mensaje usuarioDatosPersonales_json.xml

Se encuentra bajo el directorio **templates** y tiene el siguiente contenido:

```
{
  "datosPersonales":
      {
            "id_usuario" : "{ID_USUARIO}",
```

```
            "login" : "{LOGIN}",
            "nombre" : "{NOMBRE}",
            "apellidos": "{APELLIDOS}",
            "titulo": "{TITULO}",
            "firma": "{FIRMA}",
            "correorem": "{CORREOREM}",
            "copiacorreos": "{COPIACORREOS}"
      },
  "mensajes": [
<!-- BLOQUE POSIBLE INICIO MENSAJE -->
<!-- BLOQUE REPETITIVO INICIO MENSAJE -->
      {
            "nivel": "{MENSAJE.NIVEL}",
            "ts": "{MENSAJE.TS}",
            "cuerpo": "{MENSAJE.CUERPO}"
      },
<!-- BLOQUE REPETITIVO FIN MENSAJE -->
<!-- BLOQUE POSIBLE FIN MENSAJE -->
      'VALORFINAL'
      ]
}
```

Como se puede observar se crea un objeto que tiene a su vez dos propiedades:

- **datosPersonales:** Estructura que contiene cada uno de los valores de los campos de la base de datos de usuarios para el usuario en cuestión
- **mensajes:** Como en cada una de las comunicaciones JSON con el servidor se añade el objeto mensajes para poder enviar los mensajes al usuario final que la aplicación servidora crea necesarios. Siempre en cada tratamiento de objetos JSON el cliente analiza si el servidor ha enviado algún mensaje.

5.1.4.1 Resumen de métodos externos

Todas las peticiones van validadas por el flujo de seguridad del gestor de aplicaciones. Estas peticiones son tratadas por una aplicación del sistema, por lo que si el usuario no se ha autenticado en el mismo, no podrá utilizarlas.

La raíz de la petición http que el cliente tiene que hacer tiene el siguiente formato

```
http://HOST/index.php?app=usuajax&act=ACCION&json=1&VARIABLES
```

donde se cada parte es para:

- **HOST:** Servidor que contiene la aplicación
- **app=usuajax**: Aplicación que se solicita y que es un alias del fichero php del servidor.
- **act=ACCION**: Método a utilizar,
- **json=1**: El formato de comunicación con el servidor. Si json es 0 la comunicación es XML.

Si es 1 el formato es JSON.

- **VARIABLES**: Variables requeridas por cada método y que van anexadas a la cadena anterior. En el formato de paso de variables GET se pasarían así: VARIABLE1=VALOR1& VARIABLE2=VALOR2...& VARIABLEN=VALORN. También permite el paso por el método POST ya que en el servidor siempre se buscan las variables en los dos entornos.

En la siguiente tabla se va a mostrar los diferentes métodos externos que el cliente puede utilizar. Se van a mostrar las ACCIONES que se pueden utilizar y las variables necesarias en cada uno de ellos. En negrita se muestran las variables obligatorias.

Acción	Breve descripción	Variables Entrada	Mensajes informativos
sacaTodos	Envia todos los datos de usuario.	Ninguna	Ninguno
modificaDatos	Modifica la parte de datos personales	Nombre, Apellidos, titulo, firma	Se notifican los datos modificados.
modificaOpciones	Modifica la parte de opciones de aplicación	correorem, copiacorreos	Se notifican los datos modificados.
cambiaPassword	Facilita el cambio de password y de login del usuario	**login**, password1, password2	Se notifican los datos modificados.

Tabla 28: Resumen de los métodos externos disponibles para aplicación de usuario

La columna de mensajes informativos de la tabla anterior muestra que tipo de mensajes envía el servidor al cliente. Estos mensajes se envían en el formato de mensaje JSON bajo el array “mensajes” tal y como se puede ver en la plantilla anterior.

En la siguiente tabla se puede ver un resumen de que plantilla de mensaje se utiliza para que método externo.

Acción	Plantilla
sacaTodos	usuarioDatosPersonales_json.xml
modificaDatos	mensajes.xml
modificaOpciones	mensajes.xml
cambiaPassword	mensajes.xml

Tabla 29: Listado de las plantillas de métodos externos de aplicación usuario

5.1 Aplicación de contactos

5.1.1 Descripción general

Con la intención de darle un valor añadido a la aplicación de CITAS2.0 se ha implementado una aplicación de gestor de contactos. Un usuario registrado en la aplicación tendrá invitaciones creadas por él, una relación de las invitaciones a las que ha sido añadido y un lista de contactos que usará en las invitaciones que cree.

No es imprescindible tener contactos en el listín de contactos para crear una invitación, ya que el usuario puede introducir los contactos directamente en el formulario de creación de invitación. Si el usuario elige la opción de introducir los contactos en el formulario de alta de invitación, la aplicación introducirá automáticamente esta lista en la base de datos de contactos. Esto facilita la creación de esta base de datos de contactos la cual es modificable desde la parte de gestión de contactos.

Un contacto esta constituido por una serie de atributos y por uno o varios correos asociados al mismo. Con esta abstracción se consigue que, cuando se introduce un contacto en una invitación, la cita llegue a cada uno de sus correos asociados a el mismo y se envíe una misma clave de votación para evitar múltiples votaciones por contacto. No se limita el contacto a una cuenta de correo sino que se puede disponer de varias.

En la parte de cliente interfaz enriquecido se ha habilitado una aplicación que hace las veces de gestor de contactos. También se habilita la importación y exportación a diferentes formatos para poder reutilizar la base de datos de diferentes aplicaciones.

5.1.2 Estructura de datos utilizados

La parte de contactos cuenta con dos tablas:

- **contactos**: la cual almacena los contactos registrados en el sistema
- **contactos_email**: las cuentas de correo disponibles para cada contacto.

La tabla de datos donde se almacenan los contactos esta estructurada con los siguientes campos (los tipos son están en formato MySQL):

Campo	Tipo	Descripción
id_persona	int(5)	Identificador numérico del contacto
id_usuario	int(5)	Identificador numérico del usuario al que pertenece el contacto
nombre	varchar(100)	Nombre del contacto
apellidos	varchar(255)	Apellidos del contacto
tparticular	varchar(10)	Teléfono fijo
tmovil	varchar(10)	Teléfono móvil

Tabla 30: Resumen de campos de la tabla de contactos

Campo	Tipo	Descripción
id_persona	int(5)	Identificador numérico del contacto
id_usuario	int(5)	Identificador numérico del usuario al que pertenece el contacto
email	varchar(100)	Cuenta de correo

Tabla 31: Resumen de campos de la tabla de contactos_email

Se han definido unos atributos iniciales útiles para localizar a un contacto, pero puede ser de interés en un futuro hacer un gestor de contactos más complejo añadiendo más campos en la base de datos.

5.1.1 Aplicaciones creadas

Para realizar la gestión de contactos se han definido un conjunto de aplicaciones que se resumen en la siguiente tabla. La localización de los ficheros depende de las variables de configuración globales comentadas en apartados anteriores, por lo que las aplicaciones interfaz enriquecidos cuelgan del directorio “aplicaciones/DINORMAL.

Aplicación	Descripción
contactos.php	Aplicación que controla las diferentes capas para la gestión de usuarios. Habilita cada parte de la plantilla que da la funcionalidad requerida.
contactos_ajax.php	Interfaz para realizar modificaciones de datos asíncronamente

Tabla 32: Aplicaciones de gestión de contactos interfaz enriquecido

Las plantillas de cada aplicación colgaran de los directorios “templates/DPNORMAL”. En las siguientes tablas se resume las plantillas utilizadas para cada aplicación y el orden en el que se muestran en la salida.

Aplicación	Plantillas
contactos.php	cabecera.tpl, cuerpo_contactos.tpl, pies.tpl
contactos_ajax.php	mensajes.xml ó contactos_json.xml

Tabla 33: Plantillas de las aplicaciones de gestión de contactos interfaz enriquecido

5.1.2 Librerías al uso

Para poder realizar una gestión adecuada de los contactos que dispone el usuario se ha creado una clase de contactos que facilita acciones a realizar sobre los mismos. Esta clase se implementa en la librería **libcontactos.php**.

5.1.2.1 Resumen de propiedades

Estas son las propiedades que dispone la librería.

Propiedad	Breve descripción
$id_usuario	Identificador del usuario
$datos_usuario	Array con todos datos del usuario en el formato NOMBRE => VALOR
$datos_filtro	Array en el formato CAMPO => VALOR. Cada elemento del array nos indica un valor a añadir en el filtrado del select
$datos_filtro_orden	String que indica el campo por el que queremos ordenar
$datos_filtro_linferior	String que indica el número inferior por el que empezamos del total de los resultados
$datos_filtro_lsuperior	String que indica el número superior por el que terminamos del total de los resultados

Tabla 34: Resumen de propiedades de la clase contactos

5.1.2.2 Resumen de métodos

Método	Breve descripción
asignaDatos ($datos_conexion, $datos_usuario)	Asigna los datos de usuario iniciales y los de conexión
contactoIntroduceNuevo ($id_persona,$datos_contacto, $datos_emails)	Introduce un nuevo contacto
contactoModifica ($id_persona, $datos_contacto,$datos_emails)	Borra el actual y lo introduce de nuevo con los nuevos valores

Método	Breve descripción
contactoBorrarTodos ()	Borra todo el listado
contactoBorrar ($id_persona)	Borra un contacto
contactoListadoFiltroReset ()	Resetea los valores del filtro
contactoListadoFiltroLimites ($inferior,$superior)	Introduce el rango de contactos a extraer
contactoListadoFiltroOrden ($campo)	Indica el campo de ordenación del select.
contactoListadoFiltroVariosCampo ($filtro)	Indicamos que campos tienen que tener que valores en el select
contactoListadoFiltroUnCampo ($campo,$valor)	Metemos un campo para el filtrado
contactoListadoLanza ()	Lanzamos la extracción de contactos a la que se le aplicarán los filtros definidos
contactoListadoExtraeFila ()	Extraemos una fila del resultado del select
contactoNumeroTotal ()	Mostramos cuantos contactos tiene el usuario
contactoMuestraDatos ($campos, $id_persona)	Mostramos los datos que queramos del contacto indicado
contactoMuestraCorreos ($id_persona)	Muestra los correos del contacto indicado.

Tabla 35: Resumen de las propiedades de la clase contactos

5.1.3 Métodos externos para peticiones asíncronas

Al igual que en otras aplicaciones desarrolladas, para la aplicación de contactos, también se definen una serie de métodos externos que atienden las peticiones asíncronas del cliente. Estas aplicaciones están bajo el conjunto de aplicaciones seguras, por lo que el usuario se ha tenido que autenticar previamente.

A continuación se describen los tipos de mensajes que envía el servidor así como una relación de los métodos externos.

5.1.3.1 Formato de mensaje mensajes.xml

Se encuentra bajo el directorio **templates** y tiene el siguiente contenido:

```
<?xml version="1.0" encoding="UTF-8"?>
<mensajes>
<!-- BLOQUE REPETITIVO INICIO MENSAJE -->
<mensaje ts="{MENSAJE.TS}" nivel="{MENSAJE.NIVEL}"> {MENSAJE.CUERPO}
</mensaje>
<!-- BLOQUE REPETITIVO FIN MENSAJE -->
</mensajes>
```

Este mensaje es un objeto XML puro que el cliente tiene que tratar como tal. La utilidad del mismo es la del envío de mensajes desde el servidor al cliente con notificaciones que se vayan realizando. Se va a utilizar en muchos más métodos externos con esa finalidad.

Para el envío de mensajes se ha explicado otro método previamente en el cual se añadía un objeto mensajes a una definición JSON de un objeto global generado.

5.1.3.2 Formato de mensaje contactos_json.xml

Se encuentra bajo el directorio **templates** y tiene el siguiente contenido:

```
{
      "contactos": [
<!-- BLOQUE REPETITIVO INICIO CONTACTO -->
             {
                  "id_persona" : "{CONTACTO.ID_PERSONA}",
                  "nombre": "{CONTACTO.NOMBRE}",
                  "apellidos": "{CONTACTO.APELLIDOS}",
                  "tparticular": "{CONTACTO.TPARTICULAR}",
                  "tmovil": "{CONTACTO.TMOVIL}",
                  "emails": [ {CONTACTO.CORREO} ]
              },
<!-- BLOQUE REPETITIVO FIN CONTACTO -->
      'VALORFINAL'
      ]
}
```

Como se puede observar se crea un objeto contactos que está formado por un array de datos de contacto. Estos datos son un volcado de la información de la base de datos.

5.1.3.3 Resumen de métodos externos

Siguiendo el mismo formato que se explicó para la aplicación de usuarios, las peticiones que vayan contra la aplicación de contactos tienen el siguiente formato:

```
http://HOST/index.php?app=contactosajax&act=ACCION&json=1&VARIABLES
```

donde la utilidad de los campos es la misma que la explicada en los apartados anteriores.

Ahora para este caso el valor de app es **contactosajax.**

Los diferentes métodos externos que el cliente puede utilizar pueden verse enumerados en la siguiente tabla. Se indican las ACCIONES que se pueden utilizar y las variables necesarias en cada uno de ellos. En negrita se muestran las variables obligatorias.

Acción	Breve descripción	Variables Entrada	Mensajes informativos
sacaTodos	Extrae todos los contactos del usuario sin aplicar filtrado	Ninguna	Ninguno
nuevo	Introducción de un nuevo contacto. Si ya estaba introducido lo comunica.	-**cnombre**: Nombre del contacto -**capellidos**: Apellidos del contacto. - tparticular: Teléfono fijo - tmovil: Teléfono movil - id_persona: Identificador de persona. Si es un nuevo contacto va vacio - cemail: lista de correos separados por comas	Notificación en el caso de error o cuando el contacto ya estaba introducido.
actualiza	Actualización de los campos de un contacto.	-**cnombre**: Nombre del contacto -**capellidos**: Apellidos del contacto. - tparticular: Teléfono fijo - tmovil: Teléfono movil - id_persona: Identificador de persona. - **cemail**: lista de correos separados por comas	Notificación en el caso de error.
borrar	Borra el contacto indicado	- **id_persona**: Identificador de persona.	Notificaciones de borrado correcto o de errores encontrados
importar	Realiza la importación de los contactos. Aquí se utiliza la técnica del Iframe oculto que nos permite realizar la subida de un fichero al servidor mediante el submit de un formulario creado en este iframe.	**contactosAccionesImportarSobreescribir**: Se indica si hay que sobreescribir la lista actual de contactos.	Notifica cada uno de los contactos introducidos.
exportar	Método que nos	**tipo:** formato de exportación	Ninguna

Acción	Breve descripción	Variables Entrada	Mensajes informativos
	permite exportar la lista de los contactos en el formato indicado.	deseado. Esto se refleja en la plantilla correspondiente cuyo nombre se construye con contactos_$tipo.xml	

Tabla 36: Resumen de los métodos externos disponibles para aplicación de contactos

La columna de mensajes informativos de la tabla anterior muestra que tipo de mensajes envía el servidor al cliente. Estos mensajes se envían en el formato de mensaje JSON bajo el array “mensajes” tal y como se puede ver en la plantilla anterior.

En la siguiente tabla podemos ver un resumen de que plantilla de mensaje se utiliza para que método externo.

Acción	Plantilla
sacaTodos	contactos_json.xml
nuevo	mensajes.xml
actualiza	mensajes.xml
borrar	mensajes.xml
importar	mensajes.xml
exportar	contactos_txt.xml, contactos_xml.xml

Tabla 37: Listado de las plantillas de métodos externos de aplicación contactos

El método **exportar** permite exportar la base de datos contactos en el formato predefinido indicado. La petición de este método dará como resultado una comunicación con el servidor del tipo octet-stream con la finalidad de guardalo a fichero. El nombre que se pone por defecto al fichero es de la forma backupContactos_TIMESTAMP.txt, donde TIMESTAMP es la cuenta de los segundos desde 1/1/1970.

Las plantillas de los dos formatos de exportación son las siguientes:

contactos_txt.xml

```
<!-- BLOQUE REPETITIVO INICIO CONTACTO -->
{CONTACTO.NOMBRE};{CONTACTO.APELLIDOS};{CONTACTO.TPARTICULAR};
{CONTACTO.TMOVIL};{CONTACTO.CORREO}
<!-- BLOQUE REPETITIVO FIN CONTACTO -->
```

contactos_xml.xml

```
<?xml version=”1.0” encoding="UTF-8"?>
<contactos>
<!-- BLOQUE REPETITIVO INICIO CONTACTO -->
```

```
        <contacto>
                <nombre>{CONTACTO.NOMBRE}</nombre>
                <apellidos>{CONTACTO.APELLIDOS}</apellidos>
                <tparticular>{CONTACTO.TPARTICULAR}</tparticular>
                <tmovil>{CONTACTO.TMOVIL}</tmovil>
                {CONTACTO.CORREO}
        </contacto>
<!-- BLOQUE REPETITIVO FIN CONTACTO -->
</contactos>
```

5.2 Aplicación de invitaciones

5.2.1 Descripción general

Esta es la parte más importante del sistema CITAS2.0. Se encarga de toda la gestión de invitaciones del sistema la cual pasa por los siguientes puntos:

- **Gestión de las invitaciones creadas:** Aporta una interfaz para la creación, edición y eliminación de las invitaciones que pertenecen al usuario
- **Gestión de las invitaciones a las que el usuario a sido invitado**. También se facilita una interfaz para realizar las acciones de votación y paso a históricas de las invitaciones a las que el usuario ha sido invitado.
- **Gestión de estados de las invitaciones creadas**. Añade otra interfaz que permite acceder a un resumen del estado de las invitaciones que el usuario ha creado y a las que ha sido añadido.

5.2.1 Definición de invitación

Una invitación es un objeto que tiene las siguientes características:

- Un **conjunto de propiedades** que definen a la invitación y que incluyen las fechas propuestas y los usuarios añadidos
- Un **estado de votación** que depende de si el total de los usuarios ha votado o si el administrador ha actuado para forzar la votación
- Un **estado temporal** que es dependiente del estado de votación y la fecha elegida o a la primera fecha propuesta.
- Un **estado de gestión** que muestra si la invitación esta anulada, en el listado de activas, en el histórico, etc

5.2.1.1 Propiedades de una invitación

Una invitación tiene las siguientes características:

- Un **asunto** y una **descripción** de la cita donde se muestra cual va a ser la actividad de la invitación
- El lugar donde se va a **producir** y la **duración** global del evento. Esta duración va a poder ser modificada a nivel de fecha si se desea poner una duración diferente. Existirá la duración global y la duración particular de cada fecha.
- El **tipo de votación** deseado. Esto permite darle mayor flexibilidad a la elección de la fecha
- Unas propiedades que modifican el **comportamiento de la votación** hacia los participantes, permitiendo que cuando un participante vote vea al resto de los participantes y sus votaciones.
- Otras propiedades que modifican las **notificaciones** y **recordatorios** de las votaciones al administrador.
- Y propiedades de **gestión** de la cita donde se indica si la cita esta anulada o no, o si ha pasado al histórico de invitaciones o no.

5.2.1.1 Tipos de votaciones

Para darle una mayor versatilidad a las invitaciones se han definido distintos tipos de votaciones. Se podrá seleccionar una entre las siguientes:

- **Chequear disponibilidad:** Se pide al participante que vote la disponibilidad que tiene en cada una de las fechas propuestas. Se elegirá la fecha que más disponibilidad tenga por todos los usuarios.
- **Ordenar por preferencias:** En este caso el participante tendrá que ordenar por preferencias cada una de las fechas propuestas. Se elegirá aquella fecha que sea más preferente por todos.
- **Anunciar varias y elegir un número de preferencias:** El participante tendrá que elegir el número de fechas acordado por el administrador entre el total de fechas propuestas.
- **Anunciar solo una fecha y solicitar respuesta:** Se pide al participante que vote si la fecha que se le ha propuesto le viene bien.
- **Anunciar solo una fecha y no solicitar respuesta:** Se le notifica al participante cuando se le ha invitado a un evento fijo. No es necesario votar.

5.2.1.1 Estados por lo que pasa una invitación

Se han definido varios estados por los que puede pasar una votación atendiendo a

todos los elementos que pueden modificarla:

Estado de las Votaciones

Una votación puede pasar por los siguientes estados:

- Está **en proceso** si aún no ha finalizado la votación de todos los participantes.
- Está **completa** cuando han votado todos los participantes
- Está **duplicada** si al final han resultado elegidas más de una de las fechas propuestas.
- Está **forzada** si el administrador ha sido el que ha elegido una de las fechas propuestas.

Estado Temporal

Dependiendo de en que estado esté una invitación tenemos:

- Está **pasada** si no se ha realizado la votación y todas las fechas propuestas están pasadas.
- Está **activa** si aún no ha finalizado la votación de todos los participantes y todas las fechas propuestas son posteriores a la fecha actual.

Estado de gestión

Dependiendo del estado impuesto por el administrador, una invitación puede pasar por:

- Está **anulada** cuando el administrador lo decide
- Puede estar en el listado de invitaciones **activas**.
- Puede estar en el listado de invitaciones **históricas**.

Este último grupo simplemente modifica la propiedad de la invitación que indica en que listado esta, por lo que se incluye aquí pero no son propiamente un estado operacional sino simplemente algo administrativo.

5.2.1 Estructura de datos utilizadas

A continuación se muestran unas tablas con la estructura de datos de las tablas utilizadas. Son tipo definidos para la base de datos en uso que es MySQL.

Esta primera tabla se corresponde con el formato de las propiedades básicas de una cita.

Campo	Tipo	Descripción
id_cita	int(5)	Identificador numérico de la invitación
id_usuario	int(5)	Identificador numérico del usuario en el sistema
tipo	varchar(128)	Tipo de votación elegido para la invitación
actividad	varchar(255)	Resumen de la actividad propuesta
comentarios	text	Descripción de la actividad propuesta
localizacion	varchar(255)	Lugar donde se va a realizar el evento
duracion	varchar(255)	Duración global del evento. Se sobreescribe con las duraciones individuales si las hubiera

Campo	Tipo	Descripción
opcpart	varchar(255)	Si se desea que los contactos vean el resto de participantes y sus votaciones.
poremail	varchar(255)	Como se quiere que se notifiquen las votaciones de los contactos
recordatorio	varchar(255)	Desde cuantos días antes de la elección se envían los recordatorios.
identif	varchar(255)	Si el administrador se identifica en el correo que se envía a cada participante.
tipo_dato	varchar(255)	Campo auxiliar de uso cuando se selecciona un tipo de votación particular.
activa	int(1)	Si esta en el listado de invitaciones activas o históricas
mostrarcorreos	varchar(5)	Si se muestran los correos de los participantes
anulada	int(1)	Si la cita ha sido anulada por el administrador

Tabla 38: Listado de campos de la tabla invitaciones

Las fechas propuestas se almacenan en la siguiente estructura con características individuales como pueden ser la duración particular o si ha sido forzada como elegida por el administrador.

Campo	Tipo	Descripción
id_cita	int(5)	Identificador numérico de la invitación
id_usuario	int(5)	Identificador numérico del usuario en el sistema
fecha	varchar(20)	Fecha propuesta para la cita id_cita
duracion	varchar(20)	Duración particular para esa fecha
forzar	int(1)	Si esta forzada como la elegida por el administrador

Tabla 39: Listado de campos de la tabla invitaciones_fechas

El siguiente punto es disponer de los invitados a la cita. Para ello se implementa la siguiente tabla con el código de votación de cada invitado, el día que votó y si el invitado la tiene en el listado de activas o históricas. Esto último es solo a nivel administrativo del usuario registrado, para poder organizar mejor sus eventos en la interfaz.

Campo	Tipo	Descripción
id_cita	int(5)	Identificador numérico de la invitación
id_usuario	int(5)	Identificador numérico del usuario en el sistema
id_persona	int(5)	Identificador numérico del contacto

Campo	Tipo	Descripción
codigo	varchar(255)	Código de votación para ese contacto.
diav	varchar(255)	Timestamp de la votación del usuario
activa	int(1)	Indica si la cita está en el listado histórico o en el de activas

Tabla 40: Listado de campos de la tabla invitaciones_persona

Una vez que el invitado vota, hay que almacenar cual ha sido su votación. Puede parecer que esta tabla se podría haber incluido en la anterior, pero es necesaria ya que existen algunos tipos de votaciones que son múltiples y necesitan de varios registros.

Campo	Tipo	Descripción
id_cita	int(5)	Identificador numérico de la invitación
id_usuario	int(5)	Identificador numérico del usuario en el sistema
id_persona	int(5)	Identificador numérico del contacto
votacion	varchar(20)	Elección del usuario
peso	varchar(5)	Campo necesario para algunos tipos de votaciones.

Tabla 41: Listado de campos de la tabla invitaciones_votaciones

El administrador de la cita puede introducir los comentarios que desee en la misma y hacerlos de carácter público o no. Para ello utilizamos la tabla siguiente.

Campo	Tipo	Descripción
id_nota	int(5)	Identificador numérico del comentario
id_cita	int(5)	Identificador numérico de la invitación
id_usuario	int(5)	Identificador numérico del usuario en el sistema
id_contacto	int(5)	Identificador numérico del contacto del usuario
publico	int(1)	Indica si la nota es de carácter público o no
tiempo	int(15)	El timestamp de la introducción de la nota
comentarios	text	Los comentarios introducidos.

Tabla 42: Listado de campos de la tabla invitaciones_comentarios

5.2.2 Aplicaciones creadas

La gestión de invitaciones se realiza mediante un conjunto de aplicaciones que dan toda la funcionalidad necesaria. Para ello se han definido una serie de aplicaciones tanto para la parte accesible como para la parte de interfaz enriquecido. La localización de los ficheros

depende de las variables de configuración globales comentadas en apartados anteriores, por lo que las aplicaciones interfaz enriquecidos cuelgan del directorio “aplicaciones/DINORMAL” y las accesibles de “aplicaciones/DIACCESIBLE”.

Aplicación	Descripción
invitaciones.php	Aplicación de gestión de invitaciones. Se encarga de habilitar la parte de la plantilla que corresponde a cada acción requerida por el usuario. Pone a visible la parte que da la funcionalidad requerida.
invitaciones_ajax.php	Interfaz para realizar modificaciones de datos asíncronamente

Tabla 43: Aplicaciones de gestión de invitaciones interfaz enriquecido

Aplicación	Descripción
invitaciones.php	Aplicación de gestión de invitaciones. Habilita la plantilla adecuada para cada función.

Tabla 44: Aplicaciones de gestión de invitaciones accesible

Las plantillas de cada aplicación colgaran de los directorios “templates/DPNORMAL” y las accesibles de “templates/DPACCESIBLE”. En las siguientes tablas se resume las plantillas utilizadas para cada aplicación y el orden en el que se muestran en la salida.

