Desarrollo de Webs interactivas con filosofía AJAX: El TRIVIAL.GZ

Nieves R. Brisaboa[1], **Miguel R. Luaces**[1], **Ángeles S. Places**[1], **Diego Seco Naveiras**[2]

[1]Laboratorio de Bases de Datos
Facultad de Informática
Universidade da Coruña
Campus de Elviña, S/N 15071
A Coruña, España
e-mail: {brisaboa, luaces, asplaces}@udc.es

[2]Enxenio, S.L.
Estrada dos Baños de Arteixo
Edificio San Cristobal, Portal B
15071 A Coruña, España
e-mail: dieseco@enxenio.es

Abstract

We present in this paper the architecture and some implementation details of a web-based version of a Trivial game. Our implementation achieves such a high degree of interactivity between the players that they perceive the game as being played real-time. More importantly, no plug-in or applet is used in the architecture of the system. These properties are achieved by means of a carefully designed architecture that uses AJAX (*Asynchronous JavaScript and XML*) for data exchange. Using this approach, it is possible to develop any type of web-based collaborative software with few load on the web server. In the paper, we analyze traditional architectures for web-based applications and we show how our approach overcomes their limitations. Furthermore, we proof the efficiency of our approach by means of an empirical comparison.

Keywords: Collaborative software, Web, AJAX

Resumen

En este artículo se presenta la arquitectura y algunos detalles de la implementación de una aplicación Web (un juego virtual de tipo Trivial) que, sin el uso de ningún plug-in o applet, permite una gran interactividad entre los usuarios, hasta el punto de que estos tienen la percepción de comunicación entre ellos en tiempo real. La percepción de interactividad en tiempo real se consigue mediante una arquitectura especialmente diseñada que se apoya, además, en la filosofía de AJAX (*Asynchronous JavaScript and XML*) para el intercambio de datos. Utilizando esta aproximación es posible desarrollar cualquier tipo de software colaborativo en Web con muy poca carga de trabajo para el servidor. En el articulo se analizan las arquitecturas tradicionales para el desarrollo de aplicaciones Web y se muestra cómo nuestro enfoque supera sus limitaciones. Además, se avala la eficacia de esta aproximación con los resultados de la valoración empírica realizada.

Palabras claves: Software colaborativo, Web, AJAX

1. INTRODUCCIÓN

El nivel de madurez de los usuarios en Internet y la calidad de las conexiones y los servicios disponibles, están produciendo una demanda creciente de interactividad en las aplicaciones Web, no sólo entre el usuario y el sistema, sino también entre los propios usuarios.

Sin embargo, las características tradicionales de las aplicaciones Web dificultan el desarrollo de aplicaciones colaborativas o de juegos que requieran interacción entre los usuarios en tiempo real debido a dos factores fundamentales:

- No permiten que los diferentes clientes intercambien información entre sí. Es decir, en toda aplicación Web la comunicación se establece entre un cliente y el servidor y nunca entre dos clientes, por tanto, el intercambio de datos entre clientes debe hacerse a través del servidor.
- Un servidor Web sólo responde a peticiones de clientes y no puede nunca tomar la iniciativa de enviar información nueva a los clientes conectados. Esto significa que el servidor no puede comunicar a los demás usuarios la información que le llega de uno de ellos mientras que estos no la soliciten, dificultando así las posibilidades de interacción entre usuarios.

Como consecuencia de estas dos características propias de las aplicaciones Web, cuando se desea crear una aplicación en la que los usuarios colaboren e interactúen entre sí en tiempo real en el desarrollo de una tarea o juego, es preciso que cada uno de ellos descargue e instale en su propio ordenador un software de tipo plug-in para el navegador web que permita mantener la conexión y gestione el intercambio de mensajes entre ellos.

Una alternativa que se podría considerar, en casos en los que el intercambio de datos entre los usuarios no sea muy denso, es que cada cliente le envíe al servidor los nuevos datos generados y, en la siguiente petición de página por parte de los demás clientes, el servidor les envíe la nueva página con la información actualizada. En este caso los clientes pueden estar programados para hacer peticiones periódicas (y frecuentes) al servidor mediante scripts incluidos en la página. El problema de esta alternativa es que si el intercambio de datos es muy frecuente o, en el peor de los casos, se desea que se perciba como en tiempo real, el servidor tendrá una carga de trabajo muy elevada, ya que deberá crear y enviar nuevas páginas constantemente, lo que en la práctica se traduce en una limitación del número de usuarios que pueden interactuar. Sin embargo, esta aproximación presenta con respecto a la anterior la ventaja de liberar a los usuarios de tener que descargar, instalar y configurar un plug-in, lo que, en algunos dominios de aplicación (sistemas dirigidos a entornos en los que puede haber usuarios poco expertos), es una restricción insalvable.

