



UNIVERSITAT
POLITÀCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de aplicación cliente web para responder cuestionarios generados mediante SAKAI

Proyecto Final de Carrera

Ingeniería Informática

Autor: Fco Javier Calás Blasco
Director: David de Andrés Martínez

Fecha: 01/09/2012

Índice de contenido

1. Resumen	5
2. Presentación.....	6
3. El entorno de e-learning SAKAI.....	9
3.1. Introducción.....	9
3.2. Instalación del entorno Sakai	12
3.3. Configuración básica de Sakai.....	16
3.4. La herramienta de encuestas (POLL).....	19
3.5. La herramienta de exámenes (SAMIGO).....	22
3.6. Conexión con sistemas remotos.....	28
3.6.1. Web services SOAP.....	29
3.6.2. Servicios REST. Entity Broker.....	32
4. Frameworks web para el desarrollo de aplicaciones móviles. Estudio previo.....	36
4.1. LungoJS.....	36
4.2. JQuery Mobile.....	39
5. Diseño de la aplicación de cuestionarios	45
5.1. Conexión con servicios remotos en SAKAI.....	45
5.2. Especificación de requisitos.....	50
5.2.1. General.....	50
5.2.2. Autenticación y acceso.....	51
5.2.3. Gestión de encuestas.....	52
5.2.4. Gestión de cuestionarios/exámenes.....	53
5.3. Prototipos desarrollados.....	53
5.3.1. LungoJS.....	54
5.3.2. JQuery Mobile.....	56
5.4. Diseño de interfaces externas.....	59
6. Implementación.....	63
6.1. Proxy de conexión con SAKAI.....	63
6.1.1. El problema de las llamadas cross-domain.....	63
6.1.2. Tecnologías utilizadas.....	65
6.1.3. Servicios de login.....	69
6.1.4. Servicio de lista de encuestas.....	71
6.1.5. Servicio de encuesta.....	72
6.1.6. Servicio de cuestionario.....	74
6.2. Aplicación para terminales móviles JQuery Mobile.....	75
6.2.1. Plugins JQuery Mobile que se van a utilizar.....	76
6.2.2. Estructura de la aplicación móvil	79
6.2.3. Pantalla de Login.....	84
6.2.4. Pantalla de menú.....	88
6.2.5. Lista de encuestas.....	90
6.2.6. Pantalla de encuesta.....	93
6.2.7. Pantalla de cuestionarios/exámenes.....	95
6.2.8. Despliegue de la aplicación en el dispositivo móvil.....	96
7. Conclusiones.....	99
8. Bibliografía.....	102

Índice de ilustraciones

Ilustración 1: Consola de Tomcat con Sakai arrancado.....	15
Ilustración 2: Pantalla inicial de Sakai.....	15
Ilustración 3: Pantalla principal de Sakai tras la identificación.....	16
Ilustración 4: Formulario para dar de alta un usuario en Sakai.....	17
Ilustración 5: Editor de sitios de Sakai.....	17
Ilustración 6: Adminstrador de páginas de un sitio en Sakai.....	18
Ilustración 7: Pantalla de añadir nueva herramienta en una página de Sakai.....	18
Ilustración 8: Pantalla del nuevo sitio creado en Sakai.....	19
Ilustración 9: Pantalla de creación de encuestas mediante Sakai.Poll.....	20
Ilustración 10: Formulario para crear opciones en una encuesta de Sakai.....	21
Ilustración 11: Página con la encuesta definida en Sakai.....	21
Ilustración 12: Pantalla de gestión de exámenes de la herramienta Sakai.SAMIGO.....	22
Ilustración 13: Selección de tipo de pregunta para un examen en Sakai.SAMIGO.....	23
Ilustración 14: Pregunta tipo Opción múltiple en Sakai.SAMIGO.....	24
Ilustración 15: Pregunta tipo Encuesta en Sakai.SAMIGO.....	25
Ilustración 16: Pregunta tipo Respuesta corta en Sakai.SAMIGO.....	25
Ilustración 17: Pregunta tipo Rellenar espacios en blanco en Sakai.SAMIGO.....	26
Ilustración 18: Pregunta tipo Respuesta numérico en Sakai.SAMIGO.....	26
Ilustración 19: Pregunta tipo Relacionar en Sakai.SAMIGO.....	27
Ilustración 20: Pregunta tipo Verdadero/false en Sakai.SAMIGO.....	27
Ilustración 21: Pregunta tipo Grabación de audio en Sakai.SAMIGO.....	28
Ilustración 22: Pregunta tipo Subir ficheros en Sakai.SAMIGO.....	28
Ilustración 23: Ejemplo de visualización de un programa en LungoJS.....	39
Ilustración 24: Ejemplo de programa Hola Mundo en JQuery Mobile.....	43
Ilustración 25: Ejemplo de lista en JQuery Mobile.....	44
Ilustración 26: Prototipo LungoJS. Pantalla de login y lista de encuestas.....	55
Ilustración 27: Prototipo LungoJS. Tipos de preguntas.....	55
Ilustración 28: Prototipo LungoJS. Lista de cuestiones.....	56
Ilustración 29: Prototipo JQueryMobile. Login y lista de encuestas.....	57
Ilustración 30: Prototipo JQuery Mobile. Tipos de preguntas.....	58
Ilustración 31: Prototipo JQueryMobile. Lista de preguntas.....	58
Ilustración 32: Mockup con la pantalla de Login.....	60
Ilustración 33: Pantalla con el menú tras el login.....	60
Ilustración 34: Popup de opciones de una encuesta.....	61
Ilustración 35: Pantalla con la lista de encuestas.....	61
Ilustración 36: Mockup con pantalla de encuestas con checkboxes.....	62
Ilustración 37: Mockup con pantalla de encuestas con botones de radio.....	62
Ilustración 38: Configuración de proyecto en Eclipse para el proxy de Sakai.....	66
Ilustración 39: Pantalla de creación de proyecto web estático en Eclipse.....	76
Ilustración 40: Ejemplos de mensajes generados mediante ToastMessage.....	77
Ilustración 41: Ejemplos de gráficos generados mediante JQPlot.....	79
Ilustración 42: Estructura de la aplicación Sakai Encuestas.....	80
Ilustración 43: Sakai Encuestas. Pantalla de login.....	86
Ilustración 44: Sakai Encuestas. Pantalla de menú.....	89

<u>Desarrollo de aplicación cliente web para responder cuestionarios generados mediante SAKAI</u>	<u>4</u>
Ilustración 45: SakaiEncuestas. Pantallas de encuestas con una o varias opciones.....	94
Ilustración 46: SakaiEncuestas. Tipos de gráficos.....	95

1. Resumen

En este Proyecto Fin de Carrera vamos a construir una herramienta ejecutable en un dispositivo móvil (smartphone, tablet...) que utilice el entorno de Sakai para conseguir recuperar y votar las encuestas a las que un usuario tiene acceso, de forma que emulemos un entorno ARS (Sistema de Respuesta de la Audiencia) con un coste significativamente menor.

Después de una breve introducción, realizaremos una breve descripción del entorno Sakai, su instalación y configuración básica. Crearemos entonces un sitio en Sakai (que es el elemento que se utiliza para definir una asignatura) y le añadiremos las herramientas disponibles para la generación de encuestas y exámenes. A continuación, explicaremos dichas herramientas y las utilizaremos para definir los ejemplos que vamos a utilizar.

Posteriormente, realizaremos un breve estudio de las opciones que tenemos para la generación de la aplicación móvil, centrándonos en las opciones HTML5 para poder abarcar el mayor número de dispositivos posibles. De entre las opciones disponibles, profundizaremos en los frameworks LungoJS y JQuery Mobile.

A continuación, realizaremos el diseño de la aplicación. Estudiaremos las formas en las que una aplicación móvil se puede conectar al entorno de Sakai (mediante servicios web) y decidiremos la mejor estrategia para acceder, donde nos daremos cuenta de que precisamos realizar una aplicación que actúe de proxy entre la aplicación móvil y Sakai.

Continuaremos enumerando los requisitos que deberemos implementar, y preparemos unos prototipos con los dos frameworks móviles para decidir la opción final a utilizar. También prepararemos unos mockups con el diseño de las interfaces externas que deberá tener la aplicación.

Por último lugar, detallaremos la implementación que se ha realizado, tanto de la aplicación web para dispositivos móviles, como del proxy de conexión a Sakai. También explicaremos las distintas posibilidades que existen para instalar la aplicación en el entorno móvil y conseguir utilizarla fuera de línea.

Palabras clave: ARS, E-Learning, Poliformat, Sakai, proxy, mockup, HTML5, LungoJS, JQueryMobile, servicios web, dispositivos móviles,

2. Presentación

En el ámbito de la educación, sobre todo en el entorno universitario, es deseable que los profesores puedan recibir feedback de los alumnos. Para ello, existen los denominados Sistemas de Respuesta de la Audiencia (ARS), que se pueden explicar como un sistema basado en un ordenador en el lado del profesor, que se encarga de proyectar el contenido que el docente quiere utilizar en su exposición, mientras que permite cierta interactividad por parte de la audiencia, mediante la respuesta de preguntas sencillas, del tipo de selección de opciones, respuestas cortas....

Para conseguir esta interactividad, estos sistemas utilizan típicamente de un hardware especial (llamado *clicker*) que se pone a disposición de los participantes. El uso de este hardware supone que para sacar todo el rendimiento a estos sistemas pueda desembocar en un desembolso económico importante, sobre todo al añadirle el propio del sistema ARS no relacionado directamente con el hardware. Proponer la adopción de este sistema en un entorno universitario como puede ser la UPV, con la cantidad de aulas que existen, es sencillamente inviable económicamente.

A pesar de ello, estos sistemas tienen unos beneficios muy importantes en la docencia, tanto para los participantes como para los profesores que vayan a impartir la materia:

- Mejoran la atención de los alumnos, puesto que la escucha pasiva hace que le interés y la atención se pierda de manera muy rápida
- Mejoran la participación activa de los participantes, al deber estar pendiente de contestar las preguntas que se realizarán
- Permiten la participación anónima de los participantes. Hay ocasiones que por vergüenza o por miedo a la equivocación los alumnos no participan en la clase, mientras que si se dispone de un sistema que les garantice el anonimato sí que lo harían. Además, permite hacer un tipo de preguntas más sensibles que podrían provocar conflicto en el caso de no garantizarse el anonimato
- Permiten comprobar el nivel de atención y comprensión de los participantes, puesto que las respuestas a las preguntas realizadas estarán disponibles para el profesor para que pueda hacer el análisis que precise.
- Permite obtener de manera sistemática información como el número de asistentes, realizar el seguimiento de alumnos de manera individualizada...
- Crea un entorno de aprendizaje más atractivo y ameno para los participantes, y por lo tanto la experiencia educativa será más satisfactoria

Por lo tanto, vamos a buscar otras alternativas para conseguir estos beneficios, sin la necesidad de adoptar un caro entorno ARS. Además, vamos a intentar integrar este sistema con el sistema de LMS (*Learning Management System*, Sistema de Gestión de Aprendizaje) que se dispone en la UPV. Dicho sistema de E-Learning, llamado Poliformat, que está basado en uno de los proyectos de software libre sobre sistemas de e-learning más utilizados y potentes, el proyecto Sakai.

Este entorno, además de disponer de herramientas para la gestión de los distintos participantes, cursos, materias y asignaturas que conforman una institución educativa,

dispone de otras destinadas al apoyo de la enseñanza y a la generación de contenidos por parte tanto de docentes como de participantes. Por ejemplo, Sakai dispone de herramientas que se puede utilizar para crear encuestas y cuestionarios/exámenes para ser respondidos por los alumnos que estén matriculados.

Por otra parte, debido al gran auge del uso de los teléfonos inteligentes o smartphones, sobre todo en el rango de edad de los estudiantes universitarios, podrían ser un buen sustituto del *clicker* utilizado en los sistemas ARS. Debido a que cada vez más personas utilizan este tipo de tecnologías (sobre todo teniendo en mente que la gente joven suele ser la más entusiasta con las nuevas tecnologías) llega un momento en el que se puede presuponer que prácticamente la totalidad de los participantes dispongan de un dispositivo que pueda utilizarse como parte de la infraestructura de la clase. Esta aproximación ya se utiliza en muchas empresas como parte de las políticas BYOD (*Bring Your Own Device*), que proporciona un gran número de beneficios:

- Permite facilitar el trabajo fuera del entorno laboral o educativo, en nuestro caso
- Reducción del coste de mantenimiento de los dispositivos, pues cada usuario se encarga de su propio dispositivo (del que además es propietario)
- Reducción del coste de formación en el uso de los dispositivos
- Facilita la gestión de la gran proliferación de diferentes dispositivos

Si conseguimos construir el software apropiado, tanto para la confección del servidor como de la aplicación móvil cliente, podemos utilizar esta aproximación para emular el sistema ARS con la infraestructura que ya está disponible en la universidad. Para ello, utilizaremos las posibilidades que nos brinda el entorno Sakai para crear y gestionar los contenidos (en nuestro caso, la creación de encuestas y exámenes, que utilizaremos en realidad para emular un cuestionario).

Dicho software tiene que ser compatible con la gran mayoría de dispositivos móviles y smartphones que existen en el mercado, puesto que gran parte del éxito del desarrollo estriba en que la mayor parte de participantes puedan utilizar sus propios terminales para participar en la aplicación. Debemos tener en cuenta la gran cantidad de entornos y tecnologías existentes: iOS (Apple), Android (con una cantidad de fabricantes enorme), BlackBerry, Windows Phone, Symbian... y el ecosistema de dispositivos no hace más que crecer, además de convivir con otro tipo de dispositivos que también son susceptibles de ser utilizados para nuestros fines. Este es el ejemplo de las tabletas, en el que existe un gran mercado en continuo crecimiento por parte de fabricantes como Apple, Microsoft, Samsung, HTC... e incluso Amazon (con el Kindle Fire).

Cada uno de estos sistemas permiten programar aplicaciones de forma nativa. Por ejemplo, los dispositivos de Apple utilizan un entorno de programación basado en el lenguaje Objective-C, mientras que otros dispositivos como los Android o Blackberry disponen de potentes frameworks basados en Java, aunque incompatibles entre sí. Por lo tanto, si queremos realizar un desarrollo para estos dispositivos de forma nativa tendremos que modificar sustancialmente el código dependiendo del dispositivo, con el consiguiente sobreesfuerzo en la implementación y el problema del mantenimiento, puesto que para hacer cualquier modificación en el código, o para añadir cualquier funcionalidad que se desee, se deberá implementar tantas veces como entornos distintos queramos mantener.

Existe otra aproximación que nos evitará tener que realizar diferentes implementaciones para los distintos entornos a los que queremos dar soporte, y es utilizar alguno de los frameworks disponibles para realizar aplicaciones HTML5 preparadas específicamente para móviles. Estos entornos se basan en javascript para dar formato a una página web, que se ejecuta mediante el navegador web que se disponga en el dispositivo. Gracias a las nuevas capacidades de HTML5 y CSS3, junto con toda la funcionalidad que nos proporciona javascript en cuanto al manejo del DOM de la página, hace que sean una buena alternativa a las aplicaciones nativas. Las ventajas que obtenemos con esta tecnología son variadas:

- Permite un sólo desarrollo para los dispositivos soportados por el framework
- Se utiliza HTML y javascript, que son dos tecnologías muy asentadas y conocidas.
- La implementación es sencilla, puesto que se apoyan en herramientas y frameworks que simplifican el desarrollo.
- Es una tecnología en constante expansión y mejora, tanto en la parte de los frameworks, que cada vez ofrecen más posibilidades, como por parte de los dispositivos, con un mejor aprovechamiento de las tecnologías descritas en el estándar HTML5
- No es necesario pasar por los sistemas de gestión de aplicaciones propios de cada entorno, como el Android Market (actual Google Play) o el App Store de Apple
- Permite la conexión con sistemas heterogéneos mediante servicios web, gracias al amplio soporte que ofrecen los frameworks al tratamiento mediante AJAX.

Esta tecnología también tiene inconvenientes: muchas de las capacidades de los móviles más modernos no pueden ser aprovechados por las aplicaciones web (como por ejemplo el GPS, el acelerómetro, la cámara de fotos...) y el rendimiento general de la aplicación es bastante mejor en el caso de las aplicaciones nativas, al estar el dispositivo optimizado para ejecutar este tipo de aplicaciones. Sin embargo, la aplicación que vamos a desarrollar no va a hacer uso de estas características avanzadas. Además, su tamaño no va a ser grande (al menos lo que va a tener cargado en un momento preciso el navegador dentro del DOM), por lo que para nuestros propósitos es el sistema idóneo que precisamos.

3. El entorno de e-learning SAKAI

3.1. Introducción

Sakai CLE es un desarrollo de e-learning desarrollado en código abierto, que representa un Entorno de Colaboración y Aprendizaje (Collaboration and Learning Environment), que se dirige principalmente a estudios superiores (universidades, institutos tecnológicos, centros de investigación...). Pretende competir con otros desarrollos similares comerciales (Blackboard, WebCT...) y mejorar las capacidades de otros sistemas en código abierto (Moodle). Contiene la base tecnológica y las herramientas necesarias para administrar cursos y sus usuarios, tanto alumnos como profesores, investigadores, administradores...

Incluye un gran número de herramientas de apoyo a la enseñanza, que permiten generar sitios en donde configurar, cursos proyectos o portfolios electrónicos. Dentro de ese curso, se incluirán las herramientas que se deseen, que se pueden utilizar para crear un espacio para la comunicación entre participantes (blogs, chats, forums...), trabajos colaborativos (wikis, compartición de materiales...), recibir feedback (encuestas), mandar mensajes, (avisos, correo electrónico), realizar exámenes y calificaciones... y muchas más necesidades, que están cubiertas por las herramientas que proporciona el desarrollo, o por otras desarrolladas por la comunidad que tiene detrás y que mantiene la Fundación Sakai.

El proyecto nació a finales de 2003, cuando cuatro universidades americanas se unieron para crear un desarrollo común para un entorno de aprendizaje virtual en código libre. Las universidades que colaboraron en un primer momento fueron la Universidad de Michigan, Universidad de Indiana, Universidad de Standford y el Instituto Tecnológico de Massachusetts (MIT). Posteriormente la Fundación Andrew W. Mellon contribuyó económicament con el proyecto.

Se implementó sobre una base ya desarrollada por la Universidad de Michigan, y en junio de 2004 se liberó la versión 1.0 de Sakai, que fue adoptada en un primer momento por las universidades de Michigan e Indicana, entre el final de 2004 y el año 2005.

Para mantener un proyecto de estas características, se tenía como objetivo la creación de una comunidad para que pudiera mantenerlo por sí mismo. Para ello, además de la propia arquitectura libre del sistema, se puso en marcha la Fundación Sakai a finales de 2005, que en un primer momento se encargó de que todos los desarrollos y los recursos generados por las distintas organizaciones transfirieran sus derechos de copyright a la Fundación. Ésta se encargaría de la liberación de versiones en como software libre. Actualmente se encarga de coordinar el ciclo de liberación de versiones asegurando la calidad del producto, organiza conferencias y otras iniciativas, mantiene espacios y estimula a la amplia comunidad de desarrollo que posee, para que pueda comunicarse y organizarse, junto con muchas otras labores.

A la fundación Sakai pertenecen más de 100 organismos, a la que pertenecen más de un centenar de organizaciones entre universidades privadas, públicas y militares, colegios técnicos, instituciones online, organizaciones de investigación... incluso partidos políticos (como la Democratic Alliance of South Africa) Por el número de cursos y usuarios, algunas instituciones destacadas son las siguientes:

- Metropolitan autonom University
- Indiana University
- University of Michigan
- Yale University
- Stanford University
- Universitat de Lleida
- Universidad Politécnica de Valencia
- Universidad del Valle de Guatemala
- Universidad Complutense de Madrid
- Universidad Católica San Antonio de Murcia
- Universidad de Murcia
- Universidad Pública de Navarra
- Instituto Tecnológico de Buenos Aires

La lista completa de miembros de la Fundación se puede consultar en la dirección <http://www.sakaiproject.org/organization-list>

A continuación se explican algunos de los grandes objetivos de Sakai para la ayuda de instituciones de enseñanza e investigación:

- **Gestión del aprendizaje:** Como indican sus siglas (Collaborative Learning Environment), el objetivo de Sakai es proveer un entorno para enseñanza de forma colaborativa, y para ello se apoya de las herramientas que tiene incluidas.
- **Colaboración en investigación:** Gracias a las herramientas para trabajo compartido y de colaboración, Sakai está siendo muy utilizado para utilizarlo como el sistema con el que se gestionan los flujos de información, y para la comunicación de los distintos investigadores y grupos de trabajo. Al poder utilizar el mismo entorno para gestionar el trabajo investigador y docente, se reduce la carga administrativa que tienen que soportar, ganando su labor en productividad.
- **Colaboración en proyectos:** También es posible utilizar Sakai para gestionar la colaboración en proyectos que no necesariamente estén dedicados a la docencia o investigación. En las instituciones en las que está implantado Sakai, lo utilizan para la colaboración en diversos proyectos por parte de grupos de estudiantes, comités facultativos, comités de tesis, planificación estratégica...
- **Gestión de portfolios (cuadernos electrónicos):** Gracias a Sakai y a sus herramientas para crear y administrar portfolios, cada usuario puede disponer de cuadernos electrónicos. Esta herramienta es muy versátil y tiene una gran variedad de usos, a todos los niveles de la organización:
 - Para los estudiantes, es un modo ideal para disponer en un lugar en el que se disponen de todos los materiales de aprendizaje, y los logros del usuario en el

proceso. Se puede también publicar el portfolio para el grupo de usuarios que se desee, de forma que se convierte en la mejor herramienta para que los estudiantes puedan exhibir su trabajo.

- Para los profesores, es un buen punto de partida para poder orientar a los alumnos en los objetivos que se persiguen. Permite a los educadores dar soporte a la evaluación, pudiendo revisar los portfolios que publiquen los alumnos y permitiendo una evaluación tanto formal, como alguna indicación más informal.
- A nivel de organización, también proporciona una valiosa información para poder evaluar la metodología para el aprendizaje, comprobando el cumplimiento de objetivos por parte de los alumnos. Además, proporciona la información necesaria para poder analizar los resultados, y poder de este modo afinar en la metodología con todo el feedback que se recibe.

El sistema Sakai CLE dispone de un conjunto de herramientas por defecto, que se utiliza en prácticamente todas las instituciones y universidades que utilizan Sakai. De todas formas, existen multitud de herramientas adicionales desarrolladas por la comunidad y marcadas como “contrib”, que pueden ser adoptadas por la institución que lo crea conveniente. La lista siguiente enumera algunas herramientas incluidas en el conjunto básico:

- **Announcements:** Permite introducir anuncios, es decir, información crítica en el tiempo (avisos de administración, citas...).
- **Assignments:** Crear y calificar tareas, tanto en línea como fuera de línea.
- **Blog:** Permite utilizar capacidades blogueras a la clase.
- **Calendar:** Mantener plazos, actividades y eventos relacionados con el sitio.
- **Chat:** Participar en conversaciones en tiempo real con los participantes del sitio.
- **Discussion Forum:** Crear, moderar y gestionar los temas de discusión y grupos dentro de un curso y enviar mensajes privados a los participantes del sitio.
- **Drop Box:** Compartir archivos de forma privada con los participantes del sitio
- **Email Archive:** Acceso a un archivo de correos electrónicos mandados por los participantes del sitio.
- **Glossary:** Mantiene definiciones y ayuda contextual para los términos utilizados en el sitio.
- **Gradebook:** Calcula, almacena y distribuye la información de grado a los estudiantes.
- **News:** Muestra noticias personalizadas de fuentes dinámicas y en línea, via ficheros RSS.
- **Profile 2:** Crea un perfil y lo conecta con otros utilizando un modelo de red social.
- **Resources:** Publica, almacena y organiza los materiales referentes en el sitio.

- **Site Roster:** Muestra una lista de los participantes del sitio, junto con una fotografía.
- **Site Stats:** Muestra las estadísticas de uso del sitio, como por ejemplo visitas de usuarios, actividad de las herramientas, y actividad de los recursos puestos a disposición.
- **Syllabus:** Muestra un resumen de los requerimientos del curso.
- **Tests & Quizzes:** Crea y organiza exámenes en línea.
- **Web Page:** Muestra páginas web externas.
- **Wiki:** Creación y edición de contenido web de forma colaborativa.

Para el ámbito que nos ocupa, de entre todas las herramientas que dispone Sakai, nos vamos a centrar en dos:

- La herramienta de encuestas (POLL): Es una herramienta bastante básica, que permite definir y publicar encuestas, y recoger los resultados, que normalmente son en forma de porcentaje de usuarios que han contestado la misma opción. Sólo permite utilizar un tipo de respuesta, el de selección entre varias posibles opciones.
- La herramienta de exámenes (Test & quizzes: SAMIGO): Permite definir y publicar exámenes que serán contestados por los usuarios que tengan permiso. Existe un gran número de tipos de respuesta que se pueden utilizar: respuesta corta, numérica, completar espacios, opción múltiple...

3.2. *Instalación del entorno Sakai*

Lo primero que se necesita para poder instalar Sakai es conseguir los binarios o fuentes con los que se quiere trabajar. Existen tres tipos de instalaciones, cada una enfocada a unas necesidades distintas:

- **Versión Demo:** Esta versión contiene todo lo necesario para instalar y ejecutar Sakai sin necesidad de otros sistemas distintos, como servidor de aplicaciones web ni sistema gestor de base de datos. En su interior, además de la compilación de Sakai, dispone de un servidor Apache Tomcat preconfigurado con una configuración básica, preparado para que se pueda realizar una evaluación rápida de las capacidades del sistema.
Como contra, esta versión no está preparada para soportar un gran número de cursos ni participantes, y no se tiene control ni sobre el sistema gestor de base de datos ni sobre el servidor de aplicaciones, por lo que no se puede afinar su rendimiento, ni modificar los parámetros de despliegue (para añadirlo a un pool de servidores o un cluster de datos, por ejemplo).
- **Desplegar desde ficheros binarios:** Esta versión dispone de los ficheros binarios de Sakai pero no dispone de la preconfiguración de Tomcat, ni dependencias o ficheros de configuración extra. Por lo tanto, es la versión indicada si se quiere desplegar Sakai en una infraestructura propia, pero no se van a realizar cambios en el código fuente.
Como desventaja, obviamente es más complicado de instalar que la versión Demo,

puesto que hay que hacerse cargo de la configuración del servidor de aplicaciones y el sistema gestor de base de datos, a cambio de proporcionar mucha más flexibilidad al sistema.

- **Desplegar desde el código fuente:** Esta versión proporciona el código fuente que se utiliza para generar las versiones binaria y Demo, sin ningún tipo de configuración adicional. Es por lo tanto la versión perfecta para desarrolladores, que a partir del código fuente pueden modificar (evidentemente si dispone de los conocimientos apropiados) la funcionalidad del sistema a su gusto, y personalizarlo de la manera que la organización considere oportuna. Posteriormente, con los fuentes modificados, se podrá generar los binarios necesarios para desplegar en los servidores de producción de la organización.
Esta es la versión más complicada de instalar, puesto que a las tareas de configuración de la instalación de ficheros binarios se le suman las configuraciones propias del entorno de desarrollo necesario para trabajar: IDE de desarrollo, Apache Maven para el despliegue, Subversion para el control de versiones..

En el proyecto que nos ocupa, la versión que vamos a elegir es la version Demo, al menos mientras no haya alguna dificultad que nos obligue a usar alguna de las otras dos. En principio, al utilizar servicios web que ya estarán definidos en el sistema, no vamos a necesitar modificar el código fuente. Además, queremos que la aplicación esté alojada en el teléfono móvil, o quizás en un servidor separado de la instalación de Sakai, con el fin de ser lo más independiente posible, por lo que tampoco necesitaremos administrar servidores de aplicaciones ni gestores de datos.

De todas formas, esto quedará supeditado a que los servicios web puedan ejecutarse en un dominio separado al que está instalado Sakai. Si no fuera así, es posible que en este caso se deba cambiar el código fuente (para añadir a los servicios REST soporte a JSONP o CORS: Cross-Origin Resource Sharing), o para instalar la aplicación móvil en el mismo servidor que Sakai, para lo que al menos necesitaríamos la versión binaria.

Para poder instalar el sistema, debemos tener en cuenta de que para poder ejecutar el sistema necesitaremos tener en nuestra máquina al menos dos gigabytes de memoria RAM. De no ser así se producirán problemas en el arranque al no poder reservarse la memoria necesaria para la ejecución.

Seguidamente debemos conseguir los archivos binarios o los fuentes, dependiendo del tipo de instalación que queramos realizar, distribuidos desde el proyecto Sakai (<http://sakaiproject.com>). A día de hoy, la última versión disponible es la 2.8.2. En la página <http://source.sakaiproject.org/release/2.8.2/> se pueden conseguir los ficheros necesarios para cualquiera de los tres tipos de instalaciones, para entornos Windows, Unix/Linux o Macintosh.

A continuación, es conveniente echar un vistazo a las guías de instalación disponibles en línea. Las guías para cualquiera de los tres tipos de instalación están alojados en las siguientes direcciones:

- Versión Demo: <https://confluence.sakaiproject.org/pages/viewpage.action?pageId=75106831>
- Versión Binaria: <https://confluence.sakaiproject.org/pages/viewpage.action?pageId=75106833>
- Código fuente: <https://confluence.sakaiproject.org/pages/viewpage.action?pageId=75106836>

La instalación de la versión Demo es muy sencilla. Antes de comenzar con la instalación, se debe comprobar que se dispone en la máquina de una versión de JDK (Java Development Kit) correcta. Se debe utilizar JDK en lugar de JRE (Java Runtime Environment), puesto que hay ciertas partes del sistema que necesitan compilarse antes de ser utilizados, como por ejemplo los archivos JSP (Java Server Pages) y los archivos JWS (Java Web Services).