Aplicación	Plantillas
invitaciones.php	cabecera.tpl, cuerpo_invitaciones.tpl, pies.tpl
invitaciones_ajax.php	invitacionesResumen_json.xml ó invitacionesResumenOtros_json.xml ó invitacionesSola_json.xml ó mensajes.xml

Tabla 45: Plantillas de las aplicaciones de gestión de usuarios interfaz enriquecido

Aplicación	Plantillas
invitaciones.php	Primero: cabecera.tpl, Segundo, dependiendo de la acción se construye cuerpo_invitaciones_$act.tpl: - cuerpo_invitaciones_anular.tpl - cuerpo_invitaciones_borrar.tpl - cuerpo_invitaciones_crear.tpl - cuerpo_invitaciones_datospersonales.tpl - cuerpo_invitaciones_desforzar.tpl - cuerpo_invitaciones_forzar.tpl - cuerpo_invitaciones_historico.tpl - cuerpo_invitaciones_listarActivas.tpl - cuerpo_invitaciones_mensajes.tpl - cuerpo_invitaciones_modifDatos.tpl - cuerpo_invitaciones_modificarV.tpl - cuerpo_invitaciones_votaciones.tpl Al final se incluye pies.tpl

Tabla 46: Plantillas de las aplicaciones de gestión de usuarios accesible

5.2.3 Librerías al uso

Para poder realizar una gestión adecuada de las invitaciones que ha creado el usuario o a las que ha sido invitado, se dispone de una clase **invitaciones** que facilita acciones a realizar sobre las mismas. Esta clase se implementa en la librería **libinvitaciones.php**.

5.2.3.1 Resumen de propiedades

Estas son las propiedades que dispone la librería.

Propiedad	Breve descripción
$id_usuario	Identificador del usuario
$id_cita	Identificador de la invitación
$datos_conexion	Array con todos datos de conexión a la base de datos de usuarios
$datos_cita	Array en el formato CAMPO => VALOR con todos los datos administrativos de la cita
$datos_fecha	Array con las fechas que pertenecen a la invitación
$datos_contactos	Array con los contactos de la invitación

Tabla 47: Resumen de propiedades de la clase invitaciones

5.2.3.2 Resumen de métodos

Método	Breve descripción
asignaDatos ($datos_conexion,$datos_usuario)	Asigna los datos de usuario iniciales y los de conexión
invitacionesIntroduceCita ($datos_invitacion)	Introduce una nueva invitación en el sistema
invitacionesIntroduceContacto($id_cita,$contacto)	Añade un contacto a la invitación indicada
invitacionesBorraContacto ($id_cita,$contacto)	Borra un contacto de la invitación
invitacionesIntroduceFecha($id_cita,$fecha, $duracion)	Añade una fecha propuesta con la duración deseada
invitacionesBorraFecha ($id_cita,$fecha)	Borra la fecha indicada
invitacionesIntroduceVotacion ($id_cita,$id_persona, $peso,$fecha)	Introduce la votación de un contacto en la cita indicada
invitacionesBorraTodasVotacion ($id_cita, $id_persona)	Quita las votaciones de una persona en la cita
invitacionesBorraUnaVotacion ($id_cita,$id_persona, $votacion)	Quita solo una votación de todas las que haya podido realizar un usuario en la cita
invitacionesMuestraDatosVotaciones($id_cita)	Muestra los datos de las votaciones
invitacionesMuestraDatosVotacionesPeso($id_cita)	Muestra los datos de las votaciones con peso
invitacionesMuestraDatosResumenVotaciones($id_cita)	Saca un resumen de las votaciones realizadas hasta el momento
invitacionesMuestraEstadoVotacion($id_cita)	Muestra un string con todos los estados que tiene activos la invitación
invitacionesMuestraInicioCita($id_cita)	Muestra la primera fecha propuesta
invitacionesMuestraCandidata($id_cita)	Saca la fecha más candidata a ser elegida
invitacionesCaducaCitas($id_cita,$activa)	Cambia el estado administrativo de activa o histórico de la cita
invitacionesVotacionesMuestraForzada ($id_cita)	Indica si la cita tiene la votación forzada o no
invitacionesVotacionesPonForzada ($id_cita,$fecha)	Pone la fecha indicada como forzada en la invitación.
invitacionesVotacionesQuitaForzada ($id_cita)	Quita la fecha forzada si la hubiera
invitacionesAnulaCita ($id_cita)	Pone la cita como anulada administrativamente
invitacionesMuestraAnulada ($id_cita)	Indica si la cita esta anulada o no
invitacionesMuestraIdUsuarioCodigo ($codigo)	Busca si hay algún id_usuario con ese código
invitacionesMuestraDatosCitaPorCodigo($codigo)	Saca los datos de la cita que corresponden al código indicado
invitacionesMuestraCorreoCodigo ($id_cita)	Muestra el correo del participante
invitacionesBorraComentario($id_cita,$id_nota)	Quita el comentario indicado
invitacionesIntroduceComentario($id_cita,$publico, $tiempo,$mensaje)	Mete un comentario con el carácter indicado
invitacionesSacaComentarios($id_cita,$publico)	Saca todos los comentarios de la cita con el

Método	Breve descripción
	carácter indicado
invitacionesMuestraDatosContactos($id_cita, $campos)	Saca los campos indicados de los contactos de la cita
invitacionesMuestraCorreosContactos($id_cita)	Saca todos los correos de la cita así como su código de votación asociado.
invitacionesMuestraFechaVotacionContactos($id_cita)	Muestra en que fecha han votado los contactos de la cita indicada
invitacionesMuestraNombreContactos($id_cita)	Saca los datos de los contados de la cita
invitacionesMuestraDatosContactosSimple($id_cita)	Muestra solo los id_persona de los contactos.
invitacionesMuestraDatosFechas($id_cita)	Saca todas las fechas de la invitación
invitacionesMuestraDatosFechasDuracion($id_cita)	Muestra las fechas con duración particular de cada una
invitacionesBorraFechaPropuesta($id_cita,$fecha)	Borra una fechas del total de propuestas
invitacionesMuestraIdPersonaPorCorreoEnCita($id_cita,$correo)	Busca el identificador de contacto por el correo
invitacionesMuestraIdCita()	Muestra cual es identificador de cita actual
invitacionesMuestraTodasCitas($activa)	Muestra todas las citas del listado de activas o histórico que indiquemos
invitacionesMuestraTodasCitasUsuario($id_usuario, $activa)	Muestra todas las citas en las que aparece el usuario como invitado
invitacionesMuestraDatosCita($id_cita)	Muestra los datos de la cita indicada
invitacionesMuestraDatosOrganizador($id_cita)	Muestra los datos del organizador para enriquecer los mensajes
invitacionesMuestraActividadCita($id_cita)	Muestra la actividad de la invitación
invitacionesEliminaDatosCita($id_usuario, $id_cita)	Borra los datos de la cita
invitacionesEliminaCita($id_cita)	Borra la cita y sus votaciones

Tabla 48: Resumen de propiedades de la clase invitaciones

5.2.4 Métodos externos para peticiones asíncronas

Siguiendo la metodología anterior se muestran los métodos externos que facilitan las peticiones asíncronas desde el navegador. Al igual que el resto, estas aplicaciones están bajo el conjunto de aplicaciones seguras por lo que el usuario se ha tenido que autenticar previamente.

A continuación se describen los tipos de mensajes que envía el servidor así como una relación de los métodos externos.

5.2.4.1 Formato de mensaje mensajes.xml

Se encuentra bajo el directorio **templates** y tiene el siguiente contenido:

```
<?xml version="1.0" encoding="UTF-8"?>
<mensajes>
<!-- BLOQUE REPETITIVO INICIO MENSAJE -->
<mensaje ts="{MENSAJE.TS}" nivel="{MENSAJE.NIVEL}"> {MENSAJE.CUERPO}
</mensaje>
<!-- BLOQUE REPETITIVO FIN MENSAJE -->
</mensajes>
```

Este mensaje es un objeto XML puro que el cliente tiene que tratar como tal. La utilidad del mismo es la del envío de mensajes desde el servidor al cliente con notificaciones que se vayan realizando. Se va a utilizar en muchos más métodos externos con esa finalidad.

También para el envío de mensajes se vio que hay mensajes JSON que contemplan un objeto de mensajes anexado al objeto global que se genera. Este es otro método implementado para el envío de mensajes desde el servidor.

5.2.4.2 Formato de mensaje invitacionesResumen_json.xml

Se encuentra bajo el directorio **templates** y tiene el siguiente contenido:

```
{
  "invitacion": [
 <!-- BLOQUE REPETITIVO INICIO INVITACION -->
      {
            "cita" : "{INVITACION.CITA}",
            "fechainicio" : "{INVITACION.FINI}",
            "tipo" : "{INVITACION.TIPO}",
            "actividad": "{INVITACION.ACTIVIDAD}",
            "localizacion": "{INVITACION.LOCALIZACION}",
            "duracion": "{INVITACION.DURACION}",
            "acabada": "{INVITACION.ACABADA}",
            "contactos": "{INVITACION.CONTACTOS}",
            "votaciones": "{INVITACION.VOTACIONES}",
            "candidatas": [ {INVITACION.CANDIDATAS} ],
            "fechasp": [ {INVITACION.FECHASP} ],
            "forzada": "{INVITACION.FORZADA}",
            "invitados": [ {INVITACION.INVITADOS} ],
            "comentarios": "{INVITACION.COMENTARIOS}"
      },
<!-- BLOQUE REPETITIVO FIN INVITACION -->
      'VALORFINAL'
      ],
       "mensajes": [
<!-- BLOQUE POSIBLE INICIO MENSAJE -->
<!-- BLOQUE REPETITIVO INICIO MENSAJE -->
      {
            "nivel": "{MENSAJE.NIVEL}",
            "ts": "{MENSAJE.TS}",
            "cuerpo": "{MENSAJE.CUERPO}"
```

```
      },
<!-- BLOQUE REPETITIVO FIN MENSAJE -->
<!-- BLOQUE POSIBLE FIN MENSAJE -->
      'VALORFINAL'
      ]
}
```

Este mensaje envía un resumen de todas las invitaciones en las que el usuario es administrador. Se genera el objeto **invitacion** que se es un array de estructuras las cuales contienen la información resumida de cada invitación. Cada campo tiene el siguiente significado:

- **cita**: Identificado de la cita
- **fechainicio**: Fecha propuesta más antigua
- **tipo**: Tipo de votación
- **actividad**: Campo actividad de la invitación
- **localizacion**: Campo localización de la invitación
- **duracion**: Duración global del evento
- **acabada**: String con todos los estados de la invitación
- **contactos**: Número de contactos de la invitación
- **votaciones**: Número de votaciones realizadas
- **candidatas**: Todas la fechas que son candidatas a ser las elegidas
- **fechasp**: Fechas propuestas
- **forzada**: Si la invitación esta forzada o no
- **invitados**: Todos los identificadores de contactos
- **comentarios**: Campo comentarios de la invitación

Se aprovecha también este mensaje JSON para enviar los mensajes que considere pertinente el servidor a los clientes mediante el objeto **mensajes**.

5.2.4.1 Formato de mensaje invitacionesResumenOtros_json.xml

Para notificar los mensajes en los que el usuario aparece como invitado se hace uso de la siguiente plantilla. Se encuentra bajo el directorio **templates** y tiene el siguiente contenido:

```
{
  "invitacion": [
 <!-- BLOQUE REPETITIVO INICIO INVITACION -->
      {
            "cita" : "{INVITACION.CITA}",
            "fechainicio" : "{INVITACION.FINI}",
            "tipo" : "{INVITACION.TIPO}",
            "actividad": "{INVITACION.ACTIVIDAD}",
            "localizacion": "{INVITACION.LOCALIZACION}",
```

```
            "duracion": "{INVITACION.DURACION}",
            "acabada": "{INVITACION.ACABADA}",
            "contactos": "{INVITACION.CONTACTOS}",
            "codigo": "{INVITACION.CODIGO}",
            "votaciones": "{INVITACION.VOTACIONES}",
            "candidatas": [ {INVITACION.CANDIDATAS} ],
            "fechasp": [ {INVITACION.FECHASP} ],
            "forzada": "{INVITACION.FORZADA}",
            "invitados": [ {INVITACION.INVITADOS} ],
            "comentarios": "{INVITACION.COMENTARIOS}"
      },
<!-- BLOQUE REPETITIVO FIN INVITACION -->
      'VALORFINAL'
      ],
      "mensajes": [
<!-- BLOQUE POSIBLE INICIO MENSAJE -->
<!-- BLOQUE REPETITIVO INICIO MENSAJE -->
      {
            "nivel": "{MENSAJE.NIVEL}",
            "ts": "{MENSAJE.TS}",
            "cuerpo": "{MENSAJE.CUERPO}"
      },
<!-- BLOQUE REPETITIVO FIN MENSAJE -->
<!-- BLOQUE POSIBLE FIN MENSAJE -->
      'VALORFINAL'
      ]
}
```

La única diferencia con respecto a la anterior es que se introduce el campo **codigo** para indicar cual es el código de votación que el usuario tiene.

Se aprovecha también este mensaje JSON para enviar los mensajes que considere pertinente el servidor a los clientes mediante el objeto **mensajes**.

5.2.4.2 Formato de mensaje invitacionesSola_json.xml

Una vez que el usuario decide acceder al contenido de una invitación exclusivamente, se requiere que el servidor envíe toda la información concerniente a la invitación seleccionada. Se hace uso del siguiente mensaje que se encuentra bajo el directorio **templates** y tiene el siguiente contenido:

```
{
  "invitacion":
      {
            "cita" : "{CITA}",
            "tipo" : "{TIPO}",
            "tipodato" : "{TIPODATO}",
            "actividad": "{ACTIVIDAD}",
            "comentarios": "{COMENTARIOS}",
            "localizacion": "{LOCALIZACION}",
            "duracion": "{DURACION}",
            "opcpart": "{PARTICIPANTES}",
            "poremail": "{POREMAIL}",
```

```
            "recordatorio": "{RECORDATORIO}",
            "identif": "{IDENTIFICARME}",
            "mostrarcorreos": "{MOSTRARCORREOS}",
            "fpropuestas": [ {FP} ] ,
            "forzada": "{FORZADA}",
            "fpduracion": [ {FPDURACION} ] ,
            "contactos": [ {CONTACTOS} ],
            "contactosv": [
<!-- BLOQUE REPETITIVO INICIO CONTACTOSV -->
                  {
                        fechav: "{CONTACTOSV.FECHAV}",
                        contacto: "{CONTACTOSV.CONTACTO}"
                  },
<!-- BLOQUE REPETITIVO FIN CONTACTOSV -->
                  'VALORFINAL'
                  ],
            "votaciones": [
<!-- BLOQUE REPETITIVO INICIO VOTACIONES -->
                  {
                        fecha: "{VOTACIONES.FECHA}",
                        contactos: [{VOTACIONES.CONTACTOS}]
                  },
<!-- BLOQUE REPETITIVO FIN VOTACIONES -->
                  'VALORFINAL'
                  ],
            "notas": [
<!-- BLOQUE REPETITIVO INICIO NOTAS -->
                  {
                        id_nota: "{NOTAS.IDNOTA}",
                        publico: "{NOTAS.PUBLICO}",
                        tiempo: "{NOTAS.TIEMPO}",
                        comentarios: "{NOTAS.COMENTARIOS}"
                  },
<!-- BLOQUE REPETITIVO FIN NOTAS -->
                  'VALORFINAL'
                  ]
      },
      "mensajes": [
<!-- BLOQUE POSIBLE INICIO MENSAJE -->
<!-- BLOQUE REPETITIVO INICIO MENSAJE -->
      {
            "nivel": "{MENSAJE.NIVEL}",
            "ts": "{MENSAJE.TS}",
            "cuerpo": "{MENSAJE.CUERPO}"
      },
<!-- BLOQUE REPETITIVO FIN MENSAJE -->
<!-- BLOQUE POSIBLE FIN MENSAJE -->
      'VALORFINAL'
      ]
}
```

Un detalle más extenso del significado de cada uno de los campos es el siguiente:

- **cita:** Campo id_cita de la tabla de invitaciones y para la invitación indicada
- **tipo:** Campo tipo de la tabla de invitaciones y para la invitación indicada
- **tipodato**: Campo tipo_dato de la tabla de invitaciones y para la invitación indicada

- **actividad**: Campo actividad de la tabla de invitaciones y para la invitación indicada
- **comentarios**: Campo comentarios de la tabla de invitaciones y para la invitación indicada
- **localizacion**: Campo localizacion de la tabla de invitaciones y para la invitación indicada
- **duracion**: Campo duracion de la tabla de invitaciones y para la invitación indicada
- **opcpart**: Campo opcpart de la tabla de invitaciones y para la invitación indicada
- **poremail**: Campo poremail de la tabla de invitaciones y para la invitación indicada
- **recordatorio**: Campo recordatorio de la tabla de invitaciones y para la invitación indicada
- **identif**: Campo identif de la tabla de invitaciones y para la invitación indicada
- **mostrarcorreos**: Campo mostrarcorreos de la tabla de invitaciones y para la invitación indicada
- **fpropuestas**: Array con las fechas propuestas
- **forzada**: En el caso de que la hubiera muestra la fecha forzada
- **fpduracion**: Array con las fechas propuestas con sus duraciones
- **contactos**: Array con los identificadores de los contactos
- **contactosv**: Array que contiene las votaciones de los contactos
  - **contactosv.fechav**: Fecha en la que votó el contacto
  - **contactosv.contacto**: Identificador de contacto
- **votaciones**: Array con todas las votaciones por fecha
  - **votaciones.fecha**: Fecha propuesta
  - **votaciones.contactos**: Contactos que han seleccionado esta fecha con el peso concreto
- **notas**: Array con las notas que tenga la invitación
  - **notas.id_nota**: identificador de nota
  - **notas.publico**: carácter de la misca
  - **notas.tiempo**: cuando se introdujo
  - **notas.comentarios**: Texto de la notas

Se aprovecha también este mensaje JSON para enviar los mensajes que considere pertinente el servidor a los clientes mediante el objeto **mensajes**.

5.2.4.1 Resumen de métodos externos

Para poder acceder a los métodos externos de la aplicación de invitaciones se

construye la url de la siguiente forma:

```
http://HOST/index.php?app=invajax&act=ACCION&json=1&VARIABLES
```

donde la utilidad de los campos es la misma que la explicada en los apartados anteriores.

Ahora para este caso el valor de app es **invajax.**

En la siguiente tabla se va a mostrar los diferentes métodos externos que el cliente puede utilizar. Se van a mostrar las ACCIONES que se pueden utilizar y las variables necesarias en cada uno de ellos. En negrita se muestran las variables obligatorias.

Acción	Variables Entrada	Mensajes informativos
Breve descripción		
mostrarResumenMios	Ninguna	Ninguno
Genera el listado resumen de las invitaciones activas en las que el usuario es administrador		
mostrarResumenOtros	Ninguna	Ninguno
Muestra el resumen de las invitaciones activas en las que el usuario es ha sido añadido		
mostrarResumenMiosHistorico	Ninguna	Ninguno
Genera el listado resumen de las invitaciones históricas en las que el usuario es administrador		
mostrarResumenOtrosHistorico	Ninguna	Ninguno
Muestra el resumen de las invitaciones históricas en las que el usuario es ha sido añadido		
mostrarDatosInvitacion	**id_cita**: identificador numérico de la cita	Ninguno
Muestra el contenido detallado de la cita indicada		
crearInvitacion	- contactos: identificadores de los contactos - fechas: Fechas propuestas - fduracion: Fechas propuestas con duración **- actividad:** asunto de la cita - comentarios: descripción de la invitación - localizacion: Lugar de la misma - tipo: Tipo de votación - duracion: Duración global - id_cita: Identificador de la invitación - opcpart: Opciones de los participantes - poremail: Notificación por email - recordatorio: Si se realiza recordatorio - recordatoriodia: Cuantos días antes - identif: Si identificamos al administrador en los correos	Se notifica que se ha creado la cita.

Acción	Variables Entrada	Mensajes informativos
	- notificar: Si se notifican los cambios a los invitados - tipo_dato: Campo con valor de tipo de votación si es requerido - mostrarcorreos: Si se muestran los correos de los participantes.	
Método para la creación o edición de invitación.		
borrarCita	- **id_cita**: Identificador de la invitación - **activa**: Si está en el listado de activas o históricas	Se notifica el borrado de la cita
Realiza el borrado de la cita indicada.		
caducaCita	- **id_cita**: Identificador de la invitación - **activa**: Si está en el listado de activas o históricas - invitado: Si en la cita el usuario esta como invitado o administrador	Se indica que la cita ha pasado al histórico
Realiza el paso de una cita del listado de activas al histórico		
activaCita	- **id_cita**: Identificador de la invitación - **activa**: Si está en el listado de activas o históricas - invitado: Si en la cita el usuario esta como invitado o administrador	Se indica que la cita ha pasado al listado de activas.
Realiza el paso de una cita del histórico al listado de activas		
enviaMensaje	- **id_cita**: Identificador de la invitación - **mensaje**: Mensaje a enviar - contactos: listado de contactos a los que enviar - publico: carácter del mensaje	Se informa del envío del mensaje
Este método nos permite enviar mensajes a usuario o introducirlos como notas en la invitación		
borrarMensaje	- **id_cita**: Identificador de la invitación - **id_nota**: Identificador de la nota	Notificación de borrado
Elimina el mensaje indicado del listado de los mensajes de la invitación		
actualizaVotaciones	- **id_cita**: Identificador de la invitación	Indica el número de votaciones cambiadas
Se encarga de actualizar las votaciones modificadas por el administrador en el menú de edición. Solo modifica las cambiadas por el administrador en el menú cliente		
forzarFecha	- **id_cita**: Identificador de la invitación	Indica si se ha forzado

Acción	Variables Entrada	Mensajes informativos
	- **fechap**: Fecha propuesta a forzar. - envia: Si se envía un mensaje de la elección a todos los contactos. - comentarios: los comentarios añadidos al mensaje de elección	correctamente la invitación
Fuerza una fecha a ser la elegida		
desforzarFecha	- **id_cita**: Identificador de la invitación	Se notifica el estado del comando
Se quita la fecha forzada de la cita. Solo puede haber una		
anularCita	- **id_cita**: Identificador de la invitación - envia: Si se envía un mensaje de la anulación a todos los contactos. - comentarios: los comentarios añadidos al mensaje de anulación	Se notifica el estado del comando
Se realiza la anulación del evento y se permite el envío de mensajes a los participantes.		

Tabla 49: Resumen de los métodos externos disponibles para aplicación de invitaciones

La columna de mensajes informativos de la tabla anterior muestra que tipo de mensajes envía el servidor al cliente. Estos mensajes se envían en el formato de mensaje JSON bajo el array “mensajes” tal y como se puede ver en la plantilla anterior.

En la siguiente tabla podemos ver un resumen de que plantilla de mensaje se utiliza para que método externo.

Acción	Plantilla
mostrarResumenMios	invitacionesResumen_json.xml
mostrarResumenOtros	invitacionesResumenOtros_json.xml
mostrarResumenMiosHistorico	invitacionesResumen_json.xml
mostrarResumenOtrosHistorico	invitacionesResumenOtros_json.xml
mostrarDatosInvitacion	invitacionesSola_json.xml
crearInvitacion	invitacionesSola_json.xml
borrarCita	invitacionesResumen_json.xml
caducaCita	invitacionesResumen_json.xml
activaCita	invitacionesResumen_json.xml
enviaMensaje	mensajes.xml
borrarMensaje	mensajes.xml

Acción	Plantilla
actualizaVotaciones	invitacionesSola_json.xml
forzarFecha	mensajes.xml
desforzarFecha	mensajes.xml
anularCita	mensajes.xml

Tabla 50: Listado de las plantillas de métodos externos de aplicación invitaciones

5.3 Aplicaciones no seguras

5.3.1 Introducción

El gestor de aplicaciones permite la ejecución de aplicaciones no seguras, las cuales se definen en el fichero de configuración y están libres del control de usuarios del sistema.

Se han definido algunas para dar a la aplicación algunas funcionalidades sin necesidad de estar registrado en el sistema y que se van a ver a continuación.

5.3.2 Registro de usuarios

El sistema necesita de una aplicación para poder registrar usuarios en el mismo. Como es lógico tiene que estar fuera del sistema de control de acceso para poder permitir el registro inicial.

Se basa en un formulario de entrada en el que hay que introducir los datos básicos del usuario a crear y un login o cuenta de correo asociada. Es a esa cuenta donde se envían los datos asociados del registro y cualquier notificación. También este login sirve como indicador de búsqueda en los contactos de otros usuarios registrados, con la finalidad de mostrar en un listado inicial las citas a las que el usuario ha sido invitado.

Esta aplicación no tiene mucha más complejidad que la de introducir el usuario en la base de datos previo chequeo de que no existe un usuario con el mismo login.

5.3.3 Interfaz de votaciones de cliente

También es necesario tener un interfaz de votaciones externo. Desde aquí, los invitados que reciben la notificación del evento vía email, pueden acceder al enlace facilitado para realizar su votación pertinente. Dependiendo de las características que el administrador de la cita haya definido en la misma, el invitado podrá ver mediante este interfaz al resto de invitados, sus votaciones y los correos asociados a la etiqueta de contacto.

Se ha desarrollado una versión accesible, basada únicamente en HTML y CSS , y otra versión de interfaz enriquecido que utiliza javascript para hacer más usable el mecanismo de votación.

El invitado cuando acceda se encontrará con una interfaz donde aparecen las fechas propuestas y unos valores asociadas al tipo de votación seleccionado por el administrador. Dependiendo del trascurso de las votaciones se mostrará en distinto color las fechas propuestas que vayan siendo las candidatas a ser las elegidas.

5.4 Notificaciones

5.4.1 Introducción

La filosofía de la aplicación es la de tener un sistema de gestión de eventos que utilice notificaciones para alertar, tanto a invitados como administradores, de nuevos eventos o modificaciones en el transcurso de las citas.

Para poder abarcar todas las posibles notificaciones se han definido las siguientes notificaciones, las cuales se describen como plantillas del sistema.

Plantilla de Mensaje	Utilización
mensajeUsuario.tpl	Para envío de mensajes a usuario.
notificacionUsuarios.tpl	Envío inicial de la invitación con la descripción de la misma
mensajeVotacionTodosUsuario.tpl	Mensaje de envío al administrador con la información de todas la votaciones.
recordatorioCitaUsuarios.tpl	Si el sistema envía recordatorios a los invitados de la cita una vez que ya ha sido elegida
mensajeVotacionUnUsuario.tpl	Si el administrador decide que quiere notificaciones individuales de cada votación
recordatorioDuplicidadCita.tpl	Si en la elección de la cita hay duplicidad se envía un mensaje al administrador para informarle de la situación

Tabla 51: Listado de plantillas de notificaciones del sistema

Se encuentran en el directorio “templates/mensajes” y no hay diferenciación en parte accesible o no ya que no tiene sentido en este apartado.

En el contenido de las notificaciones toman importancia los parámetros configurados en

la invitación. De tal modo que se han definido partes que pueden o no mostrar su contenido dependiendo de si el administrador así lo ha configurado en la invitación.

Cualquier modificación en los mensajes que el sistema se simplifica hasta el punto de solo tener que tocar la plantilla adecuada.