En el Laboratorio de Bases de Datos de la Universidad de A Coruña [3] hemos desarrollado una arquitectura específicamente concebida para simular la interactividad entre usuarios, que, además, utiliza la filosofía AJAX para el intercambio de información entre el servidor y los clientes. Con esta aproximación se produce en los usuarios la percepción de que interactúan entre sí en tiempo real como si estuviesen usando una aplicación que mantuviese una conexión múltiple entre ellos. Esa arquitectura, que puede ser usada para el desarrollo de cualquier aplicación Web colaborativa, la hemos usado para implementar una versión virtual del *Trivial*, el clásico juego de mesa de preguntas y respuestas.

Esta estrategia nos permitió crear un juego que, a diferencia de otras aplicaciones de este tipo, no requiere que los jugadores descarguen e instalen un plug-in, permitiendo al mismo tiempo un número muy grande de partidas simultáneas con múltiples jugadores/as en cada una de ellas. Nuestro trivial virtual, al que llamamos *Trivial.gz* tiene, con respecto a la versión original del juego de mesa, algunas variaciones orientadas a sacarle partido al entorno virtual, minimizando el problema que

supone que los jugadores no compartan el mismo espacio físico durante las partidas. El *Trivial.gz* fue promovido por la *Asociación Socio-Pedagóxica Galega (AS-PG)* [6] para potenciar el uso de la lengua gallega en Internet y fue subvencionado por el gobierno de Galicia. El Trivial.gz fue inaugurado durante las jornadas *Xuventude Galiza Net* [2], celebradas en Santiago de Compostela durante los días 7, 8 y 9 de abril de 2006, y está actualmente disponible en la siguiente URL `http://www.as-pg.com/trivial.gz/`.

Este desarrollo nos ha permitido evaluar y comparar nuestra propuesta con las aproximaciones tradicionales de desarrollo de aplicaciones Web. En este artículo presentamos la arquitectura utilizada, una descripción básica de la filosofía AJAX y los resultados de nuestra experiencia. Como se podrá observar en el apartado de datos empíricos, el software desarrollado permite que un servidor normal (Pc monoprocesador con 1 Gb de RAM) haya sido, en nuestras pruebas, capaz de atender un número muy importante de partidas simultáneas (del orden de 1000) con numerosos jugadores (hasta 6) en cada partida.

El resto del artículo se estructura como sigue. En la sección 2 se describe el funcionamiento y las reglas del Trivial.gz con el objetivo de transmitir el nivel de interactividad que es posible implementar con este nuevo paradigma de programación Web. A continuación, en la sección 3, se presentan las diferencias entre la arquitectura tradicional de las aplicaciones web y la arquitectura de nuestra aplicación. En la sección 4 se describe con más detalle AJAX, citando las limitaciones que supera con respecto a otras tecnologías y las ventajas que ello supone para el desarrollo sistemático de aplicaciones web interactivas. La arquitectura de la aplicación se describe con detalle en la sección 5. En la sección 6 se describe el entorno en el que se ha inaugurado el Trivial.gz así construido y se presentan algunas cifras obtenidas de los accesos recibidos. Finalmente, en la sección 6 se presentan nuestras conclusiones y algunas ideas para trabajos futuros.

2. EL TRIVIAL.GZ

El Trivial.gz fue promovido por la *Asociación Socio-Pedagóxica Galega* [6] para fomentar el uso del gallego entre la gente joven y su creación fue financiada por el gobierno de Galicia. Se trataba de crear un juego para la Web basado en el clásico trivial de mesa, que fuese multipartida y multijugador y que se pudiese ejecutar directamente sobre cualquier navegador Web sin ningún tipo de plug-in.

Se modificaron ciertos aspectos del juego de mesa para adaptarlo al entorno Web, por ejemplo, el hecho de que sólo quien posee el turno puede realmente jugar en cada momento, produciría aburrimiento en la versión virtual. Fue necesario además simular en el espacio virtual la interacción entre jugadores, las acciones de tirar el dado, mover la ficha, observar las posiciones y movimientos de los demás jugadores y jugadoras, etc. Por otro lado, se adaptó el juego para que también se pudiese jugar en solitario acumulando puntos y comparándose con un ranking general de jugadores/as. Para fidelizar jugadoras/es se incentiva que se registren de modo que sólo se mantiene la acumulación de puntos en la base de datos de quienes están registrados. Además sólo las personas registradas pueden crear partidas y parametrizarlas, abriéndolas a todo el mundo o cerrándolas para que sólo puedan jugar sus amistades por invitación. Se incorporó además la posibilidad de configurar listas de amistades para facilitar la invitación a partidas concretas.

En líneas generales, el Trivial.gz consiste en contestar correctamente preguntas, previamente clasificadas en tres niveles de dificultad y en seis temas diferentes: *Cultura y espectáculos*, *Geografía*, *Historia*, *Lengua y Literatura*, *Ciencia* y *Mundo*). Todas las preguntas tienen tres respuestas de las cuales sólo una es correcta. El elemento principal del juego es un tablero en forma de hexágono dividido en casillas de diferentes colores (ver figura 1), en el que cada color va asociado a un tema concreto.