En la guía de instalación se recomienda disponer de la versión Java SE 1.6, La versión 1.5 está desaconsejada, puesto que se ha terminado su ciclo de vida, aunque en principio se podría usar, aunque por motivos de seguridad se recomienda al menos utilizar la versión 1.5 update 18 o posterior. Para comprobar la versión de instalada en el sistema, se debe utilizar la siguiente orden en la línea de comandos:

```
java -version
```

Esto nos va a dar una salida en pantalla parecida a la siguiente, que es la que se consigue en nuestro entorno:

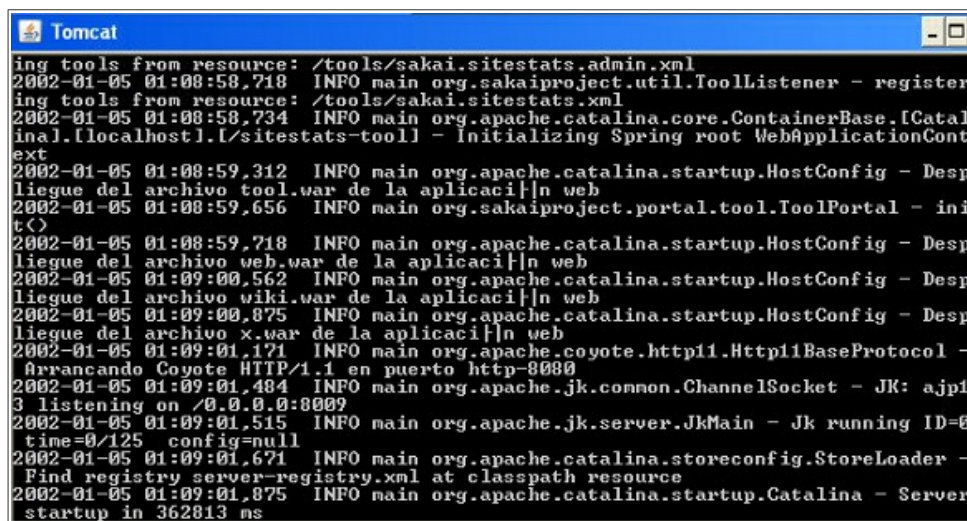
```
java version "1.7.0_04"  
Java(TM) SE Runtime Environment (build 1.7.0_04-b22)  
Java HotSpot(TM) Client VM (build 23.0-b21, mixed mode, sharing)
```

Esto significa que en nuestro sistema se encuentra instalada la versión 1.7 de JDK. A pesar de no ser la versión recomendada (esta es más moderna) vamos a utilizarla para la instalación, para comprobar así que en versiones posteriores a la 1.6 también se puede instalar Sakai sin problemas.

Es importante también comprobar que el sistema tiene configuradas correctamente las variables de entorno de Java. En concreto, la variable JAVA_HOME debe contener la ruta correcta en donde se encuentran alojada la instalación del JDK.

- Para indicar las variables de entorno en Windows, nos debemos dirigir al Panel de control – Sistema – Configuración avanzada del sistema – Variables de entorno
- Para indicarlas en un sistema Linux/Unix, se abre la línea de comandos y se utiliza la orden EXPORT.

Una vez comprobada y configurada la instalación del JDK, descomprimiremos el fichero con los binarios de Sakai, preferiblemente en una ruta que no contenga espacios, teniendo cuidado de que los directorios tengan permiso de escritura, puesto que de no ser así el arranque de Tomcat dará errores. Cuando se acabe la descompresión, sin ningún paso adicional ya tendremos preparada la instalación de la versión Demo.



```
ing tools from resource: /tools/sakai.sitestats.admin.xml
2002-01-05 01:08:58,718 INFO main org.sakaiproject.util.ToolListener - register
ing tools from resource: /tools/sakai.sitestats.xml
2002-01-05 01:08:58,734 INFO main org.apache.catalina.core.ContainerBase.[Catal
inal.[localhost].[/sitestats-tool] - Initializing Spring root WebApplicationCont
ext
2002-01-05 01:08:59,312 INFO main org.apache.catalina.startup.HostConfig - Desp
lliege del archivo tool.war de la aplicaci|n web
2002-01-05 01:08:59,656 INFO main org.sakaiproject.portal.tool.ToolPortal - ini
t()
2002-01-05 01:08:59,718 INFO main org.apache.catalina.startup.HostConfig - Desp
lliege del archivo web.war de la aplicaci|n web
2002-01-05 01:09:00,562 INFO main org.apache.catalina.startup.HostConfig - Desp
lliege del archivo wiki.war de la aplicaci|n web
2002-01-05 01:09:00,875 INFO main org.apache.catalina.startup.HostConfig - Desp
lliege del archivo x.war de la aplicaci|n web
2002-01-05 01:09:01,171 INFO main org.apache.coyote.http11.Http11BaseProtocol -
Arrancando Coyote HTTP/1.1 en puerto http-8080
2002-01-05 01:09:01,484 INFO main org.apache.jk.common.ChannelSocket - JK: ajp1
3 listening on /0.0.0.0:8009
2002-01-05 01:09:01,515 INFO main org.apache.jk.server.JkMain - Jk running ID=0
time=0/125 config=null
2002-01-05 01:09:01,671 INFO main org.apache.catalina.storeconfig.StoreLoader -
Find registry server-registry.xml at classpath resource
2002-01-05 01:09:01,875 INFO main org.apache.catalina.startup.Catalina - Server
startup in 362813 ms
```

Ilustración 1: Consola de Tomcat con Sakai arrancado

Para arrancar la aplicación, basta buscar el fichero start-sakai.bat (en Windows) o start-sakai.sh (en Linux/Unix/Mac). En este momento, se abrirá una ventana con la consola del servidor (Tomcat) en la que se puede seguir el arranque de la aplicación. Unos cuantos minutos después (la primera vez que se arranca la aplicaciónes más costosa que las siguientes, puesto que se tienen que inicializar varios sistemas y compilar archivos pendientes) en la consola se mostrará una salida similar al siguiente:

Esto significa que la aplicación ya ha arrancado. Para comprobarlo, nos dirigimos con el navegador web a la siguiente dirección:

<http://localhost:8080/portal>

Y deberemos encontrarnos con una pantalla similar a la siguiente:

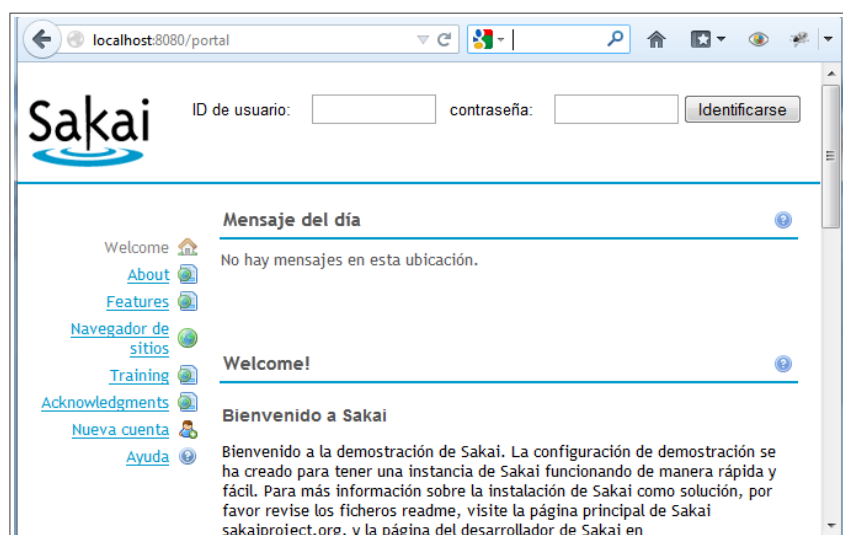


Ilustración 2: Pantalla inicial de Sakai

De esta manera hemos comprobado que la instalación de Sakai se ha producido sin problemas. Si queremos apagar la instancia que hemos ejecutado, tenemos que buscar y ejecutar los archivos stop-sakai.bat (en Windows) o stop-sakai.sh (en Linux), en el directorio de instalación de Sakai. Comprobaremos entonces en la consola de Tomcat que

se van destruyendo los componentes ejecutados hasta que la propia ventana con la consola se cierre.

3.3. Configuración básica de Sakai

Una vez hemos arrancado el entorno, se nos muestra una pantalla de bienvenida, en donde además de mostrarnos un menú con algunas opciones y mostrarnos información sobre el sistema, nos pide una clave de acceso y una contraseña. Sakai dispone por defecto de un usuario administrador. Para poder conectarnos con él, debemos indicar como nombre de usuario y como contraseña 'admin'. A continuación, se nos mostrará la pantalla inicial del usuario que se ha identificado en el sistema.

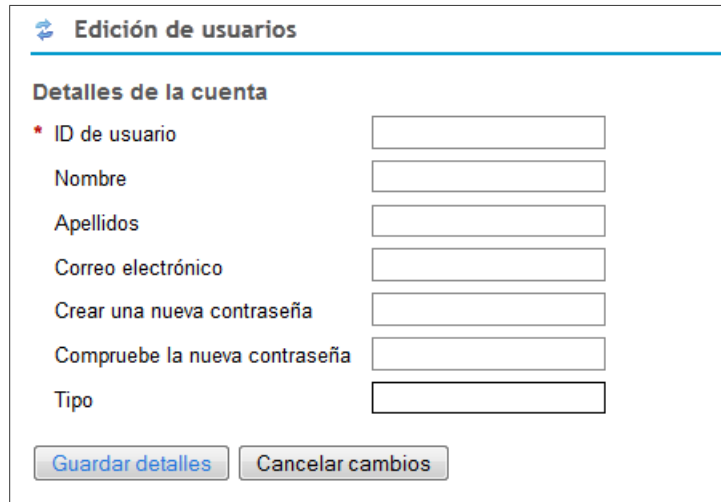


Ilustración 3: Pantalla principal de Sakai tras la identificación

La primera pestaña que se nos muestra desplegada es 'Mi sitio', que existe para todos los usuarios del sistema. Dependiendo del tipo de usuario que se conecte, se disponen de unas herramientas u otras. Además cada usuario puede personalizar su sitio con las herramientas y páginas que tenga disponible. En el caso del usuario Administrador, las herramientas que tiene disponibles en 'Mi sitio' por defecto son las mismas que las que dispone en la pestaña 'Administration Workspace' (la segunda pestaña que tiene disponible).

A partir de aquí, se pueden utilizar dichas herramientas para crear usuarios, crear y administrar sitios, añadir usuarios a sitios...

Para crear un usuario, nos dirigiremos a la opción de menú 'Edición de usuarios' y pulsamos en el enlace 'Nuevo usuario'. Se nos muestra entonces un formulario para indicar los datos del usuario: Identificador, nombre y apellidos, dirección de correo electrónico, contraseña y tipo de usuario.



Edición de usuarios

Detalles de la cuenta

* ID de usuario

Nombre

Apellidos

Correo electrónico

Crear una nueva contraseña

Compruebe la nueva contraseña

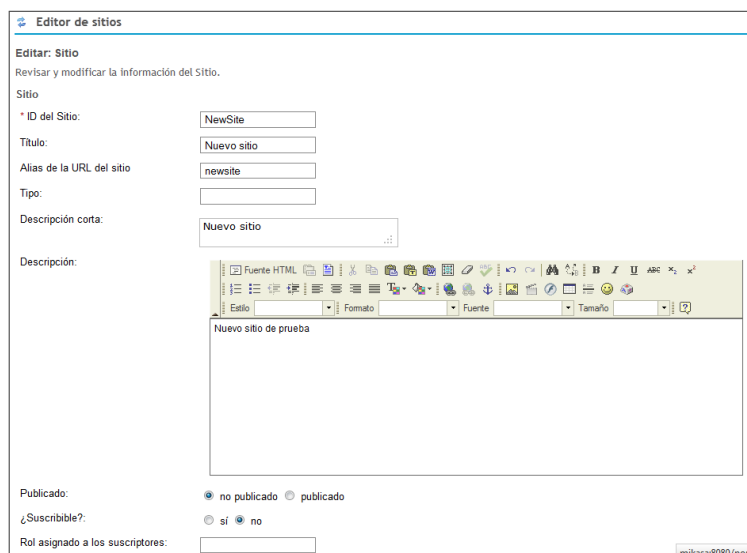
Tipo

Ilustración 4: Formulario para dar de alta un usuario en Sakai

A continuación vamos a crear un nuevo sitio, que es el elemento que se utiliza para definir y administrar un curso, proyecto o portafolio, dependiendo del tipo de sitio que se cree. Cada curso se compone de distintas páginas, para acceder a cada una de ellas se creará un enlace en el menú de la izquierda de la página principal del sitio. Dentro de cada página se pueden añadir distintas herramientas, que podrán utilizar los participantes del sitio.

Como ejemplo, crearemos un nuevo sitio con dos páginas, En la primera página indicaremos la herramienta de gestión de encuestas (sakai.poll), y en la segunda la de gestión de exámenes (sakai.samigo), que son las que vamos a utilizar en nuestro proyecto.

Lo primero que haremos será dirigirnos a la opción del 'Editor de sitios' y seleccionar el enlace 'Nuevo sitio'. Se mostrará una pantalla con un formulario para indicar los datos del nuevo sitio, como el identificador y título del sitio, el alias para la URL, tipo de sitio y descripciones (corta y larga), entre otros datos.



Editor de sitios

Editar: Sitio
Revisar y modificar la información del Sitio.

Sitio

* ID del Sitio:

Título:

Alias de la URL del sitio

Tipo:

Descripción corta:

Descripción:

Nuevo sitio de prueba

Publicado: no publicado publicado

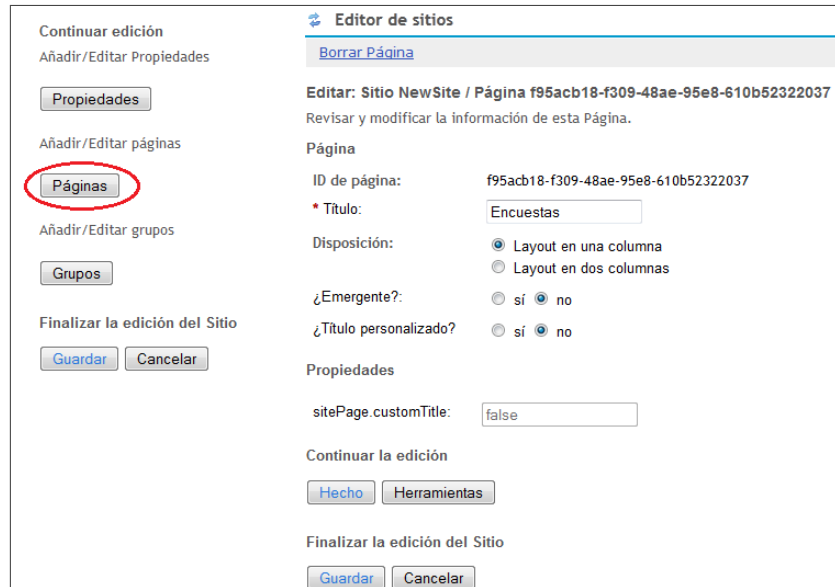
¿Suscribible?: sí no

Rol asignado a los suscriptores:

mikasa8080/porta

Ilustración 5: Editor de sitios de Sakai

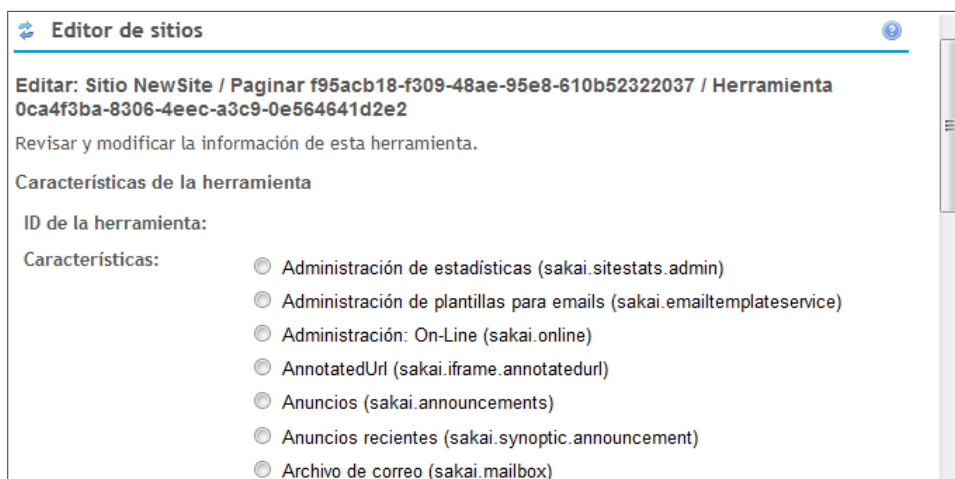
Para añadir páginas en un sitio, se debe seleccionar la opción de añadir o editar páginas, que se encuentra más abajo en el mismo programa. Después de pulsar en el enlace 'Nueva página', se accede al editor de páginas, donde se añadirá el título de la página y otras opciones. Además se puede indicar con el botón correspondiente las herramientas que va a contener la página.



The screenshot shows the 'Editor de sitios' interface. On the left, there are several sections: 'Continuar edición' with a 'Propiedades' button; 'Añadir/Editar Propiedades'; 'Añadir/Editar páginas' with a 'Páginas' button circled in red; and 'Añadir/Editar grupos' with a 'Grupos' button. Below these are 'Finalizar la edición del Sitio' with 'Guardar' and 'Cancelar' buttons. The main area on the right is titled 'Editor de sitios' and contains a 'Borrar Página' button. Below that, it shows the page ID 'f95acb18-f309-48ae-95e8-610b52322037' and the title 'Encuestas'. There are radio buttons for 'Disposición' (selected: 'Layout en una columna'), '¿Emergente?' (selected: 'no'), and '¿Título personalizado?' (selected: 'no'). At the bottom, there are 'Hecho' and 'Herramientas' buttons, and another set of 'Guardar' and 'Cancelar' buttons.

Ilustración 6: Administrador de páginas de un sitio en Sakai

Dentro de la edición de herramientas y seleccionando la opción 'Nueva herramienta', se muestra una lista con las herramientas disponibles en la instalación de Sakai:



The screenshot shows the 'Editor de sitios' interface for editing a tool. The title bar says 'Editor de sitios'. The main content area shows 'Editar: Sitio NewSite / Pagina f95acb18-f309-48ae-95e8-610b52322037 / Herramienta 0ca4f3ba-8306-4eec-a3c9-0e564641d2e2'. Below this is the text 'Revisar y modificar la información de esta herramienta.' and the section 'Características de la herramienta'. Under 'ID de la herramienta:', there is a 'Características:' label followed by a list of tools with radio buttons: 'Administración de estadísticas (sakai.sitestats.admin)', 'Administración de plantillas para emails (sakai.emailtemplateservice)', 'Administración: On-Line (sakai.online)', 'AnnotatedUrl (sakai.iframe.annotatedurl)', 'Anuncios (sakai.announcements)', 'Anuncios recientes (sakai.synoptic.announcement)', and 'Archivo de correo (sakai.mailbox)'.

Ilustración 7: Pantalla de añadir nueva herramienta en una página de Sakai

Para esta página, buscaremos la herramienta Encuestas (sakai.poll), y continuaremos la edición para crear otra página con la herramienta de Exámenes (sakai.samigo). En la imagen siguiente se muestra el resultado de la generación del nuevo sitio.

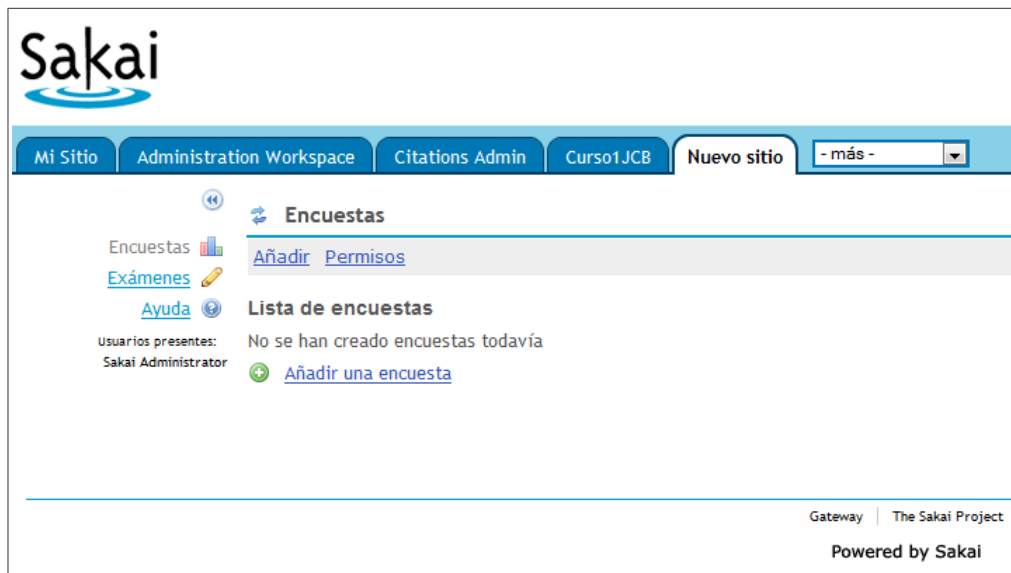


Ilustración 8: Pantalla del nuevo sitio creado en Sakai

3.4. La herramienta de encuestas (POLL)

Con la herramienta de encuestas se pueden definir sondeos sencillos, en los que se define una pregunta y se le añaden las opciones que los usuarios pueden seleccionar para responder.

Desde la pantalla inicial de la herramienta de encuestas, además de crear y administrar las encuestas generadas, se pueden gestionar los permisos que van a tener los usuarios que se conecten a las encuestas del sitio, dependiendo de su rol: access (acceso) o maintain (mantenimiento). Se pueden indicar permisos para:

- Votar
- Añadir encuestas
- Borrar las encuestas de las que sea el propietario
- Borrar cualquier encuesta
- Editar las encuestas de las que sea el propietario
- Editar cualquier encuesta

La pantalla de definición de encuestas tiene el aspecto que se muestra en la siguiente imagen

Encuestas

Añadir una encuesta

*Pregunta De todas las materias, ¿cuál te ha parecido más útil?

Instrucciones adicionales (si aplica)

Fuente HTML

Estilo Formato Fuente Tamaño

*Fecha de apertura 5/01/02 4:39

*Fecha de cierre 12/01/15 4:39

Límites

*¿Cuál es el número mínimo de opciones de respuesta que pueden seleccionarse? 1

*¿Cuál es el número máximo de opciones de respuesta que pueden seleccionarse? 1

Opciones de acceso:

Público - esta encuesta será visible al público general y los usuarios anónimos podrán votar

Los resultados son visibles:

siempre

a los participantes que han votado

tras la fecha de cierre

nunca

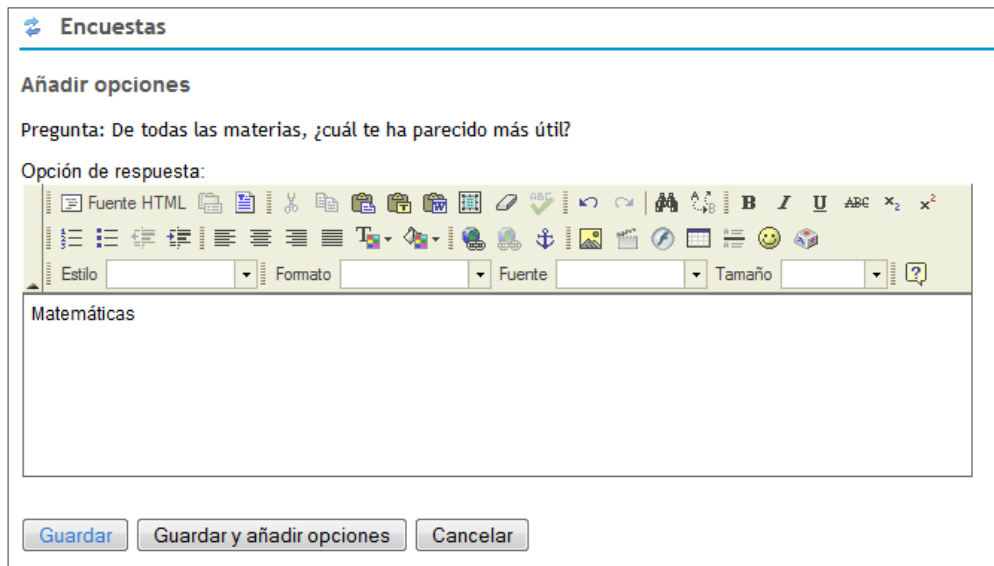
Guardar y añadir opciones Cancelar

Ilustración 9: Pantalla de creación de encuestas mediante Sakai.Poll

Las encuestas tienen una serie de atributos que se pueden configurar, además del texto y las opciones de la encuesta:

- La fecha de inicio y fin entre las que se puede contestar en la encuesta
- El número de opciones mínimo y máximo que se pueden seleccionar para contestar.
- Permite indicar si los usuarios anónimos pueden votar en la encuesta
- Permite indicar la visibilidad de los resultados, a elegir entre las siguientes opciones:
 - Los resultados son visibles siempre
 - Los resultados son visibles sólo para los participantes que han votado
 - Los resultados son visibles sólo tras la fecha de cierre de la encuesta
 - Los resultados no son visibles nunca.

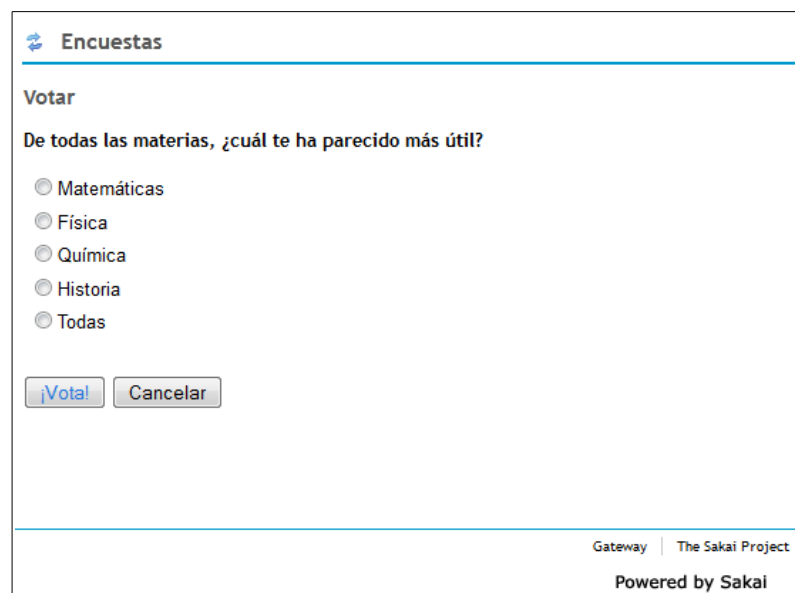
Para añadir las opciones de la encuesta se dispone de un editor de texto, como se muestra en la imagen siguiente. Al pulsar el botón 'Guardar y añadir opciones' se guarda la opción que se ha indicado y se vuelve a abrir automáticamente el editor, para añadir la siguiente opción.



The screenshot shows the 'Encuestas' (Surveys) interface. At the top, there is a header 'Encuestas' with a blue icon. Below it, the section 'Añadir opciones' (Add options) is active. The question is 'Pregunta: De todas las materias, ¿cuál te ha parecido más útil?' (Question: Of all subjects, which one do you find most useful?). Underneath, the 'Opción de respuesta:' (Response option) section features a rich text editor with a toolbar containing icons for text formatting (bold, italic, underline, text color, background color), alignment, bulleted and numbered lists, indentation, link, unlink, image, and table. Below the toolbar are dropdown menus for 'Estilo' (Style), 'Formato' (Format), 'Fuente' (Font), and 'Tamaño' (Size). The main text area contains the word 'Matemáticas' (Mathematics). At the bottom, there are three buttons: 'Guardar' (Save), 'Guardar y añadir opciones' (Save and add options), and 'Cancelar' (Cancel).

Ilustración 10: Formulario para crear opciones en una encuesta de Sakai

En la imagen siguiente se muestra la pantalla de votación de la encuesta que hemos definido, tal y como se ve desde el portal de Sakai.



The screenshot shows the 'Encuestas' (Surveys) interface for voting. The section 'Votar' (Vote) is active. The question is 'De todas las materias, ¿cuál te ha parecido más útil?' (Of all subjects, which one do you find most useful?). Below the question, there are five radio button options: 'Matemáticas' (Mathematics), 'Física' (Physics), 'Química' (Chemistry), 'Historia' (History), and 'Todas' (All). At the bottom, there are two buttons: '¡Vota!' (Vote!) and 'Cancelar' (Cancel). In the bottom right corner, there is a footer that reads 'Gateway | The Sakai Project' and 'Powered by Sakai'.

Ilustración 11: Página con la encuesta definida en Sakai

3.5. La herramienta de exámenes (SAMIGO)

Mediante la herramienta de gestión de exámenes Samigo se pueden definir en Sakai exámenes en un sitio y publicarlos, de forma que los participantes del sitio lo tengan disponible y puedan responderlo. La pantalla inicial de la herramienta es la siguiente:

Ilustración 12: Pantalla de gestión de exámenes de la herramienta Sakai.SAMIGO

La pantalla está dividida en varias secciones. En la barra superior se puede seleccionar qué elemento se va a administrar, entre los tres disponibles: exámenes, plantillas o baterías de preguntas. Las plantillas consisten en una serie de valores por defecto y una subselección de la configuración del examen para un tipo particular de examen. Las baterías de preguntas son conjuntos de preguntas que se pueden utilizar en varios exámenes.

En la siguiente sección se puede crear un nuevo examen. Para ello, se puede utilizar el asistente que dispone la herramienta o utilizar el lenguaje de marcado propio de Sakai. Cada tipo de pregunta tiene un formato específico. Si se selecciona esta opción de creación, en la pantalla siguiente se pueden consultar ejemplos para cada tipo de pregunta. También se puede importar de fichero un nuevo examen, que debe ser del tipo IMS QTI-compliant XML. Este fichero puede haberse generado desde el propio Sakai o desde Respondus, que es una poderosa herramienta para a creación y manipulación de exámenes, que pueden ser publicados en diversos LMS.

Las siguiente secciones muestran las evaluaciones generadas en el sistema, y el estado en el que están. Existe una sección para las evaluaciones que todavía no se han publicado, otra para los que están publicados y activos, y otros para los inactivos, a los que los participantes ya no tienen acceso.

En la siguiente imagen se muestra la pantalla de creación de un nuevo examen. Un examen consta de una o varias secciones, que a su vez está formada por una o varias preguntas. Las preguntas pueden tener asociada una puntuación para la evaluación. Los aspectos y opciones que se pueden configurar en cada una dependen del tipo que fue indicado en la creación de la misma.

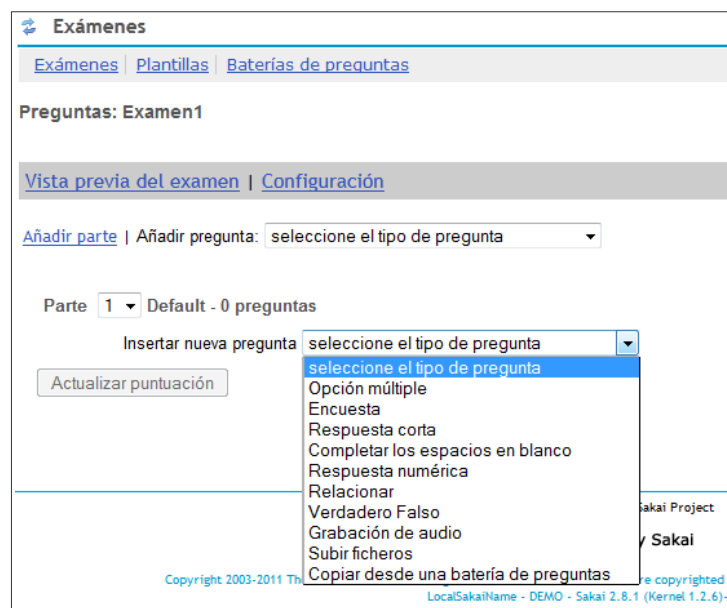


Ilustración 13: Selección de tipo de pregunta para un examen en Sakai.SAMIGO

Estos son los tipos de preguntas que se pueden utilizar para la definición de las distintas preguntas de un examen mediante SAMIGO:

- **Opción múltiple:** Se indican las respuestas posibles, de forma que el usuario debe elegir cuál o cuales son las correctas. Los atributos que se pueden configurar para este tipo de pregunta son los siguientes:
 - Texto de la pregunta: Permite definir también el formato de caracteres (tipo de fuente, tamaño...) con el que se quiere mostrar el texto.
 - Tipo de respuesta. Se puede elegir entre las siguientes opciones:
 - Selección única, con una respuesta correcta. Se muestran las respuestas mediante botones de radio.
 - Selección múltiple con una respuesta correcta. Las respuestas se muestran mediante checkboxes, a pesar de que sólo una de las respuestas será la correcta.
 - Selección múltiple con varias correctas. Como en el caso anterior, se muestran las respuestas mediante checkboxes.
 - Aleatorizar respuestas: Las respuestas se mostrarán en un orden aleatorio cada vez que se muestren al usuario.
 - Requiere razonamiento: Se indicará en pantalla un cuadro de texto para que los usuarios indiquen la explicación de por qué han seleccionado esa respuesta o respuestas.