5.5 Programas externos de mantenimiento

5.5.1 Introducción

Para poder realizar diferentes tareas del sistema sin afectar al funcionamiento de la aplicación se han desarrollado algunos scripts que dan funcionalidad al mismo. Se ejecutan periódicamente gracias al cron del servidor y están desarrollados en php para reutilizar el código ya realizado del sistema.

Se ha normalizado la ubicación de los scripts en el directorio **script** que cuelga del raíz de la aplicación. Para darle seguridad se ha añadido un .htaccess que no permite la visualización de este directorio por parte del servidor web. Este uso de filtrados de ficheros también se ha utilizado en algunos directorios de la estructura para evitar el acceso directo a la información contenida en los mismos.

5.5.1.1 enviaRecordatorio.php

Este script se ejecuta cada 8 horas y tiene como finalidad el envío de los recordatorios de las citas. El funcionamiento se puede resumir en los siguientes puntos:

- Se extraen todas las invitaciones activas del sistema
- Se comprueba el estado de cada una y si sus votaciones están completadas o la elección ha sido forzada se calcula la diferencia de tiempo respecto a la primera fecha elegida. Se tienen en cuenta la fecha actual, la primera fecha elegida, ya que puede darse el caso de duplicidad, y el número de días anteriores a partir de los cuales comienzan los recordatorios.
- Si solo existe una fecha elegida se envía un recordatorio a cada uno de los contactos con un resumen de la cita y la url de la votación.
- Si existe duplicidad en la votación se envía un mensaje de duplicidad de fecha al administrador con el fin de que el administrador fuerce una fecha.

5.1 Comparativa con sistemas de elección de eventos actuales

5.1.1 Introducción

En este apartado se va a comparar las prestaciones que da la aplicación CITAS2.0 frente a otras alternativas de aplicaciones de organización de eventos del mercado.

Se va a realizar la comparativa en base a los siguientes objetivos:

- **Accesibilidad**: Ver que nivel de adecuación consigue con herramientas como TAW
- **Requiere javascript**: Ver si la aplicación es funcional para dispositivos que no utilicen javascript
- **Coste**: Ver si es una alternativa totalmente gratuita, con funcionalidad limitada gratuita o totalmente de pago
- **Funcional**: Si es sencillo utilizarla
- **Intuitiva**: Si fácilmente se descubre su uso sin necesidad de explicaciones
- **Documentación**: Si dispone de una ayuda online que facilite su uso
- **Exportabilidad**: Si la información es exportable mediante estándares a otros sistemas
- **Añadidos**: Si tiene funcionalidades extras a las necesarias
- **Clasificación de eventos**: Si la aplicación dispone de una clasificación clara de tipos de eventos

5.1.1 AgreeAdate

Esta aplicación esta ubicada en la url http://www.agreeadate.com.

Para poder acceder a la aplicación es necesario realizar un registro previo en la misma.

Nada más entrar se muestra un listado con el estado de las invitaciones creadas y se brinda la posibilidad de crear un evento eligiéndolo entre una lista de tipos de evento. Diferencia entre un tipo de evento u otro ofreciendo un interfaz de creación distinto para cada tipo.

En el proceso de creación del evento se separa se diferencia en páginas cada una de los pasos que se necesitan para crear el evento haciendo que sea muy guiada. Llama la atención que permite la inclusión de una encuesta en la creación del evento, de tal modo que se puede preguntar que tipo de comida les gusta, que tipo de estancia prefieren, etc.

Los contactos se meten directamente en el proceso de entrada en base al correo electrónico del mismo no disponiendo de la posibilidad de tener varios correos para un contacto. Si que permite la opción de agrupar los contactos y trabajar directamente con el

grupo de personas.

Si se analiza la aplicación en base a los puntos anteriores tenemos:

- **Accesibilidad**: Con TAW AA
- **Requiere javascript**: Si. Si el dispositivo no ejecuta javascript no se puede utilizar la aplicación.
- **Coste**: Tiene funcionalidad limitada. Si se desea tener más de 100 contactos en la agenda hay que pagar.
- **Funcional**: Es bastante sencillo de utilizar y ágil gracias a las mejoras introducidas con Javascript
- **Intuitiva**: El proceso de creación de evento es guiado y fácil de seguir
- **Documentación**: Dispone de ayuda online
- **Exportabilidad**: No se puede exportar la información a sistemas externos mediante iCAL o RSS. Si que se indica que es compatible con Outlook
- **Añadidos**: Es interesante el envío de una encuesta en el mensaje de invitación al evento para aclarar alguna cuestión.
- **Clasificación de eventos**: Separa los tipos de eventos que ofrece el sistema.

5.1.1 Meet-o-Matic

Esta herramienta se encuenta en la siguiente URL http://beta.meetomatic.com/calendar.php. Su principal atractivo es la simplicidad que ofrece la aplicación en la creación de eventos ya que permite crear un evento simplemente con el nombre del mismo, un correo y la selección de unas fechas. Esto envía un correo al administrador para que lo reenvíe a los participantes que considere. Al participante le llega una dirección donde aparece un formulario sobre el que se tiene que inscribir pasando a formar parte del grupo de votantes. No cuenta con una gestión de contactos que permita la inclusión directa de contactos cuando se crea la invitación. Tampoco se puede definir que tipo de votación se desea para los participantes. Solo se consulta la disponibilidad al evento

Si se analiza la aplicación en base a los puntos anteriores tenemos:

- **Accesibilidad**: No consigue ningún nivel de adecuación con TAW
- **Requiere javascript**: Si. Si el dispositivo no ejecuta javascript no se puede utilizar la aplicación.
- **Coste**: Es totalmente gratuito
- **Funcional**: Es bastante sencillo de utilizar y ágil gracias a las mejoras introducidas con Javascript

- **Intuitiva**: El proceso de creación de evento va desde una visión muy sencilla para crear eventos de forma rápida, a otra visión más compleja si queremos modificar más detalles de la aplicación.
- **Documentación**: Dispone de ayuda online
- **Exportabilidad**: No se puede exportar la información a sistemas externos mediante iCAL o RSS.
- **Añadidos**: La aplicación se basa en la sencillez de la creación de eventos por lo que no cuenta con ningún añadido como gestor de contactos, mensajería, tipos de votaciones, etc
- **Clasificación de eventos**: Todos los eventos son tratados igual

5.1.1 Meeting Wizard

Esta herramienta esta disponible en la url http://www.meetingwizard.com/.

Se muestra como una herramienta de trabajo para poder gestionar los eventos.

Dispone de un interfaz “express” que agrupa en una misma página todas las partes de la creación del evento.

Se definen distintos tipos de eventos así como distintos tipos de votación.

También cuenta con la posibilidad de tener una agenda de contactos.

Si se analiza la aplicación en base a los puntos anteriores tenemos:

- **Accesibilidad**: No consigue ningún nivel de adecuación con TAW
- **Requiere javascript**: No. Aunque cuenta con funcionalidad mejorada con javascript, se puede trabajar directamente sobre los inputs del formulario.
- **Coste**: Su uso es gratuito
- **Funcional**: Es bastante sencillo de utilizar pero el interfaz no esta muy depurada. Por ejemplo en la vista de reuniones, si se dispone de varias requiere de mucha pantalla para mostrar la información.
- **Intuitiva**: Es muy sencilla de utilizar.
- **Documentación**: Dispone de ayuda online
- **Exportabilidad**: No se puede exportar la información a sistemas externos mediante iCAL o RSS. Si que se puede importar y exportar la agenda de contactos.
- **Añadidos**: Cuenta con una agenda de contactos.
- **Clasificación de eventos**: Si que la tiene

5.1.1 Doodle

Esta aplicación esta ubicada en la url http://www.doodle.com.

Lo más interesante de esta herramienta es que da la posibilidad de organizar eventos sin registro. Mediante tres sencillos pasos la herramienta organiza el evento y da al administrador un enlace para que los participantes voten y otro para realizar la administración del evento. Los participantes no son introducidos previamente, sino que serán a los que el administrador envíe el correo.

Si por el contrario el usuario decide registrarse podrá optar a un listado de las invitaciones que ha creado así como a las que ha sido invitado. El sistema ofrece herramientas de administración para modificar la cita, las votaciones , etc.

Si se analiza la aplicación en base a los puntos anteriores tenemos:

- **Accesibilidad**: Nivel A con TAW para la mayoría de las páginas.
- **Requiere javascript**: No. El uso de javascript es simplemente para enriquecer la interfaz. Se trabaja de la misma manera sin javascript
- **Coste**: Es gratuito
- **Funcional**: Es bastante sencillo de utilizar pero al ser más accesible no cuenta con demasiados enriquecimientos de interfaz.
- **Intuitiva**: El proceso de creación de evento es guiado y fácil de seguir
- **Documentación**: Dispone de ayuda online
- **Exportabilidad**: No se puede exportar la información a sistemas externos mediante iCAL o RSS.
- **Añadidos**: No cuenta con agenda de contactos. Solo gestiona las invitaciones creadas
- **Clasificación de eventos**: Solo tiene dos tipos de votaciones ya que puede solicitar disponibilidad en determinadas fechas y realizar encuestas.

5.1.1 Google Calendar

Se este apartado se va a analizar la posibilidad que da Google Calendar de crear eventos invitando a contactos.

Cuando se crea un evento directamente Calendar te permite invitar a cualquier persona, bien de la lista de contactos como cualquier correo electrónico introducido. Permite configurar también las opciones que pueden tener los invitados de invitar a otros a su vez, ver la lista de invitados, y hacer la invitación comunitaria al permitirles editarla.

La votación que pueden realizar los invitados solo se basa en la disponibilidad de tienen para el evento no dando la opción de varias alternativas en fechas. Para conseguir eso habría

que realizar varios eventos.

Si se analiza la aplicación en base a los puntos anteriores tenemos:

- **Accesibilidad**: Nivel A con TAW para la mayoría de las páginas.
- **Requiere javascript**: Si. Un dispositivo sin javascript no puede utilizar la aplicación.
- **Coste**: Es gratuito
- **Funcional**: Es bastante sencillo de utilizar.
- **Intuitiva**: Es muy sencillo crear un evento. En una página se muestran todos los objetos posibles.
- **Documentación**: Dispone de ayuda online
- **Exportabilidad**: Si que se puede exportar e importar eventos mediante estándares.
- **Añadidos**: Tiene todos los de google.
- **Clasificación de eventos**: Realmente no es un gestor de invitaciones. Es un añadido al calendar. La única clasificación es la de invitación y la votación esta fijada a tres estados que son Si, No y Quizás.

5.1.1 Comparación con CITAS2.0

CITAS2.0 incorpora la todas las funcionalidades básicas descritas en los ejemplos anteriores. Añade alguna aplicación más como una agenda de contactos completa, la posibilidad de tener varios correos para un contacto, la exportación de los calendarios siguiendo los estándares actuales y respetando la privacidad del usuario al hacer una exportación basada en una autenticación en el sistema y un sistema de mensajería que permite tanto introducir comentarios en el evento como enviárselos a los participantes.

Aplicando el mismo criterio que a sus competidores podemos ver cada uno de los puntos:

- **Accesibilidad**: Nivel AA con TAW para la parte de interfaz enriquecido y AAA para la parte accesible.
- **Requiere javascript**: No. Un dispositivo sin javascript puede utilizar la aplicación.
- **Coste**: Su uso es gratuito y de código libre
- **Funcional**: Se maneja de una forma muy sencilla cada una de las vistas de la aplicación. Gracias a AJAX se ha conseguido dar rapidez al sistema.
- **Intuitiva**: Es muy sencilla de utilizar.
- **Documentación**: Dispone de ayuda online
- **Exportabilidad**: Se puede exportar la información de los eventos a sistemas externos mediante iCAL o RSS. También se puede importar y exportar la agenda de contactos.

- **Añadidos**: El sistema cuenta con una agenda de contactos más o menos completa, una sistema de envío de mensajes a los participantes, un sistema de anotación en el evento
- **Clasificación de eventos**: Todos los eventos son tratados de la misma manera. Solo cambia el tipo de votación deseada.

5.1.1 Resumen de comparaciones

En la tablaq siguiente se muestra un resumen de la comparativa atendiendo a los siguientes indicadores

- **Accesibilidad:** Da el nivel de adecuación proporcionado en distintas partes de la aplicación dado por TAW. Evalúa solo el código HTML.
- **Coste de uso**: Si su uso es gratuito
- **Obligatorio javascript**: Si es necesario javascript para que la aplicación funcione
- **Aumento de la usabilidad mediante javascript**: Si javascript hace la aplicación más funcionalidad y usable.
- **Intuitiva**: Es muy sencilla de utilizar.
- **Documentación**: Dispone de ayuda online
- **Exportabilidad invitaciones**: Se puede exportar la información de los eventos a sistemas externos mediante iCAL o RSS.
- **Exportabilidad contactos**: Se puede importar y exportar la agenda de contactos.
- **Añadidos extras**: Si el sistema cuenta con aplicaciones extras a la mera creación de invitaciones.
- **Gestor de contactos**: si cuenta con esta aplicación
- **Gestor de votaciones**: si cuenta con esta aplicación
- **Clasificación de eventos**: Todos los eventos son tratados de la misma manera. Solo cambia el tipo de votación deseada.

  Si se agrupan todas las comparaciones en la tabla siguiente:

	Citas2.0	Agree A Date	Meet-o-Matic	Meeting Wizard	Doodle	Google Calendar
Accesibilidad – Nivel de adecuación	AA/AAA(1)	AA	Ninguno	Ninguno	A	A
Coste de uso	No	Parcial(2)	No	No	No	No
Obligatorio javascript	No	Si	Si	No	No	Si
Aumento usabilidad por javascript	Mucho	Normal	Normal	Poco	Poco	Mucho
Intuitiva	Si	Si	Si	Regular	Si	Si
Documentación	Si	Si	Si	Si	Si	Si
Exportabilidad Invitaciones	Ical, RSS	No	No	No	No	Ical,RSS
Exportabilidad Contactos	CSV	Outlook	No	CSV, Outlook	No	CVS,vCard
Añadidos extras	Mensajería, anotaciones	Encuesta	No	No	No	No
Gestor de contactos	Si	Si	No	Si	No	Si
Gestor de votaciones	Si	Si	No	Si	Si	No
Clasificación de eventos	Si	Si	No	Si	No	No

Tabla 52: Comparativa con aplicaciones externas

(1) Clasificación AA para versión enriquecida y AAA para versión accesible. Fuente TAW

(2) Si se quieren más de 100 contactos en la agenda hay que pagar

Estas mejoras sobre las aplicaciones tratadas hacen que CITAS2.0 sea una alternativa muy buena a las actuales e incluso la única en algunos aspectos como la exportación de los datos de los eventos de forma remota.

Capítulo 6. Descripción de las aplicaciones de cliente de CITAS2.0

6.1 Introducción

Como ya se ha comentado en capítulos anteriores se han realizado dos implementaciones distintas atendiendo al userAgent dado por la petición cliente:

- **Parte de interfaz enriquecido**: Es una parte más rica en funcionalidad y basada principalmente en JavaScript. En una petición inicial al sitio, el cliente se baja la plantilla de la aplicación solicitada y el programa javascript que lo gestiona. El programa se encargará de la modificación de esta plantilla en base a los datos obtenido mediante la llamada remota a los métodos externos de forma asíncrona, los cuales mandarán la información en JSON ó XML.
- **Parte accesible**: Esta parte es la de compatibilidad con el resto de periféricos y versiones antiguas de navegadores. Esta basada en HTML y CSS y sigue un uso tradicional de la interacción web con un servidor. El cliente solicita una página con unos valores y se baja toda la información en una página web completa.

6.1 Parte de interfaz enriquecido

6.1.1 Introducción

Esta parte esta construida básicamente en javascript. Dependiendo de la aplicación solicitada el servidor devolverá una plantilla u otra, las cuales llevan asociadas un programa javascript particular.

La programación se realiza en base a eventos, siendo el evento inicial la carga de la página. Otros eventos que se utilizan para darle dinamismo al entorno son temporizadores, cambio de estado de elementos de formularios, clicks en enlaces o zonas, movimientos del ratón por zonas concretas, etc.

Aunque la base es fundamentalmente javascript se ha perseguido el delimitar el uso a parte pequeñas de la aplicación evitando que toda una aplicación controle muchas capas de

información.

Se necesita de un directorio que tenga visibilidad pública dada por servidor web. En la estructura de directorios se ha creado un directorio, “exportar”, al cual se le ha dado la posibilidad de lectura externa. Dentro del mismo se han creado otros directorios para:

- **exportar/**: aquí se guarda la hoja de estilos.
- **exportar/images/**: Las imágenes que el sistema pueda utilizar.
- **exportar/js/**: las librerias javascript básicas y frameworks utilizados.
- **exportar/jsapp/**: Las aplicaciones javascript creadas.
- **exportar/ayudas/**: Diversas páginas html con contenido que sirve de ayuda.

6.1.1 Frameworks javascript utilizados

Para facilitar la tarea de la programación, la cual es un poco ardua en javascript, se ha utilizado algunos frameworks de javascript que lo facilitan. También se han utilizado frameworks que facilitan efectos gráficos para utilizar algunas de sus funcionalidades que permiten agilizar la aplicación y darle mayor vistosidad.

6.1.1.1 Prototype

Este framework ayuda a la hora de programar mediante funciones, clases y macros implementadas en el mismo.

De todas las funciones que implementa la librería se van a destacar las que han sido utilizadas en el desarrollo de las aplicaciones cliente.

La versión utilizada en el desarrollo del proyecto ha sido la 1.5.0. Se encuentra en el servidor en la ubicación exportar/js/prototype.js.

6.1.1.1.1 Uso de macros

Prototype implementa un conjunto de macros muy cómodas:

- **$('id')** => Esto implementa el document.getElementById('id') obteniendo el objeto.
- **$A(array)** => Genera un objeto array al que le podemos aplicar los métodos implementados en la librería.
- **$F(idVariableFormulario)** => Devuelve el valor de la variable del formulario identificada con un id.
- **$H()** => Permite el uso de hash en javascript.
- **$R(inicio,fin)** => Permite crear un rango numérico

6.1.1.1.1 Objeto Ajax

Prototype ha implementado un objeto Ajax que sirve de manejador de las peticiones asíncronas que se realicen. Este objeto nos ofrece tres métodos de comunicaciones ajax que se identifican con tres subobjetos a su vez:

- **Ajax.Request**; Inicia y maneja todos los pasos de una comunicación Ajax.
- **Ajax.Updater**: Actualiza un contenedor identificado con un id con los datos resultado de una petición Ajax. Es una especialización de la petición Request.
- **Ajax.PeriodicalUpdater:** Igual que el anterior pero con posibilidad de hacerlo de forma periódica.
- **Ajax.Responders:** Permite registrar eventos comunes que se lanzarán cuando se produzca un determinado evento para cualquier petición.

El objeto básico de una comunicación Ajax es el Ajax.Request, el cual permite el control de todo el ciclo de vida de la petición al servidor. Es el que se ha utilizado en las aplicaciones desarrolladas.

El ciclo de vida de la petición XHR pasa por los siguientes estados:

1. Creado
2. Inicializado
3. Petición enviada
4. La petición empieza a recibirse
5. Petición completada.

Para controlar cada uno de los puntos del ciclo se han mapeado unas funciones de callback que van a permitir desarrollar los comportamientos que se deseen para cada caso y que son ejecutadas en el siguiente orden:

1. **onCreate**: Reservada para los responders globales de Ajax
2. **onUninitialized**: Punto 1 del ciclo de vida
3. **onLoading**: Punto 2 del ciclo de vida
4. **onLoaded**: Punto 3 del ciclo de vida
5. **onInteractive**: Punto 4 del ciclo de vida
6. **onXYZ**, **onSuccess** ó **onFailure**: Para un código concreto resultante de la petición o para el código de éxito o para el código de error.
7. **OnComplete**: Punto 5 del ciclo de vida

Para el objeto Ajax.Request también han implementado diversas propiedades que nos

permiten configurar del todo una petición:

- **asynchronous** (true | false) - Realizar la petición en modo síncrono o asíncrono, por defecto *true*
- **contentType** (string) - Tipo mime de la petición, por defecto *'application/x-www-form-urlencoded'*
- **encoding** (string) - Codificación de caracteres de la petición, por defecto *'UTF-8'*
- **method** (string) - Método de la petición (*'GET'* o *'POST'*)
- **parameters** (string u objeto) - Parámetros como string tipo *'?num=1&page=0'* o tipo *{ 'num':1, 'page':0 }*
- **postBody** (string) - Cuerpo de la petición en caso de usar **method** *'POST'* .
- **requestHeaders** (array u objeto) - Parámetros HTTP adicionales de la petición como array u objeto tipo *{ 'Accept':'text/javascript' }*
- Eventos sobre la petición: **onComplete**, **onException**, **onFailure**, **onInteractive**, **onLoaded**, **onLoading**, **onSuccess**, **onUninitialized** y **on**NNN (donde NNN es un HTTP Status Code).

Un ejemplo de la utilización de un objeto Ajax.Request puede ser la siguiente:

```
var url="index.php";
var param="app=contactosajax&act=sacaTodos&json=1";
var peticion = new Ajax.Request( url, {
method: 'get',
parameters: param,
asynchronous: true,
onComplete: cargaContactosAux,
onLoading: cargando,
onInteractive: cargando,
onFailure: falloXHR });
```

6.1.1.1.1 Uso de otros objetos

Existen otros objetos muy útiles que se van a enumerar por encima destacando los métodos más utilizados:

- **Objeto Array**: Dispone de métodos de búsqueda de elementos, ordenación, eliminación de elementos, y de posición. Un método muy interesante es el **Array.each** el cual permite ejecutar una función sobre cada elemento del array
- **Objeto Hash**: Tiene funciones similares al objeto array y algunas nuevas asociadas a los valores y claves del objeto.
- **Objeto PeriodicalExecuter**: Permite crear un objeto que ejecuta una función periódicamente.

- **Objeto String**: Añade nuevas funciones a las originales de javascript. Entre ellas las funciones propias de reemplazo, búsqueda de cadenas, etc.

Para ver un listado más detallado y extenso de los objetos se recomienda la lectura de la documentación de programación de prototype.

6.1.1.1 Script.aculo.us

Esta librería esta basada en prototype y aporta mejoras para enriquecer el interfaz web. Da un motor de efectos visuales, una librería de drag & drop la cual incluye listas ordenadas, controles de autocompletado, edición en caliente, sliders y más. El autor original de la librería es **Thomas Fuchs**, aunque actualmente recibe contribuciones de numerosos programadores, ya que la librería se distribuye de forma completamente gratuita y dispone de una buena documentación.

La librería está dividida en varios módulos:

- **Efectos**: permite añadir de forma muy sencilla efectos especiales a cualquier elemento de la página. La librería incluye una serie de efectos básicos y otros efectos complejos construidos con la combinación de esos efectos básicos. Entre los efectos prediseñados se encuentran el parpadeo, movimiento rápido, aparecer / desaparecer, aumentar / disminuir de tamaño, desplegarse, etc.
- **Controles**: define varios *controles* que se pueden añadir directamente a cualquier aplicación web. Los tres controles que forman este módulo son: "arrastrar y soltar", que permite definir los elementos que se pueden arrastrar y las zonas en las que se pueden soltar elementos; "autocompletar", que permite definir un cuadro de texto en el que los valores que se escriben se autocompletan con ayuda del servidor; editor de contenidos, que permite modificar los contenidos de cualquier página web añadiendo un sencillo editor AJAX en cada elemento.
- **Utilidades**: la utilidad principal que incluye se llama `builder`, que se utiliza para crear fácilmente nodos y fragmentos complejos de DOM.
  Se encuentra ubicada en exportar/js/scriptaculous.js.

6.1.1.1 Aim

AIM son las siglas de AJAX IFRAME METHOD. Esta librería ayuda en la subida de ficheros al servidor de forma asíncrona. La técnica utilizada en este objeto es la de la creación de un iframe oculto que va a servir de target del formulario a ejecutar. Los pasos son los

siguientes:

- Se crea un formulario del tipo multipart/form-data
- A la función submit del formulario se le indica que es un objeto AIM
- El objeto AIM se encargará de realizar el submit en un iframe oculto y procesar la respuesta del servidor con la función indicada.

Se encuentra ubicada en exportar/js/aim.js

6.1.1.1 Tooltip

Esta librería esta desarrollada por Jonathan Weiss y también esta basada en prototype. Da la funcionalidad de poder utilizar ventanas emergentes cuando se pasa el cursor del ratón por alguna zona. Ayuda a la hora de ahorrar espacio en la interfaz y mostrar datos solo cuando se pase por el área indicada.

Se encuentra ubicada en exportar/js/lightbox/tooltips/tooltip.js

6.1.1.2 Básicas

Durante la programación de la aplicación han surgido mucho código reutilizable que se ha agrupado a modo de funciones en una librería de básicas.

Entre las funciones que se pueden utilizar existen las siguientes:

- Funciones para la modificación de tablas mediante DOM
- Funciones de conversiones de datos
- Funciones de manejo de tiempo
- Funciones para el manejo de notificaciones provenientes del servidor.
- Funciones de ordenación y comparación de selects.
- Funciones de popUp

Las funciones introducidas han sido las que se han ido necesitando. Puede utilizarse este repositorio como centro para introducir nuevas funciones.

6.1.1 Estructura de cada una de las partes

En este apartado vamos a comentar como se han constituido las diferentes partes del interfaz de cliente de la aplicación CITAS2.0. Se va a tratar de mostrar de forma escueta el funcionamiento de la aplicación si bajar demasiado al nivel de las funciones de cada una de ellas, ya que es objeto de este texto dar un conocimiento tan exhaustivo del código de la aplicación. En cualquier caso se puede recurrir al código fuente de cada una de las partes para complementar más en profundidad el conocimiento de las mismas.

6.1.1.1 Contactos

Esta aplicación es la que se encarga de dar la funcionalidad a la gestión de contactos. En la tabla siguiente se muestra un resumen de los ficheros utilizados.

URL	Index.php?app=contactos
Aplicación	contactos
Plantilla bajada	cabecera.tpl, cuerpo_contactos.tpl, pies.tpl
Aplicación Javascript	exportar/jsapp/contactos.js

Tabla 53: Resumen de la aplicación cliente de contactos

En esta aplicación se ha definido una plantilla donde cada aplicación se separa en capas controladas por eventos por el programa. El control de las capas se basa en la ocultación o activación de cada una de ellas bajo demanda de cliente por eventos. Es útil ya que de golpe se baja todo el esqueleto de la página y se puede actualizar partes ocultas. Por contra se da demasiado control al javascript, no hay demasiadas recargas del servidor para actualizar las aplicaciones y esqueleto, no se dispone de la posibilidad de dar a la página atrás ya que solo se ha accedido a una url, etc,

Las capas se han definido acorde con la funcionalidad del interfaz cliente. De tal modo que existen las siguientes:

Identificador	Capa
contactosImportarCapa	Capa donde aparece el formulario para la importación / exportación de los contactos en el sistema
contactosEditarCapa	Capa donde aparece el esqueleto de la edición / creación de un contacto en el sistema
contactosListarCapa	Capa donde se muestra el listado de los contactos que pertenecen al usuario.