Figura 1: Pantalla durante el juego

Inicialmente, todos los jugadores/as parten con sus fichas de la casilla central. Quien tiene el turno tira el dado (pulsando sobre una animación que representa el dado girando) y se mueve por el tablero, en cualquier dirección, saltando tantas casillas como indique el dado (el sistema controla que sólo se pueda mover a casillas válidas para el valor del dado). Cuando se elige una casilla, se presenta a todos los jugadores una pregunta del tema asociado al color de la casilla. Siempre que se acierta una pregunta (se tenga o no el turno) se acumulan los puntos correspondientes. Si no se acierta se pierde el turno que pasa al siguiente jugador/a.

Para ganar la partida hay que conseguir reunir los seis *pentágonos* (los *quesitos* del juego de sobremesa) y, después de haberlos conseguido, acertar una pregunta sobre el hexágono central del tablero. Así las casillas que se buscan, al moverse durante el juego, son las de los vértices del hexágono ya que cada vez que se acierta la pregunta asociada a un vértice se consigue el *pentágono* del color correspondiente. Evidentemente, hay un vértice de cada color.

Consideramos que los elementos que se le han incorporado al Trivial de mesa en el Trivial.gz son imprescindibles para que cualquier juego de mesa tenga éxito en la Web, donde quienes juegan una misma partida no comparten el mismo espacio físico. Sin embargo, estos cambios aumentan notablemente el nivel de interacción entre usuarios que debe soportar la arquitectura de la aplicación. Así, durante las partidas, se informa a todos los jugadores/as de lo que marca el dado en cada tirada, de la casilla a la que mueve quien tiene el turno y de que pregunta salió. Además, se mantienen actualizadas las posiciones de las fichas de cada jugador/a en el tablero (indicando quien tiene el turno), las puntuaciones y los pentágonos conseguidos por cada uno, de manera que se perciba que se sigue la evolución de la partida en tiempo real. Además, se ha incluido un chat para que quienes juegan puedan hablar entre sí. En resumen, las características del juego virtual Trivial.gz, que en ocasiones no coinciden con el juego de mesa, son:

- *Se ve el tablero con las fichas, el valor del dado y se indica el turno*. Evidentemente en el juego de mesa también se ve todo esto que aquí hay que reproducir virtualmente para que haya sensación de compartir el espacio virtual de la partida.

- *Se puede "hablar":* Durante la partida, los jugadores pueden comunicarse entre sí a través de un chat. De nuevo ese chat trata de simular la interacción verbal de los jugadores/as del juego de mesa.

- *Los pentágonos también se pierden:* Para hacer el juego más dinámico y divertido hemos introducido la modificación de que los pentágonos también se puedan perder si se falla la pregunta de la casilla del vértice al que corresponda.

- *Se facilita caer en los vértices:* Si se cae en la casilla en forma de pentágono que hay en medio de cada lateral del hexágono se salta automáticamente al vértice del color correspondiente facilitándose así la obtención o la perdida de pentágonos.

- *Todos juegan:* Todos los jugadores/as pueden intentar contestar la pregunta que le salga a quien tiene el turno y aumentarán su puntuación si eligen la respuesta correcta, aunque sólo podrá conseguir/perder pentágonos quien tenga el turno. Además, mientras se está pensando la respuesta a una pregunta, se puede ver como otros jugadores/as aumentan sus puntos por haberla contestado bien. La respuesta correcta no la verá cada quien hasta que agota el tiempo o responde erróneamente.

- *Tiempo límite:* Quien tiene el turno debe contestar la pregunta en un tiempo máximo de 30 segundos. Los restantes jugadores tienen solo el tiempo que tarde en contestar quien tiene el turno. Es decir, quien tiene el turno, en cuanto acierta una pregunta, puede volver a tirar el dado y mover su ficha generndo así una nueva pregunta que es enviada de nuevo a todos los jugadores/as de la partida, cortándoles el tiempo disponible para contestar la pregunta anterior, que en todo caso será como máximo de 30 segundos también. Así, quien tiene el turno puede dificultar que los demás jugadores consigan acumular puntos respondiendo rápidamente.

- *Historial de juego:* Se anota la actuación de cada jugador/a registrado en cada partida, de manera que la aplicación ofrece datos sobre el número de partidas jugadas, ganadas y perdidas, puntos por temas y clasificaciones globales.

Cualquier persona registrada puede comenzar una partida y, al hacerlo, se convierte en el *director/a* de la misma. Los parámetros que debe establecer para configurar la partida antes de comenzar a jugar son:

- *Número de jugadores:* es posible jugar al Trivial.gz en solitario o se puede competir con otras personas. En una partida puede haber hasta un máximo de seis jugadores/as. Quien crea la partida puede restringirla a personas concretas o dejarla abierta a cualquier persona (registrada o invitada) que acceda al juego. Puede además tener una lista de amistades para facilitar el restringir la partida a esas personas concretas

- *Número de temas:* es posible jugar con los seis temas o restringir su número a tres que podrá elegir.