Parte 1 de 1 -

Preguntas 1 de 6

Seleccione la opción que contenga la igualdad correcta

A. $1 + 1 = 3$

B. $3 / 2 = 10$

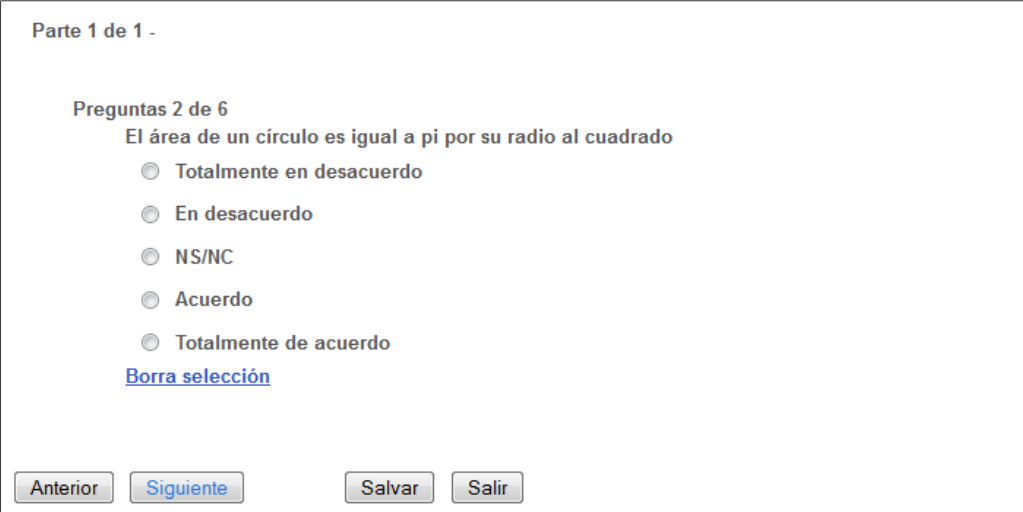
C. $2 * 0 = 0$

D. $3 - 4 = 9$

[Borra selección](#)

Ilustración 14: Pregunta tipo Opción múltiple en Sakai.SAMIGO

- **Encuesta:** El tipo de pregunta es similar al de opción múltiple, pero las respuestas ya están definidas, para adaptarlas a las que se utilizan habitualmente en la definición de encuestas. Los aspectos configurables son los siguientes:
 - Texto de la pregunta: Como en el caso anterior (y en todos los tipos de pregunta, porque este atributo va a ser común a todos los tipos de pregunta), se puede indicar el formato con el que se mostrará el texto.
 - Tipo de respuesta. Existen las siguientes opciones:
 - Sí, No: Se utilizarán dos botones de radio para indicar los valores Si y No.
 - En desacuerdo, de acuerdo: Como en el caso anterior, se utilizarán dos botones de radio para mostrar los dos valores
 - En desacuerdo, NS/NC, de acuerdo: En este caso se utilizarán tres botones de radio con las opciones anteriores.
 - Por debajo de la media - Por encima de la media: Se muestran tres botones de radio con los siguientes valores: Por debajo de la media, Media y Por encima de la media.
 - Totalmente en desacuerdo - Totalmente de acuerdo: Se utilizan cinco botones de radio para mostrar las opciones Totalmente de acuerdo, En desacuerdo, NS/NC, De acuerdo y Totalmente de acuerdo.
 - Inaceptable - Excelente: Se utilizan cinco botones de radio, como en las dos opciones anteriores, con los siguientes valores: Inaceptable, Por debajo de la media, Media, Por encima de la media y Excelente.
 - 1 - 5: Se utilizan cinco botones de radio, uno por cada valor numérico entre el 1 y el 10.
 - 1 - 10: Similar a la opción 1 – 5, pero con diez botones de radio.



Parte 1 de 1 -

Preguntas 2 de 6

El área de un círculo es igual a pi por su radio al cuadrado

Totalmente en desacuerdo

En desacuerdo

NS/NC

Acuerdo

Totalmente de acuerdo

[Borra selección](#)

Ilustración 15: Pregunta tipo Encuesta en Sakai.SAMIGO

- **Respuesta corta:** En este tipo se utiliza un cuadro de texto para que el usuario indique la respuesta. Estas son las opciones de configuración
 - Texto de la pregunta, con el formato de caracteres que se puede indicar de forma opcional.
 - Modelo de respuesta (opcional)
 - Comentario (opcional)



Parte 1 de 1 -

Preguntas 3 de 6

Exponga el enunciado de la segunda ley de Newton

(Número máximo de caracteres: 60000)

[Mostrar/Ocultar el editor de texto](#)

Ilustración 16: Pregunta tipo Respuesta corta en Sakai.SAMIGO

- **Completar espacios en blanco:** Dentro del enunciado de la pregunta se dejan en blanco algunas palabras o frases, que deberán ser completados por el usuario. Los parámetros a configurar en este tipo son:
 - Texto de la pregunta: Además del texto y formato, se debe indicar las palabras que se van a dejar en blanco. Para ello, se engloban entre llaves en donde se quiera insertar el espacio en blanco. Ejemplo: *Las rosas son {rojas} y las violetas {azules}*
Si se quieren utilizar sinónimos o respuestas múltiples, se pueden separar las distintas opciones mediante el carácter “[]”. Ejemplo: *Las hojas tienen {anverso|reverso}*
Se puede usar el asterisco, para indicar que uno o más caracteres de comodín.

Por ejemplo: *Llueve a {c*} y hace un frío que {p*}*

- Opción distinguir mayúsculas/minúsculas: Para que la respuesta se de por válida, si esta opción está activa se deben indicar correctamente las mayúsculas y minúsculas.
- Opción Mutuamente exclusivas: Al activarlo, las cuestiones que incluyen más de un espacio a rellenar con opciones de respuesta idénticas deben tener respuestas únicas. Por ejemplo: *Las caras de una moneda son {cara|cruz} y {cara|cruz}*. Respuesta correcta: cara, cruz. Respuesta medio correcta: cara, cara.

Parte 1 de 1 -

Preguntas 4 de 6

Complete los espacios en blanco: La suma de los de los es igual al cuadrado de la hipotenusa

Anterior [Siguiente](#)

Ilustración 17: Pregunta tipo Rellenar espacios en blanco en Sakai.SAMIGO

- **Respuesta numérica:** Como en la opción de Completar espacios en blanco, se indica en el texto de la pregunta el lugar en el que se indicará la respuesta numérica. Se puede configurar el siguiente aspecto:
 - Texto de la pregunta: Además del formato (opcional), se utilizan llaves ({} alrededor de un valor numérico requerido en un espacio en blanco. Por ejemplo: $3*3= \{9\}$ y $2+2= \{4\}$
Para indicar un rango numérico, se inserta el símbolo (|) entre los valores límite del rango. Por ejemplo: *El precio es {12.2|14.5}*.
En este caso, cada valor entre 12.2 y 14.5 será válido en la solución.

Parte 1 de 1 -

Preguntas 5 de 6

Complete las siguientes operaciones aritméticas:

$2 * 4 =$

$3 + 5 =$

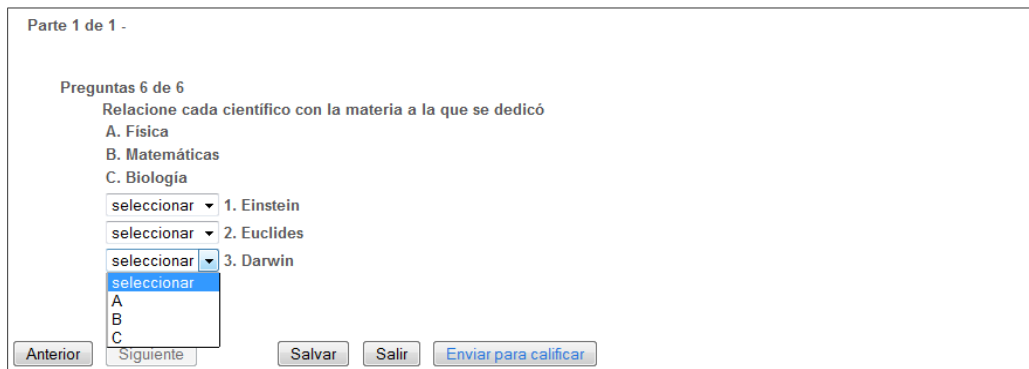
$10 -$ $= 2$

Anterior [Siguiente](#)

Ilustración 18: Pregunta tipo Respuesta numérico en Sakai.SAMIGO

- **Relacionar:** En este tipo se indica una serie de pares opción/respuesta, y el usuario debe relacionarlos entre sí de forma correcta. Se mostrará la opción y una lista de valores posibles, del que el usuario seleccionará uno, que dará lugar a su respuesta. Los parámetros a configurar son:
 - Como en el resto de tipos, el texto de la pregunta y opcionalmente su formato

- Pares a relacionar: Valores opción/respuesta correcta. El conjunto de todas las respuestas será el que se muestre al usuario en una lista desplegable, de manera que tenga que seleccionar la correcta para cada opción.



Parte 1 de 1 -

Preguntas 6 de 6

Relacione cada científico con la materia a la que se dedicó

A. Física
B. Matemáticas
C. Biología

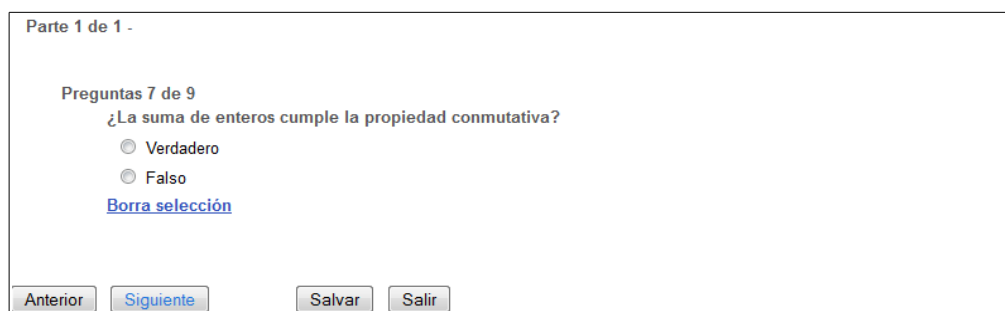
seleccionar 1. Einstein
seleccionar 2. Euclides
seleccionar 3. Darwin

seleccionar
A
B
C

Anterior Siguiete Salvar Salir Enviar para calificar

Ilustración 19: Pregunta tipo Relacionar en Sakai.SAMIGO

- **Verdadero/falso:** Este tipo de pregunta es muy similar al tipo Si/No del tipo Encuesta, en donde se mostrarán dos botones de radio, uno para el valor Verdadero y otro para Falso.
 - Como en el resto de las preguntas, se debe configurar el texto de la pregunta, y el formato en el que se mostrará.
 - También existe la opción de requerir razonamiento.



Parte 1 de 1 -

Preguntas 7 de 9

¿La suma de enteros cumple la propiedad conmutativa?

Verdadero
 Falso

[Borra selección](#)

Anterior Siguiete Salvar Salir

Ilustración 20: Pregunta tipo Verdadero/false en Sakai.SAMIGO

- **Grabación de audio:** En este curioso tipo, se permite al usuario que grabe su respuesta en formato audio. Para ello, se le abrirá una grabadora de sonidos que el usuario puede utilizar para realizar la grabación. Los parámetros a configurar son:
 - Texto de la pregunta: Como en el resto de tipos de pregunta, con el formato si se quiere
 - Tiempo de grabación: Se indica el tiempo máximo de la grabación medido en segundos.
 - Número de intentos: Número de veces que los usuarios pueden regrabar su respuesta.



Parte 1 de 1 -

Preguntas 8 de 9

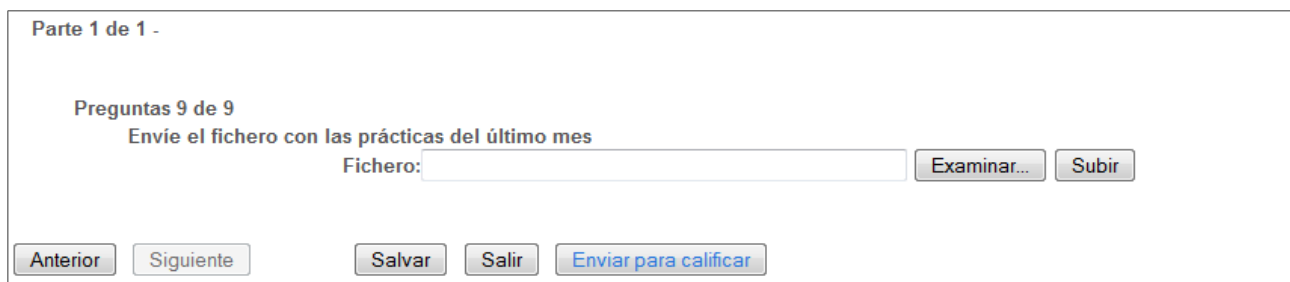
Enuncie la tercera ley de Newton

[Haga clic para grabar su respuesta...](#)

Anterior Siguiente Salvar Salir

Ilustración 21: Pregunta tipo Grabación de audio en Sakai.SAMIGO

- **Subir ficheros:** Permite a los usuarios indicar un fichero como respuesta. Para ello, les abre un explorador de archivos que les permite seleccionar el fichero a subir que será utilizado como su respuesta. Sólo se puede configurar:
 - Texto de la pregunta, de la misma forma que en el resto de tipos.



Parte 1 de 1 -

Preguntas 9 de 9

Envíe el fichero con las prácticas del último mes

Fichero: Examinar... Subir

Anterior Siguiente Salvar Salir Enviar para calificar

Ilustración 22: Pregunta tipo Subir ficheros en Sakai.SAMIGO

3.6. Conexión con sistemas remotos

Cuando Sakai se desarrolló, no se conocían las necesidades ni estándares de conexión de las organizaciones (que incluían a universidades con miles de estudiantes), aunque se tenía claro que las organizaciones debían ser capaces de conectarse y obtener información de Sakai, para utilizarlo en sus propios sistemas externos.

La aproximación que se adoptó fue mediante servicios web, que son fáciles de utilizar y fáciles de crear, en el caso de necesitar alguna información no incluida en los servicios web básicos que existen en el sistema.

Existen dos tipos principales de servicio web que se utilizan en Sakai, ambos basados en estándares existentes:

- **Servicios web SOAP:** Es un protocolo de paso de mensajes basado en XML, y que en el caso de Sakai, se utiliza con HTTP (el protocolo que utilizan los navegadores web para obtener las páginas).
- **Servicios REST. Entity Broker:** Es un servicio interno que ayuda a manipular las entidades internas de Sakai. Estas entidades son las representación de partes del sistema como usuarios, cursos... y se exponen en forma de servicio RESTful.

Existe también otra forma de conexión llamada WSRP, que se utiliza para exponer información a aplicaciones tipo Portal mediante portlets, pero no nos es útil en el ámbito de la aplicación que vamos a desarrollar, por lo que no profundizaremos en esta tecnología.

3.6.1. Web services SOAP

En este protocolo de paso de mensajes mediante XML, el cliente prepara una petición de datos en un formato conocido y estandarizado, por supuesto en XML, utilizando HTTP como protocolo de transporte. En el archivo están codificados los datos que necesita el sistema para obtener la información que pide el cliente. El servicio, posteriormente, responde con otro mensaje XML con la información requerida, o en su caso con un mensaje de error, si se ha producido alguno.

Para el desarrollo de esta tecnología mediante Sakai, se han utilizado las bibliotecas de un proyecto de la Apache Foundation especializado en la generación de aplicaciones de servicios web SOAP: Apache Axis. Este framework permite implementar un servicio web escribiendo en un archivo el código Java que lo especifica, indicarlo en el directorio raíz donde se alojan los servicios web y automáticamente ya se encuentra disponible. El servicio se compilará la primera vez que un sistema realice una llamada sobre el mismo, y todo esto sin que haya sido necesario reiniciar el sistema ni ninguna parte del mismo, lo que es una gran ventaja para los administradores.

Al publicar un servicio web, se genera automáticamente (gracias a Apache Axis) un archivo de descripción llamado WSDL (Web Service Description Language), que contiene la información de los métodos que se disponen en el servicio, junto con las estructuras de datos que se utilizan. Con ese archivo, en los lenguajes de programación que soportan llamadas remotas a servicio web existen parsers que generan las clases con los stubs y proxys necesarios para que los programadores hagan uso de los servicios, sin necesidad de tener que generar el mensaje SOAP desde cero. También se encargan de recoger la respuesta del servidor y extraer la información útil para el cliente, o gestionar el error en el caso de que se hubiera producido en la llamada.

Los servicios web SOAP están deshabilitados por defecto en Sakai. Para habilitarlos, se debe indicar mediante una propiedad en el fichero sakai.properties:

```
webservices.allowlogin=true
```

También se puede indicar en el mismo archivo una propiedad con una lista de IPs que se les permite la conexión. Se pueden indicar también expresiones regulares, como en el ejemplo siguiente:

```
webservices.allow=localhost,127\\.0\\.0\\.1,192\\.168\\. [0-9.]+,domain\\.somewhere\\.ac\\.uk,123\\.45\\.678\\.90
```

Se puede indicar acceso libre a todos los servicios web, aunque no se recomienda su uso en absoluto en un entorno de producción. Esto solamente es útil en entornos de desarrollo, para no preocuparse posibles errores en llamadas a servicio web por no tener permiso explícito. Para conseguir este acceso no restringido, se indica en el fichero sakai.properties:

```
webservices.allow = .*
```

También se puede permitir o denegar el acceso a los ficheros de log utilizando las siguientes propiedades:

```
webservices.log-allowed=true
webservices.log-denied=true
```

Sakai dispone de algunos servicios web implementados por defecto, aunque también existen servicios generados por la comunidad, en el repositorio de código, en la sección para las contribuciones. En la tabla siguiente se detallan los servicios disponibles por defecto en todas las instalaciones de Sakai, junto con los métodos de cada uno y una pequeña descripción.

Servicios	Métodos	Descripción
SakaiLogin	login, logout	Servicios que se utilizan para hacer login en el sistema. Sin una sesión válida no se puede llamar a otros servicios
SakaiPortalLogin	login, loginAndCreate, UsageSessionService_loginDirect	Servicio web de apoyo a las conexiones desde el portal (como uPortal)
SakaiScript	checkSession, addNewUser, removeUser, changeUserInfo, changeUserName, changeUserEmail, changeUserType, changeUserPassword, getUserEmail, getUserDisplayName, addNewAuthzGroup, removeAuthzGroup, addNewRoleToAuthzGroup, removeAllRolesFromAuthzGroup, removeRoleFromAuthzGroup, allowFunctionForRole, disallowAllFunctionsForRole, setRoleDescription, addMemberToAuthzGroupWithRole, removeMemberFromAuthzGroup, removeAllMembersFromAuthzGroup, setRoleForAuthzGroupMaintenance, addMemberToSiteWithRole, addNewSite, removeSite, copySite, addNewPageToSite, removePageFromSite, addNewToolToPage, addConfigPropertyToTool, checkForUser, checkForSite, checkForMemberInAuthzGroupWithRole, getSitesUserCanAccess	Servicio web con los servicios principales para manipular usuarios, sitios, membresía y permisos en los sitios.
SakaiSession	checkSession, getSessionUser	Devuelve la información de la sesión que se pide
SakaiSigning	establishSession, testsign, verifysign, getsession, touchsession	Permite a aplicaciones externas verificar que un usuario es utilizado normalmente, junto con la

		herramienta Rutgers Link tool
SakaiSite	establishSession, getUserSite, getSiteList, joinAllSites, getSitesDom, getToolsDom	Servicios de manipulación de sitios. Los métodos que contienen la cadena DOM devuelven la información en un formato XML específico.

A continuación mostramos un ejemplo en jsp/java, que realiza una llamada al servicio de login y guarda la sesión obtenida para futuros usos (como por ejemplo llamar a otros servicios web). Con esto mostramos lo sencillo que es el código que se necesita en el cliente para utilizar los servicios web expuestos en Sakai.

```
<%@ page import="org.apache.axis.encoding.XMLType" %>
<%@ page import="javax.xml.rpc.ParameterMode" %>
<%@ page import="org.apache.axis.client.Call" %>
<%@ page import="org.apache.axis.client.Service" %>
<c:if test="${empty sakai_session}">
  <%
    try {
      Service service = new Service();

      String id = (String) application
        .getAttribute("sakai_admin_username");
      String pw = (String) application
        .getAttribute("sakai_admin_password");

      Call nc = (Call) service.createCall();

      nc.setTargetEndpointAddress((String) application.
        getAttribute("sakai_login_soap_URI"));

      nc.removeAllParameters();
      nc.setOperationName("login");
      nc.addParameter("id",XMLType.XSD_STRING, ParameterMode.IN);
      nc.addParameter("pw",XMLType.XSD_STRING, ParameterMode.IN);
      nc.setReturnType(XMLType.XSD_STRING);

      String results = (String) nc.invoke(new Object [] { id, pw });
      session.setAttribute("sakai_session",results);

    } catch (Exception e ) {
      pageContext.setAttribute("ex",e);
      session.removeAttribute("sakai_session");
    }
  %>
</c:if>
```

Como desventajas de este sistema, encontramos los siguientes:

- No es especialmente intuitivo de utilizar (al menos comparado con los servicios REST que vamos a tratar a continuación).
- Parsear y construir los mensajes SOAP introduce un sobreesfuerzo en el sistema,

tanto en la parte cliente como en el servidor.

- Los mensajes SOAP son bastante pesados al mantener toda la estructura del XML en el que está codificada la información, que se podría ahorrar en el caso de utilizar alguna plataforma de paso de mensajes más ligera.
- Los servicios se tienen que desarrollar en Java, por lo que los programadores de otros lenguajes tendrán dificultades para implementarlos.
- Es complicado de usar en el mundo javascript/AJAX que se utiliza en la mayoría de las webapps

3.6.2. Servicios REST. Entity Broker

La filosofía que se ha utilizado para construir los servicios web en Sakai es totalmente distinta en el caso de los servicios REST, en comparación con los webservices SOAP.

- En el caso de los servicios SOAP, se necesita que un programador escriba servicios web propios para nueva herramienta que se desarrolle en el sistema. Cada nueva herramienta tendrá sus propias estructuras y elementos, de manera que se hace difícil dicha integración.
- En los servicios REST mediante Entity Broker, es el kernel el que se responsabiliza del la exposición de datos y del manejo de los servicios web REST. Como el programador no tiene que preocuparse de los detalles del servicio, se reduce la duplicación de código y el esfuerzo de programación, mientras que se mejora la escalabilidad, calidad y mantenibilidad del código. Además las entidades se exponen en varios tipos MIME como HTML, XML, JSON, para facilitar el consumo de los datos.

Para acceder a los servicios, se utiliza una URL que apunta a la parte principal del servicio. A partir de esta dirección base, dependiendo de la entidad a la que se está accediendo, se sigue construyendo utilizando la información de la entidad concreta a la que queremos hacer referencia (si es el caso). Por ejemplo, una URL para acceder a un item (en este caso el item de código 3) del servicio de agenda podría ser la siguiente:

```
http://<sakai_root>/direct/addressbook_item/3
```

Como se puede comprobar, esta dirección es muy intuitiva de entender y utilizar, por lo que se simplifica el código que se utiliza. Una vez visto cómo se accede a una entidad, vamos a ver cómo indicar la operación que queremos hacer con ella. Para ello, se utilizan los métodos ya disponibles en el protocolo HTTP, siguiendo las recomendaciones del W3C explicadas en la siguiente web: <http://microformats.org/wiki/rest/urls>

En esta recomendación se utilizan los verbos HTTP para implementar la semántica estándar de bases de datos CRUD: Create (crear), Retrieve (obtener), Update (actualizar), Delete (borrar). Los métodos primarios son:

- POST: Crea un recurso de una colección (entidad) dada
- GET: Obtiene un recurso

- PUT: Actualiza un recurso
- DELETE: Borra un recurso

También se puede indicar en la llamada el tipo de datos que queremos que el servicio nos devuelva. Para ello existen dos maneras de indicarlo:

- Indicando el tipo MIME apropiado en la cabecera HTTP, que es la forma que se prefiere
- Indicándolo en la URL como extensión, codificando en la misma el tipo de datos que queremos utilizar. Por ejemplo, si hacemos un GET a la URL `http://<sakai_root>/direct/addressbook_item/3.json`, pedimos al servicio que nos devuelva en formato JSON la entidad 3 de la agenda.

Una vez realizada la llamada, el sistema retornará una respuesta utilizando el protocolo HTTP, y codificará el tipo de respuesta mediante los códigos de respuesta ya estandarizados. Los códigos de respuesta que se utilizan mayormente en los servicios REST se detallan a continuación:

- 200: OK. La petición es correcta y se devuelve el contenido
- 201: CREATED. La petición ha creado nuevo contenido. Se indica la URL y la ID del contenido en la cabecera
- 204: NO CONTENT: La petición es correcta pero no hay contenido que devolver
- 400: ERROR. Error general en la petición, probablemente parámetros o datos inválidos
- 401: UNAUTHORIZED. Se necesita autenticación de usuario para esta petición
- 403: FORBIDDEN. Se necesita autorización, privilegios insuficientes o el usuario ya se ha autenticado
- 404: NOT FOUND. Recurso no encontrado, URL inválida de alguna forma, o por identificador o acción
- 405: METHOD NOT ALLOWED. El método no está soportado para este tipo de entidad
- 406: NOT ACCEPTABLE. El fomrato de datos de la petición no está disponible para este tipo de entidad
- 500: INTERNAL SERVER ERROR. Fallo general en el servidor, probablemente fallo en el proveedor de datos
- 501: NOT IMPLEMENTED. Indica que el prefijo es inválido

Los servicios están registrados bajo la URL siguiente:

```
http://<sakai_root>/direct/
```

Para descubrir los servicios que están disponibles en el sistema existe una dirección en donde se indica una descripción legible de todos los que están registrados:

`http://<sakai_root>/direct/describe`

A partir de esta dirección se puede navegar por los servicios para obtener información de uso, aunque hay que decir que muchas veces esta información es bastante escueta. En la versión demo de Sakai 2.8.1, los servicios REST que estaban disponibles se detallan en la siguiente lista:

- **Announcement:** Representa los anuncios de una asignatura concreta (incluyendo los anuncios combinados) o todos los los anuncios de un usuario. También incluye Mensaje del Día y anuncios públicos.
- **Assignment:** Representa una tarea y sus propiedades asociadas
- **Basiclti-events:** Permite el informe de uso básico de LTI, para su uso en las estadísticas del sitio. No existen en este lugar interfaces utilizables.
- **Batch:** Manejador especial para proveer proceso batch de varias peticiones de una vez.
- **Chat-channel:** Representa un canal de chat
- **chat-message:** Representa un mensaje de chat
- **forum:** Representa las propiedades de un foro
- **forum-message:** Representa un mensaje concreto de un foro
- **forum-topic:** Representa un tema de un foro determinado
- **gradebook:** Representa el libro de grados de un alumno
- **matrixcell**
- **membership:** Representa una membresía de un usuario en una localización (sitio, grupo, etc.) del sistema y trabaja con las entidades del sitio/grupo y las entidades de usuario
- **my:** Es un conjunto de direcciones que se utilizan como accesos directos a páginas y vistas de la herramienta Profile2.
- **Poll:** Representa un sondeo (un único elemento de una encuesta) y opcionalmente los votos (respuestas) al sondeo y las opciones.
- **Poll-option:** Representa una opción de un sondeo
- **Poll-vote:** Representa el voto de un usuario en un sondeo
- **Profile:** Representa el perfil de un usuario. Existe un sistema de seguridad que permite indicar las partes del perfil que se van a devolver, dependiendo del usuario que está pidiendo los datos, por razones de privacidad.
- **Profile-events:** Permite el reporte anónimo de la herramienta de Profile para las estadísticas del sitio. No hay interfaces que sean utilizables.
- **Profile-message:** Permite el acceso al sistema de mensajería disponible en la herramienta de Profile2

- **Profile-status**: Representa el estado de un usuario
- **sam_pub**: Permite realizar un examen de la herramienta SAMIGO
- **search**
- **server-config**: Representa la configuración del servidor
- **session**: Representa la sesión de un usuario o sistema que está actualmente activa. Las sesiones inactivas o información acerca de las misma no pueden ser accedidas. La sesión es inmutable, y borrarla equivale a desactivarla
- **Site**: Representa una sitio (una colección de usuarios y herramientas) en Sakai
- **Sitestats-metrics**: Añade un medidor de eventos para el nodo de un servidor actual.
- **Sitestats-report**
- **synopticmsgcntritem**
- **url**: Permite el acortado de direcciones mediante el servicio ShortenedUrlService
- **User**: Representa una usuario del sistema. Los datos de usuario son inmutables y borrar usuarios no sobrescriben aquellos provistos externamente
- **UserPrefs**: Reprerenta la preferencias de usuario
- **Validation**:

En el futuro, además de añadir y mejorar las entidades que dan soporte a las llamadas RESTful, se pretende añadir más soporte a estándares como OpenSearch 1.1 URL, añadir formatos de salida como RSS y ATOM, añadir integración con GWT (Google Web Toolkit, el framework para aplicaciones web de Google) e ir haciéndolo cada vez mas modular.

4. Frameworks web para el desarrollo de aplicaciones móviles. Estudio previo.

4.1. LungoJS

LunjoJS es un framework de desarrollo para aplicaciones móviles web, que sigue el estándar HTML5/CSS3 para la creación del código fuente. Está desarrollado por TapQuo, que es una empresa española afincada en Bilbao, especializada en desarrollo web y sobre todo en tecnologías javascript.

El desarrollo del framework estuvo motivado cuando el CEO de la compañía, Javier Jiménez Villar, se puso a investigar sobre tecnologías de desarrollo para dispositivos móviles, una vez consiguió su primer iPhone (que no llegó a venderse en España). Al desarrollar primeramente en nativo (Objective-C) y darse cuenta de que en este caso su desarrollo no sería compatible con distintos dispositivos (para Android debería usar su API Java, para Microsoft sus propias tecnologías...), decidió volver a sus raíces y probar el desarrollo de webapps. En ese momento empezaban a sonar las tecnologías HTML5 y CSS3, y al disponerse de un estándar de futuro para estos desarrollos se abrieron las opciones.