Tabla 54: Listado de capas de la aplicación cliente de contactos

El **flujo de ejecución de la aplicación,** sin interacción de ningún evento, puede resumirse en los siguientes pasos:

1. El cliente accede a la URL
2. El servidor procesa las plantillas asociadas a la aplicación
3. El cliente procesa la plantilla y ejecuta el código javascript que se encuentra en las mismas. En este caso es la ejecución del código contactos.js
4. Se realiza la petición de todos los contactos al servidor mediante una petición asíncrona al método index.php?app=contactosajax&act=sacaTodos&json=1

5. Cuando se obtengan todos los contactos se crea la variable de contactos.
6. Cuando se produce el evento de carga de la página se activa la capa por defecto y se inicializan eventos de diferentes ids de la página
7. Como es el inicio del programa nuestra capa inicial es la de listado de contactos por lo cual se activa esta capa.
8. Se muestran todos los contactos obtenidos de la petición asíncrona del servidor en una tabla formateada por página y con el añadido de las acciones que se pueden realizar sobre cada elemento de la misma.
9. Se queda a la espera de que se produzca cualquier evento del cliente.

Una vez que el sistema se ha ejecutado por primera vez, y en ausencia de eventos periódicos que ejecuten funciones, ya sólo quedan los eventos que el usuario introduzca por la iteración con la aplicación.

Muchos de los eventos configurados no desencadenan en una petición asíncrona al servidor. Los eventos que si desencadenan estas peticiones hacia el servidor son los siguientes:

Identificador	URL
Recargar Contactos	- URL: index.php? app=contactosajax - Parametros: act=sacaTodos&json=1
EditarUpdate	- URL: index.php? app=contactosajax - Parametros: act=actualiza, json=0, cnombre, capellidos, id_persona, tparticular, tmovil, cemail
NuevoUpdate	- URL: index.php? app=contactosajax - Parametros: act=nuevo, json=0, cnombre, capellidos, id_persona, tparticular, tmovil, cemail
Borrar	- URL: index.php? app=contactosajax - Parametros: act=borrar, json=0, id_persona

Tabla 55: Resumen de peticiones asíncronas en la aplicación cliente de contactos

Otro de las cosas a observar de la usabilidad de esta aplicación es el interfaz de búsqueda de contactos en la base de datos. Al tener todos los contactos bajados del servidor en una estructura, es fácil implementar una búsqueda lanzada cada vez que el usuario va tecleando el patrón que quiere obtener. La tabla de listado se va actualizando conforme el usuario teclea lo que busca. Esta tabla a su vez es ordenable por varios de los campos.

6.1.1.1 Invitaciones

Esta aplicación se ha divido en varias plantillas dependiendo de la acción que se solicite al servidor. La finalidad de esto es la de evitar las capas, lo cual va a permitir darle a la aplicación un histórico de páginas visitadas.

Por lo tanto dependiendo de cada acción solicitada se van a acceder a distintos programas javascript como es puede ver en la tabla siguiente.

Acción	Programa javascript
listadoActivas	exportar/jsapp/invitacionesListar.js
listadoHistorico	exportar/jsapp/invitacionesListar.js
crearInvitacion	exportar/jsapp/calendario.js exportar/jsapp/invitacionesNuevo.js
modifDatos	exportar/jsapp/usuario.js
mensajes	exportar/jsapp/mensajes.js
votaciones	exportar/jsapp/votaciones.js
forzar	exportar/jsapp/forzar.js
desforzar	exportar/jsapp/desforzar.js
anular	exportar/jsapp/anular.js

Tabla 56: Listado de scripts para la aplicación cliente de invitaciones

La URL para acceder a cada una de las acciones es “index.php?app=invitaciones&act=ACCION”.

En los siguiente apartados se va a explicar cada una de las aplicaciones y sus relaciones asíncronas con el servidor.

6.1.1.1.1 calendario.js

Esta librería se ha realizado para brindar la posibilidad de crear un calendario en la parte de la página que indiquemos. Se apoya en prototype y muestra en resaltado tanto el día actual como los domingos.

Su uso es muy simple, basta con incluirla al principio del script y llamar a la función calendarioCrea con la fecha que se desea marcar como actual. Un ejemplo sería el siguiente;

```
calendarioCrea('calendarioId',calendarioDameFechaActual());
```

Si se quiere introducir otra fecha basta con meterla en el formato DD-MM-YYYY.

El calendario se crea con los siguientes clases de estilos

- **calendario**: la clase de la tabla y formato por defecto para cada celda.
- **calendarioNoMes**: para los días que no están en el mes actual.
- **calendarioDiaActual**: para el día actual.

Cuando se crea la celda de cada día se le añade el identificador en el formato div_IF_YYYY-MM-DD. Así para el ejemplo anterior las celdas tendrían el id siguiente en el formato "calendarioId_IF_YYYY-MM-DD". Esto se hace con la intención de luego modificar la propiedad onclick de cada una de las celdas, para que cuando el usuario pulse la fecha que desee, se procese este click con una función que captura la fecha y hace lo programado con ella. La función a la que apuntan los onclick es la "submitInvitacionesCrear" que es parte del código de invitacionesCrear.js.

Esta librería solo implementa el calendario, no realiza ninguna petición asíncrona al servidor.

6.1.1.1.1 invitacionesListar.js

Mediante esta aplicación se va a controlar la aplicación del listado de invitaciones del usuario. Existen dos listados de invitaciones si se atiende a su clasificación administrativa, uno el de las **invitaciones activas** y otro el del **histórico de las invitaciones**. Para reutilizar el código javascript, ya que es lo mismo para ambos listados, se apoya en una variable que se escribe en la plantilla y que tiene distinto valor para cada tipo de listado. La variable se llama **tipo** y puede valer **activas** o **histórico**.

Luego, en cada uno de los tipos, tenemos una segunda clasificación en la que se muestran por un lado las invitaciones que ha creado el usuario y por otro las invitaciones a las que ha sido añadido.

El flujo del programa sería el siguiente:

1. Cuando la página se ha cargado hacemos peticiones asíncronas al servidor para bajarnos la lista de invitaciones creadas por el usuario y del tipo indicado y la lista de invitaciones a las que el usuario ha sido añadido del tipo indicado.
2. Se utiliza el objeto PeriodicalExecuter para ejecutar cada una de las cargas anteriores periódicamente con la finalidad de darle viveza al listado ofrecido y mostrar los cambios más recientes.
3. Una vez que se realiza con éxito las peticiones anteriores se actualizan las variables de listado del programa y se llaman a unas funciones que crean la tabla con el listado de invitaciones pasado como parámetro. Para cada una de las invitaciones les añaden unas acciones particulares que nos permiten tener control sobre ellas.

Ya una vez que estamos en esta situación de estabilidad, los únicos eventos que la modifican son, los generados por el usuario o los que se producen con cada una de las peticiones periódicas del cliente.

Los únicos eventos javascript que realizan peticiones asíncronas en esta URL son:

Acción	URL
cargaInvitaciones	- URL: index.php? app= invajax - Parametros: act=mostrarResumenMios & json = 1 ó act = mostrarResumenMiosHistorico & json=1
cargaInvitacionesOtros	- URL: index.php? app= invajax - Parametros: act=mostrarResumenOtros & json = 1 ó act = mostrarResumenOtrosHistorico & json=1
Borrar una cita	- URL: index.php? app= invajax - Parametros: act=borrarCita, id_cita, activa, json=1
Caducar/Activar una cita	- URL: index.php? app= invajax - Parametros: act=caducaCita, id_cita, activa, json=1, invitado 0 ó 1

Tabla 57: Acciones que realizan peticiones asíncronas invitacionesListar

6.1.1.1.1 invitacionesNuevo.js

Mediante esta aplicación vamos a controlar toda la interfaz de creación de una cita o de edición de una creada.

La creación / edición de la cita esta dividida en los siguientes bloques de información:

- **Datos de la invitación**: En este bloque se indican todos los datos administrativos de la cita tales como el asunto, los comentarios, como se quiere que se haga la votación, la localización, si se quiere recordatorios, como se quiere que se hagan las notificaciones, etc.
- **Selección de contactos**: Mediante un select o un bloque de texto se van a elegir los contactos que se quieren añadir a la cita. El select contiene los contactos que el usuario tiene almacenados y se brinda la posibilidad de introducir nuevos mediante un input text.
- **Introducción de las fechas**: Mediante el calendario se van a ir seleccionando las fechas propuestas y añadiendo las horas a las mismas
- **Notas**: En esta parte se van a ver todas las notas que se han introducido en la cita.

El flujo del programa es el siguiente:

1. Se solicita asíncronamente la descarga de los datos de los contactos del usuario.
2. Se solicita asíncronamente la descarga de los datos de la cita si estamos en el caso de edición
3. Se crea el calendario en la div adecuada
4. Cuando se recibe la respuesta del servidor con los contactos se actualizan las variables adecuadas y el select de la parte de “Selección de contactos”
5. Si se está en el modo edición, cuando se reciben los datos de la cita se actualizan los datos pertinentes de todas las áreas.
6. Se queda en espera de eventos del usuario.

El usuario interaccionará con la aplicación introduciendo nuevas fechas, horas , contactos, modificando valores, etc.

Acción	URL
Carga de contactos	- URL: index.php? app=contactosajax - Parametros:act=sacaTodos&json=1
Carga de datos de la cita	- URL: index.php? app= invajax - Parametros: act=mostrarDatosInvitacion, id_cita=CITA, json=1
Crear nuevo contacto	- URL: index.php? app=contactosajax - Parametros: act=actualiza, json=1, cnombre, capellidos, id_persona, tparticular, tmovil, cemail
Borrar mensaje	- URL: index.php? app= invajax - Parametros: act=borrarMensaje, id_cita = CITA, id_nota = NOTA, json=1
Crear invitación	- URL: index.php? app= invajax - Parametros: act=crearInvitacion, json = 1, contactos, fechas, fduracion

Tabla 58: Acciones que realizan peticiones asíncronas invitacionesNuevo

Una vez que el usuario da al submit del formulario se realiza un chequeo de los datos, se comprueba que los datos son los necesarios para las votaciones y se envía la petición al servidor para introducir o editar la invitación pertinente.

6.1.1.1.1 usuario.js

En la parte de invitaciones también se le da la opción al usuario de cambiar sus datos

personales. Se ha elegido ponerlo aquí ya que estos datos afectan claramente a las notificaciones que el sistema envía.

Los datos del usuario se han dividido en tres partes claras:

- **Datos personales**: En este apartado se pueden modificar el nombre y apellidos, el título que nos ponemos y la firma que queremos poner.
- **Datos de aplicación**: Cambiar el login y la password
- **Opciones de las citas**: Si se quiere aparecer como remitente o si se quiere incluir otros correos en copia de las notificaciones.

En este caso el flujo de programa es muy sencillo ya que solo carga los datos personales asíncronamente del servidor y los actualiza cuando el usuario da un submit.

Acción	URL
Carga de datos personales	- URL: index.php? app=usuajax - Parametros:act=sacaTodos&json=1
Actualización de la parte Personales	- URL: index.php? app= usuajax - Parametros: act=mostrarDatosInvitacion, Nombre, Apellidos, titulo, firma,json=1
Actualización de la parte Aplicación	- URL: index.php? app= usuajax - Parametros: act = cambiaPassword, login, password1, password2, json = 1
Actualización de la parte de invitaciones	- URL: index.php? app= usuajax - Parametros: act=modificaOpciones, correorem, copiacorreos, json=1

Tabla 59: Acciones que realizan peticiones asíncronas usuario

6.1.1.1.1 mensajes.js

Este script va a permitir controla el envío de mensajes a usuarios y las anotaciones que se quieran realizar en la cita.

Se pueden enviar los mensajes a :

- Los contactos que se seleccionan con el select.
- A todos los contactos.
- A los que no han votado todavía

Como en casos anteriores el flujo del programa es muy sencillo:

1. Se bajan todos los contactos

2. Se bajan los datos de la cita para obtener los mensajes de la misma

   El resto del programa se queda esperando a los eventos que produce el usuario.

Acción	URL
Carga de contactos	- URL: index.php? app=contactosajax - Parametros:act=sacaTodos&json=1
Obtención de los datos de la invitación	- URL: index.php? app= invajax - Parametros: act = mostrarDatosInvitacion, id_cita=CITA, json=1
Envío del mensaje	- URL: index.php? app= invajax - Parametros: act = enviaMensaje, json=1, id_cita = CITA , contactos, publico, mensaje

Tabla 60: Acciones que realizan peticiones asíncronas mensajes

6.1.1.1.1 forzar.js , desforzar.js y anular.js

Estos scripts van a dar la funcionalidad de forzar o desforzar la elección de una votación o anular una cita. Se meten todos en el mismo apartado por que realizan la misma funcionalidad que es la de hacer submit del formulario.

Simplemente espera a que el usuario confirme la acción. Las peticiones asíncronas que se realizan en estos scripts se realizan cuando el usuario realiza la acción dando al submit de la misma.

Acción	URL
Forzar elección	- URL: index.php? app=invajax - Parametros:act=forzarFecha, json=1, id_cita, fechap, envia, comentarios
Desforzar elección	- URL: index.php? app=invajax - Parametros:act=desforzarFecha, json=1, id_cita, envia, comentarios
Anular cita	- URL: index.php? app=anularCita - Parametros:act=anularCita, json=1, id_cita, envia, comentarios

Tabla 61: Acciones que realizan peticiones asíncronas forzar, desforzar y anular

6.1.1.1.2 votaciones.js

Mediante este script se realiza el control de las votaciones dentro del modo edición de

una invitación.

El menú de votaciones se divide en dos partes:

- **Datos de la invitación**: Aquí aparecen los datos administrativos de la invitación
- **Estado de la votación**: Se muestra una matriz con las votaciones de todos los invitados.

La matriz de votaciones es una tabla donde aparece en las filas los invitados y en las columnas las fechas propuestas. El cruce de ambas será la votación elegida por el contacto con posibilidad de ser modificada por el administrador.

El flujo del programa es muy sencillo:

1. Se solicitan los datos de los contactos
2. Se solicitan los datos de las votaciones
3. Cuando se reciben los datos de las votaciones se genera la matriz de votaciones en donde se calculan cuales son las fechas propuestas candidatas y se marcan las columnas de distinto color. También se muestra la columna que esta forzada.

Como en el resto de aplicaciones se realizan peticiones asíncronas que se pueden ver en la siguiente tabla:

Acción	URL
Carga de contactos	- URL: index.php? app=contactosajax - Parametros:act=sacaTodos&json=1
Obtención de los datos de la invitación	- URL: index.php? app= invajax - Parametros: act = mostrarDatosInvitacion, id_cita=CITA, json=1
Actualiza votaciones	- URL: index.php? app= invajax - Parametros: act = actualizaVotaciones, id_cita=CITA, json=1, datos del formulario de votaciones

Tabla 62: Acciones que realizan peticiones asíncronas de la votaciones

6.1.1.1 Votaciones de cliente

También en la interfaz de votación de los invitados se le ha dado más funcionalidad con el uso de javascript.

El flujo del programa es igual que el caso de la votación en la edición de cita, solo que en este caso la información que van a tener las funciones esta condicionada a las opciones que el administrador ha marcado en la invitación.

No se realizan peticiones asíncronas al servidor porque la información necesaria se escribe en el cuerpo de la página.

6.2 Parte accesible

6.2.1 Introducción

Uno de los objetivos del proyecto era la de contemplar la accesibilidad en el sistema. La interfaz de cliente se ha realizado principalmente siguiendo pautas de web 2.0 que no son del todo accesibles para todos los posibles dispositivos. Para ello nace la necesidad de hacer que el servidor interprete el origen de la petición y sepa que tipo de aplicación cliente tiene que servir.

Para saber si el cliente es accesible o no usa la variable UserAgent de la petición. Si se encuentra en alguno de los siguientes considera que la petición se realiza desde un dispositivo accesible:

> Windows CE, WebTV, AvantGo, Blazer, PalmOS, lynx, Go.Web, Elaine, ProxiNet, ChaiFarer, Digital Paths, UP.Browser, Mazingo, Mobile, T68, Syncalot, NetFront, Danger, Symbian, Nokia, Xiino, AU-MIC, EPOC, BlackBerry, Wireless, Handheld

Para ampliar el número de dispositivos accesibles basta con modificar el fichero de configuración.

No hay un código de cliente para esta parte. El cliente simplemente interpreta las salidas XHTML compuestas por la aplicación servidora.

Capítulo 7. Ejemplos prácticos de CITAS2.0

7.1 Introducción

En este capítulo se va a mostrar la aplicación gráficamente en cada uno de sus puntos principales. Se mostraran los aspectos más importantes para no tratar de alargar demasiado el capítulo.

El proceso que se va a elegir es ir desde el registro en el sistema hasta, la entrada en el mismo, introducción de unos contactos y la realización de una cita y sus votaciones.

Para ello se van a crear cuatro usuarios de ejemplo citas201@yahoo.es, citas202@yahoo.es, citas203@yahoo.es y citas204@yahoo.es.

7.2 Ejemplo para la parte de interfaz enriquecido

7.2.1 Ventana de inicio

La página de inicio del portal tiene la siguiente forma:

Ilustración 8: Página de inicio versión interfaz enriquecido

donde se pueden ver las partes de la estructura de la información. Se pueden diferenciar dos partes:

- La parte de “Acceso al sistema” desde donde el usuario se autentica para acceder a la aplicación. También en este bloque se muestran enlaces hacia la aplicación de cambio de contraseña y hacia el registro de usuarios.
- La parte de menú horizontal es un menú de acceso directo que lleva de manera rápida a las los grandes bloques de la aplicación

7.2.1 Registro de usuario

Para poder acceder al sistema es necesario un usuario. Para ello se accede al menú de registro el cual tiene el siguiente aspecto.

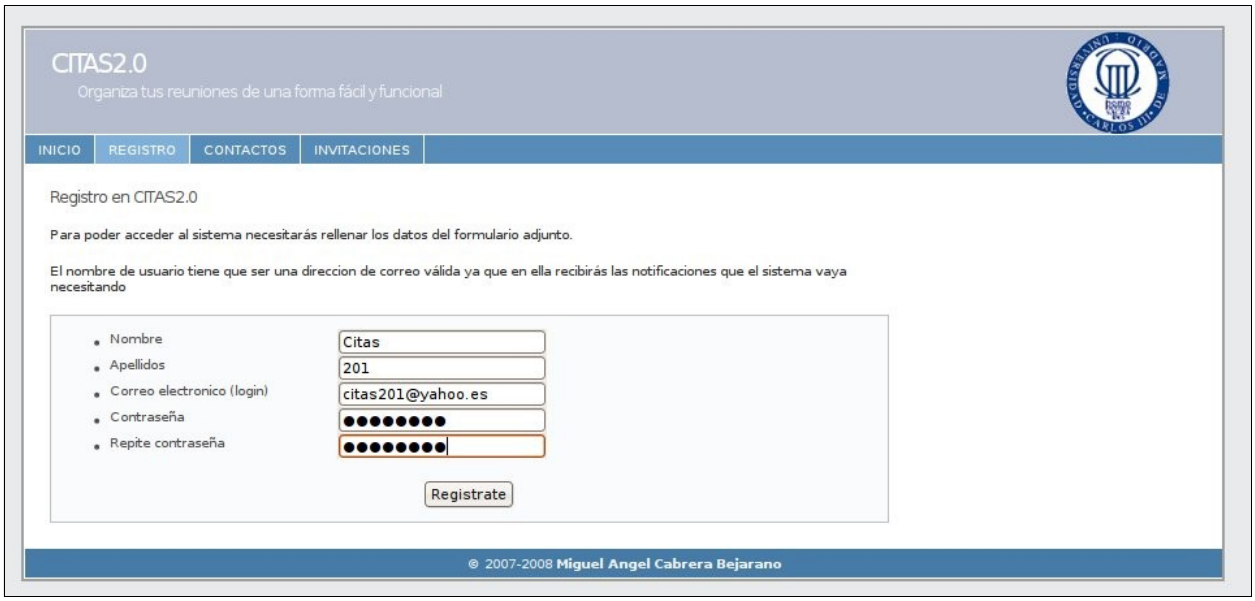

Ilustración 9: Registro de usuario para aplicación interfaz enriquecido

Una vez rellenados los datos solicitados, y si no existe en el sistema un usuario con el mismo login, se dará de alta automáticamente el usuario solicitado. Con este usuario se podrá ir a la parte inicial y acceder mediante el bloque de “Acceso al sistema”.

Si alguna vez se pierde la contraseña, el sistema en el bloque de “Acceso al sistema” facilita un enlace hacia la aplicación de cambio de contraseña. Este sistema envía una nueva contraseña a la cuenta de correo asociada al login.

7.2.2 Aplicación inicial

Si el acceso es correcto se pasará directamente a la aplicación invitaciones en su

listado de invitaciones activas. En los listados de invitaciones se ven dos parte bien diferenciadas:

- Las invitaciones que el usuario ha creado
- Las invitaciones a las que el usuario ha sido añadido como invitado.

Ilustración 10: Listado de invitaciones activas versión interfaz enriquecido

Si se pasa el ratón sobre los enlaces donde pone “Leyenda de códigos” y “Leyenda de colores” se muestra el significado de cada uno de los colores o códigos que se van a mostrar en el listado.

También se puede ver como hay enlaces de ayuda en cada una de las partes. Esto va a acompañar al usuario durante toda la aplicación y va a permitir acceder a la ayuda online de la aplicación.

Aparece otro submenú de la parte de invitaciones en donde se puede ir al listado de invitaciones activas, el histórico de invitaciones, modificar los datos personales o acceder a toda la ayuda online. Este submenú también va a acompañar durante toda la aplicación de invitaciones al usuario.

Como mejora con respecto a otras aplicaciones se da la opción de exportar en formato RSS o iCAL la lista de invitaciones deseada. Se facilitan los enlaces a esta información para que se puedan añadir en cualquier aplicación externa compatible.

7.2.1 Aplicación de contactos

Antes de empezar con las invitaciones se va a introducir algún contacto para poder realizar citas. Para ello hay que meterse en la aplicación de contactos pulsando en el enlace de contactos.

Ilustración 11: Listado de contactos en versión interfaz enriquecido

La aplicación de contactos tiene un submenú para poder acceder a las distintas funcionalidades: Importar/exportar lista de contactos e Introducir nuevo o editar.

Como se puede ver no tenemos ningún contacto introducido.

En las siguientes figuras se puede ver como se introduce un contacto y el listado de

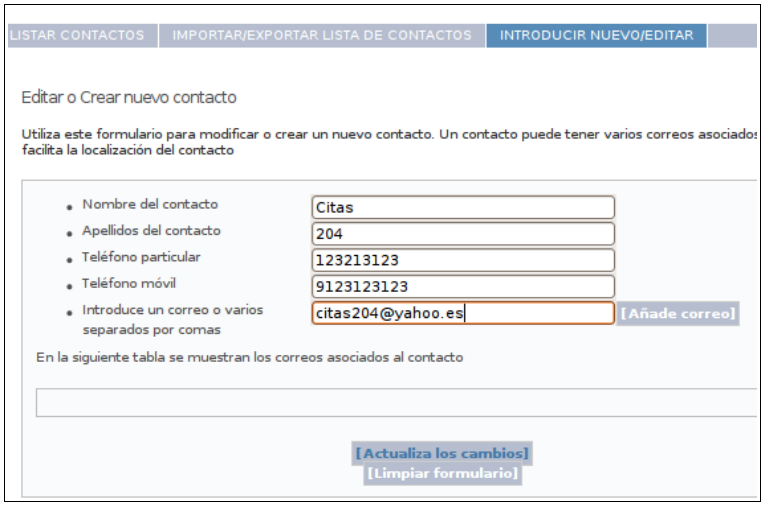

Ilustración 12: Vista para introducir un contacto

contactos introducidos.

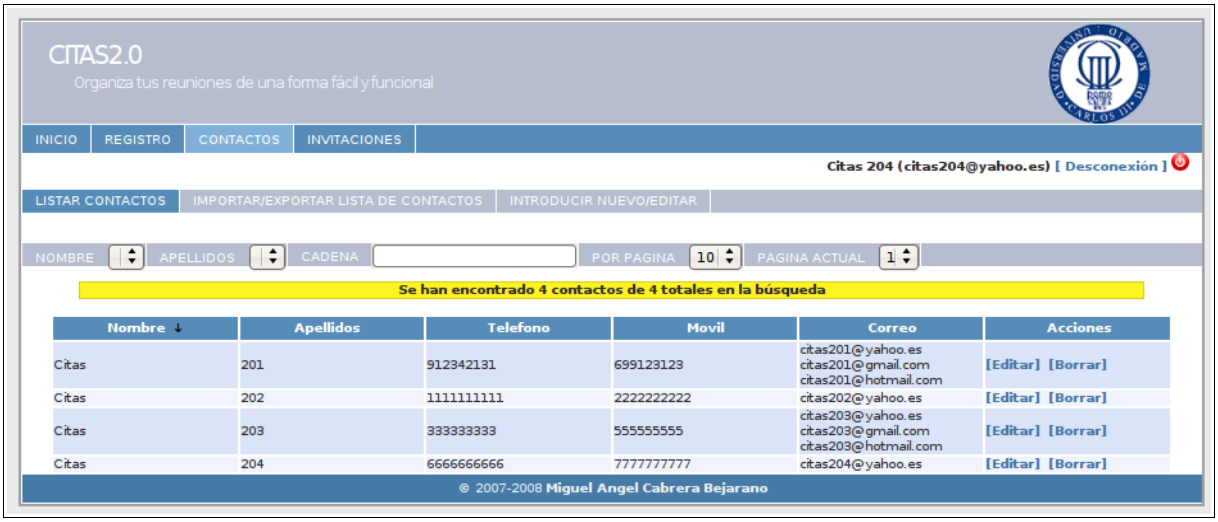

Ilustración 13: Lista de contactos introducidos

En el listado se puede ver los datos de los usuarios, los correos asociados a cada contacto y las acciones definidas para cada usuario. Se podrá editar el contacto modificando sus datos, añadir correos al contacto e incluso borrar el contacto seleccionado.

Otra de las opciones que nos da esta aplicación es la de exportar e importar los contactos.