- *Dificultad:* se puede elegir el nivel de dificultad de una partida: fácil, intermedia o difícil. La elección que se haga implicará que aproximadamente el 60 por ciento de las preguntas serán de ese nivel de dificultad.

En la figura 1 puede verse una captura de pantalla durante el desarrollo de una partida. En ella pueden distinguirse una serie de elementos estáticos (por ejemplo, el tablero de juego) y otros que cambian según los datos que se reciben (por ejemplo, la posición de las fichas de los participantes, el valor del dado, las puntuaciones, las preguntas, etc.).

En la actualidad el Trivial.gz está instalado en el servidor web de la Asociación Socio-Pedagóxica Galega (`http://www.as-pg.com/trivial.gz`), y tiene más de 500 usuarios registrados y de 2.500 preguntas.

3. DIFERENCIAS CON LAS APLICACIONES WEB TRADICIONALES

Hasta el momento la arquitectura de la mayor parte de las aplicaciones Web sigue una de las dos filosofías habituales:

- *Aplicaciones ejecutadas del lado del servidor:* Son las clásicas aplicaciones Web. Todo el procesamiento recae en el servidor. Así, cada petición de un usuario le supone al servidor un tiempo de procesamiento y el envío de una página Web completa al cliente. Debido a que el número de páginas que el servidor tiene que procesar y enviar a los clientes crece cuanto mayor sea la interactividad (usuario-aplicación) permitida por la aplicación y según aumenta el número de clientes simultáneos, este tipo de arquitecturas son poco escalables ya que se hacen insostenibles en aplicaciones con mucho intercambio de datos y/o que requieran atender a muchos usuarios simultáneos.

- *Aplicaciones del lado del cliente:* En este tipo de arquitectura, todo el proceso recae sobre el equipo del cliente, minimizando el intercambio de información con el servidor que, de este modo, se libera de procesar tantas peticiones de usuario y de generar tantas nuevas páginas Web para enviar cada respuesta. Este tipo de aplicaciones puede implementarse mediante programas plug-in que es preciso descargar, instalar y configurar, o, en el mejor de los casos mediante Applets incrustados en la página Web que requieren, también, que esté instalada y activada la Java Virtual Machine (JVM). En cualquier caso, esta filosofía hace que las aplicaciones sean incómodas de configurar para los usuarios y requieren cierto grado de destreza informática para la descarga e instalación del plug-in o de la JVM y su configuración, que limitan su uso generalizado.

Una alternativa intermedia es el uso de simples scripts en las páginas Web de modo que la aplicación cliente pueda tener cierta capacidad de proceso sin necesidad de instalar y configurar la JVM u otro plug-in. AJAX (*Asynchronous JavaScript and XML*), es una nueva filosofía para crear aplicaciones Web que se basa precisamente en las posibilidades de JavaScript. Con AJAX todo el proceso que pueda ser realizado en el cliente se programa mediante JavaScript, que, como es sabido, no precisa ningún tipo de configuración especial ni una Java Virtual Machine (JVM) para poder ser ejecutado. La comunicación con el servidor se implementa mediante el intercambio de mensajes cortos formateados con XML [7]. Esos mensajes son interpretados en el servidor que formatea y envía una respuesta, también en XML, en vez de una página Web completa. El mensaje XML recibido por el cliente es interpretado mediante código JavaScript y utilizado para hacer las modificaciones oportunas en la página Web actual. Google [5] ha sido pionero en el uso de AJAX [4]. Así, podemos verlo en Google Suggest, Google Maps o Gmail.

El uso de AJAX facilita el trabajo de los servidores de aplicaciones Web que tienen que atender cada vez a más usuarios potenciales por lo que el coste de procesamiento del servidor para la generación de nuevas páginas se convierte en un factor crítico en el desarrollo de dichas aplicaciones.