Al investigar los dos frameworks dominadores en el desarrollo de webapps de estos momentos (Sencha y jQuery Mobile), se encontró con algunas dificultades e inconvenientes que le hicieron plantearse la implementación de su propio framework. Estos inconvenientes son derivados de la filosofía que han adoptado los dos grandes desarrollos, al adaptar sus bibliotecas de desarrollo para aplicaciones de escritorio al entorno móvil. Esto hace que las bibliotecas sean relativamente pesadas al utilizar mucho código que en realidad no se necesita en el mundo móvil. Además los eventos táctiles de estas pantallas son distintos a los eventos de ratón que se utilizan en los entornos de escritorio, y no siempre están bien resueltos.

Otro de los aspectos que quería conseguir es tener un entorno realmente HTML5/CSS3 con el que trabajar. Según el mantra que el propio Javier se impuso para el desarrollo de Lungo: *“LungoJS solo dará soporte a dispositivos que den soporte real a HTML5/CSS3/JavaScript”*. En el caso de jQuery Mobile, debido a que intenta ser compatible con la mayoría de dispositivos del mercado, aunque estén comenzando a ser obsoletos, lastra su evolución debido a que la limitación la pone el dispositivo con menos características que se ha decidido incluir en el grado-A de compatibilidad.

La filosofía de Lungo es totalmente distinta: el framework implementará las características que estén descritas en el estándar HTML5, y por lo tanto sólo dará soporte a aquellos dispositivos que tengan una compatibilidad real con este estándar. Presumiblemente, cada vez los sistemas van a tener una compatibilidad con HTML5 más depurada, por lo que a priori parece una buena estrategia de crecimiento. Además, a día de hoy la compatibilidad con los sistemas más utilizados (Blackberry, Android e iOS, el incluso Chrome como navegador de escritorio) está asegurada.

Las principales características del framework LungoJS son:

- Creación sencilla de aplicaciones web para dispositivos iOS (Apple), Android y

Blackberry

- Desarrollo de aplicación HTML5 con estructura semántica en todo el proyecto, comenzando por la plantilla en HTML, apoyándose en los estilos CSS y terminando con su API Javascript.
- No necesita líneas de código en Javascript para realizar prototipos de aplicaciones: solamente con el HTML se puede renderizar el resultado en el navegador o dispositivo móvil.
- Utiliza una librería para el manejo del DOM de la página extremadamente ligera, llamada QuoJS y desarrollada también por TapQuo, de manera similar a como trabaja jQuery.
- Aprovecha las capacidades de los móviles actuales
- Implementación sencilla de características HTML5 como WebSQL, Orientación, Conexión...
- Captura eventos táctiles como swipe, tap, doble tap...
- Puede extenderse la funcionalidad del frameworks mediante plug-ins, que en este entorno se llaman Sugars.
- Diseño totalmente personalizable
- Permite distribuir las aplicaciones tanto en sitios web como en las distintas stores (Google Play, App Market de Apple...).

La licencia de este framework es GPL v3 para desarrollo de aplicaciones (es decir, software libre). Si se quiere usar de forma empresarial, se debe contratar una licencia comercial de pago.

A la hora de comenzar con un desarrollo en este framework, en los tutoriales y en la misma descarga de las bibliotecas, se nos recomienda usar una serie de ficheros javascript para organizar el desarrollo. Por supuesto, se puede usar otra organización si lo creemos necesario, pero la propuesta es realmente cómoda de usar y efectiva:

- `app.js`: Para inicializar la aplicación
- `data.js`: Para las llamadas al sistema de almacenamiento webSQL
- `events.js`: Para los manejadores de eventos
- `services.js`: Para las llamadas a servicios remotos (por ejemplo, llamadas AJAX)
- `view.js`: Para la definición de vistas, mediante plantillas (templates)

Para inicializar la aplicación, se debe indicar en el fichero `app.js`, mediante las siguientes líneas de código:

```
var App = (function(lng, undefined) {  
  
    lng.App.init({  
        name: 'LungoJS Test Source',  
        version: '1.2'  
    });  
});
```

```
} ) (LUNGO) ;
```

En cuanto a la definición del marcado en la página HTML, se utiliza la nomenclatura recomendada para las páginas en HTML5, es decir, se definen como secciones que en su interior contendrán cabeceras, pies de página y artículos. Una sección puede contener varios artículos. Si pensamos en la organización que pueda tener un blog, las secciones de la página pueden ser por ejemplo una lista con el archivo por meses de las entradas, y otra sería el sitio en el que se indica el texto de las entradas. Los artículos serían cada una de las entradas del blog.

En LungoJS, definiríamos la sección principal mediante el código indicado a continuación:

```
<section id="main">
  <header data-back="home blue" data-title="Título de la sección"></header>

  <footer class="toolbar"></footer>

  <!-- content -->
</section>
```

En esta sección se definen además la cabecera, en la que se indica el botón de atrás, que al estar marcado semánticamente, se define con el icono de HOME y de color azul, mediante el atributo data-back. También se indica el título de la sección, mediante el atributo data-title. Para el footer, se le indica en la clase que se va a comportar como una barra de herramientas, por lo que estaría preparada para que indicáramos botones en su interior.

Además de cabeceras y pies de sección, en su interior se pueden indicar artículos, tal y como se muestra en el ejemplo siguiente:

```
<section id="main">
  <!-- header -->

  <!-- footer -->

  <article id="first_article">
    <!-- content -->
  </article>

  <article id="second_article">
    <!-- content -->
  </article>

</section>
```

El artículo se puede configurar para mostrarse como una lista indicándolo mediante atributos en el elemento article, tal y como se puede ver:

```
<article id="first_article" class="list" data-search="blue">
  <!-- content -->
  <ul>
    <li>Primer elemento de la lista</li>
    <li>Segundo elemento de la lista</li>
  </ul>
```

```
</article>
```

En la imagen siguiente se puede observar el resultado en el navegador:

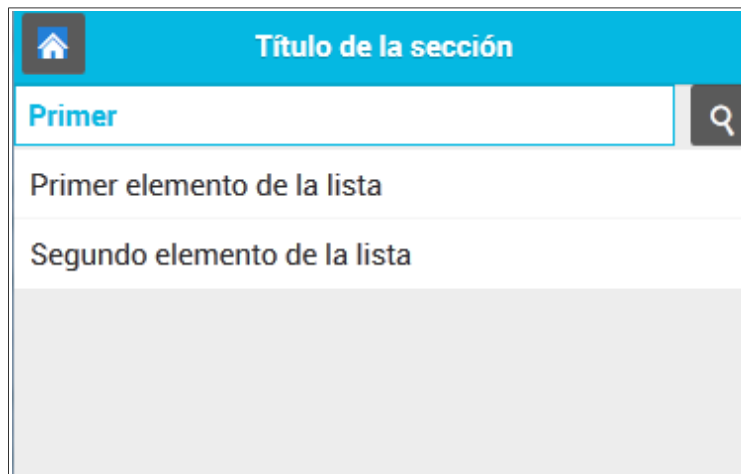


Ilustración 23: Ejemplo de visualización de un programa en LungoJS

4.2. JQuery Mobile

JQuery Mobile es un framework para la creación de aplicaciones móviles nacido al amparo de la JQuery Foundation, una asociación sin ánimo de lucro que nació para el desarrollo, soporte y documentación de los proyectos JQuery, JQuery UI y JQuery Mobile.

Su framework para aplicaciones javascript de escritorio (JQuery) es con mucha diferencia el más utilizado del mundo para simplificar y organizar el desarrollo en javascript necesario para crear aplicaciones web ricas y con efectos visuales vistosos. Además, se puede apoyar en la biblioteca JQuery UI, que dispone de diversos elementos de interfaz de usuario (botones, calendarios, acordeones, popups...) y una herramienta para generar temas visuales, simplificando en gran medida la creación de estilos y ficheros CSS, y permitiendo que toda la interfaz tenga una representación homogénea. Además, existe un inmenso número de plug-ins desarrollados por su extensa comunidad, que permiten extender la funcionalidad tanto del core de la biblioteca, como incluir el elementos y widgets de interfaz para utilizar en los desarrollos (árboles, tablas, galerías de imágenes, menús, mapas...)

En cuanto a JQuery Mobile, es un framework que se apoya en la biblioteca de JQuery oficial, por lo que hay que incluirla en la cabecera de la aplicación web. Su funcionamiento se basa en definir la interfaz mediante código HTML, indicando mediante elementos como DIVs y atributos propios cómo se va a renderizar la página. La biblioteca JQuery Mobile se encarga entonces de retocar el DOM de la página para que el navegador muestre el contenido de la página de la forma que el programador espera.

Como se puede ver, este enfoque es similar al que se adoptó en LungoJS, y bastante diferente a como lo afronta Sencha Touch, puesto que la interfaz en este framework se realiza íntegramente en javascript.

Las características principales del framework, según la propia página web del proyecto, son las siguientes:

- **Construido mediante el core de jQuery:** Al ser el framework más utilizado, la sintaxis es conocida y por lo tanto la curva de aprendizaje es pequeña
- **Compatible con las plataformas más utilizadas en móviles, tablets, e-readers y navegadores de escritorio:** Es el punto fuerte del framework, con multitud de sistemas en los que está probado: iOS, Android, Blackberry, Palm, Windows Phone y Mobile, Kindle, Firefox, Opera... y muchísimos sistemas más exóticos.
- **Reducido tamaño** y mínimas dependencias con imágenes, para aumentar la velocidad de carga
- **Arquitectura modular:** Se pueden seleccionar las características que se quieren incluir al descargar la biblioteca, para reducir todavía más el tamaño.
- **Configuración HTML5 dirigida por el marcado:** de páginas y comportamiento, para desarrollo más rápido y reducir la necesidad de programar scripts.
- **Aproximación mediante ampliaciones progresivas:** Permite al contenido del core y de la funcionalidad adaptarse fácilmente a las nuevas plataformas móviles
- **Responsive design:** Técnicas y herramientas que permiten al mismo código base escalar automáticamente desde interfaces de usuario diseñadas para smartphones hasta pantallas en equipos de escritorio.
- **Sistema de navegación AJAX:** Permite transiciones animadas entre páginas mientras mantiene la navegación mediante el botón Atrás, permite utilizar marcadores y direcciones URL limpias, mediante pushState.
- **Accesibilidad:** Se incluyen características como WAI-ARIA, para asegurarse de que las páginas funcionan en lectores de pantalla y otras tecnologías para asistencia.
- **Soporte a eventos táctiles y de ratón:** Se dispone de un API mediante la cual se pueden manejar eventos táctiles, de ratón y basados en el foco del cursor.
- **Widgets de interfaz de usuario unificados:** Se amplían los controles nativos con controles optimizados para eventos táctiles, a los que se les pueda configurar fácilmente el tema visual en cualquier plataforma soportada.
- **Framework para la generación de temas potente:** Existe una aplicación web (ThemeRoller) que facilita la generación de temas visuales, fabricando el archivo CSS que se incluirá en el desarrollo.

Las plataformas soportadas por el framework se muestran en la tabla siguiente. Se indican las plataformas que estaban soportadas en la última versión definitiva que estaba disponible en el momento de escribir este proyecto (1.1.1), pero en cada iteración se van añadiendo nuevas.

Las plataformas se dividen en varios grados de cumplimiento (A, B y C), dependiendo de las características que están soportadas en cada uno. La Fundación jQuery Mobile hace un gran esfuerzo, no sólo para soportar la mayor cantidad de sistemas posible, sino para que tengan el mayor grado de cumplimiento posible (el que sería grado A).

Grado	Entorno	Probado en
Grado A. Experiencia completa, incluyendo animaciones en transiciones de páginas y navegación AJAX	Apple iOS 3.2-5.0	iPad original (4.3 / 5.0), iPad 2 (4.3), iPhone original (3.1), iPhone 3 (3.2), 3GS (4.3), 4 (4.3 / 5.0), 4S (5.0)
	Android 2.1-2.3	HTC Incredible (2.2), Droid (2.2), HTC Aria (2.1), Google Nexus S (2.3). Funcional en 1.5 y 1.6, pero con un rendimiento a veces inadecuado (probado en Google G1 (1.5))
	Android 3.1 (Honeycomb)	Samsung Galaxy Tab 10.1 y Motorola XOOM
	Android 4.0 (Ice Cream Sandwich)	Galaxy Nexus S. Nota: rendimiento en las transiciones puede ser pobre.
	Windows Phone 7-7.5	HTC Surround (7.0) HTC Trophy (7.5), LG-E900 (7.5), Nokia Lumia 800
	Blackberry 6.0	BlackBerryTorch 9800 y Style 9670
	Blackberry 7	BlackBerry Torch 9810
	Blackberry Playbook (1.0-2.0)	BlackBerry PlayBook
	Palm WebOS (1.4-2.0)	Palm Pixi (1.4), Pre (1.4), Pre 2 (2.0)
	Palm WebOS 3.0	HP TouchPad
	Firefox Mobile (10 Beta)	Dispositivo Android 2.3
	Chrome for Android (Beta)	Dispositivo Android 4.0
	Skyfire 4.1	Dispositivo Android 2.3
	Opera Mobile 11.5	Dispositivo Android 2.3
	Meego 1.2	Nokia 950 y N9
	Samsung bada 2.0	Samsung Wave 3, Navegador Dolphin.
	UC Browser	Dispositivo Android 2.3
	Kindle 3 and Fire	Navegador WebKit incluido
	Nook Color 1.4.1	Nook Color original, no en Nook Tablet
	Chrome Desktop 11-17	OS X 10.7 y Windows 7
Safari Desktop 4-5	OS X 10.7 y Windows 7	
Firefox Desktop 4-9	OS X 10.7 y Windows 7	
Internet Explorer 7-9	OS X 10.7 y Windows 7	
Opera Desktop 10-11	OS X 10.7 y Windows 7	
Grado B. Experiencia	Blackberry 5.0	Blackberry Storm 2 9550, Bold 9770
	Opera Mini (5.0-6.5)	Probado en iOS 3.2/4.3 y Android 2.3

Grado	Entorno	Probado en
mejorada, sin navegación AJAX	Nokia Symbian^3	Nokia N8 (Symbian^3), C7 (Symbian^3), N97 (Symbian^1)
Grado C. Experiencia básica, aunque funcional	Blackberry 4.x Windows Mobile Otros smartphones y teléfonos más antiguos	Blackberry Curve 8330 HTC Leo (Windows Mobile 5.2) Cualquier dispositivo que no soporte consultas de medios

En la versión 1.2 (en estos momentos en fase alpha) está previsto añadir al grado A los siguientes sistemas: Android 4.1 (Jellybean), Tizen, Firefox for Android, así como las últimas versiones de Firefox y Chrome en versión escritorio.

Para comenzar con el desarrollo con jQuery Mobile, lo primero que se necesita es escribir la página HTML que define la interfaz que se va a mostrar en el navegador. Para ello, tenemos que indicar en la cabecera las bibliotecas javascript de jQuery y jQuery Mobile, y el archivo CSS de jQuery Mobile. Estas bibliotecas se pueden haber descargado previamente e incluido en el proyecto en local, o se pueden utilizar las que están alojadas en los servidores de la jQuery Foundation o Google (mediante su proyecto Google Code). El siguiente ejemplo muestra la definición de una página con el tradicional Hola Mundo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Página JCB</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href=
      "http://code.jquery.com/mobile/1.1.1/jquery.mobile-1.1.1.min.css" />
    <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
    <script src=
      "http://code.jquery.com/mobile/1.1.1/jquery.mobile-1.1.1.min.js"></script>
  </head>
  <body>

    <div data-role="page">
      <div data-role="header">
        <h1>Mi Titulo</h1>
      </div><!-- /header -->

      <div data-role="content">
        <p>Hola Mundo!!!!</p>
      </div><!-- /content -->

    </div><!-- /page -->

  </body>
</html>
```

Como se puede apreciar, en la cabecera se indican las bibliotecas a cargar (en este caso desde la dirección de jquery.com), además de la definición del viewport, que son

atributos que utilizará el dispositivo móvil para el dibujado de la página. En este caso, se le indica que el tamaño de la página será el mismo que el tamaño de la pantalla del dispositivo y que el escalado inicial de la página será de 1, es decir, sin escalado.

En el cuerpo de la página se indica en primer lugar un DIV con el atributo `data-role="page"`, lo que indica a jQuery Mobile que su contenido define una página completa. Dentro de la página existen dos DIVs que definen la cabecera y el contenido, indicando los atributos `data-role="header"` y `data-role="content"`, respectivamente.

En la imagen siguiente se muestra el resultado del HTML de ejemplo.



Ilustración 24: Ejemplo de programa Hola Mundo en JQuery Mobile

También como ejemplo, si se quiere indicar una lista de valores, se insertaría el siguiente código, por ejemplo en el DIV que define el contenido en la página anterior:

```
<ul data-role="listview" data-inset="true" data-filter="true">
  <li><a href="#">Acura</a></li>
  <li><a href="#">Audi</a></li>
  <li><a href="#">BMW</a></li>
  <li><a href="#">Cadillac</a></li>
  <li><a href="#">Ferrari</a></li>
</ul>
```

El resultado obtenido se muestra en la siguiente imagen:

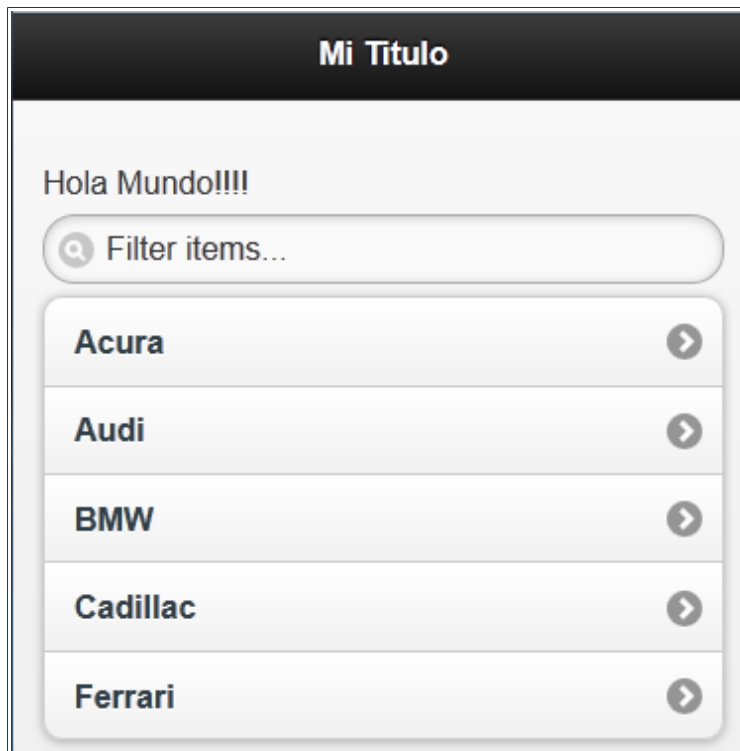


Ilustración 25: Ejemplo de lista en JQuery Mobile

5. Diseño de la aplicación de cuestionarios

5.1. Conexión con servicios remotos en SAKAI

Antes de realizar la especificación de requisitos de la aplicación, vamos a profundizar en los servicios web que nos permiten conectar con Sakai. Así podremos comprobar las posibles limitaciones que nos vamos a encontrar, que pueden ser importantes a la hora de especificar la aplicación.

Como ya se ha explicado anteriormente, existen dos técnicas principales para la conexión de sistemas remotos con Sakai: Los servicios web SOAP y los servicios RESTful mediante Entity Broker. De entre estas dos opciones, la que vamos a preferir es el uso de los los servicios RESTful, por una serie de razones que explicamos a continuación:

- Los servicios REST son STATELESS, es decir, el cliente no tiene que mantener información de estado, lo que facilita el uso de las llamadas AJAX.
- El protocolo SOAP utilizan XML para el envío de mensajes, lo que añade complejidad al cliente y al servidor, al tener que manejar y parsear las peticiones y repuestas. REST, en cambio, utiliza la propia dirección URL de acceso a la entidad, junto con los parámetros de la llamada (en la propia URL cuando se utiliza GET o como variables de formulario en el método POST).
- Los servicios web SOAP necesitan generar distintos métodos para las acciones disponibles con la entidad. REST utiliza los métodos estándar HTTP para indicar la acción que queremos realizar con la entidad, lo que hace más fácil la implementación.
- Al utilizar XML sobre HTTP, los mensajes SOAP son mucho más grandes que sus equivalentes REST. En SOAP se debe utilizar el formato XML definido, mientras que en REST se puede seleccionar el tipo de datos en la respuestas (HTML, XML, TEXT, JSON...) Esta respuesta tiene el formato propio de cada entidad, por lo que al no tener que utilizarse una estructura estándar puede ser mucho más compacta que su equivalente SOAP. En el contexto de aplicaciones móviles, es todavía más importante que el trasiego de información sea el más pequeño posible, puesto que la red móvil puede ser lenta dependiendo de la cobertura que se disponga. Además, casi todas las operadoras disponen de planes de datos con limitación en la cantidad de datos descargados a máxima velocidad, por lo que es importante optimizar al máximo el trasiego de información entre el dispositivo móvil y el servidor.
- Si se devuelven los datos directamente en JSON mediante REST, se puede parsear automáticamente la respuesta a un objeto javascript, lo que es ideal para el uso con aplicaciones basadas en web, como la que vamos a utilizar en este proyecto.

Por lo tanto, la opción que vamos a utilizar son las llamadas RESTful. Lo primero que hay que hacer es identificar los servicios que nos van a ser útiles de entre los que dispone Sakai. Para ello, con la instancia de la versión DEMO de Sakai arrancada,

accedemos a la URL que nos da acceso a las descripciones de los servicios REST, que es la siguiente:

http://<SAKAI_ROOT>/direct/describe

De entre todos los servicios que nos muestra (y que hemos detallado anteriormente) vamos a centrarnos en los que nos ocupan para la implementación del proyecto:

- **SESSION** (http://<SAKAI_ROOT>/direct/session): Representa la sesión de un usuario o sistema que está actualmente activa. Con esta entidad se puede acceder a la sesión, borrarla (lo que equivale a desconectarte del sistema), crear una sesión (lo que quiere decir que te vas a conectar al sistema), o actualizarla, para refrescar la sesión. No se puede acceder a las sesiones inactivas. Las acciones que soporta son las siguientes:
 - **GET**: Retorna los datos de la sesión. Los tipos de datos para respuesta que soporta son html, xml y json. Se puede indicar en la misma URL la ID de la sesión que queremos consultar. Si la petición se procesa de forma correcta, en la respuesta se indicará el código 200 y los datos pedidos. Si no se encuentra, se indicará el código 404 y si el formato no está disponible se indicará el código 406.
 - **POST**: Crea una nueva sesión. En la llamada se deben utilizar los parámetros *_username* y *_password*, indicando el nombre de usuario y contraseña. Si se ha creado correctamente la sesión, se devolverá en la respuesta el código 201 junto con la identificación (EntityId). Si no se ha podido crear la sesión porque los datos indicados son inválidos, en la respuesta se indicará el código de error 400. Los administradores pueden crear sesiones para cualquier usuario con sólo enviar una petición con datos de sesión válidos.
 - **PUT**: Actualizamos la sesión, de forma que conseguimos que no expire. Si la petición se ha procesado correctamente en la respuesta se le indicará el código 204. En caso contrario, se indicará el código de error 400.
 - **DELETE**: Borrarnos la sesión, es decir, nos desconectamos de la sesión. Como en el caso del método PUT, si se ha procesado correctamente se envía el código 204 en la respuesta, y si se ha producido un error el código de error que se utiliza es el 404.
- **POLL** (http://<SAKAI_ROOT>/direct/poll): Representa una encuesta. Opcionalmente se pueden obtener los votos (respuestas) a la encuesta y las opciones.

Las encuestas están representadas mediante una estructura de datos, que contiene los siguientes atributos:

Nombre	Tipo de datos	Descripción
CreationDate	Fecha	Fecha de creación
Description	String	Descripción de la encuesta
Details	String	Detalles de la encuesta
DisplayResult	String	Indica si se deben mostrar los

resultados. Puede tomar los valores siguientes:

- **open**: Los resultados se pueden mostrar siempre
- **afterVoting**: Los resultados no se mostrarán hasta que el usuario haya votado.
- **afterClosing**: Los resultados se mostraran sólo cuando haya vencido la fecha máxima en la que se permitía votar en la encuesta.
- **never**: Los resultados nunca se muestran.

Id	String	Identificación de la encuesta
IsPublic	Lógico	Indica si la encuesta es pública
LimitVoting	Lógico	Indica si la votación está limitada
MaxOptions	Entero	Número de opciones que se pueden indicar en la respuesta como máximo
MinOptions	Entero	Número de opciones que se pueden indicar en la respuesta como mínimo
Owner	String	Identificador del propietario de la encuesta
PollId	Entero largo	Identificador de la encuesta numérico
PollText	String	Texto de la encuesta. Enunciado
Properties	String	Propiedades de la encuesta
Reference	String	Identificador de la referencia
Siteid	String	Identificador del sitio al que pertenece la encuesta
Text	String	Texto de la encuesta. Enunciado
Url	String	Dirección web en donde está alojada la encuesta en el portal.
VoteClose	Fecha	Fecha hasta la que se puede votar en la encuesta
VoteOpen	Fecha	Fecha desde la que se puede votar en la encuesta
CurrentUserVoted	Lógico	Indica si el usuario logueado ya ha votado
EntityReference	String	Referencia de la entidad REST
EntityUrl	String	Dirección URL de la entidad REST
EntityId	String	Identificador de la entidad REST. Normalmente coincidirá con la

		identificación de la encuesta.
CurrenUserVotes	Lista de votos	Lista con los votos del usuario actual en la encuesta
Votes	Lista de votos	Lista total de votos de la encuesta
PollOptions	Lista de opciones	Lista de opciones de la encuesta

Los métodos que soporta este servicio son los siguientes:

- GET: Obtiene todas las encuestas que un usuario actual puede realizar en todos los sitios. También se puede limitar a una encuesta específica indicando la identificación en la URL, a un sitio específico indicando los parámetros *siteId* o *siteReference*, o indicar que sólo se quieren obtener las encuestas que el usuario puede administrar. Para ello, hay que indicar el parámetro *admin* a Verdadero. Si se utiliza el parámetro *includeOptions* y se indica a Verdadero, junto con la respuesta se mandará la lista de opciones. Si se utiliza el parámetro *includeVotes* y se ponen a Verdadero, se mandará también la lista de votos de la encuesta. Si la petición se procesa de forma correcta, en la respuesta se indicará el código 200 y los datos pedidos. Si no se ha encontrado la dirección, se indicará el código 404, y si el formato no está disponible el código indicado será el 406.
 - POST: Crea una nueva encuesta con los datos indicados. Si se ha creado correctamente, en la respuesta se mandará el código 201 junto con la identificación de la entidad. Si se ha producido un error, se indica el código 400.
 - PUT: Actualizamos la encuesta, con los datos que se le indican. Se recibe en la respuesta el código 204 si la petición se ha procesado correctamente y el código 400 si se ha producido algún error.
 - DELETE: Borrarnos la encuesta indicada. Se devuelve el código 204 si se ha borrado correctamente y 404 si no se ha encontrado.
- POLL_OPTION (http://<SAKAI_ROOT>/direct/poll_option): Representa las opciones de una encuesta concreta. La estructura de datos asociada a la entidad es la siguiente:

Nombre	Tipo de datos	Descripción
Deleted	Lógico	Indica si la opción ha sido borrada
OptionId	Entero largo	Identificador de la opción
OptionText	String	Texto de la opción
PollId	Entero largo	Identificador de la encuesta a la que pertenece la opción
Status	String	Estado de la opción
Text	String	Texto de la opción
UUID	String	UID de la opción

Los métodos disponibles para el servicio REST son los siguientes:

- GET: Devuelve las opciones de la encuesta indicada, o si se indica en la URL la opción concreta indicada. Devuelve en la respuesta el código 200 y los datos si la petición se ha procesado correctamente, que se pueden pedir en formato json o xml. Devuelve el código 404 si no se ha encontrado o 406 si el formato pedido no está disponible.
 - POST: Crea una nueva opción para la encuesta que hemos indicado mediante sus identificadores. Si la petición se ha procesado correctamente, en la respuesta se mandará el código 201 junto con la identificación de la opción. Si se ha producido un error, se indica el código 400.
 - PUT: Actualiza la opción de la encuesta indicada. Devuelve el código 204 si se ha procesado correctamente, o el código 404 si los datos de entrada son inválidos.
 - DELETE: Borra la opción de la encuesta. Devuelve el código 204 si la opción se ha borrado correctamente, y 404 si no se ha encontrado.
- POLL_VOTE (http://<SAKAI_ROOT>/direct/poll_vote): Representa los votos que los usuarios envían como respuesta de la encuesta. Tiene asociada la siguiente estructura de datos:

Nombre	Tipo de datos	Descripción
Id	Entero largo	Identificador del voto
Ip	String	Dirección IP desde la que se ha emitido el voto
PollId	Entero largo	Identificador de la encuesta que estamos procesando
PollOption	Entero largo	Identificador de la opción de la encuesta seleccionada por el usuario para votar
SubmissionId	String	Identificación de la acción asociada al voto por parte del usuario.
UserId	String	Identificador del usuario que ha realizado el voto
VoteDate	Fecha	Fecha de emisión del voto

Para este servicio se pueden realizar dos acciones:

- GET: Devuelve los votos emitidos de una encuesta, o si se indica el identificador puede devolver un voto concreto. Los formatos soportados son json y xml. Como en el resto de servicios, en la respuesta se devuelve el código 200 y los datos si se ha procesado la petición correctamente. Si no, devolverá el código

404 si no se ha encontrado o el 406 si el formato no está disponible.

- POST: Emite un voto de una encuesta determinada. También como el resto de métodos POST, se devolverá el código 201 y la identificación de la entidad si la petición ha tenido éxito, o el código 400 si los datos enviados son inválidos.
- SAM_PUB (http://<SAKAI_ROOT>/direct/sam_pub): Este servicio REST es el único que existe para el tratamiento de exámenes generados mediante la herramienta SAMIGO, y es un método muy simple, que no dispone de estructura de datos asociada, y sólo dispone de un método que se puede utilizar:
 - GET: Se debe mandar la identificación del examen que queremos mostrar, y en realidad lo que se devuelve es un enlace hacia una página donde se puede realizar el examen.

Esto hace que este servicio no sea nada recomendable para una aplicación móvil, puesto que no nos va a permitir cambiar el formato que se va a mostrar al usuario, y que no está optimizada para los dispositivos que vamos a utilizar.

5.2. Especificación de requisitos

Uno de los objetivos que se buscan con el desarrollo es que la aplicación sea fácilmente ampliable y mantenible, para poder implementar nuevas funcionalidades y mejorar las existentes. Hemos organizado los requisitos en varios grupos, que en la aplicación se verá reflejado como distintas unidades funcionales:

- **Autenticación y acceso:** Lo formarán la pantalla de login y el menú de la aplicación, que dará al resto a todas las funcionalidades.
- **Encuestas:** Esta unidad contendrá la funcionalidad de gestión de encuestas. Permitirá mostrar una lista de las encuestas disponibles, responder, ver los resultados de una encuesta, permitir el trabajo fuera de línea... La lista completa de requisitos se detalla más adelante en este capítulo.
- **Cuestionarios/exámenes:** En esta unidad funcional se mantendrá la gestión de exámenes. El objetivo debería ser mostrar una lista de exámenes y permitir la respuesta de cada uno, teniendo en cuenta el tipo de cada pregunta, También debería permitir guardar los exámenes fuera de línea, para subir las respuestas posteriormente cuando exista conexión, pero vamos a estar limitados a lo que se pueda hacer mediante el servicio web que se dispone, por lo que va a ser difícil conseguirlo tal y como está diseñado en este momento.
- **General:** En este apartado se indicarán los requisitos comunes a todas las unidades funcionales, como requisitos de estilos y la internacionalización (I18N).

A continuación detallaremos los requisitos que deberemos implementar en cada unidad funcional.

5.2.1. General

1. Todas las ventanas de la aplicación seguirán los mismos estilos para dar sensación de homogeneidad al usuario.

2. La aplicación debe permitir internacionalización (I18N). Todos los textos, tanto los propios del sistema que utilicemos como los que se definan dentro de la aplicación deben de estar parametrizados para poder indicar las traducciones en ficheros separados.
3. Si es posible, la internacionalización se realizará dependiendo del idioma que tenga configurado el navegador, de forma que al usuario se le muestren automáticamente los textos en el idioma de su dispositivo.

5.2.2. Autenticación y acceso

Los requisitos que debe cumplir el módulo de autenticación y acceso serán los siguientes:

Pantalla de login

1. Debe existir una pantalla de autenticación de usuarios (login). En esa pantalla se permitirá indicar el usuario y contraseña, y debe existir un botón para realizar la conexión con Sakai, con las credenciales indicadas.
2. El campo de contraseña debe ser de tipo Password, de forma que no se muestren los caracteres introducidos.
3. Si no se indica usuario se mostrará una opción avisando de que es obligatorio indicarlo.
4. Al pulsar el botón para realizar la conexión, si se la llamada a la creación de sesión es correcta, guardaremos el identificador como variable de sesión.
5. Se debe permitir guardar usuario y contraseña cuando se produzca una conexión correcta. Existirá una opción de selección para indicar si quieres guardar los datos.
6. Antes de mostrar la pantalla de login se comprobará si existen credenciales guardadas previamente. Si existieran, se deberían indicar en los campos correspondientes, o preferiblemente crear una sesión automáticamente y redirigir a la pantalla del menú.
7. En una conexión correcta, si la opción de guardar la contraseña está marcada se guardarán. Si no está guardada, se borrarán si existieran previamente.
8. Si la conexión devuelve un error, se mostrará por pantalla al usuario, mediante una ventana emergente.
9. Debe haber una opción que permita indicar que se va a trabajar fuera de línea, de forma que no se necesite sesión (y por lo tanto tampoco usuario ni contraseña), y se trabaje solamente con los elementos almacenados anteriormente.
10. Si se selecciona la opción de trabajo fuera de línea, se deben proteger los campos de usuario y contraseña.
11. En la parte superior habrá un enlace para acceder a la pantalla de Acerca de... donde se mostrarán el nombre y versión de la aplicación, y los créditos.
12. En la pantalla del menú, existirán opciones para acceder a las funcionalidades de encuestas y cuestionarios.

Pantalla de menú

13. Las opciones tendrán forma de botón, y debe indicarse una imagen distintiva en su interior.
14. Existirá un botón de logout, que debería realizar una desconexión con Sakai, y mostrar la pantalla de login. Si el usuario y contraseña están guardados, se deben indicar en los campos de usuario y contraseña, y marcar la opción de Guardar contraseña.

5.2.3. Gestión de encuestas

Dividiremos los requisitos entre las dos pantallas que van a formar este módulo, que serán la lista de encuestas y la pantalla que permite responderla y consultar las respuestas.

Lista de encuestas

1. Se debe mostrar en primer lugar una lista con las encuestas disponibles para el usuario que se ha logueado en Sakai.
2. Si el usuario se ha conectado para trabajar fuera de línea, se mostrarán solamente las encuestas que previamente se hayan guardado en el dispositivo.
3. En cada elemento de la lista se debe indicar si la encuesta está guardada offline o si estamos trabajando online.
4. Las encuestas online se recuperarán llamando al servicio web correspondiente. Las encuestas offline se recuperarán del sistema de almacenamiento del dispositivo.
5. Se debe permitir guardar offline cada encuesta, o volver a indicar que se va a dejar de trabajar offline para hacerlo online.
6. La opción de guardar offline las encuestas debe estar desactivada cuando se haya logueado para trabajar online.
7. Al pulsar sobre un elemento de la lista, debe ir a la pantalla que permita trabajar con la encuesta
8. Debe existir un botón en la cabecera para volver a la pantalla del menú.

Trabajando con una encuesta

9. En la pantalla para trabajar con la encuesta se mostrará la pregunta y las opciones permitidas para contestarla
10. Si se permite más de una respuesta, se mostrará una frase que avise del número máximo de respuestas posibles
11. Si se permite una sola respuesta posible, las opciones se formatearán mediante botones de radio. Si se puede indicar más de una respuesta, se utilizarán casillas de selección (checkboxes)
12. Existirán tres botones para realizar acciones relativos a la encuesta: Votar, Vaciar opciones, Mostrar resultados

13. Si se ha logueado el usuario en el sistema para trabajar offline, el botón de Votar estará oculto.
14. Al pulsar el botón de Votar, se enviará la respuesta al servidor para registrar el voto emitido
15. Si una encuesta ya ha sido votada, las opciones se deben proteger, indicando la que el usuario votó. Además el botón de Votar y el de Vaciar opciones debe de ocultarse, y mostrar un mensaje al usuario diciendo que el voto se ha emitido correctamente.
16. Si se intenta votar y se han seleccionado más opciones de las permitidas, se debe mostrar un mensaje de error al usuario “Ha seleccionado demasiadas opciones”
17. Si pulsa el botón de Vaciar opciones, todas las opciones que se hayan seleccionado se desmarcarán.
18. Si se pulsa el botón de Mostrar resultados, se mostrará en un popup un gráfico con el número de respuestas para cada opción de la encuesta con la que estamos trabajando
19. Se debe poder seleccionar el tipo de gráfico que queremos mostrar: gráfico de barras o tipo tarta.

5.2.4. Gestión de cuestionarios/exámenes

Debido a que el servicio web del que disponemos para atacar a la herramienta de exámenes (SAMIGO) es tan básico, nos va a dar muchos problemas para desarrollar una aplicación móvil, al contrario que sucede con la herramienta de encuestas (POLL). El servicio web del que disponemos (SAM_PUB) se comporta como un enlace al programa de respuesta de exámenes, y no tiene entidades implementadas, por lo que no tenemos ninguna forma de recoger sus propiedades. Tampoco tenemos ninguna forma de obtener la lista de exámenes que tenga disponibles un alumno, porque el servicio nos obliga a indicar su identificación.

Por lo tanto, como prueba de concepto vamos a preparar una página que contenga un marco. A ese marco le indicaremos que cargue lo que nos devuelva el servicio SAM_PUB. Indicaremos la identificación de un examen que tengamos definido, para comprobar al menos que podemos usarlo para mostrar el resultado en una pantalla de dispositivo móvil. También podremos entonces comprobar hasta qué punto podemos personalizar la respuesta del servicio.

5.3. Prototipos desarrollados

Para poder hacernos una idea del funcionamiento y el modo de desarrollo utilizando los frameworks móviles, se han realizado unos prototipos de la aplicación final. Además de permitirnos evaluar con conocimiento de causa los entornos, nos servirá como entrenamiento para cuando tengamos que desarrollar la aplicación completa, con la conexión a la herramienta Sakai. Los frameworks que vamos a poner a prueba son los que hemos detallado anterioremente: LingoJS y jQuery Mobile.

Para realizar los prototipos hemos relajado las especificaciones que vamos a implementar, y los vamos a dejar en los siguientes:

- Dispondrán de una pantalla de login para indicar usuario y password, pero no realizarán conexión a ningún sistema
- A continuación se mostrará una lista de cuestionarios que se pueden seleccionar. Dicha lista se obtendrá mediante llamadas AJAX que los frameworks deben realizar, aunque los datos estarán alojados en el entorno local en archivos de tipo JSON. La estructura del archivo JSON se preparará de forma que nos de toda la información necesaria, en el formato más sencillo para la implementación.
- Al seleccionar un cuestionario se mostrará una nueva pantalla con la primera pregunta del cuestionario. Las cuestiones se obtendrán también de archivos JSON alojados en local, mediante llamadas AJAX
 - Los tipos de pregunta que se van a tener en cuenta son las siguientes:
 - Respuesta corta: Para que los usuario respondan se dispondrá de un cuadro de texto
 - Completar los espacios: Se utilizarán tantos cuadros de texto como espacios se puedan rellenar
 - Respuesta numérica: Se utilizará un control tipo rango para indicar la respuesta.
 - Selección: Si sólo se puede seleccionar una respuesta se utilizarán botones de radio o controles de selección. Si se puede seleccionar más de una, se utilizarán checkboxes
 - Se deberán disponer de botones para navegar entre las distintas cuestiones. Además, en la cabecera se indicará un botón que mostrará una lista con todas las cuestiones. Se permitirá seleccionarlas para navegar rápidamente hasta la cuestión seleccionada.

5.3.1. LungoJS

Como ya hemos indicado anteriormente, hemos preparado unas pantallas de ejemplo en este framework. La programación es bastante sencilla, y además al estar basado en las nuevas sentencias diseñadas en el estándar HTML5 da una sensación de producto con tecnología puntera.

En las imágenes siguientes se puede ver el aspecto que tiene la aplicación del prototipo realizada con este framework. El tema visual que se ha utilizado es el que viene por defecto, y se puede modificar con mucha facilidad, así que no vamos a tener en cuenta este aspecto (que la verdad es que no me gusta demasiado) a la hora de evaluarlo.

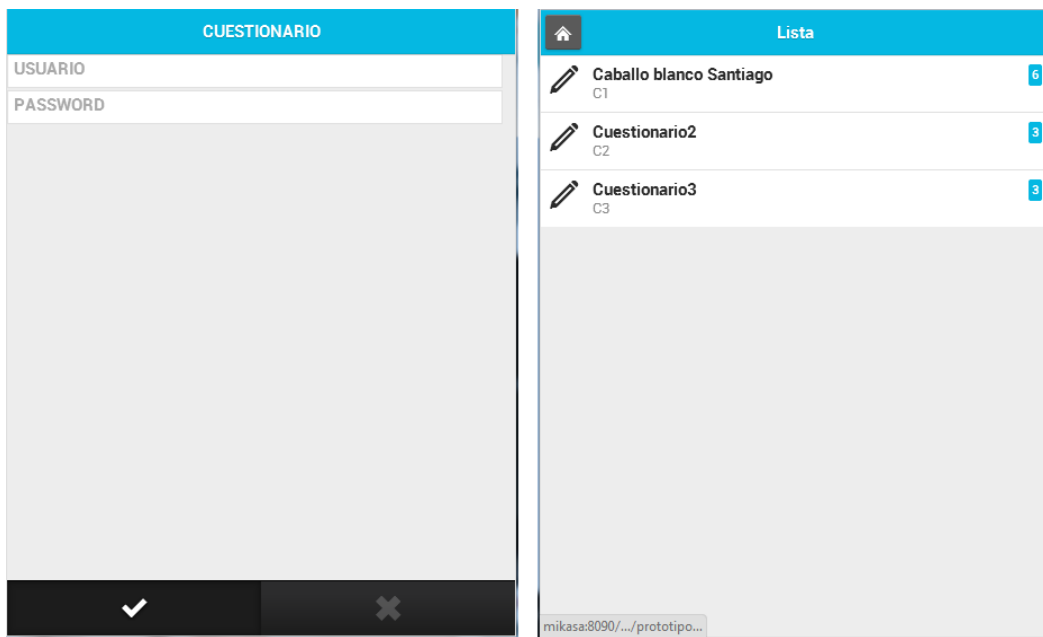


Ilustración 26: Prototipo LungoJS. Pantalla de login y lista de encuestas

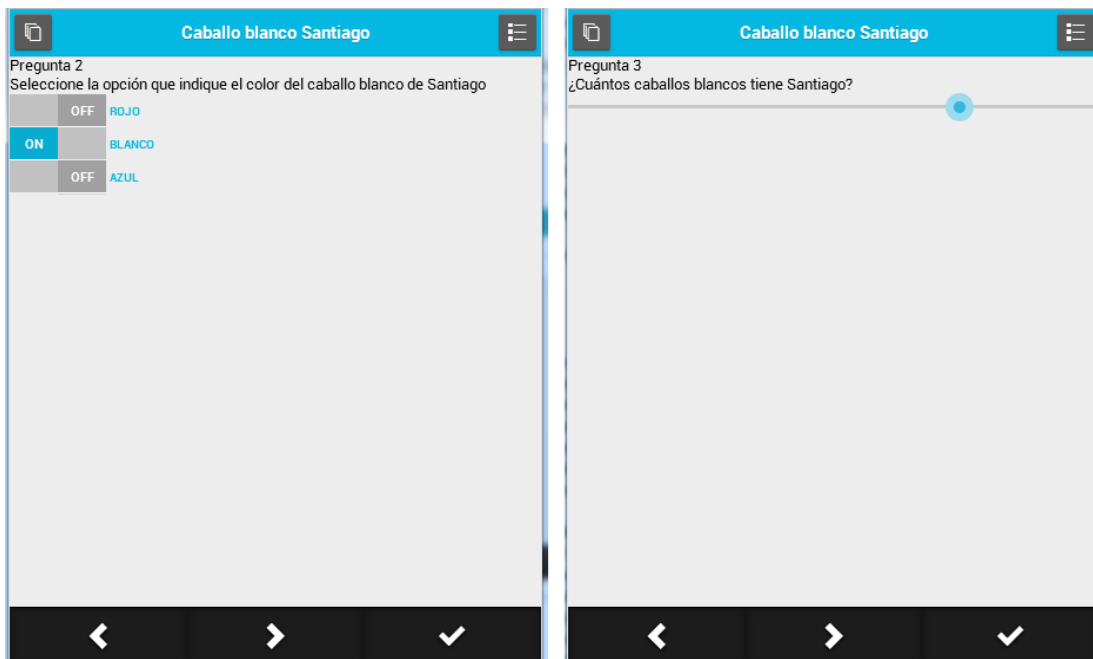


Ilustración 27: Prototipo LungoJS. Tipos de preguntas

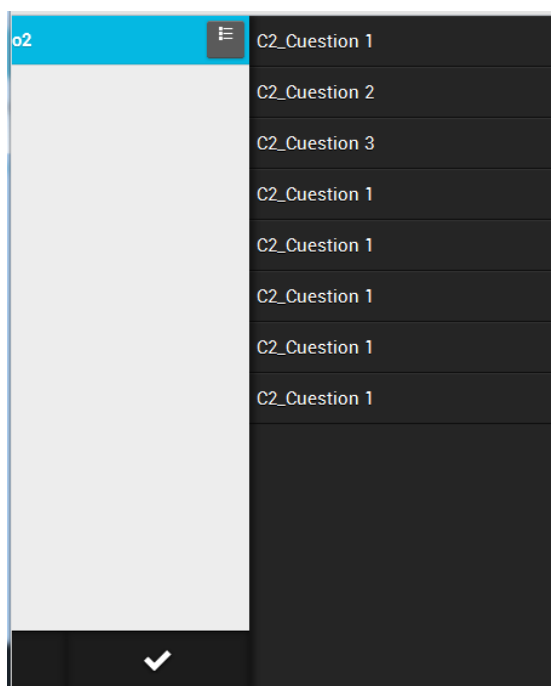


Ilustración 28: Prototipo LungoJS. Lista de cuestiones

El control que se ha utilizado para mostrar la lista de cuestiones es un ASSET, y su comportamiento me ha impresionado. Si la ventana disponible tiene un tamaño lo suficientemente grande (como por ejemplo en una tableta) se muestra directamente el menú. En cambio, si la ventana tiene poco tamaño (como en un smartphone) te aparece un botón en la barra de título, que al pulsarlo despliega el menú con un efecto.

Desgraciadamente, al ser un framework muy nuevo tiene algunos fallos e inconveniente que seguro que serán corregidos en futuras versiones. Por ejemplo, al desplegar el ASSET la imagen del botón que lo contiene se redimensiona y se hace más pequeña. El uso de la base de datos web es bastante engorroso, puesto que todas las operaciones que puedes hacer con ella son asíncronas, por lo que no puedes hacer más de una acción sin tener que encadenar funciones de callback.

Pero el mayor problema que he tenido ha sido al intentar crear artículos de manera dinámica, puesto que quería crear tantos artículos como preguntas tuviera mi cuestionario. En este caso, Lungo no proporciona ningún método para redibujar una página que hayas introducido en el DOM, por lo que, a pesar de que se intenta dibujar correctamente, se producen fallos en los controles que hacen que en algunas situaciones no funcionen correctamente, por ejemplo en las listas desplegadas.

Por lo tanto, la conclusión es que aunque es un framework prometedor, todavía le falta tiempo para convertirse en un sistema robusto para realizar desarrollos serios para aplicaciones móviles.

5.3.2. JQuery Mobile

JQuery Mobile no tiene una filosofía tan radical sobre HTML5 como LungoJS, seguramente debido a que intenta soportar una amplísima mayoría de dispositivos móviles que existen en el mercado. Su programación por lo tanto es ligeramente más

engorrosa a la hora de programar el HTML de la página estática. En cambio, la biblioteca javascript en la que está sostenida (jQuery) impresiona por la sencillez con la que se puede tomar el control total de la página y la extraordinaria potencia que tiene.

En cuanto a los temas visuales, los que vienen por defecto son mucho más elegantes y sobrios que el proporcionado por Lungo. Pero es que jQuery Mobile dispone de una herramienta (Themroller) que te ayuda a crear el estilo que desees. Y no sólo eso, puesto que permite indicar múltiples estilos, que se identifican mediante una letra (de la A a la D, lo que hace un total de 4 estilos simultáneos). De todas formas, para nuestro prototipo utilizaremos los estilos estándar proporcionados. Estos son los ejemplos de cómo queda el prototipo implementado en jQuery Mobile

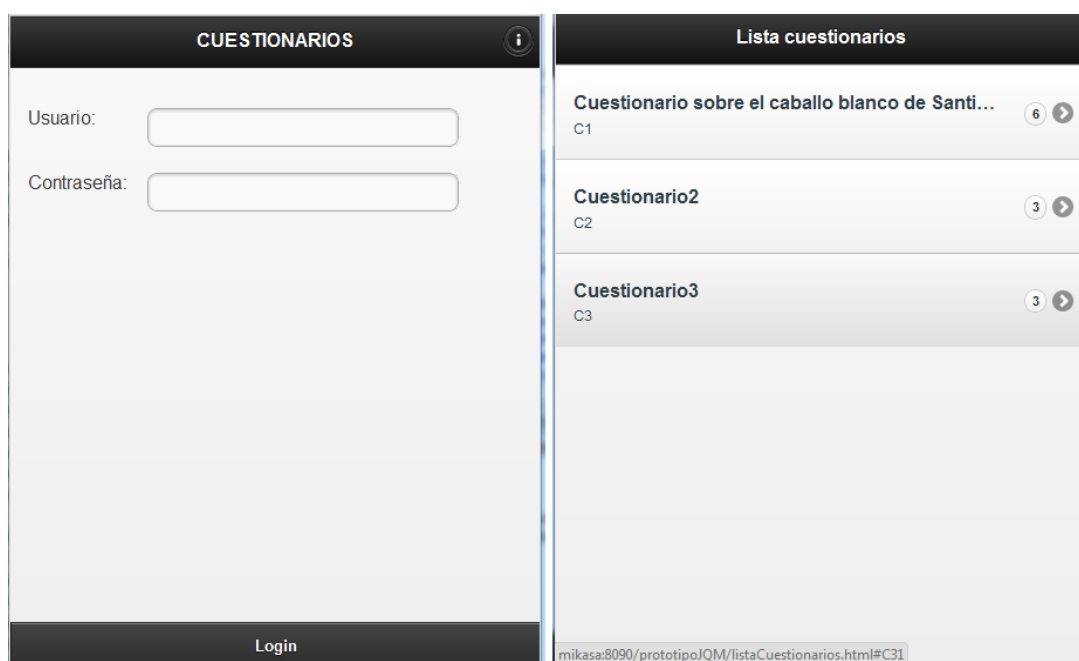


Ilustración 29: Prototipo JQueryMobile. Login y lista de encuestas



Ilustración 30: Prototipo JQuery Mobile. Tipos de preguntas

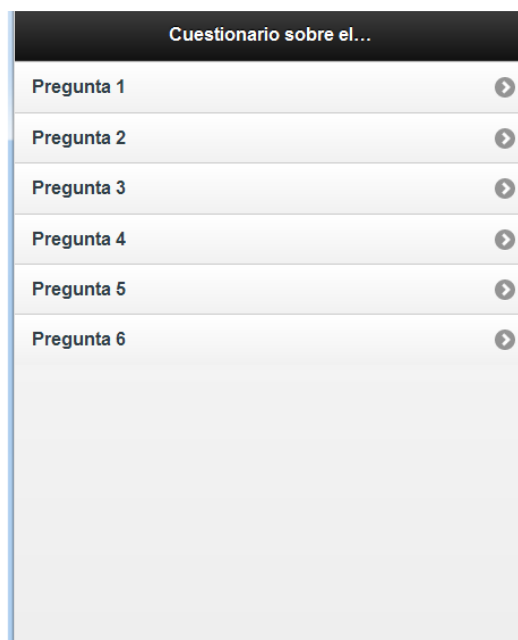


Ilustración 31: Prototipo JQueryMobile. Lista de preguntas

A diferencia de LungoJS, JQueryMobile sí que permite definir páginas enteras de manera totalmente dinámica, pues dispone de un mandato que, dado un código en HTML, analiza la página y la transforma para añadirle los estilos que tendrá una página tras ser mejorada mediante el framework. Esto, unido a la potencia que te da JQuery, hace que tengas un control absoluto de la forma en que se va a dibujar la ventana

Por contra, no existe un control parecido a lo que nos ofrecía el ASSET de Lungo. De todas formas, si para la aplicación final nos hiciera falta, podríamos buscar entre la gran base de plugins disponibles, por lo que no sería un gran problema, mientras que la base de plugins de Lungo sí que es reducida de verdad.

Por lo tanto, creemos que JQueryMobile tiene más puntos a favor que LungoJS para implementar el proyecto, por lo que este será el framework elegido.

5.4. Diseño de interfaces externas

Para diseñar la interfaz vamos a utilizar una herramienta que nos permita hacer un mockup de la aplicación. Un mockup en un borrador de la interfaz que queremos conseguir, es decir, una forma de plasmar la idea que tenemos en la cabeza en un boceto con la representación de la interfaz a realizar.

Existen diversas herramientas que permiten hacer mockups de aplicaciones, algunas con un aspecto más formal y otras con un acabado llamado *wireframe*, que hacen que el resultado se asemeje más a un dibujo realizado en una servilleta, de forma que se acentúe la sensación de estar delante de un boceto. Hemos elegido la herramienta Pencil (<http://code.google.com/p/evoluspencil/>), que tiene las siguientes características:

- Es gratuita
- Existe como aplicación de escritorio para windows, o como una extensión del navegador Firefox
- Se le pueden añadir nuevas galerías de elementos, entre las que se encuentran controles para realizar bocetos de interfaces móviles

La primera pantalla que diseñaremos será la de la página de identificación. Debemos tener campos de entrada para indicar usuario y password, dos botones de selección para las opciones de Guardar password y Trabajar offline, y un botón para realizar el login:

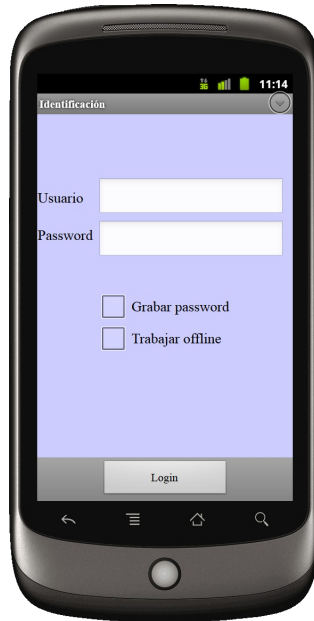


Ilustración 32: Mockup con la pantalla de Login

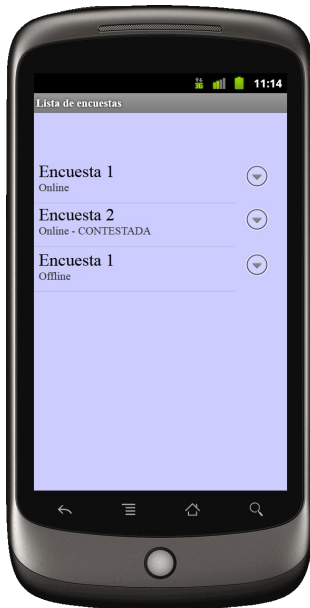
A continuación diseñaremos la pantalla con el menú de la aplicación, que dispondrá de dos botones, uno para acceder a las encuestas y otro a los cuestionarios.



Ilustración 33: Pantalla con el menú tras el login

La siguiente pantalla mostrará una lista con las encuestas. En cada opción de la lista se mostrará un botón que al pulsarlo, abrirá una ventana de tipo popup con las

propiedades de la encuesta. Estas propiedades de limitarán, por ahora, a indicar si vamos a guardar la encuesta de manera offline o no.



*Ilustración 35:
Pantalla con la lista de encuestas*



Ilustración 34: Popup de opciones de una encuesta

La pantalla de la encuesta concreta puede variar. Si existe una sólo opción que un usuario pueda indicar como voto, se mostrarán las opciones mediante botones de radio. Si existe más de una, se indicarán checkboxes, junto con un texto que indique el número de opciones que se pueden seleccionar.

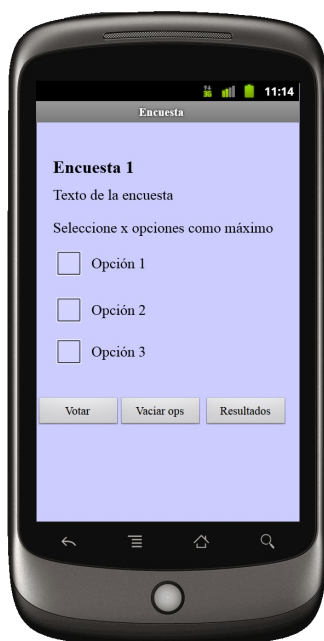


Ilustración 36: Mockup con pantalla de encuestas con checkboxes

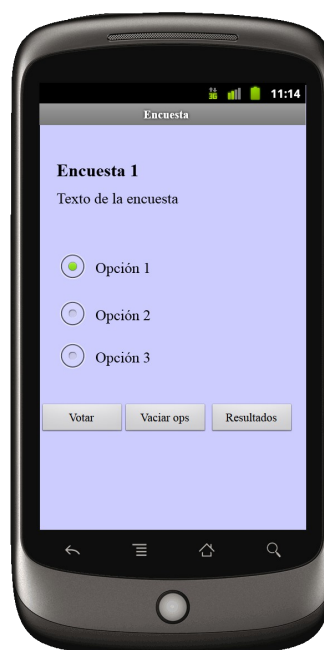


Ilustración 37: Mockup con pantalla de encuestas con botones de radio

En cualquier caso, en la parte inferior de la ventana se indicarán tres botones: uno para votar, otro para limpiar la ventana y otro para mostrar los resultados de la encuesta

6. Implementación

Para la implementación de este proyecto vamos a necesitar realizar dos aplicaciones distintas. La primera será un proxy de conexión con los servicios web disponibles en Sakai, que vamos a necesitar para sortear el problema de las llamadas cross-domain (que explicaremos a continuación). La segunda será la aplicación HTML5 con la interfaz móvil, desarrollada mediante el framework JQuery Mobile.

6.1. Proxy de conexión con SAKAI

6.1.1. El problema de las llamadas cross-domain

Cuando se realiza una llamada AJAX, o desde una aplicación con tecnologías como Flash o Siverlight a un servicio desde cualquier aplicación web, si el servidor en el que está alojado el servicio es el mismo que el que aloja la aplicación web no existirá ningún problema, puesto que se considera el funcionamiento normal. En el caso de que el servicio resida en otro servidor, los navegadores modernos (que por supuesto incluyen los que están disponibles en las plataformas móviles de los smartphones) disponen de mecanismos de seguridad que evitan realizar directamente este tipo de llamadas. De este modo se intenta evitar dos tipos de ataques muy habituales que utilizan este mecanismo:

- Cross-Site Request Forgery (XSRF o session riding): Se produce cuando un sitio web malicioso se apodera de información de contexto disponible en el navegador (como cookies o información del localStorage) y la utiliza para suplantar al usuario y acceder a otra aplicación en su nombre (como servicios de banca on-line, redes sociales...)
- Cross-Site Scripting (XSS): Este ataque consiste en que desde un sitio web se inyecta un código javascript que se encarga de robar la información de contexto y enviarla a un sitio de un tercero.

De todas formas, existen diversas estrategias que se pueden seguir para poder realizar este tipo de llamadas. Las más utilizadas son las siguientes:

- JSONP (JavaScript Object Notation with Padding): Se basa en la capacidad que tienen los navegadores de añadir scripts de otros dominios para acceder a la información. De esta forma se puede solucionar el problema de las llamadas a otro dominio.
 - Cuando un servidor expone un servicio para ser utilizado mediante JSONP, lo que hace es encapsular la información que va a devolver dentro de una función, de forma que el cliente al recibir la respuesta, podrá ejecutarla y recoger la información requerida.
 - La función debe existir en el entorno cliente, por lo que el servidor aceptará un parámetro en la llamada (es decir, un parámetro GET) que le indique el nombre de la función que debe utilizar. Usualmente, el parámetro que se utilizar se llama callback.
 - JQuery permite llamadas a JSONP sin más que indicarle el tipo de datos "jsonp" a las llamadas AJAX. Si no se indica otro nombre, se generará el nombre de

manera aleatoria, por lo que no hará falta siquiera crear la función de callback en el entorno cliente.

- De todas formas, existen varias desventajas en este método:
 - Sólo permite llamadas GET, por lo que sólo se pueden pasar parámetros mediante la URL. Por lo tanto, no se podrá hacer de este modo la subida de ficheros, puesto que requiere una petición POST con un encriptado de tipo “mutipart/form-data”.
 - Debido a las limitaciones de los navegadores, existen unos límites para el tamaño de las URL que hay que tener en cuenta a la hora de pasar parámetros: en Internet Explorer oficialmente son 2083 caracteres (aunque parece que en la práctica el límite es mayor), en Firefox 65.536 en teoría y más de 100.000 en la práctica, en Safari 80.000...
 - No se permiten realizar llamadas síncronas. Todas las llamadas JSONP en jQuery se tratarán de modo asíncrono.
 - Sólo permite devolver como tipo de datos JSON, por lo que no se puede usar para devolver contenido en HTML, XML, ATOM...
 - Esta técnica no deja de ser un hack que se ha desarrollado de forma que se pueda “engañar” a los navegadores para aceptar llamadas cross-domain, no es una técnica expresa de seguridad.
- CORS (Cross-Origin Resource Sharing): Es una especificación del W3C, mediante la que durante la llamada al servicio se envía la información del origen de la solicitud. El servidor entonces aprobará o denegará la solicitud de servicio tras analizar la información de origen que se había mandado.
 - Para implementar esta técnica, en la respuesta de una llamada se indican parámetros en la cabecera para indicar los orígenes y métodos permitidos:
 - El parámetro “Access-Control-Allow-Origin” indica los dominios de origen permitidos. Por ejemplo, el valor “Access-Control-Allow-Origin:*” indica que se permite el acceso a cualquier dominio (lo que no está recomendado porque también permite ataques de tipo XSS), mientras que el valor “Access-Control-Allow-Origin: dominio1 dominio2” indica que sólo los dominios “dominio1, dominio2” están permitidos.
 - El parámetro “Access-Control-Allow-Methods” indica las operaciones que se pueden realizar mediante la llamada cross-origin. Por ejemplo, el valor “Access-Control-Allow-Origin: GET,POST,DELETE” permite hacer peticiones de tipo GET,POST y DELETE, pero no permitirá las peticiones de tipo PUT o OPTIONS.
 - Esta técnica es más moderna que JSONP, y supera todas las limitaciones que poseía esta otra técnica:
 - Soporta todos los métodos de petición (GET, POST, DELETE, PUT...) y no sólo GET como en JSONP
 - Permite utilizarse para devolver cualquier tipo de datos, no está limitado al

formato JSON.

- Permite a los programadores hacer un uso normal del objeto XMLHTTPRequest, por lo que se realiza un mejor tratamiento de errores que mediante JSONP.
- Permite a los sitios web generar las respuestas para asegurar la seguridad, ante posibles ataques XSS
- Como inconveniente, el navegador debe permitir esta técnica, por lo que navegadores antiguos sin soporte a CORS no podrán usarlo. De todas formas, los navegadores más utilizados (Firefox, Safari, Chrome, Internet Explorer...) soportan, aunque sea parcialmente, esta especificación.

Para el proyecto hemos decidido utilizar la técnica Cross-Origin Resource Sharing. No sólo porque sea una técnica más moderna y potente, sino que como lo vamos a utilizar para generar un proxy de llamadas REST, vamos a buscar que las llamadas a este proxy sean lo más parecidas posibles a lo que sería hacerlas directamente a Sakai. Por lo tanto nos será necesario utilizar no sólo el método GET (para recoger los resultados) sino también el POST (para por ejemplo enviar los votos al sistema) o el DELETE (para desconectarnos de la sesión). Además, los dispositivos móviles con los sistemas más utilizados (Android, iPhone, Windows Phone...) disponen de navegadores que soportan CORS, por lo que en principio no habrán problemas técnicos que nos impidan utilizar esta técnica.

6.1.2. Tecnologías utilizadas

Para la realización de la aplicación del proxy con Sakai, hemos decidido preparar una aplicación Java que se pueda publicar en un servidor web. Se ha elegido esta alternativa en lugar de otras (como por ejemplo un servidor PHP o .NET) debido a que queríamos tener una aplicación realizada en una tecnología no muy diferente al que se debe disponer para poner en marcha Sakai. Esto nos dará una serie de ventajas como las siguientes:

- Como Sakai es una aplicación J2EE programada en su mayor parte en Java que se ejecuta en un servidor de aplicaciones (como por ejemplo Tomcat, que es el que se utiliza para su versión DEMO), al preparar una aplicación Java susceptible de instalarse en un servidor de aplicaciones similar al que está desplegado Sakai nos permitirá, en el caso que nos interese, desplegar Sakai y el proxy en el mismo servidor.
- En el caso de que se quiera utilizar el código desarrollado en el proxy para incluirlo directamente en Sakai como una modificación, será más sencillo realizarlo que si se ha preparado en otra tecnología distinta.

Como entorno de desarrollo (IDE), vamos a utilizar Eclipse edición EE, que está mejor preparada para aplicaciones web y Enterprise, en su versión Indigo, que es un entorno bien conocido, además de software libre.

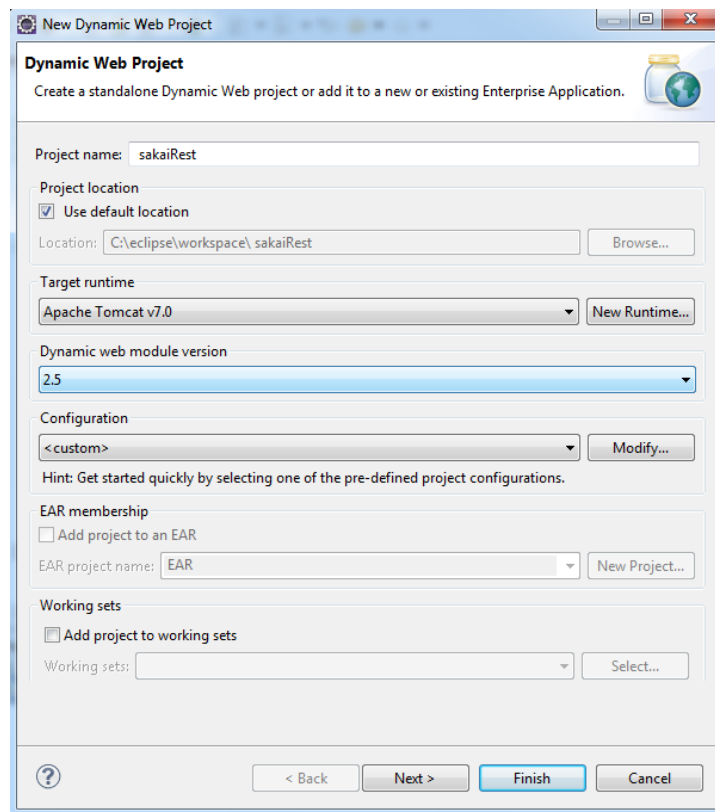


Ilustración 38: Configuración de proyecto en Eclipse para el proxy de Sakai

Para comenzar el proyecto, prepararemos un nuevo proyecto de tipo Dynamic Web Application. Como nombre le indicaremos **SakaiRest**. En la pantalla siguiente se muestra las propiedades que hemos seleccionado para la creación del proyecto. En el apartado Dynamic Web Module Version, hemos seleccionado la 2.5, y el motivo es que esta versión es compatible con la versión 5 de Java, mientras que la versión 3.0 de Dynamic Web Module Versión lo es con la 6. Debido a que en la instalación de Java nos recomendaban utilizar la versión 5, por motivos de compatibilidad hemos preferido que todo el software que desarrollemos sea también compatible con esta versión.

Para implementar el proxy de los servicios REST, utilizaremos el framework Jersey. Este desarrollo es la implementación de referencia (aunque existen otras como Apache CXF) de la especificación JAX-RS, definida por SUN, para el desarrollo de servicios REST en Java mediante clases java y anotaciones. El documento de especificación que implementa es el JSR-311.

Para poder utilizar Jersey, primeramente deberemos descargar las bibliotecas que vamos a incluir el proyecto. Para ello nos dirigiremos a la siguiente dirección y descargaremos el archivo:

<http://download.java.net/maven/2/com/sun/jersey/jersey-archive/1.4/jersey-archive-1.4.zip>

En su interior se encuentran los siguientes archivos, que incluiremos en la carpeta WEB-INF/lib de nuestro proyecto Eclipse:

- asm-3.1.jar

- jackson-core-asl-1.5.5.jar
- jackson-jaxrs-1.5.5.jar
- jackson-mapper-asl-1.5.5.jar
- jackson-xc-1.5.5.jar
- jersey-client-1.4.jar
- jersey-core-1.4.jar
- jersey-json-1.4.jar
- jersey-json-1.4.jar
- jettison-1.1.jar
- jsr311-api-1.1.1.jar

Ahora vamos a configurar los parámetros de la aplicación para registrar el servlet que se va a ejecutar al desplegarla en el servidor. Para ello, nos dirigiremos al fichero WEB-INF/lib/web.xml y le añadiremos el siguiente código XML, dentro del apartado web-app:

```
<servlet>
  <servlet-name>Jersey REST Service</servlet-name>
  <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>com.sun.jersey.config.property.packages</param-name>
    <param-value>com.jcb.rest.servicios</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>Jersey REST Service</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

En esta configuración, indicamos la clase que se va a ejecutar como servlet mediante el apartado **<servlet-class>**. En este caso, se trata de una clase contenedora definida en la implementación jersey.

También definimos el parámetro **com.sun.jersey.config.property.packages**, que indica el nombre de los paquetes donde se encontrarán las clases java con la definición de los servicios REST que vamos a desarrollar. En nuestro caso, vamos a incluir en el paquete **com.jcb.rest.servicios**, por lo que indicaremos este valor en el parámetro.

El apartado **<load-on-startup>** va a indicar que se carguen las clases en el momento de arrancar la aplicación en el servidor. Esto hará que la ejecución sea más rápida.

Por último, vamos a definir la base de las direcciones que vamos a estar escuchando. En el apartado **<servlet-mapping>** vamos a definir el parámetro **<url-mapping>** con el valor **"/rest/*"**. Esto hará que la aplicación acepte las peticiones que se dirijan al siguiente patrón de dirección URL:

`http://<dirección_del_servidor>/<nombre_aplicacion/rest/*`

Ahora configuraremos el sistema para preparar las respuestas para utilizar Cross-Origin Resource Sharing (CORS). Para ello, dentro del apartado **<servlet>** que acabamos de crear en el archivo web.xml vamos a indicar un nuevo parámetro que configurará la clase que se ejecutará cuando se haya preparado una respuesta, antes de enviarla al cliente:

```
<init-param>
  <param-name>com.sun.jersey.spi.container.ContainerResponseFilters</param-name>
  <param-value>com.jcb.rest.utils.ResponseCorsFilter</param-value>
</init-param>
```

La clase ResponseCorsFilter del paquete com.jcb.rest.utils añade a la cabecera de las repuestas los parámetros “Access-Control-Allow-Origin” y “Access-Control-Allow-Methods”. La implementación se muestra en el siguiente listado:

```
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.ResponseBuilder;

import com.sun.jersey.spi.container.ContainerRequest;
import com.sun.jersey.spi.container.ContainerResponse;
import com.sun.jersey.spi.container.ContainerResponseFilter;

public class ResponseCorsFilter implements ContainerResponseFilter {

    @Override
    public ContainerResponse filter(ContainerRequest req, ContainerResponse contResp) {

        ResponseBuilder resp = Response.fromResponse(contResp.getResponse());
        resp.header("Access-Control-Allow-Origin", "*")
            .header("Access-Control-Allow-Methods", "GET, POST, OPTIONS, DELETE");

        String reqHead = req.getHeaderValue("Access-Control-Request-Headers");

        if(null != reqHead && !reqHead.equals(null)){
            resp.header("Access-Control-Allow-Headers", reqHead);
        }

        contResp.setResponse(resp.build());
        return contResp;
    }
}
```

También vamos a preparar una clase en el paquete com.jcb.rest.utils con algunas constantes que vamos a utilizar, indicando:

- La URL donde se encuentran los servicios web del entorno de Sakai
- Una variable que indica si vamos a buscar las encuestas que el usuario puede administrar o en las que puede participar.

```
public class Constantes {

    public static String SakaiBaseURL = "http://192.168.0.101:8080/direct/";

    //0: para buscar todas las encuestas a las que tengas acceso. NO FUNCIONA
```

```
// CORRECTAMENTE
//1: para buscar las encuestas que puedes editar
public static String SakaiPollAdmin = "1";
}
```

En cuanto al servidor de aplicaciones, utilizaremos un servidor Apache Tomcat v7, que se integra perfectamente en el IDE Eclipse. Este servidor permite el despliegue automático de la aplicación en cuanto el sistema detecta que se ha producido algún cambio, lo que nos facilitará el trabajo cuando estemos escribiendo el código fuente.

6.1.3. Servicios de login

Vamos a preparar el servicio que se va a encargar de gestionar las sesiones con Sakai. Lo primero que haremos es preparar una clase en el paquete **com.jcb.rest.servicios**, al que vamos a llamar **JSONServiceLogin**. Esta clase va a ser un POJO (Plain Old Java Object), es decir, es un objeto que no extiende ninguna clase o implementa ninguna interfaz en especial, lo que facilita el mantenimiento al no tener que preocuparse de posibles dependencias. El listado siguiente muestra la creación de la clase, junto los Imports que vamos a necesitar para generar los servicios posteriormente.

```
import javax.ws.rs.DELETE;
import javax.ws.rs.FormParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.MultivaluedMap;

import com.jcb.entidades.ClientMensaje;
import com.jcb.rest.utils.Constantes;
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.ClientResponse;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.ClientConfig;
import com.sun.jersey.api.client.config.DefaultClientConfig;
import com.sun.jersey.core.util.MultivaluedMapImpl;

@Path("/login")
public class JSONServiceLogin {

}
```

La anotación `@Path("/login")` indica a Jersey que las peticiones que sean de la forma `URL_BASE` (definida en el archivo `web.xml`) + `/login'` serán tratadas por los métodos de esta clase.

A continuación vamos a definir dos métodos dentro de la nueva clase, uno con el servicio que realizará el login en Sakai y otro que se encargará del logout.

Vamos a crear entonces un método llamado `login`, que tendrá el siguiente código:

```
@POST
@Produces(MediaType.APPLICATION_JSON)
public ClientMensaje login(@FormParam("user") String user,
    @FormParam("password") String password) {
```

```

//Preparamos la llamada al servicio SAKAI
String url = Constantes.SakaiBaseURL + "session";
ClientConfig config = new DefaultClientConfig();
Client client = Client.create(config);
WebResource service = client.resource(url);

//Prepara los parámetros que se van a mandar con la petición a SAKAI
MultivaluedMap<String, String> formData = new MultivaluedMapImpl();
formData.add("_username", user);
formData.add("_password", password);

//Ejecutamos el servicio y recogemos la respuesta
ClientMensaje mensaje = null;
try {
    String response = service.post(String.class, formData);
    mensaje = new ClientMensaje(false, response);
} catch (Exception e){
    mensaje = new ClientMensaje(true, "Error al crear la sesión");
}
return mensaje;
}

```

La anotación POST significa que este método se ejecutará cuando el método HTTP de la petición sea POST. La anotación `@Produces(MediaType.APPLICATION_JSON)` indica que la respuesta que se va a generar va a ser de tipo JSON. Las anotaciones `@PathParam` indican que en la petición van a existir dos parámetros de formulario, donde uno contendrá el usuario y otro con la contraseña. Por lo tanto, la petición que deberá hacer la aplicación cliente si quiere ejecutar este servicio será la siguiente:

POST `http://<servidor>/sakaiRest/rest/login`; e indicar dos parámetros de formulario, uno llamado 'user' y otro 'password'

A continuación, prepararemos la llamada al servicio REST en Sakai, preparando un objeto de la clase `WebResource` (service) al que le pasamos la URL del servicio, y los parámetros de la llamada, mediante un objeto de la clase `MultiValuedMap`

Seguidamente ejecutamos el servicio, utilizando el método `post` del objeto `service`, y tratamos la respuesta que hemos recibido. Si no se produce ninguna excepción, prepararemos un objeto de tipo `ClientMensaje`, que no es más que una clase que hemos creado con dos parámetros: un booleano para indicar si se ha producido un error o no, y el mensaje que vamos a enviar al cliente. En el caso de no producirse ningún error, se devolverá el resultado de la llamada a Sakai, que contiene una cadena de caracteres con la identificación de la sesión. Si se produce alguno, típicamente porque se ha indicado un usuario o contraseña erróneos, se enviará el mensaje "Error al crear la sesión".

A continuación vamos a definir el método que se encargará de las desconexiones de sesiones en Sakai. Para ello, prepararemos un método llamado `logout` que tendrá el siguiente código:

```

@DELETE
@Produces(MediaType.APPLICATION_JSON)
@Path("/{sessionId}")
public ClientMensaje logout(@PathParam("sessionId") String sessionId) {
    String url = Constantes.SakaiBaseURL + "session/" + sessionId +
        "?sakai.session="+sessionId;
}

```

```
ClientConfig config = new DefaultClientConfig();
Client client = Client.create(config);
WebResource service = client.resource(url);

ClientMensaje mensaje = null;
try {
    ClientResponse response = service.delete(ClientResponse.class);
    if (response.getStatus() == 204) {
        mensaje = new ClientMensaje(false, "Logout correcto");
    } else {
        mensaje = new ClientMensaje(true,
            "Se ha producido un error en el logout");
    }
} catch (Exception e){
    mensaje = new ClientMensaje(true, "Se ha producido un error en el logout");
}

return mensaje;
}
```

El esquema de la clase es muy parecido al del servicio de login: se prepara el servicio con la URL del servicio REST de Sakai, se ejecuta y se prepara la respuesta (en este caso no hay parámetros que mandar junto con la petición). Las diferencias que existen son las siguientes:

- La anotación al inicio de la clase `@DELETE` indica que el método que estamos implementando va a responder ante peticiones del método HTTP DELETE
- La anotación `@Path({sessionId})` indica que a continuación de la URL de la petición se va a indicar un parámetro, que va a introducirse en el método logout mediante el parámetro `sessionId`. Esto se indica mediante la anotación `@PathParam` dentro de la signature del método logout. Por lo tanto, la petición que tendrá que hacer el cliente para ejecutar este servicio tendrá la siguiente forma:

DELETE http://<servidor>/sakaiRest/rest/login/<identificador de la sesión>

- Se ejecutará el método delete del servicio, y para analizar la respuesta se comprobará el estado de la misma. Si ese estado es 204 (NO CONTENT), significa que la respuesta es correcta. Si no, devolvemos la respuesta al cliente como un error.

6.1.4. Servicio de lista de encuestas

Para la implementación de este servicio vamos a crear la clase `JSONServiceListaEncuestas`, en el paquete `com.jcb.rest.servicios`. Anotaremos el path del servicio para tratar las peticiones a la dirección `listaEncuestas`, e implementaremos el método GET para recoger la lista de encuestas disponible para la sesión. En este servicio sólo tendremos este método, que es el único que necesitamos para gestionar la lista de encuestas:

```
@Path("/listaEncuestas")
public class JSONServiceListaEncuestas {

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public Object getEncuestasJSON(@QueryParam("sessionId") String sessionId) {
```

```

String url = Constantes.SakaiBaseURL + "poll?admin=" + Constantes.SakaiPollAdmin +
    "&sakai.session="+sessionId;

ClientConfig config = new DefaultClientConfig();
Client client = Client.create(config);
WebResource service = client.resource(url);

try {
    ClientResponse response =
        service.accept(MediaType.APPLICATION_JSON).get(ClientResponse.class);
    String resp = response.getEntity(String.class);
    return resp;
} catch (Exception e){
    ClientMensaje mensaje = new ClientMensaje(true,
        "Error al recoger la lista de encuestas");
    return mensaje;
}
}
}

```

Este método se ejecutará cuando se reciba una petición GET, y espera recibir un parámetro mediante la dirección, que se denota mediante la anotación `@QueryParam` en la signatura del método. Ese método se llamará `sessionId`, y contendrá la cadena con la identificación de la sesión con la que previamente nos hemos identificado. Por lo tanto, para ejecutar el servicio, el cliente debe preparar la siguiente petición al proxy:

GET `http://<servidor>/sakaiRest/rest/listaEncuestas?sessionId=<sessionId>`

La ejecución del servicio es igual que los servicios anteriores. Los parámetros se indican directamente en la URL de la llamada a Sakai al ser una petición GET, e indicamos dos:

- `admin`: Si vale 1, indica que vamos a recoger las encuestas que un usuario puede administrar. Si vale 0 o no se indica, significa que vamos a recoger todas las encuestas en las que el usuario puede participar.
- `sakai.session`: Aquí tenemos que indicar la identificación de la sesión con la que estamos ejecutando el servicio.

A continuación ejecutaremos el servicio, analizaremos la respuesta y mandaremos el resultado al cliente. Si la llamada se ha ejecutado correctamente, Sakai nos devolverá una cadena de caracteres con la lista de encuestas en formato JSON. Para ello, utilizaremos el objeto `ClientMensaje` como en todos los servicios que vamos a implementar. Si se produce un error, utilizaremos el mismo objeto de `ClientMensaje`, pero marcando la bandera de error y enviando el mensaje del mismo.

6.1.5. Servicio de encuesta

Para este servicio nuevamente vamos a crear una nueva clase en el paquete `com.jcb.rest.services` llamada `JSONServiceEncuesta`. Vamos a anotar el path de este servicio con la dirección `"/encuesta"`, y en su interior habrán dos métodos: uno para recoger una encuesta y otro para realizar un voto en la encuesta.

```

@Path("/encuesta")
public class JSONServiceEncuesta {

```



```
}
```

Para recoger una encuesta concreta, vamos a preparar un método llamado `getEncuestaJSON`, que va a devolver el resultado en formato JSON. Este método va a tener dos parámetros, que se van a indicar de formas distintas:

- Un parámetro se indicará en el path de la URL, y contendrá la identificación de la encuesta que vamos a pedir. Para ello, utilizaremos las anotaciones `@Path("{pollId}")` en el método y `@PathParam("pollId")` en la signatura.
- Otro se indicará como parámetro de tipo GET, `sakai.session`, con la identificación de la sesión con la que estamos identificados.

La llamada entonces que tiene que realizar el cliente para ejecutar el servicio será la siguiente:

```
GET http://<servidor>/sakaiRest/rest/encuesta/<pollId>?sessionId=<sessionId>
```

La implementación del servicio es la siguiente:

```
@GET
@Path("{pollId}")
@Produces(MediaType.APPLICATION_JSON)
public Object getEncuestasJSON(@QueryParam("sessionId") String sessionId,
                               @PathParam("pollId") String pollId) {

    String url = Constantes.SakaiBaseURL + "poll/" + pollId +
        "?sakai.session="+sessionId + "&includeOptions=1&includeVotes=1";

    ClientConfig config = new DefaultClientConfig();
    Client client = Client.create(config);
    WebResource service = client.resource(url);

    try {
        ClientResponse response =
            service.accept(MediaType.APPLICATION_JSON).get(ClientResponse.class);
        String resp = response.getEntity(String.class);
        return resp;
    } catch (Exception e){
        ClientMensaje mensaje = new ClientMensaje(true,
            "Error al buscar la encuesta " + pollId);
        return mensaje;
    }
}
```

La implementación es muy similar a la del servicio de obtención de la lista de encuestas: primero prepara el objeto para llamar al servicio con la URL del servicio REST de Sakai, ejecuta el servicio y devuelve el resultado, o un mensaje de error si se produce. Señalar que a la hora de construir la URL, además de indicar los parámetros con las identificaciones de la encuesta y la sesión, indicamos dos parámetros más:

- `includeOptions`: Si se indica el valor 1 se añade a la respuesta el vector con las opciones disponibles para seleccionar en la encuesta. Si se indica 0 no será así. En nuestro caso vamos a querer mostrar la mayor cantidad de información posible, por lo que indicaremos el valor 1.

- `includeVotes`: Si se indica el valor 1, de añadirá el vector con todos los votos que se han realizado en la encuesta, mientras que se indica el vector en blanco si el valor es 0. Por la misma razón que en la opción anterior, indicaremos también el valor 1.

En cuanto al servicio que se encargará de enviar los votos a Sakai, prepararemos un método que acepte las peticiones POST. Recibirá como parámetros de formulario (mediante la anotación `@FormParam` en la signature del método) la identificación de la sesión, la identificación de la encuesta y la lista de votos que se van a enviar.

```
@POST
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
public ClientMensaje votarEncuesta(
    @FormParam("sessionId") String sessionId,
    @FormParam("pollId") String pollId,
    @FormParam("votos[]") List<String> pollOptions) {

    String url = Constantes.SakaiBaseUrl + "poll-vote/vote?sakai.session="+sessionId;
    System.out.println(pollId);
    System.out.println(pollOptions);

    Form form = new Form();
    form.add("pollId", pollId);
    for(int i=0; i < pollOptions.size();i++) {
        System.out.println(pollOptions.get(i));
        form.add("pollOption", pollOptions.get(i) );
    }

    ClientConfig config = new DefaultClientConfig();
    Client client = Client.create(config);
    WebResource service = client.resource(url);

    try {
        service.accept(MediaType.APPLICATION_JSON).post(String.class, form);
        ClientMensaje mensaje = new ClientMensaje(false, "Voto OK");
        System.out.println(mensaje.getMensaje());
        return mensaje;
    } catch (Exception e){
        ClientMensaje mensaje=new ClientMensaje(true, "Error al votar en la encuesta " + pollId);
        e.printStackTrace();
        return mensaje;
    }
}
```

La implementación, como en el resto de casos, prepara primero el servicio indicándole la URL del servicio en Sakai y añadiéndole el parámetro `session.id` con la identificación de la sesión. A continuación se preparan los parámetros con la identificación de la encuesta y las identificaciones de las opciones seleccionadas, y se realiza la llamada mediante el método `post` del servicio. Si se produce un error, se enviará el mensaje marcándolo como error. Si no ha sido así, se marcará el mensaje como correcto, junto con un pequeño mensaje de confirmación.

6.1.6. Servicio de cuestionario

Este servicio lo vamos a implementar como prueba, puesto que es probable que no

lo utilicemos directamente. De todas formas, debido a que es un servicio muy sencillo y muy similar a los anteriores, vamos a implementarlo de la misma forma que el resto: prepararemos una clase llamada `JSONServiceCuestionario` en el paquete `com.jcb.rest.servicios` y la anotamos con el path `"/cuestionario"`. En su interior, prepararemos un método para tratar las peticiones GET que recibamos. La implementación será como sigue:

```
@Path("/cuestionario")
public class JSONServiceCuestionario {

    @GET
    @Path("{testId}")
    @Produces(MediaType.TEXT_HTML)
    public Object getCuestionario( @QueryParam("sessionId") String sessionId,
        @PathParam("testId") String testId) {

        String url = Constantes.SakaiBaseURL +
            "sam_pub/" + testId + "?sakai.session="+sessionId;

        ClientConfig config = new DefaultClientConfig();
        Client client = Client.create(config);
        WebResource service = client.resource(url);

        try {
            ClientResponse response = service.accept(MediaType.TEXT_HTML)
                .get(ClientResponse.class);
            String resp = response.getEntity(String.class);
            return resp;
        } catch (Exception e){
            ClientMensaje mensaje = new ClientMensaje(true,
                "Error al buscar el examen " + testId);
            return mensaje;
        }
    }
}
```

El servicio recogerá dos parámetros de la URL: la identificación de la sesión y la del examen. Indicaremos en la URL hacia Sakai el parámetro `sakai.session` con la identificación de la sesión, aunque aparentemente no tiene ningún efecto. Como se puede ver, la implementación es igual que el resto, con la salvedad de que en lugar de devolver un objeto `ClientMensaje` estamos devolviendo directamente el string con el código HTML que nos devuelve el servicio al ejecutarlo. De esta forma comprobaremos si podemos usar este código de alguna forma en el entorno de la aplicación móvil.

6.2. Aplicación para terminales móviles *jQuery Mobile*

Para preparar la aplicación web que utilizaremos en el móvil utilizaremos también el IDE Eclipse, que podemos utilizar perfectamente para crear una aplicación web estática, es decir, sin Java ni parte de servidor, sino únicamente tecnologías de cliente como Javascript, HTML, CSS y sus librerías correspondientes (jQuery, YUI, EXT...). Para ello, crearemos un nuevo proyecto estático (desde el punto de menú `New – Static Web Project`), indicándole el nombre que queramos para el proyecto (en nuestro caso `cuestionariosjQueryMobile`) y dejando el resto de opciones por defecto.

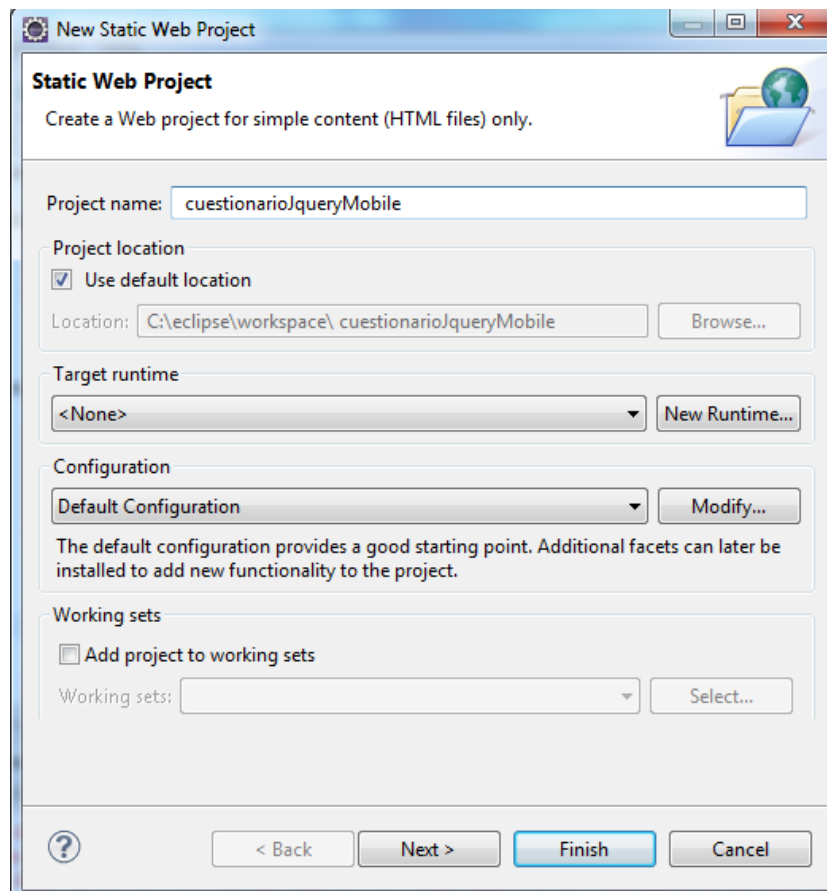


Ilustración 39: Pantalla de creación de proyecto web estático en Eclipse

En nuestro proyecto dispondremos de una carpeta llamada WebContent, que será la carpeta raíz de la aplicación. A partir de ella vamos a preparar la estructura de páginas y directorios que van a organizar la aplicación.

6.2.1. Plugins JQuery Mobile que se van a utilizar

JQuery Mobile nos proporciona un potente framework para comenzar a preparar una aplicación móvil, pero muchas veces tenemos necesidades que no están cubiertas por las capacidades básicas. Gracias a la comunidad que está detrás del framework existen multitud de plug-ins y ampliaciones que permiten ayudarnos a implementar casi cualquier necesidad que tengamos, tanto a nivel de utilidades como al de widgets y elementos de interfaz. Además, si es necesario se puede intentar utilizar los desarrollados para la versión de JQuery de escritorio, aunque no está asegurado que funcionen correctamente por el tratamiento que hace JQuery Mobile del DOM.