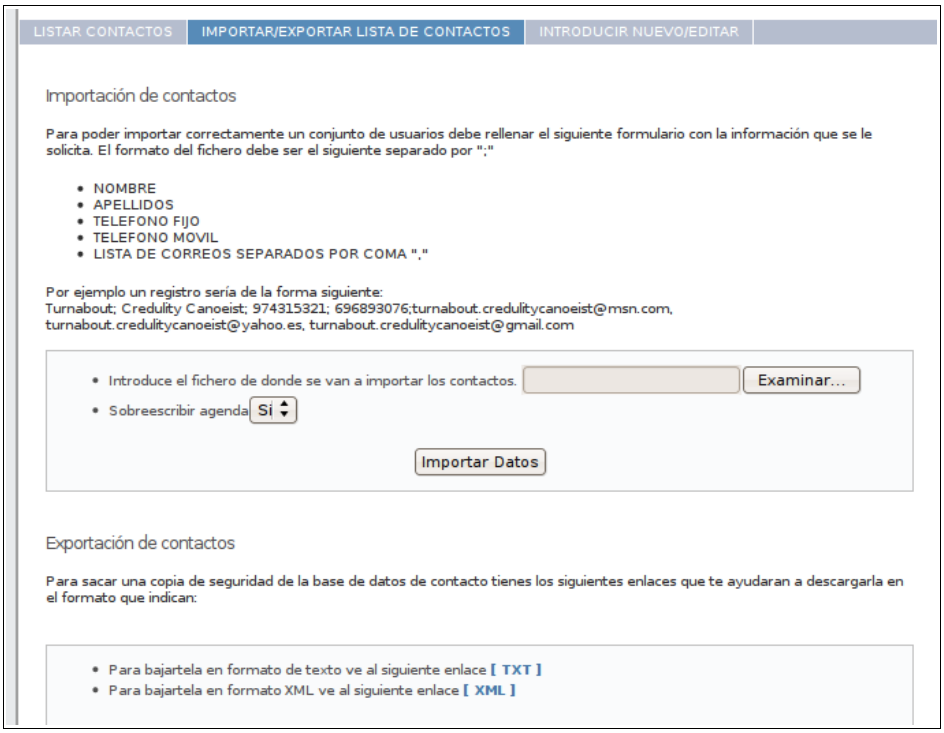

Ilustración 14: Menú de importación y exportación de contactos

El formato del fichero bajado en la exportación o el subido en la importación tiene los siguientes campos separados por puntos y comas;Nombre, Apellidos, Teléfono fijo, Teléfono móvil y Lista de correos separados por comas.

Para el ejemplo de los contactos introducidos previamente se obtiene el siguiente fichero de volcado, el cual vale para importar los mismos:

```
Citas;201;912342131;699123123;citas201@yahoo.es,citas201@gmail.com,citas201
@hotmail.com
Citas;202;1111111111;2222222222;citas202@yahoo.es
Citas;203;333333333;555555555;citas203@yahoo.es,citas203@gmail.com,citas203
@hotmail.com
Citas;204;6666666666;7777777777;citas204@yahoo.es
```

Como aparece en el menú se puede elegir si se quiere sobreescribir o no la información de la base de datos de contactos.

Durante el uso de la aplicación el usuario se encontrará con diversos mensajes procedentes del servidor o del chequeo de datos de la aplicación cliente que avisarán del estado de sus operaciones. También le indicarán está haciendo las cosas correctamente.

7.2.2 Aplicación de invitaciones

Ya una vez que se ha seguido el proceso normal de la ejecución de la aplicación, en el cual se ha introducido primero alguno de los contactos de prueba, el paso siguiente es ver como se crea una cita en el sistema.

Como se esta viendo se recomienda tener primero los contactos introducidos, pero a nivel de aplicación no es necesario. Se pueden introducir simplemente direcciones de correo cuando se crea la invitación. Internamente el sistema creará un contacto temporal asociado a la cita creada.

Durante la aplicación de invitaciones va a acompañar al usuario un submenú con desde el cual se puede acceder a los diferentes modos de la aplicación.

Ilustración 15: Submenú general de la aplicación de invitaciones

Estos modos son:

- **Activas**: Si se quiere acceder al listado de invitaciones activas.

- **Históricas**: Si se quiere acceder al listado de invitaciones históricas.
- **Nueva Invitación**: Si se va a crear una nueva invitación. Este el mismo acceso que en el modo de edición solo que en este caso no se rellena el identificador de cita.
- **Datos personales**, para la edición de los datos personales y de algunas opciones que afectan a la como nos mostramos a los contactos en las notificaciones que envía el sistema por citas.
- **Ayuda Invitaciones**, para acceder a la ayuda online de la aplicación, Se accede al raíz de la misma desde aquí. Luego se verá que desde diferentes puntos de las aplicaciones se puede acceder a la zona de la ayuda de interés en ese punto.

7.2.2.1 Creación o edición de invitaciones

Ya una vez que se ha registrado en el sistema vamos a proceder a la introducción de una invitación. En esta sección se va a comentar cada una de las secciones que se han de superar para crear una invitación.

Mientras que se esté en el entorno de creación o edición de cita va a aparecer otro submenú desde el cual se va a acceder a funciones que actúan sobre la cita en cuestión.

Ilustración 16: Submenú de acciones en el modo edición/creación de invitación

Las acciones que se pueden acometer son:

- **Editar la cita**
- **Votaciones de la cita** si se quiere entrar en modo administrador en la gestión de las votaciones.
- **Añadir comentario ó enviar mensaje** si se quiere enviar mensajes de carácter público, a los contactos, o carácter privado, como anotación, sobre la cita.
- **Anular cita**.

También, y con la finalidad de facilitar el acceso a las distintas partes del documento de edición, se facilitan accesos directos a las partes del mismo como se puede ver en la siguiente figura.

Ilustración 17: Accesos directos del menú edición de invitaciones

7.2.2.1.1 Datos de la invitación

Como se puede ver en la figura se dispone de todas las opciones para caracterizar una

invitación.

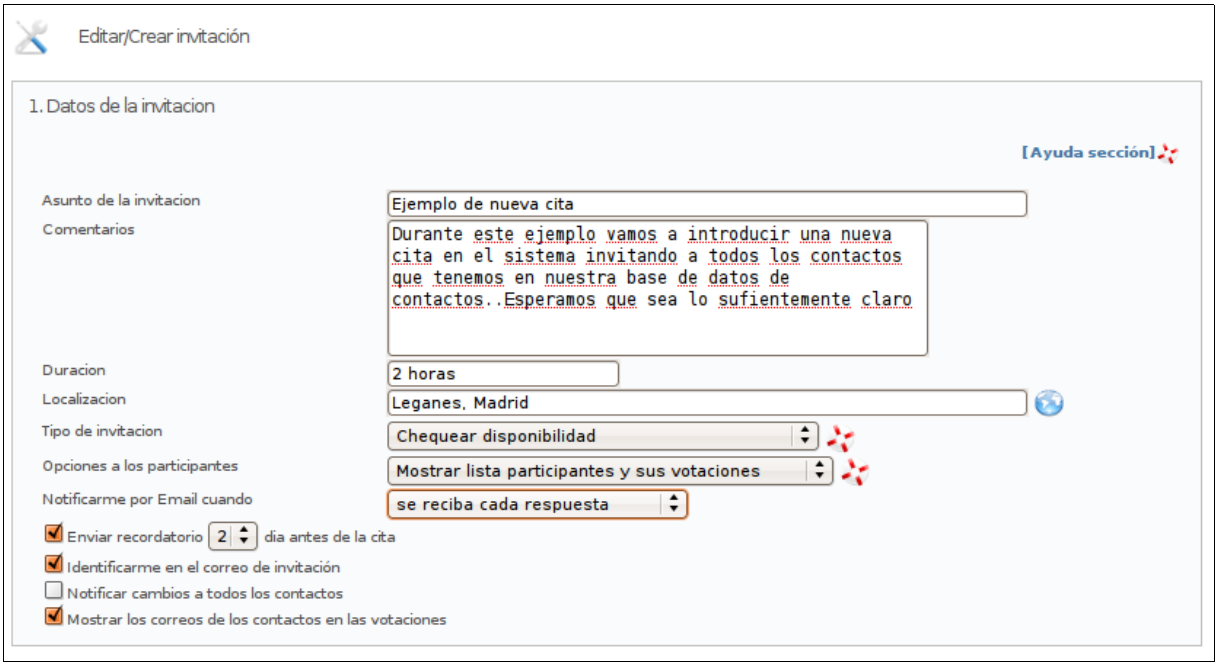

Ilustración 18: Edición/Creación de una invitación: Datos de la invitación

Donde :

- Se puede poner el **asunto** de la invitación
- Una descripción de la misma en los **comentarios**
- Que **duración** estimada va a tener
- En que **lugar** se va a producir. En esta opción, si la cadena esta en formato Google Maps, se podrá acceder, mediante el enlace que aparece seguido, al mapa de la localización en Google Maps.
- **Tipo de invitación** donde se indica que tipo de votación se quiere que los invitados realicen.
- También se podrá definir que van a ver los invitados en su página de votaciones mediante el **opciones a los participantes**.
- Y cómo se quiere que se me **notifiquen** como administrador las votaciones.

También se definen otras **opciones administrativas**:

- Si se quiere que se envíen **recordatorios** desde los días indicados antes de la cita.
- Si se desea aparecer como remitente en las notificaciones en vez de el sistema
- Si se quiere que se notifiquen los cambios que se van a hacer en la invitación ahora. Cuando es una nueva creación siempre se marca para realizar la acción. Cuando se está en modo edición se deja a elección del administrador el querer notificar los cambios realizados.

- Si se quiere que en la pantalla de votaciones se vean los correos asociados al contacto. Esto es útil ya que el administrador puede tener un alias de contacto no reconocible por el resto de invitados. El mostrar el correo puede ayudar a su reconocimiento.

7.2.2.1.1 Selección de contactos

Una vez que se ha rellenado la información de la cita se tiene que añadir los contactos deseados.

Cuando se carga la página se rellena un select, que esta en esta sección, con todos los contactos que se tienen en la base de datos de contactos. Un cambio del select hace que el contacto se seleccione como invitado. También se tiene la posibilidad de añadir el contacto con el textarea que esta debajo del select.

Ilustración 19: Edición/Creación de una invitación: Selección de contactos

Según se van seleccionando los contactos se va construyendo una lista con la selección escogida. Se facilita una acción de quitar por si se quiere quitar alguno de la selección.

7.2.2.1.2 Introduce las fechas del evento

Ya solo queda elegir las fechas propuestas para la votación. Mediante un calendario y las pulsaciones sobre los días elegidos se van proponiendo fechas. Mediante un select multiple que se tiene a la derecha del calendario se va a elegir las horas de las fechas que se deseen. Por defecto aparecerán marcadas las fechas que no tienen hora, es decir, las que aparecen SH.

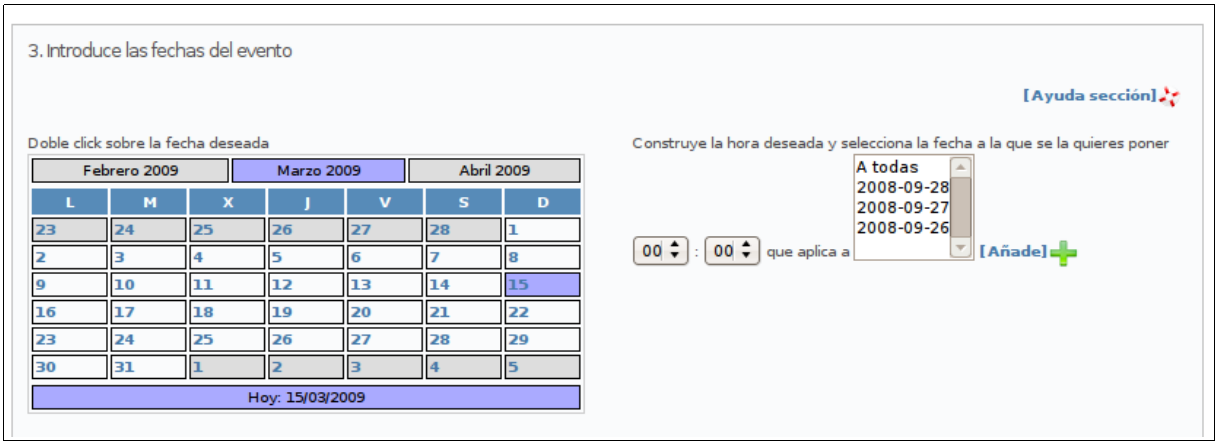

Ilustración 20: Edición/Creación de una invitación: Selección de fechas

Si se quiere poner hora a las fechas basta con seleccionar la hora y las fechas deseadas. También se nos permite el poner diferentes duraciones para cada una de las fechas así como la facilidad de quitar las que no se quieran.

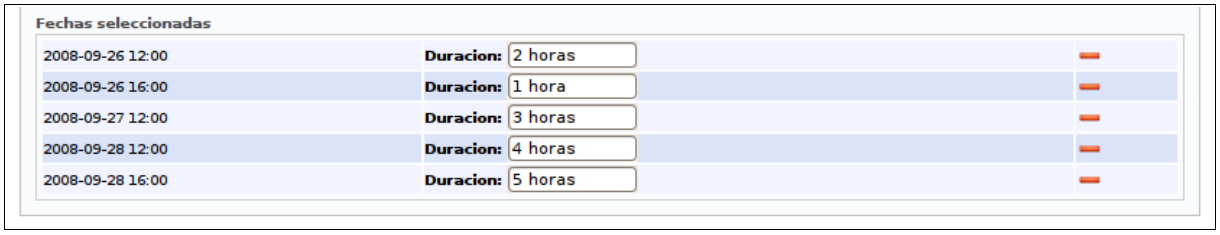

Ilustración 21: Edición/Creación de una invitación: Fechas con hora y duración

Ya en este punto se tiene toda la información definida por lo que solo queda pulsar el botón de abajo de “Actualiza Invitación” para que el evento tenga lugar.

7.2.2.1.3 Notas introducidas

Desde el modo edición se puede ver las notas que alberga la cita. Se ha introducido una para mostrar como se vería, pero más adelante se contará como se introducen en el sistema.

Dependiendo del carácter que tengan aparecerán también en el menú de votaciones.

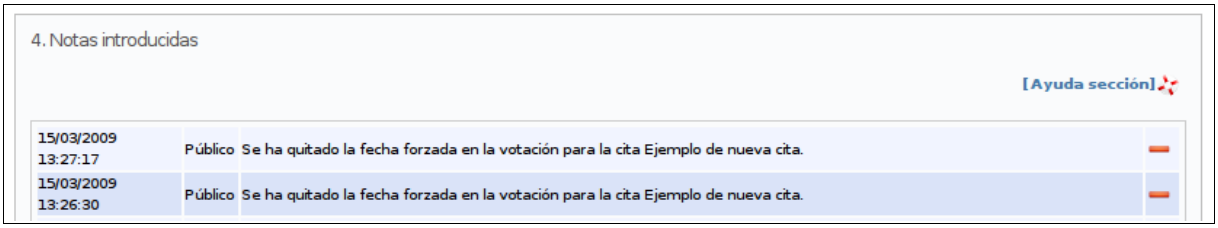

Ilustración 22: Edición/Creación de una invitación: Notas introducidas

7.2.2.2 Listado de invitaciones

Con la invitación creada ya no se tiene el listado vacío como podemos ver en la siguiente figura,

Ilustración 23: Listado de invitaciones activas

Este listado se corresponde con el de invitaciones activas aunque es igual que el de históricas. Solo cambia una de las acciones en las que en un caso hace el paso de activas a históricas y en el otro lo contrario.

Los campos que se pueden ver para la parte de invitaciones creadas son :

- **Actividad**: Se corresponde con el asunto de la invitación
- **Cont.**: Es el número de invitados a la cita.
- **Vot.**: Es el número de votaciones realizadas.
- **Localización**

- **Candidatas**: Se muestran las fechas propuestas que van ganando como candidatas
- **Acciones**: las acciones que podemos realizar sobre la cita

Como se puede observar la invitación tiene un color característico que indican en que estado temporal, de votación y administrativo esta la cita. Para que el usuario sepa que significa cada color se hace uno de un popup que se activa cuando se pasa por el enlace de "[Leyenda de colores]" y como se puede ver en la siguiente figura.

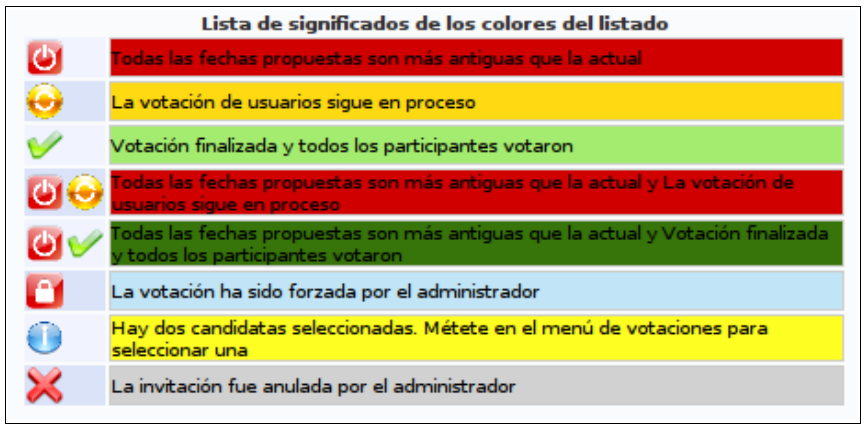

Ilustración 24: Listado de invitaciones: Leyenda de colores

También se hace uso de unos códigos para las acciones y que se pueden descifrar cuando el usuario pasa el ratón por el enlace de "[Leyenda de codigos]".

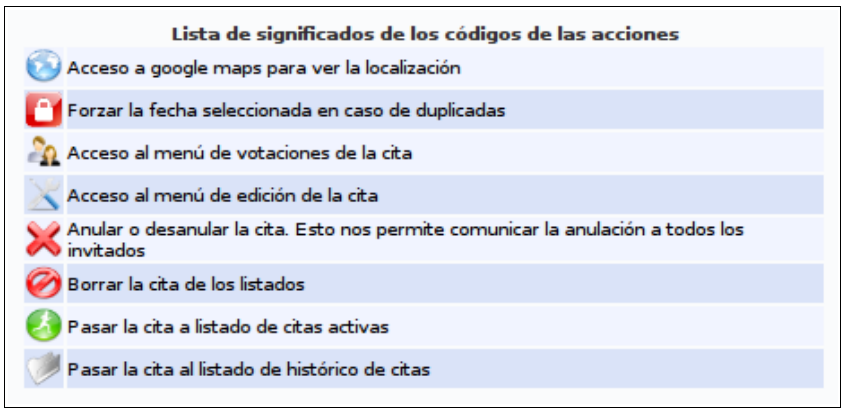

Ilustración 25: Listado de invitaciones: Leyenda de códigos

Todo esto se hace gracias al uso de la librería tooltip, la cual también la se utiliza para lanzar un popup con la información extendida de la cita cuando pasamos el ratón sobre el campo Actividad de la misma. Esta herramienta ayuda a condensar la información.

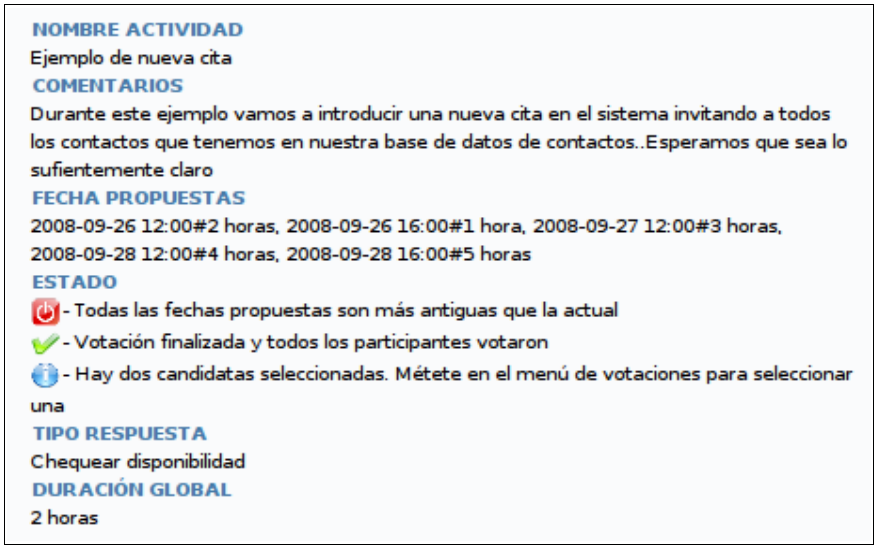

Ilustración 26: Listado de invitaciones: Información extendida

También se tiene visibilidad de las invitaciones a las que el usuario ha sido invitado por el sistema. En ese caso se tienen los mismos datos que en el caso de las que ha creado, pero se ven reducidas las acciones, ya que solamente están la de acceder al interfaz de cliente de las votaciones, y a la de pasarla al histórico.

7.2.2.1 Edición de invitaciones

El menú de edición de una cita es exactamente igual que el apartado de creación de una cita. En este apartado se va a contar el resto de aplicaciones que aparecen dentro del conjunto de acciones del modo edición.

7.2.2.1.1 Anular la cita

Se facilita al usuario una aplicación para poder anular o desanular una cita. Desde esta aplicación podrá añadir el comentario que considere oportuno al mensaje de anulación de la misma.

Ilustración 27: Edición/Creación de una invitación: Anulación

Si el administrador marca la casilla donde se indica que se va a enviar una notificación de anulación, los invitados la recibirán por parte del sistema. También se ofrece la posibilidad de añadir algún comentario más.

La acción de anular quedará registrada como una nota de carácter privado para llevar un log de las acciones que se han realizado sobre la cita y que se podrán ver desde el menú de edición de la misma.

7.2.2.1.2 Borrar la cita

La acción de borrar una cita simplemente es a nivel administrativo. La quita del listado de invitaciones si el administrador responde afirmativamente a un popup de confirmación.

7.2.2.1.3 Añadir comentario o enviar mensaje

Como se ha comentado antes, se posibilita al administrador de una herramienta para poder registrar comentarios en la invitación, o enviar nuevos mensajes a los invitados de la cita.

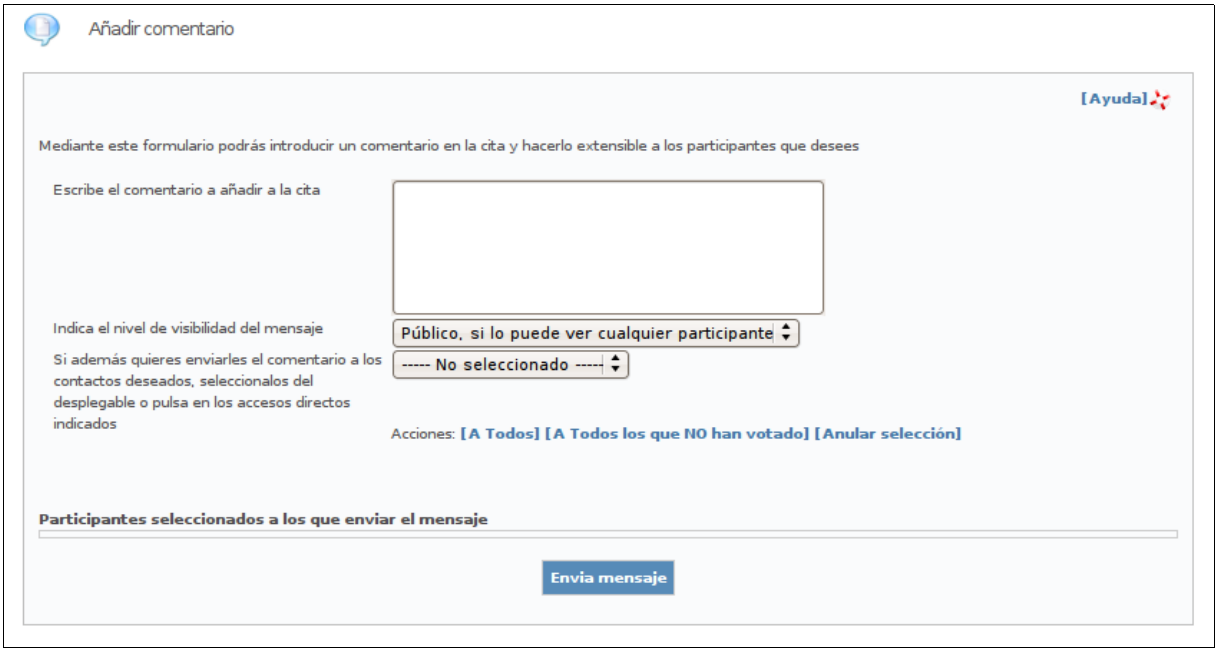

Ilustración 28: Edición/Creación de una invitación:Enviar mensajes

Se podrá seleccionar el carácter del mensaje, donde público es para que lo vea todo el mundo, y privado es una mera anotación en la cita.

Los contactos se irán seleccionando como en el menú de creación, moviéndose con el select sobre el contacto deseado lo cual irá generando un listado con todos los invitados seleccionados. Para facilitar la tarea se han creado dos accesos directos para incluir todos los invitados o solo los que no han votado. Se orienta la aplicación para mandar recordatorios de que tienen que votar a aquellos que aún no lo han hecho.

El mensaje realizar no solo se envía a los contactos seleccionados, sino que también se anotan en la cita.

7.2.2.1.4 Votaciones de esta cita

Con el fin de tener control sobre todas las votaciones de los usuarios facilitando la modificación de ellas, se ha creado un interfaz de administrador de las votaciones.

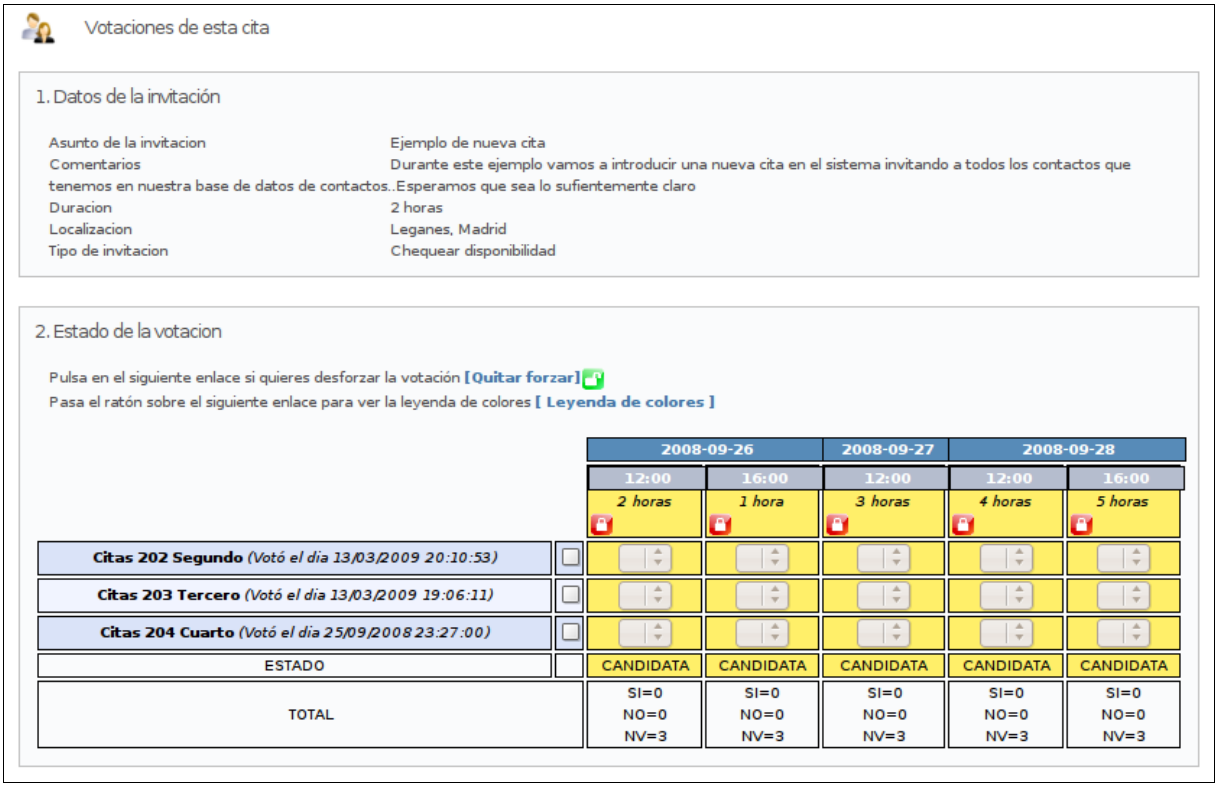

Ilustración 29: Edición/Creación de una invitación:Administración de votaciones

Se puede observar que hay dos bloques bien diferenciados, uno con los datos administrativos de la invitación y que ya se ha desarrollado en apartados anteriores, y otro con el estado de la votación. Es este último el que se va a desarrollar.