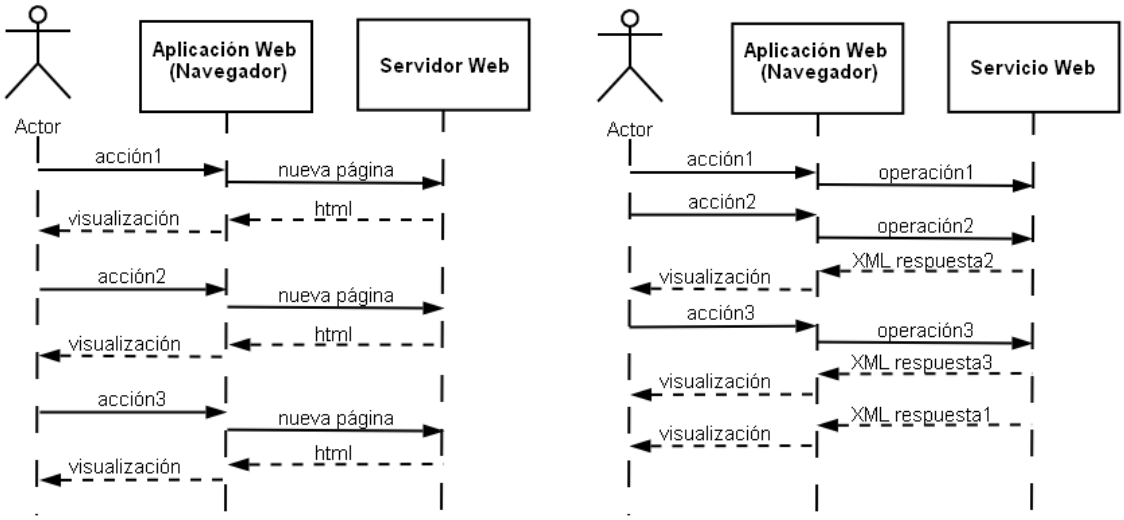
Pero el uso de AJAX por sí mismo no resuelve la creación de aplicaciones Web que permitan la interactividad entre usuarios en tiempo real, ya que, aunque agiliza el intercambio de datos servidor-clientes, no cambia la arquitectura de las aplicaciones Web. Es decir, el intercambio de datos sigue realizándose entre servidor y clientes (y no entre clientes) y el servidor sigue sin poder tomar la iniciativa de enviar a los demás clientes los datos que acaba de recibir de uno en concreto.

El *Trivial.gz* está soportado por una arquitectura diseñada para permitir/simular interactividad entre usuarios, que, además, utiliza la filosofía AJAX para el intercambio de información entre el servidor y los clientes. Básicamente, la arquitectura implementada consiste en que cada cliente cursa peticiones al servidor cada 4 segundos de modo que cada 4 segundos recibe del servidor toda la

información relevante de la actividad de los demás jugadores de la misma partida (el servidor puede atender un número indefinido de partidas simultáneas). Además, el servidor actúa de modo diferente según el estado en el que se encuentra la partida y cambia de estados según las acciones que le comunica el jugador que tiene el turno en esa partida. Evidentemente, esta filosofía produce un elevado intercambio de datos entre el servidor y los clientes que sería inmanejable si no fuera por el uso de AJAX. Recuérdese que AJAX permite la utilización de un protocolo ligero (XML) de intercambio de datos entre el servidor y los clientes, que libera al servidor de tener que crear las nuevas páginas Web cada vez que le llegan peticiones desde los clientes. La idea fundamental es no centralizar toda la lógica en el servidor, sino delegar parte del proceso en los clientes, aunque codificado en Javascript de modo que no es necesario descargar ningún software ni configurar el navegador de modo especial ni tener instalada la *Java virtual machine*. Resumiendo, el cliente sigue realizando peticiones al servidor pero no de páginas completas, sino sólamente de aquellos datos que necesita. Los clientes, una vez reciben esos datos formateados en XML, los decodifican e interpretan y modifican la página actual que esté viendo el usuario.

4. LA FILOSOFÍA AJAX

AJAX (*Asynchronous JavaScript and XML*) es el nombre que recibe una nueva técnica de reciente aparición en el ámbito de desarrollo de aplicaciones web. En realidad, AJAX no es una nueva tecnología sino que es el resultado de la combinación de distintas tecnologías ya existentes. El elemento central es la utilización de forma asíncrona del API *XMLHttpRequest* presente en los navegadores web de última generación. Esto permite que, utilizando un lenguaje de tipo script, la página web que se está visualizando en el navegador del cliente haga una petición de información a un servicio web sin bloquear la actividad del usuario. Cuando el servicio web devuelve la información, el navegador invoca una función específica del lenguaje script que puede procesar la respuesta y modificar la página en consecuencia.



(a) Aplicación web tradicional (b) Aplicación web AJAX

Figura 2: Interacción cliente-servidor en distintos modelos de aplicaciones

En la Figura 2 se muestra un diagrama de secuencia representado mediante el lenguaje UML que describe este funcionamiento. En ambas figuras el usuario invoca tres acciones sobre el interfaz de

usuario de la aplicación web. En la figura 2(a) se muestra el funcionamiento en el caso de aplicaciones tradicionales. En este caso el usuario tiene que esperar a que una acción termine antes de poder invocar otra. Además, la cantidad de procesamiento requerido en el servidor web y la cantidad de información transferida es bastante elevada. En la figura 2(b) se muestra la interacción en el caso de una aplicación que utiliza la tecnología AJAX. En este caso el usuario aprecia una mayor velocidad de respuesta en la aplicación porque no tiene que esperar a que termine la anterior petición. Además, la cantidad de procesamiento e información que se transfiere entre el servidor web y el cliente es menor.

Este modo de funcionamiento es mucho más ágil para el usuario que las tecnologías tradicionales de desarrollo de aplicaciones web en las que el peso de la interacción recaía en el servidor web. Con estas tecnologías, cualquier actualización del contenido de la página requiere una recarga completa de una parte de la misma y que el usuario espere a que esa recarga se complete. Utilizando AJAX ya no es necesario recargar la página sino que se sólo se altera su contenido, y el usuario no tiene que esperar a que este cambio se produzca.