De este modo, se pueden conseguir plug-ins para mostrar tablas enriquecidas, galerías de imágenes, añadir efectos visuales, calendarios...

Para nuestro desarrollo, hemos utilizado unos cuantos plugins que nos ayudarán en la implementación. A continuación vamos a detallar cuáles son los que hemos elegido y para qué van a sernos útiles:

- **Localize.js:** Es una plugin para JQuery que permite modificar los textos estáticos

de una página web por los que existan en un fichero dependiente del idioma que indiquemos, o del idioma del sistema si no se especifica. Esto nos permitirá modificarlos a voluntad, e implementar la internacionalización de las páginas web estáticas.

Para utilizarlo, lo que se hace es marcar el elemento de HTML que queremos traducir mediante el atributo `data-localize`, como por ejemplo:

```
<h1 data-localize="saludo"> Hola </h1>
```

En el programa javascript, al iniciar el renderizado de la página (que en jQueryMobile se puede realizar en el manejador del evento `pageCreate`), se indica que se va a utilizar el plugin y el fichero que se utilizará mediante la siguiente sentencia:

```
$("#[data-localize]").localize("example")
```

En un navegador configurado por ejemplo en inglés, se cargará el fichero `example-en.json`. En este ejemplo, el fichero tendrá un aspecto como éste:

```
{
  "greeting": "Hola!"
}
```

Este plugin está disponible en la siguiente dirección web:

<https://github.com/coderifous/jquery-localize>

- **ToastMessage:** Este plugin permite mostrar mensajes que aparecen y desaparecen pasado un tiempo, al estilo de los que se muestran en las aplicaciones móviles nativas de Android e iPhone.



Ilustración 40: Ejemplos de mensajes generados mediante ToastMessage

Para utilizarlo, no hay más que añadir a nuestro proyecto el javascript, css e imágenes que lo forman, añadir en las cabeceras de la página HTML los enlaces correspondientes y, una vez en el programa javascript, utilizar una sentencia similar

a la siguiente:

```
$.toastmessage('showToast', {
  text      : mensaje.mensaje,
  sticky    : false,
  type      : 'success',
  position  : 'middle-center'
});
```

Se pueden ver ejemplos, documentación y descargar los ficheros que forman la librería desde la dirección web siguiente:

<http://akquinet.github.com/jquery-toastmessage-plugin/>

- **JQuery Templates:** Este plugin nos permite crear plantillas con código HTML utilizando un sencillo lenguaje de marcado propio. Para crear una plantilla, en la página HTML donde la vamos a incluir se indicará un elemento de la forma:

```
<script id="identificación_de_la_plantilla" type="text/x-jquery-tmpl"></script>
```

En el interior de este elemento se especificará la plantilla, que no es más que código HTML al que se le añaden varias instrucciones propias, como por ejemplo las siguientes:

- `${variable}`: Escribe el valor de la variable
- `{{each coleccion}}{}/each}}`: Permite recorrer un elemento colección, como por ejemplo un vector
- `{{if condicion}}{}/if}}`: Permite indicar si el contenido contenido dentro de la etiqueta se renderiza al cumplirse la condición

Para utilizar el plugin, además de indicar la biblioteca en la cabecera HTML, en el programa javascript se indicará una orden como la que sigue:

```
var html = $("#identificacion_de_la_plantilla").tmpl(objeto);
```

Esto lo que hace es renderizar la plantilla con los datos que contenga el objeto, y guarda el resultado en la variable html. A continuación, utilizando JQuery podemos hacer lo que queramos con este resultado, desde modificarlo a nuestro antojo a introducirlo en el DOM de la página.

Este plugin estuvo propuesto para pertenecer a la versión oficial de las librerías JQuery para escritorio, aunque al final no se implantó, aunque desde el sitio oficial de JQuery continúa existiendo la documentación, en la dirección <http://api.jquery.com/category/plugins/templates>. Actualmente se puede conseguir desde la siguiente dirección:

<https://github.com/jquery/jquery-tmpl>

- **JQPlot:** Es un plugin gratuito y open source, que nos permitirá realizar gráficos fácilmente. Gracias a este plugin vamos a darle una visualización más agradable a los resultados de las encuestas.

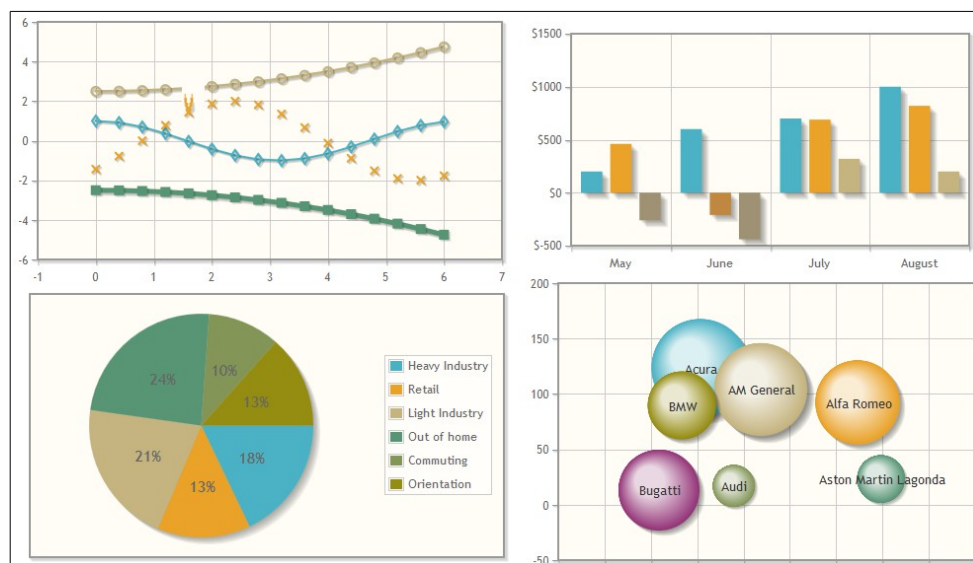


Ilustración 41: Ejemplos de gráficos generados mediante JQPlot

Cada tipo de gráfico se utiliza de una forma particular, Para no sobrecargar la librería, con la consiguiente sobrecarga en el sistema, se ha separado cada tipo de gráfico en un archivo separado, de forma que sólo carguemos los tipos de gráficos, configuraciones de ejes, y otros plugins necesarios, que de verdad vayamos a utilizar. En la versión que estamos utilizando existe en la carpeta de plugins más de 50 archivos con los distintos añadidos.

La página oficial del proyecto es la siguiente: <http://www.jqplot.com/>

6.2.2. Estructura de la aplicación móvil

Una vez creado el proyecto, vamos a crear la estructura que habrá en la carpeta WebContent para organizarlo. Vamos a utilizar una estructura bastante clásica para las aplicaciones web estáticas, y tendrá básicamente los siguientes apartados:

- Una carpeta denominada **css** que contendrá los estilos de las páginas y los que necesiten las librerías que vamos a utilizar, tanto de los frameworks JQuery y JQuery Mobile como los plugins que lo precisen. En su interior también contendrá una carpeta con las imágenes a las que se hagan referencia directamente desde los archivos css, que es un requisito de los mismos.
- Una carpeta llamada **js** que contiene todos los archivos con el código javascript necesario para el funcionamiento de la aplicación. No existirá código javascript escrito directamente en las páginas HTML, para así poder mejorar el mantenimiento. A su vez, esta carpeta la vamos a dividir en otras dos que contendrán cada una lo siguiente:
 - La carpeta **app** contendrá los archivos javascript de la aplicación, es decir, aquí se colocará toda la programación javascript de la aplicación.
 - La carpeta **lib** va a contener los archivos javascript de las librerías JQuery y JQuery Mobile, y los de los plugins que vamos a utilizar
- Una carpeta llamada **images**, que va a contener las imágenes que vamos a utilizar

en las páginas HTML

- En la raíz de la aplicación, situaremos las páginas HTML que van a formar la parte estática de la aplicación. Aquí vamos a añadir también los archivos que contendrán los textos de internacionalización (i18N), que estarán codificados en el formato JSON.

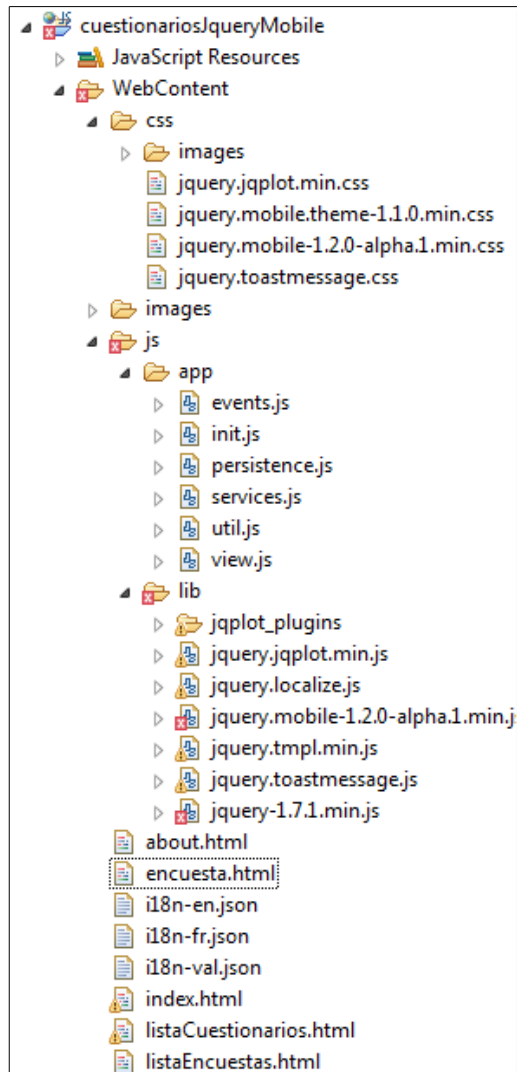


Ilustración 42: Estructura de la aplicación Sakai Encuestas

En la carpeta con los archivos javascript de la aplicación (js/app) vamos a crear una serie de archivos en donde vamos a realizar la implementación. Cada archivo va a encargarse de la definición de un aspecto de la aplicación, de forma que todo el contenido de cada uno sea funcionalmente similar. Los ficheros que vamos a crear son los siguientes:

- **init.js:** En este archivo vamos a incluir la configuración inicial de la aplicación móvil. Para ello, vamos a manejar el evento **mobileinit**, que se lanza una sola vez cuando empieza la aplicación móvil. Según la documentación de JQuery Mobile, este es el lugar idóneo para configurar las opciones por defecto del framework (de hecho este fichero se debe incluir antes en la página web que el de la librería de

jQuery Mobile). Por ejemplo, podemos cambiar los textos de los elementos predefinidos (como el mensaje de Cargando..., los textos de los botones de Cerrar y Volver...), modificar la transición por defecto entre páginas, el comportamiento de AJAX, la clase por defecto de la página activa...

También utilizaremos este evento para la parametrización de la aplicación de encuestas. Para ello, crearemos un objeto javascript global llamado **app**, que tendrá varios atributos que vamos a necesitar:

- nombre: Nombre de la aplicación
- version: Versión de la aplicación
- proxoSakaiRest: Aquí configuraremos la dirección donde se encuentra alojada la aplicación java con el proxy de conexión a Sakai
- urlSakai: La dirección donde está alojado el sistema Sakai. En principio no nos hace falta para utilizar la aplicación de encuestas, pero la vamos a usar para hacer pruebas con la aplicación de cuestionarios/exámenes.
- LocalStorage: Aquí vamos a indicar el prefijo de las claves que se van a utilizar para el LocalStorage, el almacenamiento en el dispositivo móvil de los datos de la aplicación (en este caso las encuestas). La forma que van a tener las claves que usaremos será *<prefijo><usuario>*, de esta forma vamos a poder guardar las encuestas dependiendo del usuario conectado en el sistema. Por defecto vamos a indicar el valor *sakai.poll_collection_*

También crearemos un objeto llamado **global**, en el que guardaremos las variables globales que vamos a utilizar en la aplicación, por lo que este objeto nos servirá para tenerlas controladas (en javascript es muy fácil que pierdas de vista el alcance de las variables, al no ser un lenguaje fuertemente tipado). En esta variable definiremos los siguientes atributos:

- listaEncuestas: Aquí guardaremos la lista de encuestas con la que estamos trabajando, cuando la devolvamos del servicio (o del localstorage si estamos trabajando offline)
 - idxEncuesta: Índice del elemento de la lista de encuestas actual, cuando seleccionemos alguno de la página con la lista.
 - grafico_resultados_encuesta: Objeto con el gráfico que se mostrará con los votos registrados en una encuesta.
 - Offline: Variable lógica que nos indica si la aplicación se está ejecutando en modo offline u online.
 - Language: Nos permite definir el idioma que usaremos para los textos del sistema.
 - Mensajes: Objeto con todos los mensajes de texto para el idioma seleccionado. Previamente los hemos recogido de un fichero que tenemos alojado en la raíz de la aplicación web, mediante una llamada AJAX.
- **events.js**: En este fichero vamos a indicar los manejadores de los eventos que

vamos a capturar. Los eventos que vamos a necesitar capturar van a ser los siguientes:

- Creación de la página (pagecreate): Lo utilizaremos para modificar los textos de la pantalla para la internacionalización, o para añadirle contenido antes de que se le de formato y se muestre al usuario
- Inicio de la página (pageinit): Se lanza el evento cuando ya se ha mejorado el DOM de la página mediante las librerías de JQuery Mobile, pero antes de mostrarse al usuario. Es el equivalente al metodo ready() de JQuery de escritorio. Lo utilizaremos para modificar propiedades de los widgets del framework que vayamos a usar, puesto que no lo podremos hacer hasta que estén activos.
- Pulsación de botones (tap): Aquí indicaremos la lógica que se debe ejecutar al pulsar el botón capturado
- Cambio de valor (change): Lo utilizaremos cuando necesitemos capturar el cambio de valor en checkboxes o menús de selección.
- Eventos del application cache: Podemos capturar los eventos que se lancen cuando se ejecute la caché de la aplicación. De esta forma podremos detectar si existe conexión o no y proponer el modo offline automáticamente.

Dividiremos el fichero en varios apartados, para organizar el código según la página de la que estamos capturando los eventos.

- **persistence.js**: En este fichero vamos a indicar las funciones que se encargarán de guardar los datos en el sistema de almacenamiento local del dispositivo móvil. En el estándar HTML5 existen varias opciones para implementar el almacenamiento local:

- LocalStorage y SessionStorage: Es el sustituto en HTML5 al concepto de cookies, con seguridad mejorada. Los datos no se incluyen en cada petición al servidor, y se utilizan sólo cuando se quiere acceder a ellas. Una página sólo puede acceder a los datos que ha guardado ella misma.

La información se guarda en pares de clave/valor, siendo el valor una cadena de caracteres. De todas formas, para añadir potencia siempre se pueden parsear los objetos utilizando JSON, de forma que tenemos una forma fácil de mantener grandes cantidades de información estructurada.

Existen dos clases de almacenamiento web: localStorage, que guarda la información permanentemente hasta que se elimina de manera explícita, o SessionStorage, que se mantiene en el sistema mientras la sesión está abierta (típicamente mientras no se cierre la pestaña o la ventana con la aplicación web).

El mayor inconveniente viene en el espacio disponible. Casi todos los navegadores permiten un espacio de sólo 5 megas para el almacenamiento web.

- Web Database: Mediante esta especificación se define un API para utilizar una variante de las bases de datos SQL para mantener los datos. Para la base de

esta especificación se basaron en el formato SQLite. Se permiten crear bases de datos, crear tablas, y todo el manejo que normalmente se hace de una base de datos relacional (consultas, inserciones de datos, actualizaciones, borrados...)

La implementación que hacen los navegadores suele ser muy fácil de utilizar, y se permiten espacios de datos mucho mayores que en el caso del web storage.

Como puntos negativos, todas las operaciones sobre la base de datos son asíncronas, lo que puede dificultar la programación cuando necesitamos esperar el resultado de alguna consulta para realizar alguna acción. Pero sobre todo el mayor inconveniente es que esta especificación ha sido abandonada del estándar HTML5, por lo que aunque el soporte por parte de los navegadores es bastante amplio, no se va a seguir desarrollando en favor de otras alternativas como indexedDB.

- IndexedDB: Esta especificación HTML5 permite mantener una base de datos de registros manteniendo valores simples y objetos jerárquicos. Cada registro consistirá en una clave y alguna clase de valor, y la base de datos se encarga de manejar índices sobre los registros que forman parte de ella.

El programador podrá acceder a los valores mediante la clave o mediante el índice generado, mediante un lenguaje de consultas propio. Para implementar esta especificación se utilizan estructuras persistentes de tipo árbol-B.

El mayor inconveniente de esta especificación es el bajo índice de aceptación que tiene por parte de los navegadores. A día de hoy, en la página web CANIUSE (<http://caniuse.com/#feat=indexeddb>) se puede comprobar que de entre los navegadores más utilizados sólo Firefox y Chrome tienen soporte a este estándar, mientras que ninguno de los navegadores para móvil dispone de él. El futuro Internet Explorer 10 sí que dispondrá de su implementación correspondiente.

De entre estas alternativas, nos vamos a decidir por utilizar localStorage, puesto que no necesitamos una gran cantidad de espacio para guardar los datos de las encuestas, por lo que con 5 megas debería ser suficiente. Además, los inconvenientes del resto de opciones sí que son importantes para el desarrollo actual de la aplicación móvil, aunque teniendo el desarrollo de la persistencia separado en un fichero no sería un gran esfuerzo modificar la implementación para adoptar la nueva solución.

- **services.js:** En este fichero implementaremos las llamadas AJAX que vamos a realizar al servidor, es decir, al proxy que hará de intermediario entre Sakai y la aplicación móvil.

Las llamadas se van a realizar, siempre que sea posible, de forma asíncrona, para no bloquear la ejecución del navegador en el dispositivo. Por lo tanto, en este fichero también estará la definición de las funciones que se ejecutarán al recibir la respuesta del servidor, tanto si se ha realizado correctamente como si se ha producido un error. También se realizará el manejo de errores de la propia llamada, para no perder el control en caso de que, por ejemplo, se pierda la conexión y no se pueda acceder al servidor.

- **view.js:** En este fichero vamos a incluir el código que se encargará de construir las partes de las páginas que se vayan a generar dinámicamente dependiendo de los datos que se estén manejando. Para ello, se utilizará el plugin JQuery Templates, que junto con los datos que vamos a recoger del servicio se generara el código HTML que vamos a inyectar en el DOM de la página.

En nuestro caso, vamos a usarlo para montar las opciones de la lista de encuestas y para las opciones de respuesta que se van a mostrar por cada pregunta. Sería interesante usar esta misma técnica para montar las páginas para la contestación de cuestionarios, pero debido a los problemas con el servicio anteriormente comentados no será posible.

- **util.js:** En este fichero incluiremos diferentes funciones de utilidades que serán utilizadas por el resto del sistema, de manera que estén encapsuladas y sean fáciles de usar. Entre las funciones que vamos a utilizar se encuentran las funciones para el manejo de gráficos mediante JQPlot, de la herramienta de traducción de literales para la internacionalización, o para el serializado de objetos javascript en formato JSON, básico para el módulo de persistencia mediante el localStorage.

6.2.3. Pantalla de Login

La pantalla con la que vamos a iniciar la aplicación será la de Login. Para definirla, empezaremos con el esquema del código HTML que hemos utilizado, no sólo en esta pantalla sino en el resto de las de la aplicación

```
<div data-role="page" id="pageLogin">
  <div data-role="header">
  </div><!-- /header -->

  <div data-role="content" >
  </div><!-- /content -->

  <div data-role="footer" data-position="fixed">
  </div><!-- /footer -->
</div>
```

Como se puede observar, la página consta de un DIV con el atributo data-role=page, Dentro de ese DIV hay otros tres, uno por cada área de la página: Uno para la cabecera, otro para el cuerpo y otro para el pie. Se ha indicado que la posición del pie sea fija, lo que indica que no se moverá incluso si en la página aparece un scroll vertical

El header de la página mantiene el título y un botón, con el que se abrirá el popup con el Acerca de... de la aplicación. Nótese que el literal que acompaña al span del acerca de está anotado con el atributo data-localize, para que el plugin Localize pueda modificar el contenido por el del idioma que se disponga

```
<div data-role="header">
  <h1 style="margin-left: 5px;margin-right: 5px;">
    <span id="nombreAplicacion">Cuestionarios</span>
```

```

</h1>
<a href="#pageAbout" class="ui-btn-right" data-rel="popup" data-transition="pop"
  data-role="button" data-icon="info" data-iconpos="notext">
  <span data-localize="LoginAcercaDe">Acerca de...</span>
</a>
</div><!-- /header -->

```

En el content de la página indicamos los dos campos con el usuario y password (mediante el div con el rol fieldcontain) y los checkboxes para guardar el password en el dispositivo y para activar el modo offline.

```

<div data-role="content" >
  <div data-role="fieldcontain">
    <label for="usuario" data-localize="LoginUser">Usuario</label>
    <input type="text" name="usuario" id="usuario" value="" data-mini="true" />
  </div>

  <div data-role="fieldcontain">
    <label for="pwd" data-localize="LoginPwd">Contraseña</label>
    <input type="password" name="pwd" id="pwd" value="" data-mini="true" />
  </div>
  <label>
    <input type="checkbox"
      name="savePassword"
      id="savePassword"
    />
    <span data-localize="LoginSavePwd">Guardar contraseña</span>
  </label>

  <label>
    <input type="checkbox"
      name="cbxModoOffline"
      id="cbxModoOffline"
    />
    <span data-localize="LoginOffline">Trabajar offline</span>
  </label>
</div><!-- /content -->

```

En el pie de la página únicamente se indica un botón que nos permitiera realizar el login en la aplicación

```

<div data-role="footer" data-position="fixed">
  <div data-role="navbar">
    <ul>
      <li><a id="btnLogin" href="#" data-inline="false">
        <span data-localize="LoginBoton">Login</span>
      </a></li>
    </ul>
  </div><!-- /navbar -->
</div><!-- /footer -->

```

Existe también un DIV que va a crear la ventana de Popup con el contenido de los créditos del programa. Esta ventana se abrirá cuando se pulse el botón correspondiente en la cabecera de la página

```

<div data-role="popup" id="pageAbout" data-overlay-theme="a"
  data-theme="c" class="ui-corner-all">
  <div data-role="header" data-theme="a" class="ui-corner-top">

```

```
<h1 style="margin-left: 5px;margin-right: 5px;">
  <span id="nombreAplicacion">Cuestionarios</span>
</h1>
</div>
<div data-role="content" data-theme="d" class="ui-corner-bottom ui-content">
  <h3 class="ui-title" data-localize="LoginAlumno">Alumno</h3>
  <p>Fco Javier Calás Blasco</p>
  <h3 class="ui-title" data-localize="LoginDirector">Director</h3>
  <p>David de Andrés Martínez</p>
  <h3 class="ui-title" data-localize="LoginColaboradores">Colaboradores</h3>
  <p>David Roldán Martínez</p>
  <p>Alberto Palomares </p>
</div>
</div>
```

Este es el aspecto final de la página. La segunda imagen muestra cómo se ve el popup con la ventana de créditos

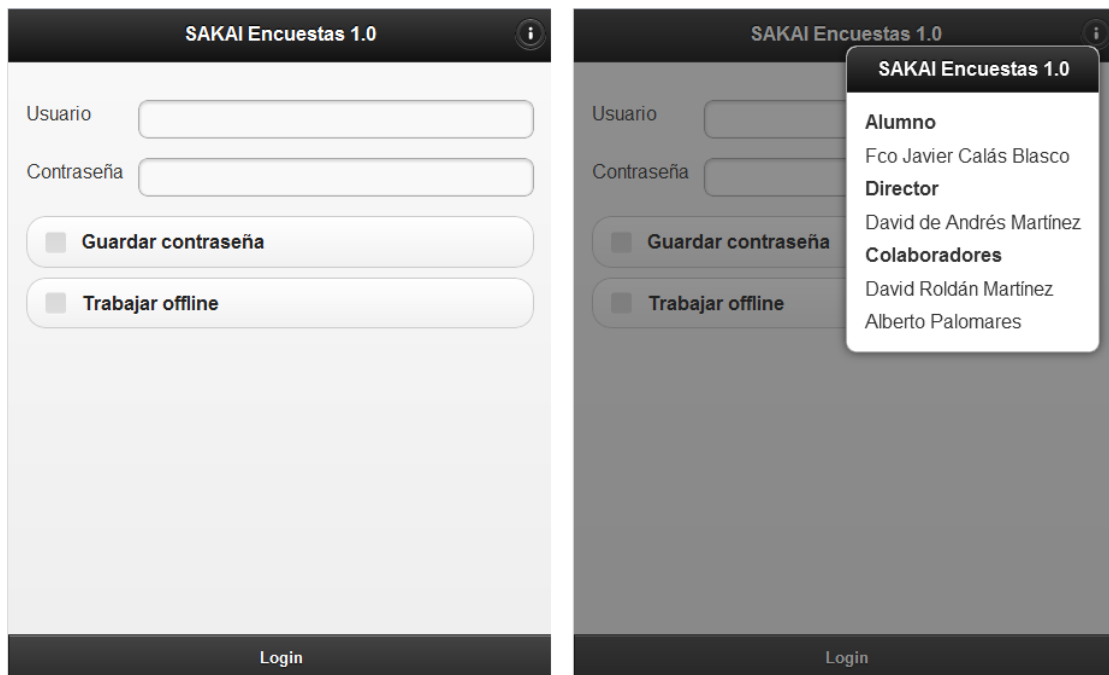


Ilustración 43: Sakai Encuestas. Pantalla de login

Para el manejo de esta ventana, vamos a capturar algunos eventos, :

- Inicio de página (pageinit): Se va a encargar de rellenar los campos con los valores por defecto: Si el usuario y password están guardados en el dispositivo, rellenará los campos con sus valores correspondientes
- Cambio del check de trabajo offline (check de cbxModoOffline): Si se desmarca, protegerá el campo Password, porque es irrelevante. Si se marca hará la operación contraria
- Pulsación del botón de login (tap de btnlogin): En este caso, se distinguirá si estamos en modo offline u online.

- En el modo offline, simplemente guardaremos el nombre del usuario en el sessionstorage, construimos el índice donde están los datos de persistencia (en el localStorage) y vamos a la página de menú
- En el modo online, llamaremos al servicio de login pasándole el usuario y password, o mostramos error si no se indica el usuario.

```
//TAP del botón del LOGIN
$(document).on('tap', 'div#pageLogin #btnLogin', function(event){
    event.stopImmediatePropagation();
    event.preventDefault();
    console.log('tap');

    var usuario = $("input#usuario").val();
    var password = $("input#pwd").val();
    global.offline = $("input#cbxModoOffline").is(":checked");

    //Llamamos al servicio de Login
    if (global.offline){
        sessionStorage.setItem('userId', usuario);
        persistence.indexLocalStorage = app.localStorage + usuario;
        $.mobile.changePage("menu.html");
    } else {
        if (usuario != "") {
            services.login(usuario, password);
        } else {
            view.mostrarMensaje({
                error: true,
                mensaje: global.mensajes.loginUsuarioObligatorio
            });
        }
    }
});
```

- El servicio de Login es un servicio asíncrono, lo que quiere decir que la ejecución de javascript no esperará a que termine y continuará. Esto nos obligará a indicar las acciones que hay que hacer cuando se realice una llamada AJAX en una función de callback, e indicarla como parámetro en la propia llamada AJAX.:
 - En el caso del login, prepararemos la llamada con la URL y los parámetros correspondientes (usuario y password), y realizaremos la llamada AJAX, indicando como función de callback una a la que hemos denominado con el nombre de login_success:

```
services.login = function(usuario, password){
    $.mobile.loading('show');
    var url = app.proxiSakaiRest + "login";
    var params = {"user" : usuario,
                 "password" : password
    };

    $.ajax({
        url: url,
        type: 'POST',
        //dataType: 'json',
        async: true,
        cache: false,
```

```
    data: params,  
    success: login_success,  
    error: services.errorCallback  
  });  
};
```

- La función de callback lo que hará es guardar la sesión y el usuario en el sessionstorage, guardar en el localStorage usuario y password si se ha indicado en la pantalla (o quitarlo en caso contrario), construir el índice del localStorage para la persistencia y cambiar a la página de menú. Si se ha producido un error en la llamada, se muestra al usuario:

```
var login_success = function(respuesta){  
  if (respuesta != null && !respuesta.error){  
    sessionStorage.setItem('sessionId', respuesta.mensaje);  
    sessionStorage.setItem('userId', usuario);  
  
    //Guardamos datos de offline y usuarios y contraseña  
    if($("#input#savePassword").is(":checked")){  
      localStorage.setItem("sakai.user", usuario);  
      localStorage.setItem("sakai.password", password);  
    } else {  
      localStorage.removeItem("sakai.user");  
      localStorage.removeItem("sakai.password");  
    };  
    persistence.indexLocalStorage = 'sakai.poll_collection_' + usuario;  
    $.mobile.changePage("menu.html");  
  } else {  
    var mensaje = global.mensajes.loginErrorConexion;  
    if (respuesta != null){  
      mensaje = respuesta.mensaje;  
    }  
  
    view.mostrarMensaje({  
      error: true,  
      mensaje: mensaje  
    });  
  }  
  $.mobile.loading('hide');  
};
```

6.2.4. Pantalla de menú

La pantalla del menú de la aplicación constará de dos botones, a los que les hemos añadido una imagen para que resulten más atractivos, junto con un botón en la parte superior que realiza el logout de la aplicación. El código HTML es muy sencillo, y nos remitimos al código fuente que acompaña este proyecto si lo quieren consultar

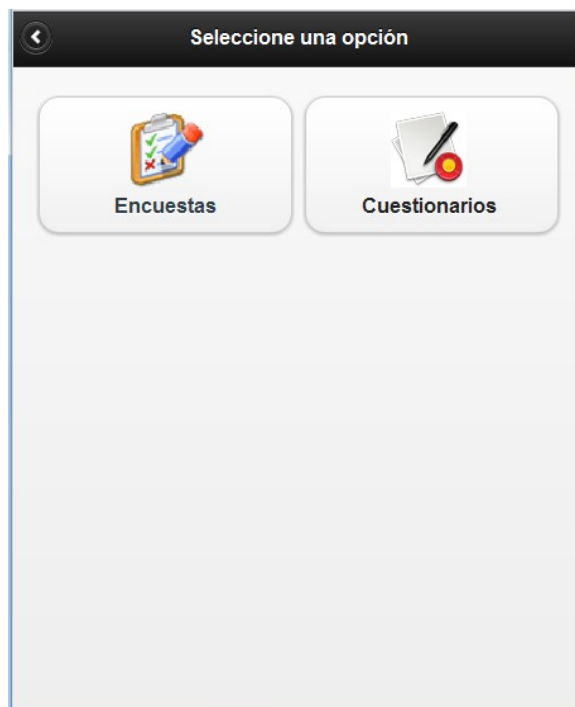


Ilustración 44: Sakai Encuestas. Pantalla de menú

En esta ventana se van a manejar dos eventos: uno para la pulsación del botón de logout y otro para el de la lista de encuestas. Cada uno en realidad lo único que hará es llamar al servicio correspondiente:

- El botón de logout llamará al servicio Logout() que se encargará de vaciar las variables globales y de sesión utilizadas y, en el caso de estar online, de realizar una llamada AJAX al servidor para salir de la sesión, utilizando para ello el método DELETE
- El botón de Lista de encuestas llamará al servicio correspondiente, que se encargará de una serie de cosas, también dependiendo de si estamos en modo offline u online:
 - Si estamos fuera de línea, recogeremos la lista de encuestas de las que estén disponibles en persistencia, mediante el localStorage
 - Si estamos en línea, recogeremos las dos listas, la de persistencia, y además llamaremos al servicio. En la función de callback del servicio realizaremos lo siguiente:
 - Recorreremos todas las encuestas que nos ha devuelto el servicio
 - Si esa encuesta está contenida entre las que tenemos en persistencia, utilizaremos la que estaba en persistencia
 - Si no, volveremos a llamar al servicio, pero esta vez para recoger todos los datos de una encuesta determinada. Marcamos la encuesta como que se ejecuta online. El siguiente listado es la implementación de la función de callback del servicio

```

var listaEncuestas_success = function(json){
  $.each(json.poll_collection, function(index, poll){
    //Si la encuesta está entre las guardadas offline, la buscamos del
    //localStorage
    //Si no, utilizamos la del servicio. La marcamos como offline
    if (pollCollection != null && poll.pollId in pollCollection){
      json.poll_collection[index] = jQuery.extend(true, {},
        pollCollection[poll.pollId]);
    } else {
      json.poll_collection[index] = services.getEncuesta(poll.pollId, false);
      json.poll_collection[index]["offline"] = false;
    }
  });
  global.listaEncuestas = json;

  $.mobile.changePage("listaEncuestas.html");
  $.mobile.loading('hide');
};

```

6.2.5. Lista de encuestas

Para esta pantalla en la página HTML hemos indicado las anotaciones necesarias para dibujar en la cabecera el título y un botón para volver a la pantalla anterior. Sin embargo, el DIV con el content lo hemos dejado deliberadamente en blanco, puesto que vamos a construir en contenido de la lista a partir de una plantilla que hemos preparado para la ejecución del plugin jquery-templates. La plantilla tiene el aspecto que se muestra en el listado siguiente:

```

<script id="tmplListaEncuestas" type="text/x-jquery-tmpl">
  <ul data-role="listview" id="listaEncuestas">
    {{each poll_collection}}
    <li id="poll${pollId}">
      <a href="#" class="btnEncuesta" data-pollid=${pollId}>
        <h4>${pollText}</h4>
        <p>${siteId}</p>
        <p>
          <span id="offline" data-offline=${offline}>
            {{if offline }} offline {{else}} online {{/if}}
          </span>

          {{if currentUserVoted}}<span
            id="contestada" data-localize="listaEncuestasContestada">
              CONTESTADA
            </span>{{/if}}
        </p>
      </a>
      <a href="#propsEncuesta" class="btnPropsEncuesta" data-pollid=${pollId}
        data-rel="popup" data-position-to="window" data-transition="pop">
      </a>
    </li>
    {{/each}}
  </ul>
</script>

```

Esta plantilla espera que se ejecute con un objeto que a su vez tenga una lista (poll_collection), y mediante su propio lenguaje de marcado indica el flujo de control que

se va a llevar. En este ejemplo, recorre todas las encuestas de `poll_collection` y va añadiendo una fila a la lista (``), mientras añade información como la identificación de la encuesta, el texto, el sitio de la encuesta, si la encuesta es offline, si está contestada...