Como puede verse, se ha generado una matriz de votaciones con los contactos como filas y las fechas propuestas con columnas. Todas ellas ahora mismo están en estado amarillo o de candidata ya que todas tienen la misma puntuación en las votaciones, ninguna.

Si pasamos el ratón por la casilla de cada contacto se nos muestra un popup con la información de los correos y siempre y cuando tenga definido en la invitación el mostrar los correos de los invitados.

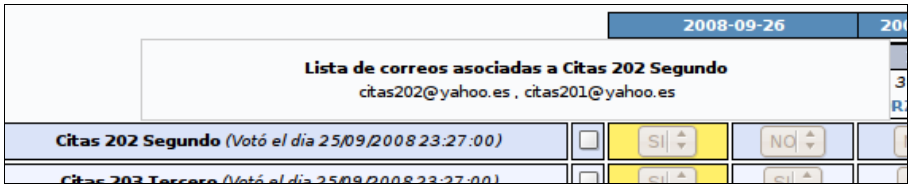

Ilustración 30: Administración de votaciones: Mostrar correos

En cada columna se muestra la fecha propuesta , las horas de cada una de las fechas así como la duración de cada una. Existe un enlace FORZAR que nos lleva a una confirmación en la elección de la votación por parte del administrador. Este caso se utilizará cuando el administrador estime oportuno, pero sobre todo será útil cuando tengamos una caso de votación duplicada.

En la base de la columna se muestra el resultado por fecha de las votaciones.

Si el administrador decide modificar la votación de un contacto tiene que marcar el checkbox que se encuentra a continuación del nombre del contacto. En ese momento se entraría en el modo de edición de la votación.

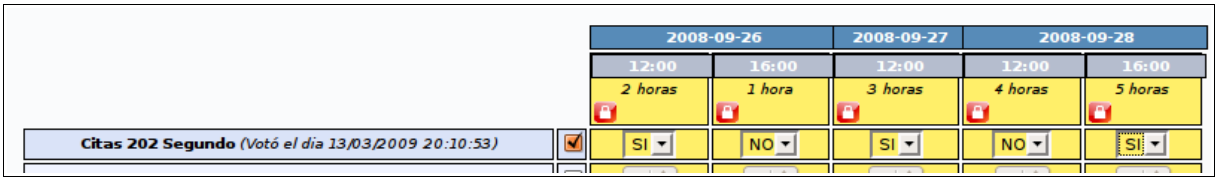

Ilustración 31: Administración de votaciones: Modificación de una votación

Para que se sepa lo que se está modificando, el color de las votaciones modificadas aparecerá en amarillo. En este caso no resalta demasiado ya que todas son candidatas.

Si se elige forzar una fecha, se podrá desforzarla con el enlace a [Quitar forzar] que tenemos en el interfaz. También se pedirá confirmación y se dará la oportunidad de enviar un mensaje.

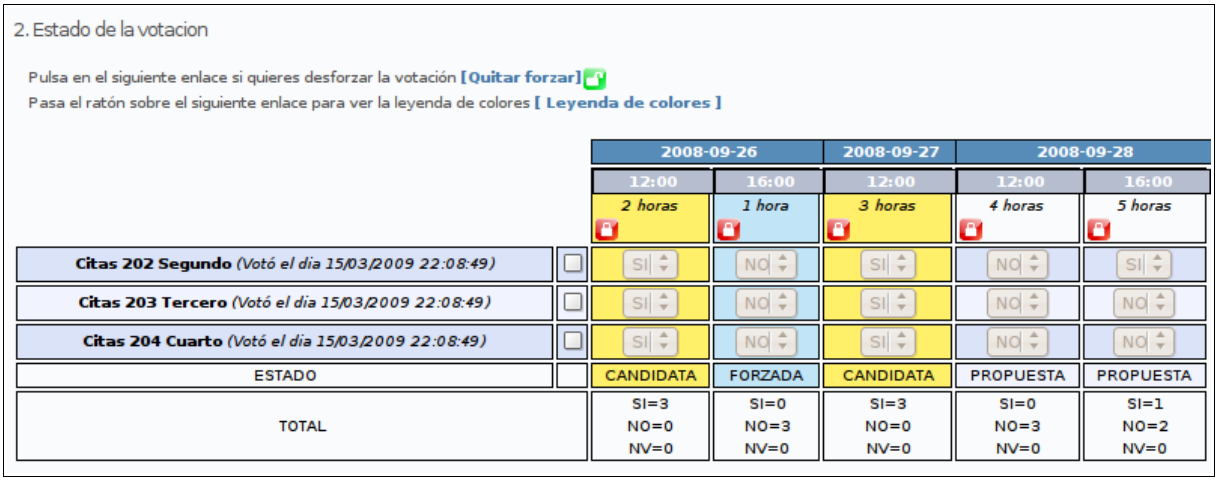

Ilustración 32: Administración de votaciones: Resumen final

En la figura de arriba podemos ver como la candidata resultó ser la fecha propuesta que está en amarillo, pero sin embargo el administrador decidió forzar la elección a la que esta en azul claro. Es un ejemplo de que el administrador es el último que tiene la palabra.

También se puede ver que cuando un contacto vota aparecerá la fecha de votación en

su casilla. Esta fecha es la de la última votación ya que se le permite al usuario modificar su votación las veces que desee.

7.2.2.1.5 Forzar o desforzar una votación

Cuando se elige forzar una cita, bien desde la vista de listado o bien desde el modo edición, se facilita un formulario donde se pide confirmación y se brinda la posibilidad de notificar el nuevo estado a los invitados. El formulario se muestra en la figura siguiente.

Ilustración 33: Forzar o desforzar una votación

Independientemente de que se notifique a los invitados la elección, se registra en las notas de la aplicación la acción tomada.

7.2.2.2 Edición de datos personales

Otra parte de la aplicación es la de edición de datos personales la cual se divide en tres bloques.

7.2.2.2.1 Datos personales

Aquí se podrá modificar el nombre y apellidos, el título que se desea tener y la firma personalizada que se añade a cada mensajes.

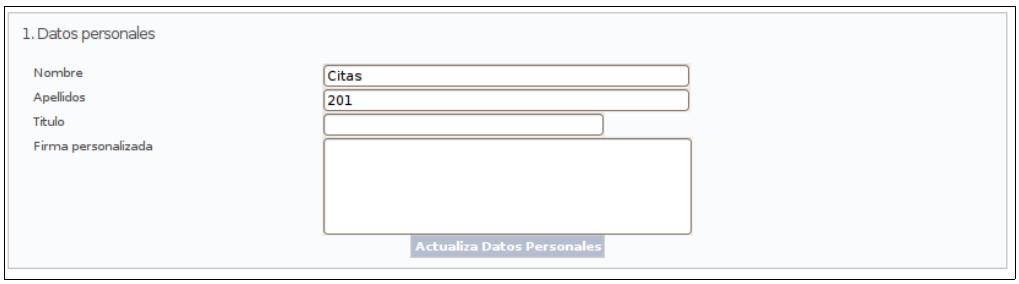

Ilustración 34: Datos personales: Bloque de datos personales

7.2.2.2.2 Datos de aplicación

Este menú permitirá modificar el login y la password del usuario

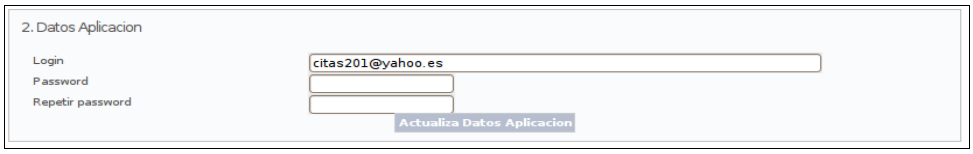

Ilustración 35: Datos personales: Bloque de datos de aplicación

7.2.2.2.3 Opciones en las citas

Con estas opciones se va a indicar si se quiere salir como el remitente en los correos que se envían desde el servidor y si se quiere poner en copia a algún otro correo siempre. Esta última opción permite tener una copia de los correos que se envían en otra cuenta de correo.

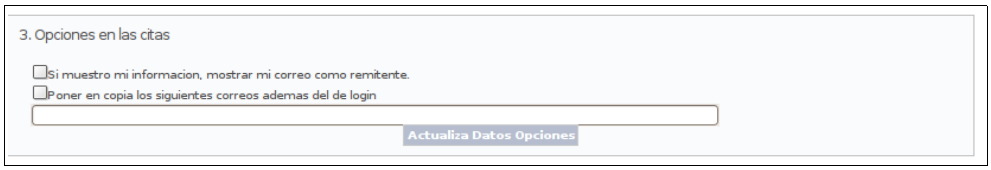

Ilustración 36: Datos personales: Bloque de opciones de la citas

7.2.3 Notificaciones recibidas

7.2.3.1.1 Notas introducidas

Después de todas las acciones que se han realizado sobre la cita se podrá ver como se van registrando los mensajes en la cita.

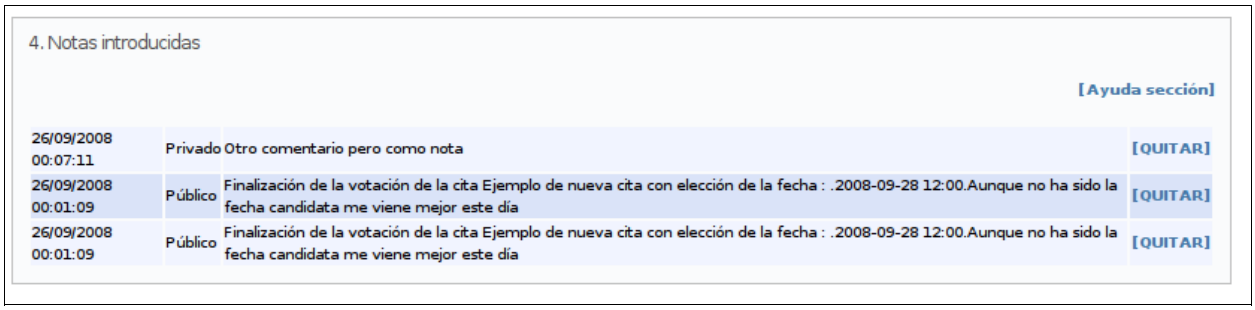

Ilustración 37: Mensajes registrados en la cita

7.2.3.1.2 Notificación de creación de invitación

Cuando se notifica un evento, los invitados reciben una notificación como la de la siguiente figura

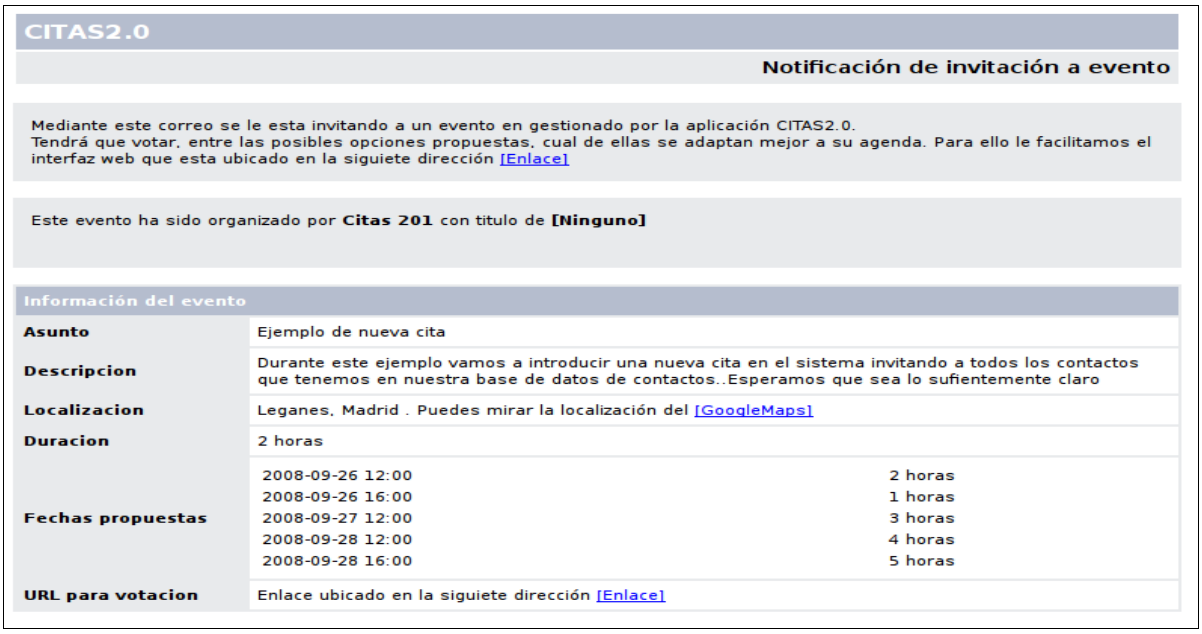

Ilustración 38: Notificación de invitación a una cita

7.2.3.1.3 Notificación de mensajes a los usuarios

La notificación de mensajes por parte del administrador se manda en mensajes del tipo de la figura siguiente. Estos mensajes se envían en casos como cuando se fuerza la votación,

se desfuerza, se anula o cuando se envía un mensaje aleatorio.

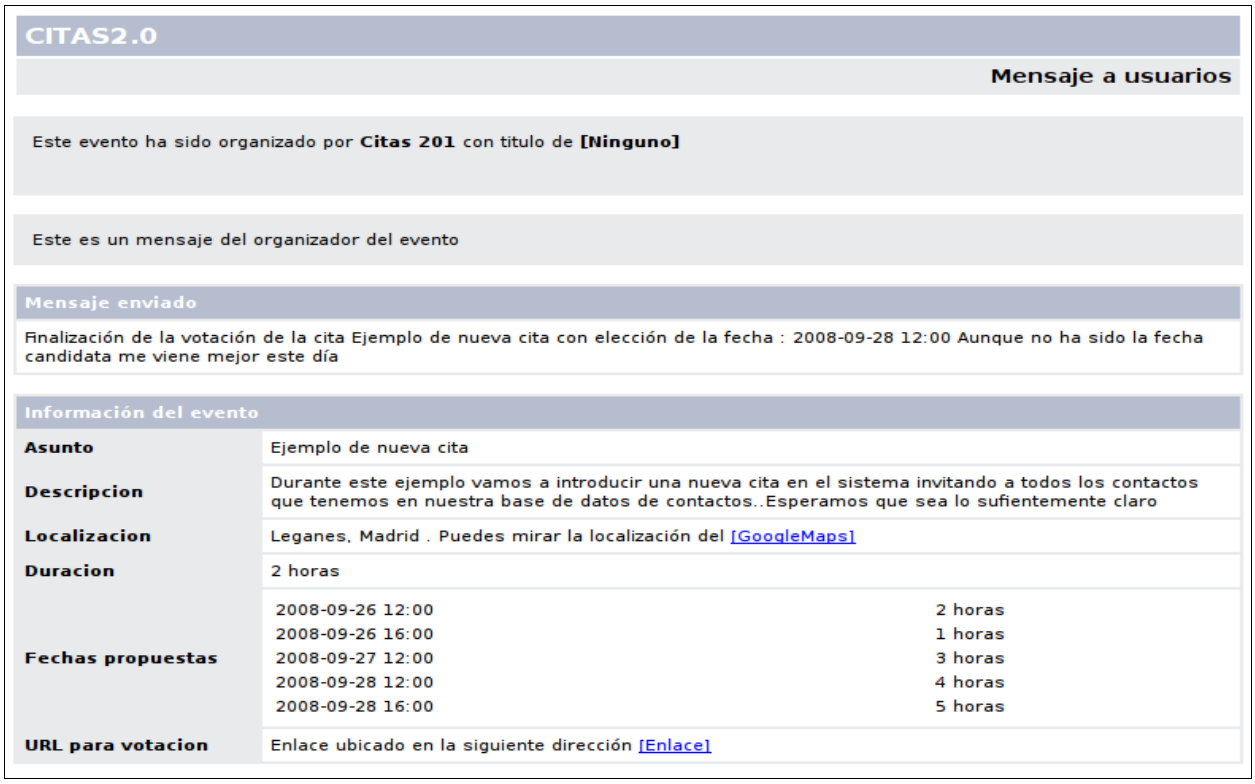

Ilustración 39: Notificación de mensajes

7.2.3.1.4 Notificación de recordatorio de cita

Una vez que la fecha para la cita ha sido elegida, y si el administrador de la misma ha seleccionado la opción de enviar recordatorio, el sistema enviará un mensaje de recordatorio desde los días previos elegidos hasta la fecha del evento.

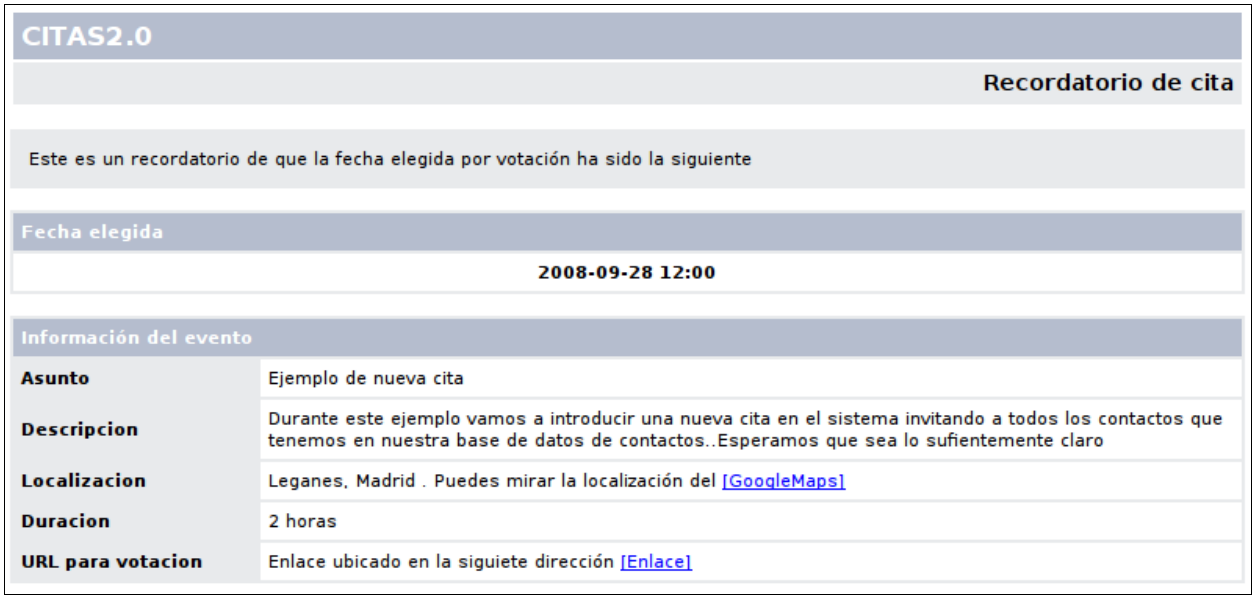

Ilustración 40: Recordatorio de cita

7.2.4 Interfaz de votaciones de cliente

Una vez que el administrador crea una cita, el sistema envía una notificación a cada contacto con un resumen de la cita y una url de votación. La aplicación que gestiona esta url es una aplicación no segura del sistema que toma la información de los datos del código pasado en el enlace.

Esta página tiene dividida la información en las siguientes partes:

- Parte de introducción a la filosofía de la aplicación
- Los datos administrativos del evento
- Los datos del organizador si él los quiere mostrar.
- Las notas que tienen carácter público
- El estado de la votación. Esto es una tabla con la matriz de votaciones del sistema. Como poco aparecerá las fechas propuestas y la fila que requiere la votación. Dependiendo de como haya sido configurada la invitación en esta matriz aparecerán el resto de invitados, sus votaciones y sus correos electrónicos.

Una vez que elegimos la votación la validamos para introducirla en el sistema.

Capítulo 8. Conclusiones

8.1 Introducción

En este capítulo se hace un balance de los objetivos propuestos frente al diseño e implementación realizado para cumplirlos. Se trata de ser objetivo y sacar los puntos fuertes y a mejorar del proyecto realizado.

8.2 Conclusiones

El objetivo principal del proyecto era la de la implementación de una aplicación de gestión de reuniones fácil de usar, flexible, robusta, de código libre y gratuita y que además participase de las definiciones de web 2.0 haciendo el contenido exportable.

Partiendo de esta idea nacen una serie de subobjetivos que se definieron en el capítulo primero y que se van a analizar a continuación

8.2.1 Realización de un juego de librerías y un gestor de aplicaciones

Este objetivo nace como hito a realizar previo a la definición de la aplicación. La idea es la de tener un conjunto herramientas y un gestor de aplicaciones que permitan construir aplicaciones web de una forma más sencilla y que tenga carácter de escalabilidad para versiones futuras.

No era necesario para la consecución del objetivo principal realizar esta parte tan definida ya que dentro del aplicativo se podría haber incluido código reutilizable adaptado a la aplicación en curso. La motivación de esta separación ha requerido más esfuerzo y tiempo, que en un proyecto real se traduce en dinero. Pero pese a este esfuerzo inicial, una vez que se ha tenido, el sistema se ha construido de un forma más sencilla y se ha ganado en disponer de herramientas genéricas reutilizables para un futuro. Se ha liberado a las aplicaciones de un control de accesos, de mezclar presentación y programación, del control de sesiones y datos de formularios, selector de tipo de dispositivo, etc.

Como punto a mejorar de este desarrollo fue el motor de plantillas creado. Tiene como ventajas que es muy rápido y sencillo, pero como desventajas la poca variedad de funciones en la sintaxis de plantilla lo cual se traduce en poca potencia. Para un proyecto real existen

mejores opciones como Smarty bajo php y como se discute en el capítulo 4.

8.2.2 Desarrollo de CITAS2.0

El siguiente objetivo era el de desarrollar la aplicación propiamente dicha. Gracias al objetivo anterior se ha podido centrar el desarrollo solo en la parte de aplicación.

Una de las características a perseguir en la implementación era que fuera una RIA pero que además pudiera ser accesible. En un principio no tiene que existir ninguna incompatibilidad entre usabilidad y accesibilidad, pero en la práctica el trasladar todas las funcionalidades de nuestra RIA a una aplicación accesible suponía tener una aplicación lenta y pesada para un dispositivo de esas características. Se optó por implementar dos aplicaciones distintas para cada tipo de dispositivo, una de interfaz enriquecido y otra centrada en accesibilidad. Esta última con menos funcionalidades que la anterior pero tratando de llegar a un equilibrio entre la rapidez de la aplicación y la funcionalidad ofrecida. Quizás el tener dos implementaciones diferenciadas pueda considerarse un error de diseño pero no se encontró otra forma que potenciara cada característica principal en cada interfaz diferente. Ejemplos similares pueden encontrarse en la web como GMAIL de Google Inc.

La parte de interfaz enriquecido se basa bastante en Javascript con la finalidad de dinamizar la interfaz. Para hacer una aplicación de estas características se ha programado mediante técnicas Ajax en donde la implementación puede dividirse en los siguientes puntos:

- Desarrollo de la aplicación cliente en javascript que controla los eventos del interfaz
- Parte de aplicación en servidor que se encarga de procesar la consulta inicial y da el contenido esquema que la interfaz de cliente rellena en base a los datos obtenidos por consultas asíncronas
- Buena parte de métodos externos que atienden peticiones asíncronas y que dan la rapidez al sistema
- Definición de plantillas iniciales XHTML y definición de mensajes JSON

Para la parte de interfaz accesible se han usado técnicas de programación tradicional basándose en contenido dinámico dado por el servidor.

Como se puede ver es un concepto diferente de estructura de una aplicación web que requiere controlar más puntos y meter más desarrollo. Esto da la idea de la complejidad del desarrollo solo de la aplicación. Si no se hubiera realizado el objetivo anterior de separación de partes comunes habría sido mucho más compleja la implementación.

También, para que esta aplicación pueda introducirse dentro de los estándares de Web 2.0 se han desarrollado interfaces que permiten exportar la información mediante iCAL y RSS. Se ha respetado la seguridad del sistema al introducir un código de autenticación en estas consultas dado que el carácter de esta información es privado.

El uso de técnicas de programación basadas en peticiones asíncronas y los interfaces abiertos, hacen a CITAS2.0 un sistema diferente a los que hay en el mercado

8.2.1 Conclusiones globales

Una vez evaluados cada uno de los objetivos se puede considerar que se ha realizado un proyecto que cumple de forma satisfactoria con los requisitos pedidos. Se ha desarrollado una aplicación distinta de las que existen en el mercado actualmente, con un conjunto de nuevas funcionalidades interesantes. Como se ha evaluado existen sus puntos de mejora que pueden irse solventando en futuros desarrollos y con esa idea se ha ido implementado y diseñando la aplicación, la de poder se mejorada y escalada a distintos sistemas.

Se han verificado fallos en el diseño como la elección del motor de plantillas, que no son absolutos pero que la utilización de aplicativos externos hubiera facilitado ahorrado tiempo. También la unificación de interfaz accesible con el enriquecido, que según el diseño y las decisiones tomadas no se veían viables en esta versión del aplicativo.

Por otra parte el alumno a cumplido su objetivo de aprender y desarrollar una aplicación basada en técnicas asíncronas creando todas las partes implicadas.

8.3 Trabajos futuros

A lo largo del desarrollo del proyecto han surgido ideas de desarrollo nuevas, de las cuales algunas partían de necesidades que surgían del uso y otras del uso de otras aplicaciones y del conocimiento de nuevas. Aunque algunas se han podido desarrollar otras solo se van a exponer aquí ya que hubieran consumido muchos recursos. Se pasa a exponer mejoras e ideas nuevas que se dejan para desarrollos posteriores u otros proyectos.

8.3.1 Hacer una única aplicación para cualquier tipo de petición

Como se ha comentado a lo largo de la memoria el diseño se a realizado en base a tener dos aplicaciones diferenciadas dependiendo del origen de la petición. Esto se puede considerar como un error de diseño, aunque se suele hacer habitualmente.