Por otra parte, el hecho de reducir la cantidad de información que se intercambia entre el cliente y el servidor permite que las aplicaciones web desarrolladas con AJAX tengan mucha más interactividad que las tradicionales ya que la capacidad de procesamiento del servidor web se puede utilizar para atender un mayor número de peticiones simultáneas.

En resumen, AJAX es una técnica que consiste en el uso de estas tecnologías:

- HTML dinámico para presentar la información al usuario.
- Cambiar el modo de funcionamiento de las aplicaciones web para que hagan uso de servicios web que contestan a peticiones específicas en lugar de servidores web que componen páginas completas.
- XML para representar las peticiones a los servicios web y las respuestas de los mismos.
- Lenguajes de tipo script, como JavaScript, para implementar la lógica de las aplicaciones web.

En torno a AJAX surgen un conjunto de herramientas de desarrollo que permiten hacer más cómodo su empleo. Una de estas herramientas es *Direct Web Remoting (DWR)* [1]. Esta librería de código abierto proporciona dos grandes ventajas sobre el uso directo de AJAX. En primer lugar, permite al código de una página web utilizar clases programadas en Java en el servidor de forma transparente. Esto se consigue mediante la generación dinámica de código JavaScript que encapsula la utilización de AJAX para efectuar desde el cliente la llamada a código del servidor. Por otra parte, proporciona utilidades para facilitar la actualización de los contenidos de las páginas web en el cliente.

Estas dos tecnologías son el centro de la arquitectura de nuestra aplicación. Sin embargo, por sí solas no son la solución a los requerimientos planteados, sino que es necesario realizar un importante trabajo de diseño de la arquitectura de la aplicación que describimos en la siguiente sección.

5. ARQUITECTURA DETALLADA DEL SISTEMA

En la figura 3 se muestra la arquitectura de la aplicación de modo genérico. Como toda aplicación web, en la arquitectura del sistema se encuentran dos módulos diferenciados: el módulo de la aplicación que se ejecuta en el servidor web utilizando *Java Server Pages (JSP)*, y el módulo que se ejecuta en el navegador web del cliente utilizando JavaScript y HTML. Además, parte de la aplicación no presenta ninguna novedad (registro de usuarios, configuración de listas de amistades, consulta de estadísticas, etc.), por lo que no es objeto de este artículo, en el que sí nos centraremos en explicar cómo se realiza el control de las partidas para simular interactividad.

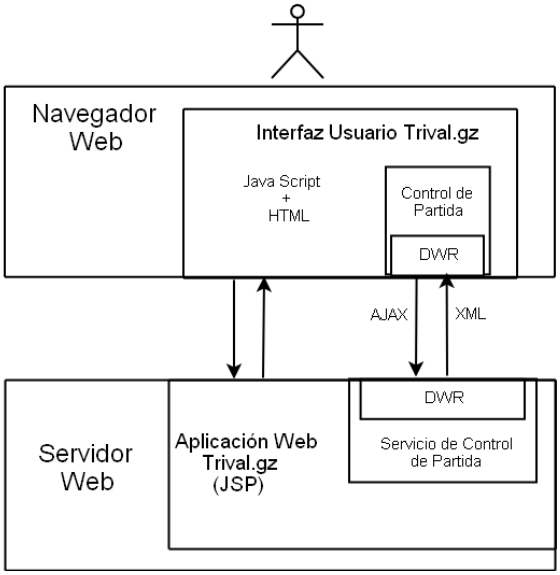


Figura 3: Arquitectura de la aplicación

5.1. Funcionalidad del cliente y del servidor

El servidor controla las partidas a través de una máquina de estados, que funciona de modo independiente para cada una de ellas, y llevando un registro de los datos relevantes de cada partida que se está ejecutando.

Durante el desarrollo de una partida, el servidor va pasando cíclicamente por una serie de estados marcados por las acciones del jugador que tiene el turno. Debido al funcionamiento del juego, hay dos tipos de jugadores diferenciados: el que tiene el turno, y los demás. El primero tiene el control de la partida y genera eventos que producen la actualización del estado de la partida (por ejemplo, tirar el dado, elegir la casilla, o contestar la pregunta). Los jugadores que no tienen el turno sólo generan un evento al contestar a la pregunta, el resto del tiempo consultan periódicamente al servicio el estado actual de la partida para actualizar el interfaz de usuario. Estos estados son los siguientes:

- *Estado inicial*. Antes de que el jugador en posesión del turno tire el dado. No hay información que enviar a los demás usuarios.
- *Dado tirado*. A los jugadores que no tienen el turno se les enviará el valor del dado cuando soliciten información de la partida.
- *Casilla movida*. A los jugadores que no tienen el turno se les enviará la casilla del jugador que sí tiene el turno junto con la pregunta actual (generada al llegarle al servidor el movimiento del jugador con el turno).
- *Pregunta contestada*. A los jugadores que no tienen el turno se les enviarán los puntos acumulados por cada jugador. También es posible que haya que enviar información relativa al cambio de turno, en caso de que el jugador que lo poseía hubiese fallado la pregunta.