Para construir esta lista, desde el evento de creación de la página (`pagecreate`), se llama al método `buildListaEncuestas` del archivo `view.js`, que contiene la lógica de creación de la lista. Se le pasa la lista de encuestas que habíamos obtenido antes de entrar en esta página. Como se puede ver en el siguiente listado, son sólo tres líneas de código:

```
view.buildListaEncuestas = function(encuestas){
  var template = $("#tmplListaEncuestas").tmpl(encuestas);
  $('#pagelistaEncuestas div[data-role="content"]').append(template);
  $('#listaEncuestas').listview();
};
```

Básicamente, la primera línea crea el HTML resultante a partir de la plantilla indicada y el objeto que contiene las encuestas, y se guarda en una variable. A continuación, incluimos el HTML resultante en el DIV del content de la página de la lista, y por último se le da formato de lista

Además de la plantilla, indicamos en el HTML un DIV que implementa el popup que se muestra al pulsar el botón de Propiedades de una encuesta, con la opción de Guardar encuesta offline

El aspecto de la ventana va a ser el siguiente. Se muestran dos imágenes, una con el aspecto inicial y otra con el popup de las propiedades de una encuesta mostrado:



Además del evento de creación, vamos a tener más manejadores de eventos en esta página:

- Pulsación del botón de propiedades de una encuesta (tap de btnPropsEncuesta): Se encargará de inicializar las opciones del popup de propiedades de la encuesta, indicando el valor correspondiente en el checkbox de Guardar encuesta offline. Si estamos en modo offline, se desactivará el checkbox.
- Pulsación del botón de OK en el popup de propiedades de una encuesta (tap de btnPropsEncuestaOK): Dependiendo de si se ha marcado o desmarcado la opción de Guardar encuesta offline:
 - Si se ha marcado, se recoge la encuesta del servicio y se guarda en persistencia.
 - Si se ha desmarcado, borraremos la encuesta de persistencia
 - En cualquier caso, actualizaremos el valor de la propiedad offline del objeto global con la lista de encuestas, para que sea coherente con el nuevo estado del objeto.

```
//BOTÓN OK DEL POPUP DE PROPIEDADES DE UNA ENCUESTA
$(document).on('tap', "div#pageListaEncuestas #btnPropsEncuestaOK", function(event){
    var pollId = $("div#propsEncuesta").data("pollid");

    //TRATAMIENTO OFFLINE
    if (global.offline == false) {
        var offline = $("div#propsEncuesta input#offline").is(":checked");

        //Marcamos el elemento de la lista
        $("li#poll" + pollId + " span#offline").data("offline", offline);

        if (offline){
            $("li#poll" + pollId + " span#offline").html("offline");
            //Recogemos la encuesta del servicio y la guardamos en el localStorage
            var poll = services.getEncuesta(pollId, false);
            persistence.guardarEncuesta(poll);
        } else {
            $("li#poll" + pollId + " span#offline").html("online");
            //Eliminamos el elemento offline en la lista del localStorage
            persistence.borrarEncuesta(pollId);
        }
    }

    $.each(global.listaEncuestas.poll_collection, function(index, encuesta){
        if (encuesta.pollId == pollId) {
            global.listaEncuestas.poll_collection[index].offline = offline;
            return false; //break
        }
    });
});
```

- Como ejemplo, vamos a ver cómo se realiza el guardado de un elemento en persistencia. El proceso es muy sencillo: recogemos toda la lista de encuestas que tengamos en persistencia, añadimos en el vector una nueva con la clave del identificador de la encuesta, y al final pasamos a JSON el objeto resultante y

lo guardamos en la clave correspondiente del localStorage.

```
persistence.guardarEncuesta = function(poll) {  
  var pollCollection = persistence.leerListaEncuestas();  
  
  poll["offline"] = true;  
  pollCollection[poll.pollId] = poll;  
  localStorage.setItem(persistence.indexLocalStorage,  
    util.JSON_stringify(pollCollection));  
};
```

- Pulsación en un item de la lista (encuesta: tap de btnEncuesta): Lo único que se hará en este evento es identificar el índice dentro del objeto global con la encuesta que se ha pulsado, y dirigirse a la página de encuesta.

6.2.6. Pantalla de encuesta

Para construir la página de la encuesta, vamos a proceder de un modo similar a como hemos hecho para el resto de ventanas de la aplicación. Primeramente crearemos el HTML con la estructura de la página. En la cabecera indicaremos el botón para volver a la pantalla anterior y el título de la ventana.

En el contenido mostraremos varios literales para indicar la encuesta que estamos tratando, así como un DIV llamado formulario en blanco. Posteriormente indicamos tres botones: Uno para votar, otro para limpiar la ventana y otro para mostrar el gráfico de resultados

En el DIV del formulario es donde vamos a indicar las opciones de la encuesta. La estrategia que vamos a seguir es similar a la usada para construir la lista de encuestas: mediante un método de la vista (buildPagEncuesta) construiremos las opciones de la encuesta. En lugar de una plantilla, existirán dos: una para cuando sólo se pueda seleccionar una opción, y por lo tanto se construyan botones de radio, y otra cuando se pueda seleccionar más de una opción, en donde se utilizarán en su lugar controles de tipo checkbox. El resultado de la ejecución de la plantilla se añadirá en el interior del DIV mencionado.

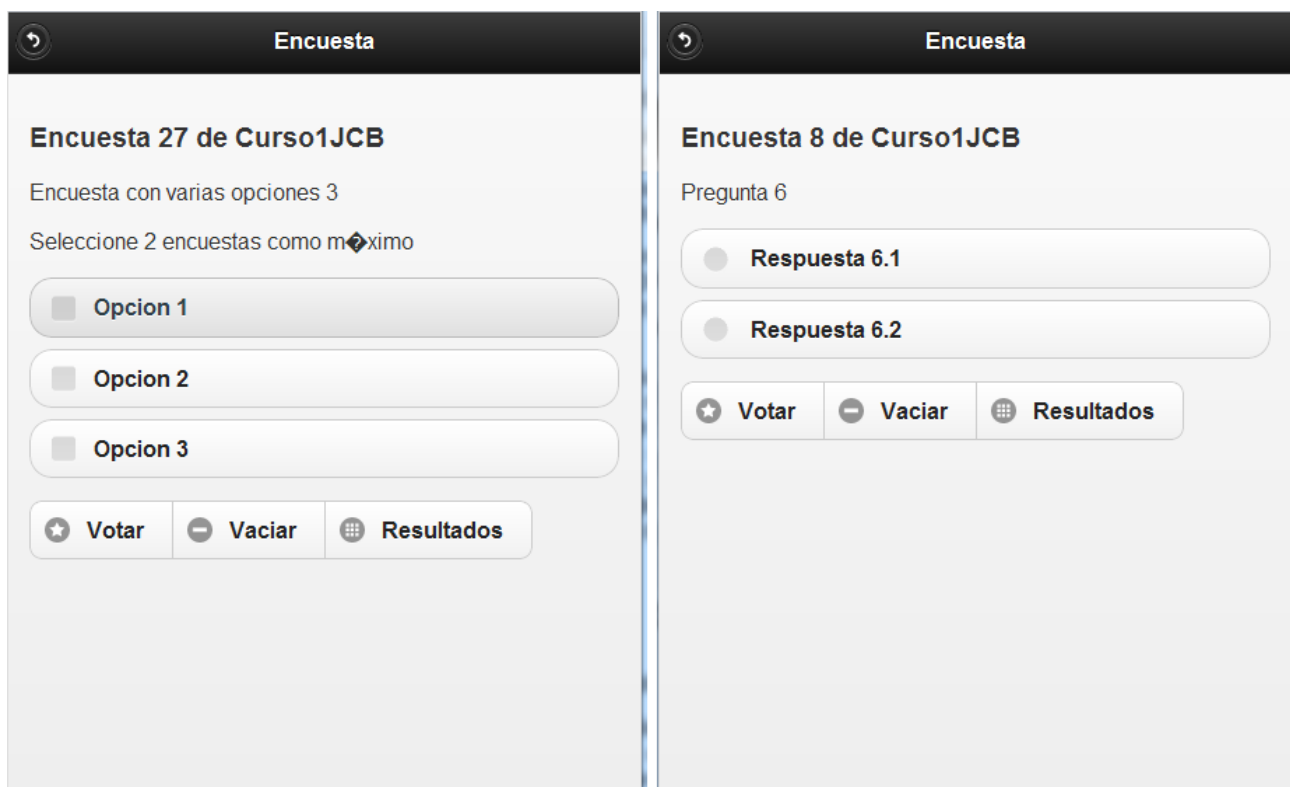


Ilustración 45: SakaiEncuestas. Pantallas de encuestas con una o varias opciones

En esta página existirán los siguientes manejadores de eventos:

- Creación de la página (pagecreate): Además de la ejecución del plugin de localize para el tema de la internacionalización (como en el resto de la aplicación), se encarga de construir el formulario con las opciones de la encuesta, como hemos indicado anteriormente.
- Evento de mostrar la página (pageshow): Inicializa el gráfico para mostrar por defecto el gráfico de barras.
- Cambio de tipo de gráfico en el pupup de resultados: Modifica el tipo de gráfico que se está mostrando para que sea de tipo de barras o tipo tarta, según corresponda por la opción seleccionada.

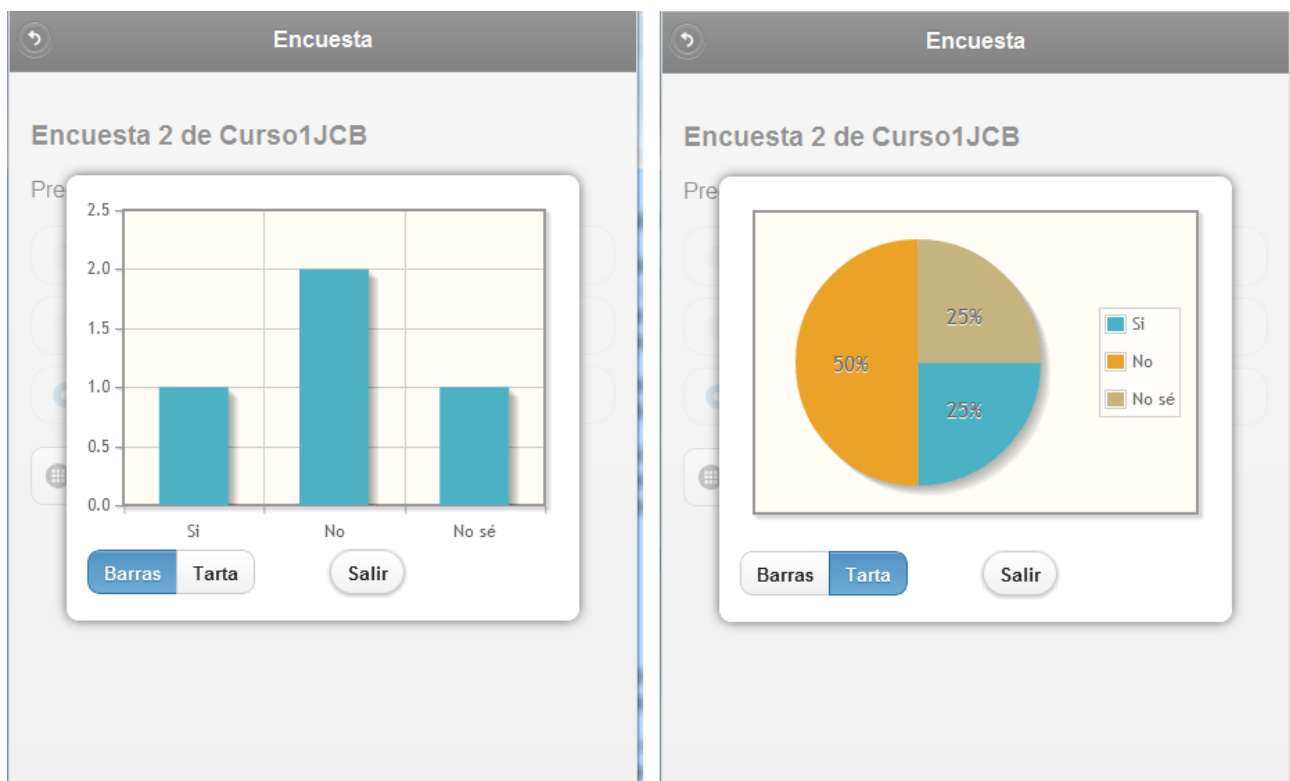


Ilustración 46: SakaiEncuestas. Tipos de gráficos

- Pulsación en el botón de inicializar opciones (tap de `initPoll`): Desmarca todas las opciones marcadas en el formulario de opciones de la encuesta.
- Pulsación en el botón de enviar la respuesta (tap de `submitPoll`): Utiliza el servicio de votar para mandar las opciones que se han seleccionado. Previamente hace una comprobación para validar si se han seleccionado demasiadas opciones, en el caso de que la encuesta permita seleccionar más de una opción.

6.2.7. Pantalla de cuestionarios/exámenes

Esta ventana la vamos a indicar como prueba de concepto, puesto que después de hacer numerosas pruebas e intentos, no hemos conseguido manejar la información de SAMIGO de la misma manera que lo hacemos con la herramienta de Encuestas.

De la única forma que hemos conseguido poder trabajar con el servicio, es indicar un IFRAME en el que vamos a indicar como destino la dirección del servicio. De todas formas, vamos a parametrizar esa dirección utilizando un evento y las propiedades configuradas en el inicio. El código HTML queda por lo tanto de la siguiente forma:

```
<div data-role="header">
  <a id="back" href="#" data-role="button" data-icon="back" data-iconpos="notext"></a>
  <h1 ><span rel="localize[listaTitle]">Lista cuestionarios</span></h1>
</div><!-- /header -->

<div data-role="content" id="divCuerpo">
  <iframe id="iframeCuestionario" style="height:300px;width:300px;padding:0px">
```

```
</iframe>  
</div><!-- /content -->
```

En el evento pageshow es donde indicamos el destino del iframe:

```
$(document).on('pageshow', "div#pageCuestionario", function(event){  
    $('#iframeCuestionario').attr("src", app.urlSakai + "direct/sam_pub/1");  
    $("#iframeCuestionario").height($(window).height() - 50);  
    $("#iframeCuestionario").width($(window).width());  
});
```

Y finalmente así es como se ve la ventana desde el navegador:

6.2.8. Despliegue de la aplicación en el dispositivo móvil

Existen varias formas de conseguir que la aplicación quede instalada en el dispositivo móvil del usuario. Vamos a distinguirlas en dos grandes grupos:

- Aplicaciones que se encargan de encapsular el desarrollo HTML5 en un paquete instalable por el dispositivo. En esta distinción se pueden clasificar tanto las aplicaciones nativas que contengan algún control para poder visualizar el contenido HTML (WebView) como el uso de frameworks completos para esta tarea, como puede ser el caso de Phonegap/Cordova. Esta solución presenta una serie de inconvenientes que nos van a hacer preferir alguna otra alternativa:
 - No es un método especialmente intuitivo de instalación, pues deberemos hacerles llegar de alguna manera esa aplicación a los participantes para que puedan instalarla en sus dispositivos
 - En el caso por ejemplo de Apple, directamente no nos va a permitir instalar aplicaciones si no han sido primeramente validadas por ellos, y han pasado por su tienda de aplicaciones (App Store)
 - Para subir una aplicación cualesquiera a la App Store, se necesita una licencia de desarrollo, que hace que subir una aplicación a la misma no baje de los 100 dólares
 - En el caso de Phonegap, estamos utilizando un framework mucho más potente y diseñado para intentar aprovechar las capacidades del teléfono que están vetadas a aplicaciones por ejemplo HTML5, por lo que no es una solución especialmente óptima.
 - Nos obliga a tener un proyecto separado para cada plataforma que vayamos a soportar. A pesar de que la mayoría del código sí que será común a todas las plataformas, existirá la parte encargada del empaquetado que debe ser obligatoriamente dependiente de la tecnología. Esto significa que si en un futuro se desarrolla otra plataforma que no tengamos soportada, hasta que no tengamos su entorno de desarrollo no podremos utilizar nuestra aplicación.
- En contrapartida, el estándar HTML5 proporciona una alternativa para que sea el propio navegador web el que mantenga la aplicación instalada en caché. Dicha

tecnología es llamada Application Cache, y se basa en la creación de un fichero de manifiesto en el que indicaremos los ficheros que deben permanecer en la caché.

Lo primero que debemos hacer es indicar en la cabecera del documento HTML el fichero de manifiesto que vamos a utilizar:

```
<html manifest="manifest.appcache">
```

A continuación indicaremos un fichero llamado manifest.appcache con una estructura como esta:

```
CACHE MANIFEST
# 2012-09-03. Version 1.0

# Explicitly cached entries
index.html
css/images/ajax-loader.gif
css/images/ajax-loader.png
css/images/icons-18-black.png
css/images/icons-18-white.png
. . .

# offline.html will be displayed if the user is offline
FALLBACK:

# All other resources (e.g. sites) require the user to be online.
NETWORK:
*

# Additional resources to cache
CACHE:
```

En cada apartado indicaremos las opciones que vayamos a utilizar: Indicar explícitamente los ficheros que se van a cachear, indicar si se va a sustituir algún fichero por otro si se encuentra offline, indicar los recursos para los que se requiere que el usuario esté online, o cacheo de recursos adicionales.

Cuando se entre en la página la primera vez, el sistema identificará el fichero de manifiesto, lo bajará y lo procesará, descargando a caché los elementos que estén referidos en el mismo. La siguiente vez que entre, se bajará de nuevo el fichero de manifiesto y si no ha sufrido cambios, utilizará los fichero que tenga alojados en caché. Si detecta algún cambio, se procederá a volverse a descargar y guardar en caché los archivos que se encuentren en el nuevo manifiesto.

Si se ha producido un error en la descarga del fichero de manifiesto (por ejemplo porque el dispositivo móvil se encuentra sin cobertura), se lanzará un evento que nos lo avise, aunque de todos modos el sistema procederá a utilizar los archivos alojados en caché. Como ejemplo, hemos preparado un manejador que indica que si ha habido un error en la descarga del manifiesto se indique obligatoriamente que nos encontramos en el modo de funcionamiento offline.

```
$(window.applicationCache).bind('error', function(event){
  $("input#cbxModoOffline").attr("checked",true).checkboxradio("refresh");
  $('input#pwd').textinput('disable');
  $('input#savePassword').checkboxradio('disable');
```

```
$('#input#cbxModoOffline').checkboxradio('disable');  
});
```

En las pruebas que hemos realizado con diferentes dispositivos, tanto los que disponían del sistema Android como los iPhone con iOS permitían el uso del Application Caché y se comportaban de la forma que hemos descrito. En los navegadores de escritorio, Chrome se comportaba de esta manera, mientras que Firefox pedía previamente confirmación al usuario cuando detectaba que una página intentaba utilizarlo.

7. Conclusiones

Mediante este Proyecto Fin de Carrera hemos podido adentrarnos en el mundo de los sistemas de e-learning y de los LMS. Hemos estudiado el entorno Sakai y hemos conseguido instalarlo, ejecutarlo y realizarle una administración básica. Además, nos hemos adentrado en las facilidades que nos da para poder explotar la información de la que dispone para sistemas externos, mediante los servicios REST y expone mediante EntityProvider/EntityBroker.

Hemos realizado el desarrollo de una herramienta Java, mediante las bibliotecas Jackson/Jersey, para explotar los servicios web que Sakai nos ofrecía, pero que a causa de las restricciones de seguridad de los navegadores no podíamos utilizar directamente. Este desarrollo, por lo tanto, actúa como proxy entre la aplicación móvil y Sakai, mientras que mediante una estrategia CORS (Cross-Origin Resource Sharing) permite que la respuesta de Sakai sea aceptable por parte de los navegadores.

También hemos realizado la aplicación para dispositivos móviles, desarrollada mediante el framework JQueryMobile, que nos permite identificarnos en el entorno de Sakai y crear una sesión. Después, mediante esa sesión, podemos consultar las distintas encuestas que el usuario identificado tiene disponibles. Seleccionando una de esas opciones, tenemos la opción de guardarla de manera offline para el trabajo cuando no existe conexión, además de seleccionarla para votar y consultar los resultados de la encuesta.

Nos hubiera gustado profundizar más en el desarrollo de esta herramienta, pero por desgracia se han producido algunas situaciones que nos han complicado el desarrollo. Las más importantes se refieren a los servicios web que nos ofrece Sakai, que han supuesto alguna dificultad en la implementación:

- Tal y como está desarrollado el servicio que nos ofrece la lista de encuestas, hemos identificado un error en su implementación. A la hora de pedir las encuestas a las que un participante puede acceder, se indicaba incorrectamente la restricción que comprobaba si la encuesta estaba abierta o no. Se seleccionaban las encuestas cuya fecha de apertura y de cierre fueran igual a la fecha actual, lo que obviamente hacía que no se seleccionara ninguna.

David de Andrés se encargó de identificar el problema y, con la ayuda de David Roldán, preparar el parche del sistema y mandarlo a la fundación Sakai para su evaluación e incorporación en una próxima versión. (<https://jira.sakaiproject.org/browse/POLL-175>)

Esto hizo que se retrasara la implementación hasta identificar el error, y comprobar que de todas formas cuando se pedían las encuestas que podías administrar en lugar de las que puedes participar sí que se devolvían correctamente, lo que permitió retomar el desarrollo.

- Relacionado con lo anterior, se propuso realizar una ventana previa a la lista de encuestas para organizarlas dependiendo de los sitios que las contengan. La idea era mostrar una lista de los sitios que tuvieran encuestas disponibles, y en el momento de la selección mostrar la lista de encuestas con las que fueran de ese sitio.

El problema es que si seleccionamos la lista de encuestas en modo administración (el único que nos devuelve datos) devuelve todas las encuestas, aunque se le haya indicado el sitio que quieres seleccionar. Como si pedimos los datos de encuestas a las que podemos acceder nos encontramos con el error anterior, este requerimiento no lo podíamos probar, por lo que esperaremos hasta que se tramite este parche

- Como hemos explicado ya anteriormente en la memoria, el servicio disponible para SAMIGO no nos ha servido para mucho, puesto que ni nos permitía obtener la lista de exámenes que tenía un usuario determinado, ni nos permitía acceder directamente a las opciones de una encuesta. Por lo tanto, hasta que no se desarrollen los servicios web necesarios, esta aplicación no se podrá utilizar para el desarrollo de aplicaciones para dispositivo móvil del estilo de la que hemos realizado para las encuestas.
- Los servicios REST de Sakai no disponen de un fichero WADL (Web Application Description Language) asociado, lo que hace que el mapeo entre los tipos de datos usados en el servicio y, por ejemplo, clases Java que podamos utilizar en el programa del proxy. Esto hace que dicho mapeo se deba hacer a mano, lo que dificulta el desarrollo de la aplicación. De hecho, el proxy en estos momentos no intenta hacer ninguna conversión de los datos que envía Sakai, cuando sabemos que existen datos que no harían falta y podríamos no enviarlos al dispositivo móvil, lo que optimizaría el trasiego de información enviada por la red móvil.

De todas formas, una vez identificados estos problemas hemos podido utilizar los servicios del sistema Sakai sin más problema que los de se han comentado, y se ha comportado de la forma en la que esperamos que lo haga un servicio REST, por lo que el desarrollo de la aplicación móvil en JQueryMobile ha sido en cierto modo confortable. Las pruebas, tanto en distintos dispositivos (basados en Android y en iOS) como en los navegadores de escritorio que teníamos disponibles (Chrome y Firefox) han sido en general positivas (salvo en determinados casos con terminales con capacidades muy pobres), por lo que concluimos que la aplicación es usable en la mayoría de dispositivos del mercado (incluidos los navegadores de escritorio más utilizados), por lo que la valoración final ha sido muy positiva.

De todas formas, la aplicación es susceptible de ser mejorada y ampliada con diversa funcionalidad. Una de las cosas más obvias es el proxy, de forma que realmente pueda mapear la respuesta que mande Sakai y poder moldearla de forma que le mandemos a la aplicación móvil la información que precise. Además, esto nos permitiría que fuera el proxy el que se encargara de organizar toda la información, llamando con una sola llamada a más de un servicio en Sakai si es necesario, lo que permitiría simplificar el código javascript de la aplicación móvil, mientras optimizamos la información que manejamos.

Además, la aplicación se ha organizado de manera que se comporte 'casi' como un portal de entrada al entorno Sakai. Es decir, en el menú principal de la aplicación hemos indicado las opciones de Encuestas y Cuestionarios, pero si existe una herramienta con una interfaz REST a la que le quisiéramos hacer una utilidad similar a la desarrollada, nada nos impide crear un nuevo botón e implementar los servicios, manejadores de eventos y páginas HTML necesarios, cogiendo como modelo la utilidad que hemos desarrollado.

También cabe decir que este proyecto servirá para realizar una presentación mediante videoconferencia en el I Congreso Iberoamericano de Sakai, que tendrá lugar del 16 al 18 de Octubre en Río de Janeiro. Los detalles del evento se encuentran en la dirección <http://sakai.ucam.edu/>

El objetivo de esta presentación será difundir el trabajo que se ha hecho, de forma que los participantes puedan comprobar el tipo de aplicaciones que se puede hacer utilizando los servicios que nos expone Sakai, en el ámbito de las tecnologías de dispositivos móviles.

Por lo tanto, este proyecto no sólo servirá en el ámbito de la docencia en la UPV, sino que se va a intentar difundir al resto de la comunidad Sakai, para que el que necesite una funcionalidad similar pueda basarse en lo realizado y ampliarla en caso necesario, lo que haría el desarrollo más rico e interesante.

Por último, no quiero terminar el proyecto sin agradecer a David Roldán y Alberto Palomares su apoyo en la realización de este proyecto.

8. Bibliografía

1. D. Roldán, R. E. Mengod, D. Merino. SAKAI. Administración, configuración y desarrollo de aplicaciones. Ed Ra-Ma, 2011
2. A. Berg, M. Korcuska. Sakai Courseware Management. The Official Guide. Packt Publishing. 2009
3. CLE Mobile - A cross-platform mobile application for the Sakai CLE, [Online] <https://confluence.sakaiproject.org/display/CLEMBL/Home>, 2011.
4. M. Pilgrim. HTML5 Up and Running. O'Reilly Press, 2010
5. E. Castledine, C. Sharkie. JQuery: Novice to Ninja. Sitepoint, 2011
6. Entrevista a Javier Jiménez Villar, autor de LungoJS [online] <http://www.genbetadev.com/entrevistas/entrevista-a-javier-jimenez-villar-autor-de-lungojs>, 2011
7. JSON example with Jersey + Jackson [online]
8. D. Sevilla. Servicios Rest con Eclipse y JAX-RS. Software como servicio y distribuído. 2011. <http://www.mkyong.com/webservices/jax-rs/json-example-with-jersey-jackson/>, 2011
9. REST with Java (JAX-RS) using Jersey – Tutorial [online] <http://www.vogella.com/articles/REST/article.html>, 2012
10. Las dichosas Cross Domain Calls [online] <http://blogs.msdn.com/b/davidsalgado/archive/2008/08/20/las-dichosas-cross-domain-calls.aspx>, 2008
11. JSONP, llamadas AJAX entre dominios [online] <http://web.ontuts.com/tutoriales/jsonp-llamadas-ajax-entre-dominios/>, 2009
12. M. Firtman. JQuery Mobile. Up and Running. O'Reilly Press, 2012
13. Guía para principiantes sobre el uso de la caché de aplicaciones [online] <http://www.html5rocks.com/es/tutorials/appcache/beginner/>, 2011
14. Traduciendo aplicaciones de JQuery Mobile [online] <http://www.funcion13.com/2012/04/03/traduciendo-aplicaciones-de-jquery-mobile/>, 2012