La forma correcta de desarrollar la aplicación respetando la accesibilidad con una única

aplicación tendría que haber seguido las siguientes directrices:

- Desarrollar la aplicación para que lo entienda el elemento menos accesible, el que tiene menos recursos.
- La aplicación debe funcionar sin javascript
- Sobre la estructura simple se añade funcionalidad mediante widgets javascript. Esto es lo que aumenta la usabilidad de la aplicación
- AJAX debe usarse de forma no intrusiva, es decir, debe aumentar la funcionalidad y no crear la información de la aplicación. Su uso tiene que dar dinamismo al contenido y agilidad a la aplicación. Debe respetar que sin javascript la aplicación tiene que funcionar.

Como primera aproximación se han desarrollado algunas partes con etiquetas noscript, introduciendo entre ellas el contenido que se crearía con AJAX de forma asíncrona. Es una solución pero parte de un diseño incorrecto. Se propone como mejora futura el desarrollo de un único interfaz.

8.3.1 Mejorar el desarrollo del gestor de aplicaciones

El gestor de aplicaciones puede mejorarse para añadir más funcionalidades como

- Mejorar el uso de plantillas añadiendo más sentencias de control e incluso otro lenguaje propietario (Smarty)
- Meter más bases de datos en el manejador de base de datos
- Mejorar el control del formulario.
- Añadir librerías de manejo de correos

8.3.1 Implementar la aplicación en Google App Engine

Una buena alternativa a todo el desarrollo del portal hubiera sido intentarlo en Google. Podemos hacer uso de sistemas de autenticación, aunque solo para cuentas google, la Big Table que es la base de datos propietaria de google, toda la disponibilidad de aplicación de sus sistemas, la escalabilidad que queramos, etc. Por contra su uso gratuito es limitado.

8.3.2 Desarrollar un sistema de envío de mensajes más completo

Quizás quede fuera del objeto de este proyecto, pero la filosofía de la aplicación creada da lugar a pensar en que cosas serían útiles si usásemos la aplicación de forma exhaustiva. Una de ellas sería el mejorar la comunicación con los participantes:

- Si el usuario esta registrado se podría mostrar el estado de los contactos.
- Habilitar mensajería instantánea entre usuarios registrados.
- Permitir el envío de mensajes por parte de los invitados desde el interfaz de votaciones para comunicar algo al administrador. Siempre se puede enviar por correo electrónico pero quedaría más compacto si se hiciera todo desde la aplicación.
- Permitir el hacer un mini blog entre los participantes de un evento

Apéndice I. Descripción de los métodos de las librerías del portal genérico

A1.1 Introducción

En este apéndice se van a mostrar las referencias de uso de cada uno de los métodos creados en las librerías. Se va a respetar la estructura mostrada en los capítulos anteriores para clasificar cada una de las librerías creadas

A1.2 Plantillas

A1.2.1 Resumen de métodos

Método	Breve descripción
asignar ($directorio,$nombre)	Asignar ficheros de plantillas
desasignar ()	Desasignar ficheros de plantillas
muestraPlantilla ()	Mostrar los ficheros de plantillas
introducirVariableEnRaiz ($etiqueta, $valor)	Introducir una VARIABLE en la raíz del documento.
introducirArrayEnRaiz ($datos)	Introducir un array de VARIABLE=>VALOR en la raíz del documento
introducirArrayRepetitivoEnBloque ($bloque,$etiqueta,$valores)	Introducir múltiples valores para una ETIQUETA dentro de un CAMINO. Realiza ITERACION.
introducirVariableEnBloque ($bloque, $etiqueta,$valor)	Introducir el valor de una ETIQUETA en un CAMINO. No realiza ITERACION.
nuevalteracion ($bloque)	Realización de ITERACION para un camino.
introducirArrayEnBloque ($bloque, $datos)	Introduce un array de ETIQUETA=>VALOR dentro de un camino. Realiza ITERACION.
volcar ()	Procesa la plantilla con la estructura de datos.

Tabla 63: Resumen de métodos de la librería de plantillas

A1.2.2 Método asignar

Variables de entrada	Descripción
$directorio	Directorio donde se encuentran las plantillas partiendo del raíz
$nombre	Puede ser el nombre un array con nombres de ficheros o un string
Salida	Devuelve true

Es método asigna los ficheros de plantillas que vamos a utilizar. Si el nombre introducido es un array, los nombres de los ficheros que vamos a utilizar son los mismos que aparecen en cada una de las posiciones del array anteponiendo el **$directorio** y siempre bajo el directorio raíz del portal.

Si por el contrario el parámetro **$nombre** es un string, la idea es que de una forma sencilla se construya un conjunto de ficheros que conformar la cabecera, el cuerpo y los pies del documento de salida. Para ello comprobamos la existencia de los ficheros "cabecera**_$nombre**.tpl", "cuerpo**_$nombre**.tpl" y "pies**_$nombre**.tpl". Si alguno de ellos no existe usamos "cabecera.tpl", "cuerpo.tpl" y "pies.tpl" respectivamente. Esto que puede parecer un poco lioso facilita la reutilización del texto ya que solo cambia el cuerpo en la mayoría de los casos.

Para el uso de un único fichero, como en el caso de la construcción de mensajes JSON y XML, basta con introducir un array con un valor que es el nombre del fichero plantilla.

A1.2.3 Método desasignar

Variables de entrada	No tiene
Salida	Devuelve true

Borra la asignación de plantillas.

A1.2.4 Método muestraPlantilla

Variables de entrada	No tiene
Salida	Devuelve el array con los ficheros de plantilla.

Muestra el array con los ficheros de platilla asignados.

A1.2.5 Método introducirVariableEnRaiz

Variables de entrada	Descripción
$variable	Nombre de la VARIABLE a introducir en la estructura de datos
$valor	Valor que tiene
Salida	Devuelve true

Este método nos permite introducir VARIABLES en el CAMINO raíz de la estructura de datos.

A1.2.6 Método introducirArrayEnRaiz

Variables de entrada	Descripción
$datos	Array con donde los índices son los nombres de las VARIABLES y el contenido el valor del las mismas
Salida	Devuelve true

Metemos la variables contenidas en el array de la forma VARIABLE => VALOR en la raíz de la estructura de datos.

A1.2.7 Método introducirArrayRepetitivoEnBloque

Variables de entrada	Descripción
$bloque	CAMINO a seguir
$etiqueta	Nombre de la ETIQUETA a introducir en la estructura de datos
$valores	Array con valores para una misma ETIQUETA
Salida	Devuelve true

Con este método introducimos de golpe todos los valores que deseemos para una ETIQUETA. Después del procesamiento de cada valor hay una iteración para ese CAMINO por lo que no podremos introducir nuevas ETIQUETAS en iteraciones anteriores.

A1.2.8 Método introducirVariableEnBloque

Variables de entrada	Descripción
$bloque	CAMINO a seguir
$etiqueta	Nombre de la ETIQUETA a introducir en la estructura de datos
$valor	Valor para una ETIQUETA
Salida	Devuelve true

Introducimos una ETIQUETA en la estructura sin incrementar la ITERACION para ese CAMINO.

A1.2.9 Método nuevalteracion

Variables de entrada	Descripción
$bloque	CAMINO a seguir
Salida	Devuelve true

Incrementa la ITERACION del CAMINO indicado.

A1.2.10 Método introducirArrayEnBloque

Variables de entrada	Descripción
$bloque	CAMINO a seguir
$datos	Array donde el indice es la ETIQUETA y su contenido el VALOR de la misma
Salida	Devuelve true

Aquí podemos introducir un conjunto de ETIQUETAS en una ITERACION. El método incrementa la ITERACION.

A1.2.11 Método volcar

Variables de entrada	No tiene
Salida	Devuelve la cadena con la plantilla procesada

Realiza el procesamiento de la plantilla y la devuelve procesada

A1.3 Seguridad

A1.3.1 Resumen de métodos

Método	Breve descripción
chequeaLogin ($salida,$datos)	Comprueba el acceso con los datos dados y devuelve los datos indicados
datosConexionChequeo ($datos, $db_tabla,$db_campos)	Configuramos el acceso a la base de datos de autenticación
destruyeSesion ()	Elimina la sesión
eliminaArrayEnSesion ($variables)	Elimina un conjunto de variables de la sesión
eliminaVariableEnSesion ($variable)	Elimina una variable concreta da la sesión
extraeArrayDeSesion ($variables)	Introduce en el ámbito global las variables de sesión indicadas dentro del array
extraeVariableDeSesion ($variable_sesion,$variable_externa)	Introduce en el ámbito global la variable de sesión indicada.
finalizaSesion ()	Fuerza la escritura de los datos de sesión y la finaliza.
guardaArrayEnSesion ($variables)	Introduce en el ámbito de sesión las variables globales indicadas dentro del array
guardaVariableEnSesion ($variable_sesion,$variable_externa)	Introduce en el ámbito de sesión la variable global indicada.
inicioSesion ()	Inicia la sesión

Tabla 64: Resumen de métodos de la librería de seguridad

A1.3.1 Método datosConexionChequeo

Variables de entrada	Descripción
$datos_conexión	Array con los valores de HOST, PUERTO, USER, PASSWD y BBDD
$tabla	Tabla de la base datos de usuarios
$campos_chequeo	Campos de la tabla que utilizamos para realizar la autenticación
Salida	true

Con este método definimos los datos del servidor de autenticación

A1.3.2 Método chequeaLogin

Variables de entrada	Descripción
$datos_extraer	Array con los datos a extraer de la tabla si el acceso es correcto
$datos_validar	Valores de los campos que hemos definido como los de autenticación
Salida	boolean

Utilizamos este método para comparar si los datos de autenticación son correctos. Si lo son, introduce en las variables globales del programa ($GLOBALS) cada uno de los campos que se han definido en el array $datos_extraer para poder utilizar su contenido en el resto del programa. El nombre de esas variables es el mismo que el nombre del campo de la tabla.

A1.3.3 Método inicioSesion

Variables de entrada	No tiene
Salida	Identificador de la sesión

Es método inicia la sesión y devuelve el identificador de la misma

A1.3.4 Método finalizaSesion

Variables de entrada	No tiene
Salida	true

Finalizamos la sesión controladamente al forzar su escritura en disco.

A1.3.5 Método destruyeSesion

Variables de entrada	No tiene
Salida	true

Limpiamos el array de variables de sesión y ejecutamos la función de destrucción de sesión implementada en php.

A1.3.6 Método extraeVariableDeSesion

Variables de entrada	Descripción
$variable_sesion	Nombre de la variable que esta almacenada en la sesión
$variable_externa	Nombre que queremos darle a la variable en el programa.
Salida	True

Primero introduce el nombre de la variable de sesión en el la propiedad de la clase que es un array de variables de sesión.

Si la variable no existe en la sesión lo que hacemos es crearla en la sesión, crearla en el grupo de variables globales e inicializarla en todos los casos a nulo. Simplemente inicializa la variable en los dos ámbitos.

Si la variable existe se actualiza o crea la variable global con el nombre externo que se indica y con el valor de la variable de sesión.

A1.3.7 Método extraeArrayDeSesion

Variables de entrada	Descripción
$variables	Array de relaciones VARIABLE_SESION => VARIABLE_EXTERNA
Salida	True

Ejecuta el método extraeVariableDeSesion para cada pareja compuesta por nombre_sesión y nombre_externo.

A1.3.8 Método guardaVariableEnSesion

Variables de entrada	Descripción
$variable_sesion	Nombre que va a tomar la variable en la sesión
$variable_externa	Nombre que tiene la variable en el ámbito global
Salida	True

Almacena la variable externa en la variable de sesión indicada.

A1.3.9 Método guardaArrayEnSesion

Variables de entrada	Descripción
$variables	Array de relaciones VARIABLE_SESION => VARIABLE_EXTERNA
Salida	True

Ejecuta el método guardaArrayEnSesion para cada pareja compuesta por nombre_sesión y nombre_externo.

A1.3.10 Método eliminaVariableEnSesion

Variables de entrada	Descripción
$variable	Nombre de la variable de sesión
Salida	boolean

Elimina el la variable indicada del conjunto de variables de sesión. Esto hará que no se almacene cuando finalice la sesión

A1.3.11 Método eliminaArrayEnSesion

Variables de entrada	Descripción
$variables	Array con nombres de variables
Salida	True

Ejecuta de forma recursiva el método eliminaVariableEnSesion para cada valor del array.

A1.4 Acceso a base de datos

A1.4.1 Resumen de métodos

Método	Breve descripción
conexion ($datos)	Conexión con la base de datos
desconexion()	Desconexión
muestraUltimoIdentificador ()	Muestra el último identificador de un insert con autoincrement
lanzaConsulta ($consulta)	Realiza la consulta

Método	Breve descripción
extraeFila ($tipo)	Extrae la fila en el formato que le indiquemos
extraeCampo ($fila,$columna)	Extrae un campo concreto
numeroFilasAfectadas ()	Muestra el número de filas afectadas
numeroFilasResultado ()	Muestra el número de filas resultado
muestraConsulta()	Muestra la última consulta realizada.

Tabla 65: Resumen de métodos de la librería de manejo de base de datos

A1.4.1 Método conexion

Variables de entrada	Descripción
$datos	Array con los datos de conexión en el formato NOMBRE => VALOR
Salida	boolean

Realiza la conexión con la base de datos remota y almacena el identificador de conexión en la propiedad creada para ese uso. También almacena los datos en las propiedades correspondientes.

A1.4.2 Método desconexion

Variables de entrada	Ninguna
Salida	boolean

Cierra la conexión con la base de datos. Aunque el servidor la cierra cuando acaba la ejecución del script puede que nos interese cerrarla antes.

A1.4.3 Método muestraUltimoIdentificador

Variables de entrada	Ninguna
Salida	Valor del último identificador

Cuando realizamos un INSERT en una tabla donde el índice es autoincrement, si no ponemos ningún valor a ese campo, la base de datos se encarga de asignar el siguiente en la lista. Es muy útil para crear registros únicos. Para poder acceder al valor de la clave generada podemos utilizar este método.

A1.4.4 Método lanzaConsulta

Variables de entrada	Descripción
$consulta	Cadena SQL a ejecutar.
Salida	boolean

Con este método realizamos consultas SQL al servidor conectado. Almacena la consulta en la propiedad interna de la clase para su posterior consulta y devuelve true o false dependiendo de como se haya dado la consulta.

A1.4.5 Método extraeFila

Variables de entrada	Descripción
$tipo	Tipo con el que indicamos el formato de salida de los datos
Salida	Array con la información de la fila

Una vez que hemos realizado una consulta podemos ir extrayendo fila a fila los datos de la misma. Dependiendo del tipo que se indique podremos mostrar la fila extraida en los siguientes formatos:

- **MYSQL_ASSOC:** Resultado en array asociativo, es decir, en el formato CAMPO=>VALOR
- **MYSQL_NUM:** Resultado en array numérico, es decir, NUMERO_CAMPO => VALOR
- **MYSQL_BOTH**: Array resultante con los dos métodos anteriores.

A1.4.1 Método extraeCampo

Variables de entrada	Descripción
$fila	Fila referenciada
$columna	Columna referenciada
Salida	Valor indicado

Podemos también extraer un campo concreto de la consulta lanzada anteriormente.

A1.4.2 Método numeroFilasAfectadas

Variables de entrada	Ninguna
Salida	Número de filas

Mostramos el número de filas que han sido afectadas por la anterior consulta. Es útil para ver rápidamente si nuestra consulta ha tenido éxito o no

A1.4.3 Método numeroFilasResultado

Variables de entrada	Ninguna
Salida	Número de filas

Mostramos el número de filas resultado de una sentencia SQL SELECT.

A1.4.4 Método muestraConsulta

Variables de entrada	Ninguna
Salida	Cadena con el SQL

Mostramos el última SQL lanzado como consulta.

A1.5 Manejo de datos

A1.5.1 Resumen de métodos

Método	Breve descripción
introduceArrayObligatorias ($variables)	Copia el array de nombres a la propiedad $variables_obligatorias
introduceArrayVariables ($variables)	Copia el array de nombres a la propiedad $variables_entorno y recarga los datos globales
recargaDatosGlobales ()	Extrae los datos de los datos GET y POST. Realiza el limpiado de los mismos
chequeaVariablesObligatorias ()	Comprueba que no hay ninguna variable vacia.

Tabla 66: Resumen de métodos de la librería de manejo de datos

A1.5.1 Método introduceArrayObligatorias

Variables de entrada	Descripción
$variables	Array con los nombres de las variables
Salida	boolean

Mediante este método indicamos que variables son las que son de introducción obligatoria. Esto nos va a ayudar cuando realicemos el chequeo de las variables que se han tenido que introducir en el formulario.

A1.5.2 Método introduceArrayVariables

Variables de entrada	Descripción
$variables	Array con los nombres de las variables
Salida	boolean

Cuando llamamos a este método con un listado de variables, se extrae del dominio de variables POST o GET las que se han indicado con el array y se introducen en el conjunto de variables globales. En el caso de no encontrarse como variable se crea con valor nulo.

También se realiza un limpiado de caracteres de control que pueden romper el código.

A1.5.3 Método recargaDatosGlobales

Variables de entrada	Ninguna
Salida	boolean

Se realiza la carga del listado de variables, indicado en métodos anteriores, en el espacio de variables globales, extrayéndolas de conjunto de variables POST y GET. Si no existe la crea como nula en el grupo de globales.

A1.5.4 Método chequeaVariablesObligatorias

Variables de entrada	Ninguna
Salida	boolean

Recorre cada una de las variables definidas anteriormente como obligatorias, y chequea que no son vacías. Si alguna lo es devuelve false. Si todas tienen contenido devuelve true.

Apéndice II. Descripción de los métodos de las librerías de la aplicación CITAS2.0

A2.1 Introducción

En este apéndice se van a mostrar las referencias de uso de cada uno de los métodos creados en las librerías. Se va a respetar la estructura mostrada en los capítulos anteriores para clasificar cada una de las librerías creadas

A2.2 Gestión de usuarios del sistema

A2.2.1 Resumen de métodos

Método	Breve descripción
asignaDatos ($datos_conexion, $datos_usuario)	Asigna los datos de usuario iniciales y los de conexión
usuarioIntroduceDatos ($datos)	Mete los nuevos datos de usuario
usuarioActualizaTSLogin ()	Actualiza el tiempo de entrada
usuarioIntroduceUnDato ($campo, $dato)	Introduce un único dato
usuarioActualizaDatos ()	Aplica los cambios realizados
usuarioMuestraId ()	Muestra el identificador de usuario
usuarioPermitido ()	Indica si el usuario es correcto
usuarioMuestraUnDato ($campo)	Muestra solo el valor de un campo
usuarioMuestraTodosDatos ()	Extrae el array de todos los datos de usuario
usuarioModificaDatos ($datos)	Sobreescribe el array de datos de usuario
usuarioModificaUnDato ($campo, $valor)	Modifica solo un campo del array
usuarioActualizaClave ()	Crea la clave del usuario
usuarioActualizaClaveSiNoEsta ()	Comprueba si esta la clave y si no la crea

Tabla 67: Resumen de las propiedades de la clase usuario

A2.2.2 Método asignaDatos

Variables de entrada	Descripción
$datos_conexion	Array con los datos de conexión con la base de datos
$datos_usuario	Array con datos de usuario iniciarles en el formato NOMBRE => VALOR
Salida	boolean

Este método se encarga de conectarse con la base de datos de usuarios y recoger todos los datos del usuario. Para ello necesita que en los datos iniciales que se le pasan en el array de datos_usuario este el campo id_usuario, clave para sacar el resto de datos. Si no existe el id_usuario en la tabla de usuarios los datos se sobreescriben con nulo.

A2.2.3 Método usuarioIntroduceDatos

Variables de entrada	Descripción
$datos	Array con los datos de usuario en el formato NOMBRE => VALOR
Salida	boolean

Mediante este método introducimos un array de datos de usuario nuevo. Actualiza la propiedades de la clase.

A2.2.4 Método usuarioActualizaTSLogin

Variables de entrada	Ninguna
Salida	boolean

Actualizamos la fecha de entrada en el sistema.

A2.2.5 Método usuarioIntroduceUnDato

Variables de entrada	Descripción
$campo	Nombre del campo de la base de datos de usuario a actualizar
$valor	Valor del mismo
Salida	boolean

Con este método podemos ir actualizando uno a uno los campos del array de datos de usuario. Al igual que el otro método solo se actualiza la propiedad de la clase.

A2.2.6 Método usuarioActualizaDatos

Variables de entrada	Ninguna
Salida	boolean

Una vez que ya hemos realizado las modificaciones pertinentes en los valores de los datos de usuario, con este método realizamos la modificación real en la base de datos.

A2.2.7 Método usuarioMuestraId

Variables de entrada	Ninguna
Salida	Entero con el identificador de usuario

Muestra la clave que es el identificador de usuario.

A2.2.8 Método usuarioPermitido

Variables de entrada	Ninguna
Salida	boolean

Si existe el id_usuario entonces decimos que el usuario es correcto. Si cuando hemos introducidos los datos iniciales el usuario no existía se anula la propiedad de la clase, por lo que dejaría de existir id_usuario y este método daría error de acceso.

A2.2.9 Método usuarioMuestraUnDato

Variables de entrada	Descripción
$campo	Campo que queremos mostrar
Salida	Valor del campo

Devuelve el valor del campo de la propiedad datos de usuario de la clase.

A2.2.10 Método usuarioMuestraTodosDatos

Variables de entrada	Ninguna
Salida	Array con datos de usuario en el formato NOMBRE => VALOR

Muestra todos los datos del array de datos de usuario devolviendo el array completo.

A2.2.11 Método usuarioModificaDatos

Variables de entrada	Descripción
$datos	Array con los datos de usuario en el formato NOMBRE=>VALOR
Salida	boolean

Sobreescribe los datos de usuario con los datos de este nuevo array.

A2.2.12 Método usuarioModificaUnDato

Variables de entrada	Descripción
$campo	Nombre del campo a modificar
$valor	Valor a introducir.
Salida	boolean

Modifica el valor de un campo del array de datos de usuario.

A2.2.13 Método usuarioActualizaClave

Variables de entrada	Ninguna
Salida	Boolean

Crea la clave de acceso al sistema de la siguiente forma:

```
$clave=md5(crypt($this->datos_usuario["id_usuario"].$this->datos_usuario["passwd"]));
```

Devuelve un booleano dependiendo del éxito de la operación.

A2.2.14 Método usuarioActualizaClaveSiNoEsta

Variables de entrada	Ninguna
Salida	Booleano

Actualiza la clave si no existe en la configuración del usuario. Devuelve un booleano dependiendo del éxito de la operación.

A2.3 Aplicación de contactos

A2.3.1 Resumen de métodos

Método	Breve descripción
asignaDatos ($datos_conexion, $datos_usuario)	Asigna los datos de usuario iniciales y los de conexión
contactoIntroduceNuevo ($id_persona,$datos_contacto, $datos_emails)	Introduce un nuevo contacto
contactoModifica ($id_persona, $datos_contacto,$datos_emails)	Borra el actual y lo introduce de nuevo con los nuevos valores
contactoBorrarTodos ()	Borra todo el listado
contactoBorrar ($id_persona)	Borra un contacto
contactoListadoFiltroReset ()	Resetea los valores del filtro
contactoListadoFiltroLimites ($inferior,$superior)	Introduce el rango de contactos a extraer
contactoListadoFiltroOrden ($campo)	Indica el campo de ordenación del select.
contactoListadoFiltroVariosCampo ($filtro)	Indicamos que campos tienen que tener que valores en el select
contactoListadoFiltroUnCampo ($campo,$valor)	Metemos un campo para el filtrado
contactoListadoLanza ()	Lanzamos la extracción de contactos a la que se le aplicarán los filtros definidos
contactoListadoExtraeFila ()	Extraemos una fila del resultado del select
contactoNumeroTotal ()	Mostramos cuantos contactos tiene el usuario
contactoMuestraDatos ($campos, $id_persona)	Mostramos los datos que queramos del contacto indicado
contactoMuestraCorreos ($id_persona)	Muestra los correos del contacto indicado.

Tabla 68: Resumen de las propiedades de la clase contactos

A2.3.2 Método asignaDatos

Variables de entrada	Descripción
$datos_conexion	Array con los datos de conexión con la base de datos
$datos_usuario	Array con datos de usuario iniciarles en el formato NOMBRE => VALOR
Salida	boolean

Este método se encarga de conectarse con la base de datos de usuarios y recoger todos los datos del usuario. Para ello necesita que en los datos iniciales que se le pasan en el array de datos_usuario este el campo id_usuario, clave para sacar el resto de datos. Si no existe el id_usuario en la tabla de usuarios los datos se sobreescriben con nulo.

A2.3.3 Método contactoIntroduceNuevo

Variables de entrada	Descripción
$id_persona	Identificador de contacto
$datos_contacto	Array con datos los datos definidos de contacto en el formato CAMPO => VALOR
$datos_emails	String con todos los correos unidos por comas
Salida	Éxito: Devuelve el id_persona; Error: false

Mediante este método introducimos un contacto nuevo en la lista de contactos del usuario. Le pasamos el identificador de persona si queremos fijarlo a un valor, los datos del contacto y los correos que le pertenecen. Si la introducción ha sido correcta, la función devuelve el id_persona en caso de éxito o false en caso de error.

A2.3.4 Método contactoModifica

Variables de entrada	Descripción
$id_persona	Identificador de contacto
$datos_contacto	Array con datos los datos definidos de contacto en el formato CAMPO => VALOR
$datos_emails	String con todos los correos unidos por comas
Salida	boolean

Elimina el contacto y lo vuelve a introducir con los datos nuevos.

A2.3.5 Método contactoBorrarTodos

Variables de entrada	Ninguna
Salida	Boolean

Elimina todos los contactos del usuario. Es útil cuando se produce cargas de lista de contactos que sobreescriben a la actual.

A2.3.6 Método contactoBorrar

Variables de entrada	Descripción
$id_persona	Identificador de contacto
Salida	boolean

Elimina un contacto identificado por id_persona de la lista de contactos que pertenecen al usuario.

A2.3.7 Método contactoListadoFiltroReset

Variables de entrada	Ninguna
Salida	Boolean

Para facilitar la búsqueda de contactos se han facilitado una serie de métodos que nos permiten definir filtrados, campos de ordenación de la salida y que grupo de elementos queremos sacar del total.

Este método forma parte de este conjunto de métodos y se encarga de inicializar las propiedades que utilizan estos.

A2.3.8 Método contactoListadoFiltroLimites

Variables de entrada	Descripción
$inferior	Valor numérico que define el límite inferior
$superior	Valor numérico que define el límite superior
Salida	boolean

Mediante este método indicamos que cuando saquemos la lista de contactos solo me muestre los que empiecen en la posición $inferior y terminen en la posición $superior del listado de contactos sacados.

A2.3.9 Método contactoListadoFiltroOrden

Variables de entrada	Descripción
$campo	String que indica que campo de la tabla usamos para ordenar
Salida	boolean

Se le pasa el campo de la tabla que se usa para ordenar en la salida total.