Internamente, este módulo mantiene una lista de las partidas que están en funcionamiento en un instante dado. Cada una de estas partidas contiene a su vez una lista con los jugadores que participan en ella y una lista de preguntas en memoria preparadas para ser enviadas (conceptualmente, un *taco*

de preguntas). Esta lista de preguntas funciona a modo de caché en memoria que permite que los accesos a la base de datos sean menos frecuentes. Así, en lugar de acceder a la base de datos cada vez que se necesita una pregunta, se accede una única vez recuperando una gran cantidad de ellas que se utilizarán durante la partida. Por otra parte, para cada jugador que participa en la partida se mantiene en memoria el número de puntos acumulados, los pentágonos ganados, y la estadística de preguntas intentadas y acertadas por tema.

En el lado del cliente consiste en una página HTML con código JavaScript incrustado en ella. Su funcionamiento se basa en un temporizador implementado con código JavaScript ejecutándose en segundo plano que hace que cada 4 segundos se compruebe si se ha producido alguna modificación en el estado de la partida y se informa al servidor de los cambios que se hayan producido en el estado de cada jugador en particular.

Para interactuar con la parte servidor del sistema se utiliza DWR, es decir, el código JavaScript del cliente no tiene que utilizar directamente el API *XMLHttpRequest*, sino que invoca direcamente los métodos y los objetos del servidor de una forma muy similar a como se hace en el caso de servicios web o de CORBA.

5.2. Funcionamiento del sistema

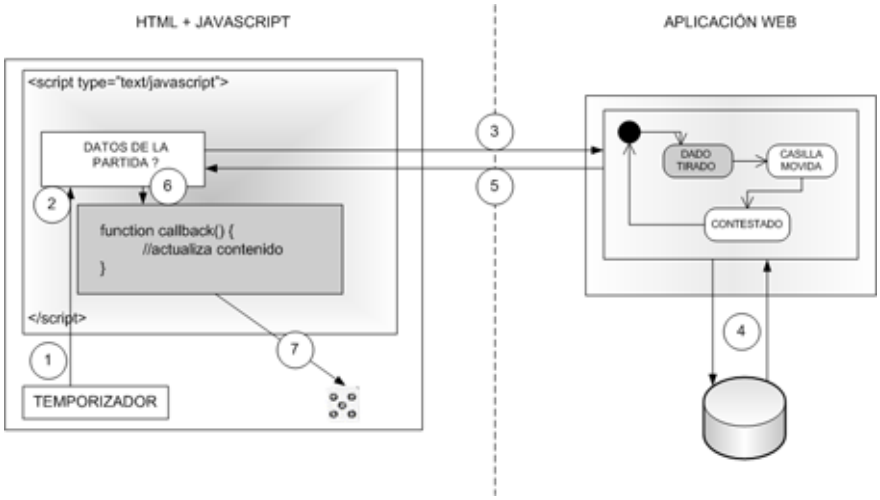


Figura 4: Protocolo de comunicación

En la figura 4 se puede ver el funcionamiento de un turno de la partida. Los números que hay en la figura se corresponden a una ordenación temporal de la secuencia de eventos y se van a emplear para organizar la explicación:

1. En primer lugar, el temporizador marca el momento de pedir información al servidor.
2. A continuación, se invoca la función del cliente que informa al servidor del estado del mismo, y solicita la información del estado de la partida.
3. Al servidor le llega la petición y se encarga de delegarla en su modelo de negocio.
4. Como ya se ha expuesto, en este contexto en concreto el servidor puede verse como una máquina de estados y como tal, en función del estado en el que se encuentre, obtendrá la

información necesaria del sistema de almacenamiento de la aplicación para generar la respuesta adecuada.

5. La respuesta se envía encapsulada al cliente. Con los datos obtenidos de la capa modelo simplemente se crea y se devuelve un objeto que representa el estado de la partida.

6. En el cliente se recibe la respuesta y se desencapsula la información de estado.

7. Finalmente, se actualiza el contenido de la página con la información de la respuesta y se vuelve a iniciar el turno.

6. VALORACIÓN EXPERIMENTAL

El Trivia.gz fue inaugurado en las jornadas *Xuventude Galiza Net* [2] de 2006. Durante estas jornadas, más de 1.500 usuarios de informática en general se dieron cita en el Palacio de Congresos de Galicia para participar en diversos concursos y juegos informáticos.

El Trivial.gz se instaló en un Pentium IV a 3.2 Ghz y con 1 GB de memoria RAM. Durante el evento, el servidor web registró casi un millón de *hits*(peticiones realizadas al servidor) y más de 600 visitas (conjunto de peticiones consecutivas desde una única IP). La mayor parte de las peticiones estuvieron concentradas en el transcurso de una hora durante la que se celebró un concurso en el que más de 200 usuarios compitieron por lograr más puntos que los demás. Durante esa hora se jugaron más de 800 partidas (776 de ellas completas, de las incompletas no se guarda registro) con una media de 3 jugadores por partida.

La tabla 1 presenta algunos datos sobre la cantidad de información que envía el servidor del Trivial.gz y su comparación con el mismo servidor si la aplicación se hubiese construído con una arquitectura web tradicional.

El código de la página web de la partida ocupa 35 Kb, sin tener en cuenta las imágenes ni el código Javascript responsable del envío de mensajes encapsulados en DWR. Por otra parte, el objeto DWR que el servidor envía a cada ordenador cliente, suponiendo un único jugador, ocupa 313 bytes, es decir, 0,31 Kb.

Como explicamos en el apartado dedicado a la arquitectura del sistema, es necesario actualizar la información de la partida en la pantalla de cada jugador de manera frecuente porque se trata de un juego en tiempo real y porque es necesario simular una interactividad entre los jugadores. Si tomamos 4 segundos como el tiempo óptimo de recarga (tal y como se está haciendo en la aplicación), eso supondría que en una aplicación web tradicional, el servidor tendría que enviar esos 35 Kb que ocupa la página 15 veces durante un minuto. Un total de 525 Kb por minuto y por jugador, cuando en el Trivial.gz suponen únicamente 4,7 Kb por minuto y jugador.

Considerando que durante 10 minutos de las jornadas en las que se inauguró el Trivial.gz, se jugaron más de 100 partidas simultáneamente (de las más de 800 que se jugaron durante la hora de competición), con una media de tres jugadores en cada una de ellas, los cálculos dicen que en una aplicación web tradicional se enviarían desde el servidor un total de 1,5 Gigabytes de información frente a los 20 Megabytes del Trivial.gz, un 75 % más de información.

Como consecuencia de este tráfico de información en la red, para una aplicación tradicional se necesitaría, en el mejor de los casos (considerando que el tráfico es uniforme), un ancho de banda de 2,56 MBytes/sg. Sin embargo, en el Trivial.gz el servidor sólo necesita un ancho de banda de 34,20 KBytes/sg.

	Aplicación tradicional	Trivial.gz
1 mensaje, 1 partida, 1 jugador	35 KBytes	3,1 KBytes
1 partida, 1 minuto, 3 jugadores	1.575 KBytes	20,52 KBytes
100 partidas, 10 minutos, 3 jugadores	1.538,09 MBytes	20,04 MBytes
Tráfico en la red en estos 10 min.	2,56 MBytes/sg	34,20 KBytes/sg

Tabla 1: Datos empíricos.

7. CONCLUSIONES

En este artículo se ha presentado la arquitectura de una aplicación web que implementa un juego de tipo trivial en el que los jugadores pueden seguir la partida en tiempo real. Esto se ha conseguido sin el uso de ningún tipo de plug-in o applet, lo que permite que el juego pueda ser utilizado en cualquier equipo sin presentar los problemas típicos asociados a la instalación de estos componentes.

Además, en la implementación se usa la filosofía AJAX para el intercambio de datos, lo que unido al protocolo de intercambio de mensajes diseñado para el Trivial.gz, junto con la arquitectura general para aplicaciones web colaborativas que proponemos, hace que el tráfico de información entre el servidor y los clientes sea mínimo. Utilizando esta aproximación es posible desarrollar cualquier tipo de software colaborativo en Web con muy poca carga de trabajo para el servidor.

Finalmente, en el artículo se ha realizado una comparación empírica entre la aplicación basada en la arquitectura que presentamos y una aplicación desarrollada utilizando la arquitectura tradicional de aplicaciones web. En esta comparación se aprecian claramente las ventajas de nuestra arquitectura en cuanto a necesidades de ancho de banda y capacidad de procesamiento del servidor web.

Como trabajo futuro podemos mencionar la mejora del Trivial.gz para permitir partidas con más jugadores, o desarrollar una versión del juego totalmente configurable por el usuario en cuanto a temas disponibles y preguntas en cada tema. Otra posible línea de trabajo es el estudio de las necesidades de adaptación de la arquitectura para otros tipos de aplicaciones colaborativas, como por ejemplo enseñanza a distancia o trabajo colaborativo.

REFERENCIAS

[1] Dwr. Accedido en Mayo de 2006 en `http://getahead.ltd.uk/dwr`.

[2] Galizanet. Accedido en Mayo de 2006 en `http://www.xuventudegaliza.net`.

[3] Laboratorio de bases de datos. universidade da coruña. Accedido en Mayo de 2006 en `http://rosalia.dc.fi.udc.es/lbd`.

[4] Servicios y herramientas de google. Accedido en Mayo de 2006 en `http://www.google.es/intl/es/options/`.

[5] Sitio web de google. Accedido en Mayo de 2006 en `http://www.google.com`.

[6] Sitio web de la as-pg. Accedido en Mayo de 2006 en `http://www.as-pg.com/`.

[7] Xml en w3c. Accedido en Mayo de 2006 en `http://www.w3.org/XML/`.