A2.3.10 Método contactoListadoFiltroVariosCampo

Variables de entrada	Descripción
$filtro	Array en el formato CAMPO=>VALOR que nos indica como hay que realizar el filtrado
Salida	boolean

Aquí recibimos del un conjunto de campos con sus correspondientes valores para el filtrado. Se introducen en la propiedad $datos_filtro y posteriormente se añaden como restricciones en el select que vuelca los contactos del usuario.

A2.3.11 Método contactoListadoFiltroUnCampo

Variables de entrada	Descripción
$campo	Nombre del campo
$valor	Valor
Salida	boolean

Metemos un elemento en la propiedad de $datos_filtro para que se añadan las restricciones en el select.

A2.3.12 Método contactoListadoLanza

Variables de entrada	Ninguna
Salida	Número de contactos encontrados

Una vez que ya hemos definido los datos del filtrado procedemos a realizar la consulta en la base de datos. Para ello realizamos los siguientes pasos:

- Para cada uno de los elementos de $datos_filtro añadimos la condición al select. Si el valor de uno elemento es un array a su vez, introducimos una condición de grupo para un campo a varios valores, es decir, usamos IN en vez de LIKE para este caso particular.

- Una vez que esta construida la select añadimos el modificador de orden con el campo indicado en $datos_filtro_orden.
- Para finalizar añadimos el rango que queremos extraer mediante las propiedades de límites.

Se realiza la consulta y se devuelve el número de elementos afectados.

A2.3.1 Método contactoListadoExtraeFila

Variables de entrada	Ninguna
Salida	Array con los datos de la fila

Devuelve un array con los datos extraídos en la consulta. El modo de devolución es MYSQL_BOTH, es decir, se devuelve un array asociativo y uno numérico de los datos de la fila.

A2.3.2 Método contactoNumeroTotal

Variables de entrada	Ninguna
Salida	Número total de contactos del usuario

Devuelve el número total de contactos que tiene el usuario en su listado.

A2.3.3 Método contactoMuestraDatos

Variables de entrada	Descripción
$id_persona	Identificador de contacto
$campos	Array con los campos que queremos extraer de los contactos
Salida	Array con los datos del contacto

Extraemos los campos identificados en el array $campos del contacto identificado por $id_persona.

A2.3.4 Método contactoMuestraCorreos

Variables de entrada	Descripción
$id_persona	Identificador de contacto
Salida	Array con los correos.

Extraemos cada uno de los emails que pertenecen al contacto identificado por

$id_persona.

A2.4 Aplicación de invitaciones

A2.4.1 Resumen de métodos

Método	Breve descripción
asignaDatos ($datos_conexion,$datos_usuario)	Asigna los datos de usuario iniciales y los de conexión
invitacionesIntroduceCita ($datos_invitacion)	Introduce una nueva invitación en el sistema
invitacionesIntroduceContacto($id_cita,$contacto)	Añade un contacto a la invitación indicada
invitacionesBorraContacto ($id_cita,$contacto)	Borra un contacto de la invitación
invitacionesIntroduceFecha($id_cita,$fecha, $duracion)	Añade una fecha propuesta con la duración deseada
invitacionesBorraFecha ($id_cita,$fecha)	Borra la fecha indicada
invitacionesIntroduceVotacion ($id_cita,$id_persona, $peso,$fecha)	Introduce la votación de un contacto en la cita indicada
invitacionesBorraTodasVotacion ($id_cita, $id_persona)	Quita las votaciones de una persona en la cita
invitacionesBorraUnaVotacion ($id_cita,$id_persona, $votacion)	Quita solo una votación de todas las que haya podido realizar un usuario en la cita
invitacionesMuestraDatosVotaciones($id_cita)	Muestra los datos de las votaciones
invitacionesMuestraDatosVotacionesPeso($id_cita)	Muestra los datos de las votaciones con peso
invitacionesMuestraDatosResumenVotaciones($id_cita)	Saca un resumen de las votaciones realizadas hasta el momento
invitacionesMuestraEstadoVotacion($id_cita)	Muestra un string con todos los estados que tiene activos la invitación
invitacionesMuestraInicioCita($id_cita)	Muestra la primera fecha propuesta
invitacionesMuestraCandidata($id_cita)	Saca la fecha más candidata a ser elegida
invitacionesCaducaCitas($id_cita,$activa)	Cambia el estado administrativo de activa o histórico de la cita
invitacionesVotacionesMuestraForzada ($id_cita)	Indica si la cita tiene la votación forzada o no
invitacionesVotacionesPonForzada ($id_cita,$fecha)	Pone la fecha indicada como forzada en la invitación.
invitacionesVotacionesQuitaForzada ($id_cita)	Quita la fecha forzada si la hubiera
invitacionesAnulaCita ($id_cita)	Pone la cita como anulada administrativamente
invitacionesMuestraAnulada ($id_cita)	Indica si la cita esta anulada o no
invitacionesMuestraIdUsuarioCodigo ($codigo)	Busca si hay algún id_usuario con ese código
invitacionesMuestraDatosCitaPorCodigo($codigo)	Saca los datos de la cita que corresponden al código indicado

Método	Breve descripción
invitacionesMuestraCorreoCodigo ($id_cita)	Muestra el correo del participante
invitacionesBorraComentario($id_cita,$id_nota)	Quita el comentario indicado
invitacionesIntroduceComentario($id_cita,$publico, $tiempo,$mensaje)	Mete un comentario con el carácter indicado
invitacionesSacaComentarios($id_cita,$publico)	Saca todos los comentarios de la cita con el carácter indicado
invitacionesMuestraDatosContactos($id_cita, $campos)	Saca los campos indicados de los contactos de la cita
invitacionesMuestraCorreosContactos($id_cita)	Saca todos los correos de la cita así como su código de votación asociado.
invitacionesMuestraFechaVotacionContactos($id_cita)	Muestra en que fecha han votado los contactos de la cita indicada
invitacionesMuestraNombreContactos($id_cita)	Saca los datos de los contados de la cita
invitacionesMuestraDatosContactosSimple($id_cita)	Muestra solo los id_persona de los contactos.
invitacionesMuestraDatosFechas($id_cita)	Saca todas las fechas de la invitación
invitacionesMuestraDatosFechasDuracion($id_cita)	Muestra las fechas con duración particular de cada una
invitacionesBorraFechaPropuesta($id_cita,$fecha)	Borra una fechas del total de propuestas
invitacionesMuestraIdPersonaPorCorreoEnCita($id_ci ta,$correo)	Busca el identificador de contacto por el correo
invitacionesMuestraIdCita()	Muestra cual es identificador de cita actual
invitacionesMuestraTodasCitas($activa)	Muestra todas las citas del listado de activas o histórico que indiquemos
invitacionesMuestraTodasCitasUsuario($id_usuario, $activa)	Muestra todas las citas en las que aparece el usuario como invitado
invitacionesMuestraDatosCita($id_cita)	Muestra los datos de la cita indicada
invitacionesMuestraDatosOrganizador($id_cita)	Muestra los datos del organizador para enriquecer los mensajes
invitacionesMuestraActividadCita($id_cita)	Muestra la actividad de la invitación
invitacionesEliminaDatosCita($id_usuario, $id_cita)	Borra los datos de la cita
invitacionesEliminaCita($id_cita)	Borra la cita y sus votaciones

Tabla 69: Resumen de propiedades de la clase invitaciones

A2.4.2 Método asignaDatos

Variables de entrada	Descripción
$datos_conexion	Array con los datos de conexión con la base de datos
$datos_usuario	Array con datos de usuario iniciarles en el formato NOMBRE => VALOR
Salida	boolean

Este método se encarga de conectarse con la base de datos de usuarios y recoger todos los datos del usuario. Para ello necesita que en los datos iniciales que se le pasan en el array de datos_usuario este el campo id_usuario, clave para sacar el resto de datos. Si no existe el id_usuario en la tabla de usuarios los datos se sobreescriben con nulo.

A2.4.3 Método invitacionesIntroduceCita

Variables de entrada	Descripción
$datos_invitacion	Array en el formato NOMBRE => VALOR con todas las propiedades de la cita necesaria para crearla
Salida	Entero. El identificador de la cita creada

Con este método generamos una nueva cita o editamos la actual. Para ello seguimos los siguientes pasos:

- Si la cita esta creada borramos los datos actuales de la cita de la base de datos e introducimos los nuevos
- De las nuevas fechas propuestas introducimos las que no estaban ya y quitamos las que estaban y que ahora no se han elegido
- Hacemos lo mismo con los contactos
- Actualizamos las propiedades de la clase
- Retornamos el id_cita

A2.4.1 Método invitacionesIntroduceContacto

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$id_persona	Identificador de contacto
Salida	Boolean

Con este método se añade un contacto a la cita indicada. Aquí es donde se genera el

código de votación que permite que cada contacto realice únicamente su votación. El código se genera de la siguiente manera:

```
$codigo=md5(time()*$id_cita*$id_contacto*rand());
```

Cuando un contacto quiere acceder a la votación, se coge el código de votación y se extrae de la tabla invitaciones_persona la cita de la que se trata y el contacto que ha realizado la solicitud.

A2.4.2 Método invitacionesBorraContacto

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$id_persona	Identificador de contacto
Salida	Boolean

Mediante este método se quita de la invitación el contacto indicado así como las posibles votaciones que haya podido introducir.

A2.4.3 Método invitacionesIntroduceFecha

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$fecha	Fecha propuesta
$duracion	Duración concreta de la fecha
Salida	Boolean

Introducimos una fecha propuesta en la invitación. También permite modificar la duración de una fecha concreta si el administrador desea tener diferente duración a la global.

A2.4.4 Método invitacionesBorraFecha

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$fecha	Fecha a borrar
Salida	Boolean

Borra la fecha indicada y las votaciones asociadas a la misma.

A2.4.5 Método invitacionesIntroduceVotacion

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$id_persona	Identificador numérico del contacto
$fecha	Fecha votada
$peso	Valor de la votación
Salida	Boolean

Cuando el administrador o el usuario realizan una votación se llama a este método. Se encarga de introducir en la tabla invitaciones_votaciones los datos de la misma y de actualizar en la tabla de invitaciones_personas el día que el contacto ha realizado la votación.

A2.4.6 Método invitacionesBorraTodasVotacion

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$id_persona	Identificador de contacto
Salida	Boolean

Borra todas las votaciones de un contacto y quita la fecha de la votación

A2.4.7 Método invitacionesBorraUnaVotacion

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$id_persona	Identificador de contacto
$votacion	Votación a eliminar
Salida	Boolean

Elimina una votación concreta de usuario en una cita dada.

A2.4.8 Método invitacionesMuestraDatosVotaciones

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array

Devuelve un array con los datos de todas las votaciones. Es un array asociativo donde,

la clave de cada índice del array es la fecha propuesta por el administrador, y su valor es un array numérico cuyos valores son los id_personas que han elegido esa fecha como una de sus votaciones.

A2.4.9 Método invitacionesMuestraDatosVotacionesPeso

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array

Al igual que el método anterior devuelve un array con los datos de todas las votaciones. Es un array asociativo donde, la clave de cada índice del array es la fecha propuesta por el administrador, y su valor es un array numérico cuyos valores están constituidos por el id_persona seguido de "_" y del peso que el contacto ha elegido para esa fecha.

A2.4.10 Método invitacionesMuestraDatosResumenVotaciones

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array con el resumen de la votación

Devuelve un array con el resumen de las votaciones hasta la fecha. El array contiene la información del número de contactos de la cita y el número de votaciones realizadas por contactos diferentes.

A2.4.11 Método invitacionesMuestraEstadoVotacion

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	string

Este método calcula el estado actual de la cita y genera un string con los diferentes estados unidos por guiones. Las palabras clave que puede generar este método son las siguientes:

- Si todas las fechas propuestas son posteriores a la fecha actual ponemos **ACTIVA**. Si no **PASADA**.
- Si el número de contactos es igual al número de votaciones de contactos distintos ponemos **COMPLETA**. Si no **ENPROCESO**

- Si existe alguna fecha forzada se añade **FORZADA**
- Si la cita esta anulada se añade **ANULADA**
- Si existe más de alguna fecha elegida como candidata se pone **DUPLICADA**.

Una vez que se tienen todos los estados posibles se unen en un string mediante guiones y se retornan.

A2.4.1 Método invitacionesMuestraInicioCita

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	String con la fecha

Devuelve la primera fecha propuesta por el administrador.

A2.4.2 Método invitacionesMuestraCandidata

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array

Mediante este método mostramos las fechas candidatas para ser las elegidas. Dependiendo del tipo de votación y de otros factores se realiza la elección:

- Si la cita tiene alguna fecha forzada se envía esta como candidata
- Si el tipo de votación es por preferencias sumamos todos los pesos elegidos para cada fecha y ordenamos por esa suma. Las candidatas serán las que tengan esa suma mayor.
- Si no es por preferencias contamos cuantos SI hay para cada fecha y ordenamos por esa cuenta. Las que tengan cuenta mayor serán las elegidas.

Se devuelve un array con las fechas candidatas según los criterios arriba expuestos.

A2.4.1 Método invitacionesCaducaCitas

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$activa	Entero que indica si estamos en activas o históricas
Salida	Boolean

Caducar una cita es simplemente pasarla del listado de invitaciones activas al de

histórico de invitaciones. No afecta a los estados de la invitación ya estos son derivados de otras causas como hemos explicado arriba. Es un método para administrarnos los listados de invitaciones.

A2.4.2 Método invitacionesVotacionesMuestraForzada

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	String con la fecha forzada

Devuelve la fecha forzada en la invitación si existe. Si no devuelve string vacio.

A2.4.3 Método invitacionesVotacionesPonForzada

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$fecha	String con la fecha a forzar
Salida	Boolean

Se pone la fecha indicada como forzada para la elección de la fecha propuesta.

A2.4.4 Método invitacionesVotacionesQuitaForzada

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	boolean

Quita el indicador de forzada a la fecha propuesta.

A2.4.5 Método invitacionesAnulaCita

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	boolean

Actualiza la base de datos para indicar que la invitación ha sido anulada.

A2.4.6 Método invitacionesMuestraAnulada

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	boolean

Devuelve true si la cita ha sido anulada.

A2.4.7 Método invitacionesMuestraIdUsuarioCodigo

Variables de entrada	Descripción
$codigo	String con el código de votación del usuario
Salida	Entero con el identificador de usuario

Este método nos ayuda a modificar los valores de la clase invitación cuando se accede por código de votación al sistema. Actualiza el id_usuario, que es el identificador del creador de la invitación, de la clase al id_usuario de la invitación al que pertenece el código pasado.

Una vez que tenemos ese id_usuario podemos acceder a todos los datos de la invitación y realizar las modificaciones pertinentes sin tener que estar activos en el sistema como el administrador de la cita. Este recurso se utiliza en el cliente que se encarga de controlar las votaciones de todos los invitados.

A2.4.8 Método invitacionesMuestraDatosCitaPorCodigo

Variables de entrada	Descripción
$codigo	String con el código de votación del usuario
Salida	Array con los datos solicitados

Cuando accedemos por código nos interesa saber de que cita estamos hablando y para que contacto concreto. Con este método obtenemos un array con esos datos.

A2.4.9 Método invitacionesMuestraCorreoCodigo

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array

Devolvemos un array con todos los contactos que pertenecen a la cita indicada. La información por contacto que se devuelve es el id_persona, el código y el email

correspondiente. Para un mismo contacto con varios emails se devolverán varias filas.

A2.4.10 Método invitacionesBorraComentario

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$id_nota	Identificador numérico de la nota en cuestión
Salida	boolean

Elimina un comentario introducido en la cita.

A2.4.11 Método invitacionesIntroduceComentario

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$publico	Si el mensaje es público o no
$tiempo	Fecha del mensaje
$mensaje	Texto del mensaje
Salida	boolean

Introducimos el mensaje deseado. La mensajería se ideo como una forma de facilitar el envío de información a los invitados. También se puede utilizar el mismo interfaz como registro de comentarios personales sobre la invitación.

A2.4.12 Método invitacionesSacaComentarios

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$publico	Si el mensaje es público o no
Salida	boolean

Extraemos los comentarios de una cita y del carácter que queramos, público o privado.

A2.4.13 Método invitacionesMuestraDatosContactos

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$campos	Campos de la tabla invitaciones_persona que queremos extraer. Es un string con los campos separados por comas
Salida	boolean

Devuelve los campos indicados de todos los contactos de la tabla invitaciones_persona.

A2.4.14 Método invitacionesMuestraCorreosContactos

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array

Devuelve un array asociativo donde la clave es el id_persona del contacto y el valor los correos separados por comas que tenga el contacto.

A2.4.15 Método invitacionesMuestraFechaVotacionContactos

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array

Devuelve un array asociativo donde la clave es el id_persona del contacto y el valor el timestamp de cuando realizó la votación.

A2.4.16 Método invitacionesMuestraNombreContactos

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array con los contactos

Busca los contactos que pertenecen a la invitación y devuelve la información extendida de cada uno de ellos. Es útil ya que buscamos por cita

A2.4.17 Método invitacionesMuestraDatosContactosSimple

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array

Devuelve un array con todos los id_persona que pertenecen a una cita

A2.4.18 Método invitacionesMuestraDatosFechas

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array

Devuelve un array con cada una de las fechas que pertenecen a la cita.

A2.4.19 Método invitacionesMuestraDatosFechasDuracion

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array

Igual que el anterior solo que se anexa a la fecha una almohadilla y la duración de la misma. El formato final para cada fecha es fecha#duración.

A2.4.20 Método invitacionesBorraFechaPropuesta

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$fecha	String con la fecha a borrar
Salida	Boolean

Eliminamos la fecha propuesta de la cita y de las votaciones.

A2.4.21 Método invitacionesMuestraIdPersonaPorCorreoEnCita

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
$correo	String con el correo
Salida	String

Devuelve el id_persona del usuario que esta invitado a la cita y que tiene el correo indicado. Devuelve cadena vacía si no encuentra nada

A2.4.22 Método invitacionesMuestraIdCita

Variables de entrada	Ninguna
Salida	String con el id_cita

Devuelve el identificador de invitación registrado en la clase invitaciones.

A2.4.23 Método invitacionesMuestraTodasCitas

Variables de entrada	Descripción
$ activa	Entero que indica si queremos sacar las activas o no
Salida	Array

Saca todas las invitaciones que pertenecen al usuario y que tienen el campo activa según se indica en el parámetro de entrada. Nos permite sacar el listado de invitaciones activas y el de históricas.

Devuelve un array con los id_cita de cada invitación que cumple la condición.

A2.4.24 Método invitacionesMuestraTodasCitasUsuario

Variables de entrada	Descripción
$id_usuario	Identificador numérico del usuario
$activa	Entero que indica si queremos sacar las activas o no
Salida	Array

Este método se encarga de sacar de un usuario que este registrado en el sistema todas las invitaciones a las que ha sido añadido por otros usuarios del sistema. Devuelve un array con el id_cita donde el aparece como invitado y el código particular de votación.

A2.4.25 Método invitacionesMuestraDatosCita

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array con los datos

Muestra en un array todos los datos de la cita

A2.4.26 Método invitacionesMuestraDatosOrganizador

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	Array

Saca los datos del administrador para utilizarlos en los correos y páginas de votaciones de cara a los contactos.

A2.4.27 Método invitacionesMuestraActividadCita

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	String

Muestra solamente el campo actividad de la cita. Es el campo resumen de la misma

A2.4.28 Método invitacionesEliminaDatosCita

Variables de entrada	Descripción
$id_usuario	Identificador numérico del usuario
$id_cita	Identificador numérico de la cita
Salida	Boolean

Elimina los datos administrativos de la cita

A2.4.29 Método invitacionesEliminaCita

Variables de entrada	Descripción
$id_cita	Identificador numérico de la cita en cuestión
Salida	boolean

Elimina la cita en cuestión y sus votaciones pertinentes.

Glosario

- **AJAX**: Asynchronous JavaScript And XML
- **Apache:** Apache es programa de servidor HTTP Web de código abierto
- **Applet:** Pequeña aplicación escrita en Java la cual se difunde a través de la red en orden de ejecutarse en el navegador cliente.
- **ASP:** Acrónimo en inglés de Active Server Pages. Páginas de Servidor Activo.
- **Atom:** Un protocolo para la sindicación y compartir contenido
- **CGI:** Common Gateway Interface.
- **CMS**: Content Management System
- **Cookie:** Un cookie es un pequeño pedazo de data enviado desde un servidor web al navegador del cliente que se guarda localmente en la maquina del usuario.
- **CSS**: Cascading Style Sheets
- **CSV:** En inglés es Comma Separated Values.
- **DHTML:** Dynamic HTML. HTML dinámico.
- **DOM:** Siglas del inglés Document Object Model (Modelo de Objetos de Documento) es un API para documentos HTML y XML
- **DTD**: Document Type Definition
- **Flash:** Creado por Macromedia, esta tecnología permite la creación de animaciones, entre otras cosas, utilizando menos ancho de banda que otros formatos, como AVI o MPEG.
- **HTML:** Siglas del inglés Hypertext Markup Language (Lenguaje de Marcado Hipertexto).
- **HTTP:** Hypertext Transfer Protocol. Protocolo de Transferencia de Hipertexto.
- **HTTPS:** Creado por Netscape Communications Corporation para designar documentos que llegan desde un servidor web seguro. Esta seguridad es dada por el protocolo SSL (Secure Socket Layer) basado en la tecnología de encriptación y autenticación desarrollada por RSA Data Security Inc.
- **JavaScript:** Lenguaje desarrollado por Netscape y se utiliza para realizar programas activos en el navegador web
- **JSP:** Siglas de Java Server Pages o Páginas de servidor de Java, es la tecnología

para generar páginas web de forma dinámica en el servidor,

- **MIME:** Siglas de Multipurpose Internet Mail Extension. Sistema que permite integrar dentro de un mensaje de correo electrónico ficheros binarios.
- **MySQL:** Base de datos gratuita
- **PHP:** Hypertext Preprocessor. Lenguaje de script diseñado para la creación de páginas web activasen la parte de servidor.
- **RIA**: Rich Internet Applications
- **RSS**: Really Simple Syndication. Sindicación Realmente Simple.
- **SGML**: Lenguaje Estandarizado de Marcado General. Estándar internacional para la definición de métodos de representación de texto en forma electrónica no ligados a ningún sistema ni a ningún dispositivo.
- **SMTP**: Protocolo Simple de Transferencia de Correo.
- **SQL**: Structured Query Language. Es un lenguaje especializado de programación que permite realizar consultas (queries) a bases de datos.
- **URL**: Acrónimo de Uniform Resource Locator. Localizador Uniforme de Recurso. Es el sistema de direcciones en Internet.
- **W3C**: W3C es la abreviación de World Wide Web Consortium
- **WAI**: Web Accessibility Initiative
- **WAI-ARIA**: Accessible Rich Internet Applications
- **WCAG 1.0**: Web Content Accessibility Guidelines
- **WCAG 2.0:** Web Content Accessibility Guidelines
- **XHTML:** Siglas del ingles eXtensible HyperText Markup Language. XHTML es basicamente HTML expresado como XML valido.
- **XML:** eXtensible Markup Language. Lenguaje Extensible de Marcado.
- **XUL:** eXtensible User-interface Language. Un lenguaje de marcación similar a HTML y basado en XML.

Referencias

- **[Cra+06]** Crane, David, Eric Pascarello y Darren James. ***Ajax in Action***. Version 1. Manning Publications. ISBN: I-932394-61-3. 2006.
- **[Gro06]** Gross, Christian. Ajax ***Patterns and Best Practices***. Apress. ISBN: 978-1-59059-616-6. 2006
- **[Asl+06]** Asleson, Ryan ***,*** Nathaniel T.Schutta***. Foundations of Ajax .*** Apress. ISBN: 1-59059-582-3. 2006
- **[Hol06]** Holzner, Steve, ***Ajax for Dummies***.Wiley Publishing. ISBN: .2006
- **[Per06]** Perry, Bruce W. ***Ajax Hacks.*** O'Reilly. ISBN: .2006
- **[Jac06]** Jacobs, Sas***. Beginning XML with DOM and Ajax: From Novice to Professional***. Apress. ISBN: .2006
- **[Mah06].** Mahemoff, Michael. ***Ajax Design Patterns***. O´Reilly. ISBN: .2006
- **[Zak+06]** Zakas,Nicholas C. Jeremy McPeak y Joe Fawcett***. Professional Ajax***. Wrox Press. ISBN: .2006
- **[Fuc08]** Fuchs, Thomas. ***Script.aculo.us*****.** [Internet]: <http://script.aculo.us/> [3 Enero 2008]
- **[Jam05]** James Garrett, Jesse . ***Ajax: A New Approach to Web Applications*****.** [Internet]: <http://www.adaptivepath.com/ideas/essays/archives/000385.php> [18 Febrero 2005]
- **[Pro06]** Prototype Core Team. ***Prototype JavaScript framework: Easy Ajax and DOM manipulation for dynamic web applications***. [Internet]: <http://prototypejs.org/> [2006]
- **[JSO98]** JSON. ***JSON***. [Internet]: <http://www.json.org/> [1998]
- **[RIC06]** RICO. ***RICO***. [Internet]: <http://openrico.org/> [2006]
- **[Her06]** Herrington, Jack D. ***Ajax and XML: Ajax for lightboxes***. [Internet]: <http://www.ibm.com/developerworks/xml/library/x-ajaxxml6/?ca=dgr-btw15AjaxLightbox> [2007]
- **[Zak05]** Zakas, Nicholas C. ***Professional JavaScript for Web Developer***. Wrox Press. ISBN: 978-0-7645-7908-0 .[2005]
- **[Fla06]** Flanagan, David. ***JavaScript: The Definitive Guide***. O'Reilly. ISBN:

9780596101992 . [2006]

- **[Hol05]** Holzschlag, Molly E. ***HTML and CSS***. Addison Wesley Professional. ISBN: 0-13-185586-7 . [2005]
- **[Mey00]** Meyer, Eric A. ***Cascading Style Sheets: The Definitive Guide***. O'Reilly . ISBN: 1-56592-622-6 . [2000]
- **[She+05]** Shea, Dave, Molly E. Holzschlag. ***The Zen of CSS Design: Visual Enlightenment for the Web***. Peachpit Press. ISBN: 1-56592-622-6 . [2005]
- **[Sch03]** Schengili-Roberts, Keith. ***Core CSS: Cascading Style Sheets, 2nd Edition***. Prentice Hall. ISBN: 0-13-009278-9 . [2003]
- **[Ced05]** Cederholm, Dan. ***Bulletproof Web Design: Improving flexibility and protecting against worst-case scenarios with XHTML and CSS.*** Peachpit Press. ISBN: 0-321-34693-9 . [2005]
- **[Iss04]** Issi Cohen, Lazaro , Joseph Issi Cohen . ***The Web Programmer's Desk Reference.*** No Starch Press. ISBN: 1593270119 . [2004]
- **[Qui03]** Quigley, Ellie**.** ***JavaScript by Example.*** Prentice Hall. ISBN: 0-13-140162-9. [2003]