Babelium Project: un entorno colaborativo y abierto para la práctica de la expresión oral de segundas lenguas

Juan A. Pereira[1], Inko Perurena[1], Silvia Sanz[1], Julián Gutiérrez[1]
Raúl Montero[2],

[1] Universidad del País Vasco / Euskal Herriko Unibertsitatea
{juanan.pereira, inko.perurena, silvia.sanz, julian.gutierrez}@ehu.es
[2] Universidad Pública de Navarra
raul.montero@upna.es

Abstract. La evaluación de la competencia oral en los alumnos de segundas lenguas es tremendamente costosa en tiempo para los profesores de la asignatura. Por otro lado, los alumnos que se enfrentan a pruebas orales, en muchas ocasiones no han podido disponer del suficiente entrenamiento. Un problema añadido en la evaluación de la competencia oral consiste en que los alumnos no disponen de herramientas preparadas que permitan la coevaluación de ejercicios. El trabajo presentado describe la implementación de una herramienta *RIA* siguiendo las necesidades de profesores de Didáctica de la Lengua. Se muestran los principales problemas técnicos que se han ido superando en la implementación de los módulos de Babelium, una aplicación *RIA* que atendiendo a las necesidades indicadas, permite la práctica de la expresión oral de segundas lenguas, mediante el uso de vídeos colaborativos. Finalmente se describe una experiencia práctica en la asignatura English II y III del Grado en Educación Infantil y Primaria.

Keywords: aprendizaje de segundas lenguas; práctica oral; web2.0, multimedia; opensource.

1 Introducción

Evaluar el conocimiento oral a un grupo de alumnos requiere una prueba individualizada y en serie a cada uno de ellos o, si esa prueba se asemeja a la realizada en las escuelas de idiomas, podría ser en parejas (un alumno comenta o discute los puntos elicitados por el otro). Para un grupo de 90 alumnos se necesitarían unas 30 horas (asumiendo 20 minutos por alumno), a lo que habría que añadirle el trabajo previo de preparación y el posterior de evaluación.

Por otro lado, los alumnos que se enfrentan a dicha prueba, en muchas ocasiones no han podido disponer del suficiente entrenamiento en un entorno real, es decir, no es viable que un profesor ejecute varias pruebas de evaluación de la competencia oral

dado que las mismas consumirían todo el tiempo disponible para impartir la totalidad de la asignatura (competencias relacionadas con la escritura, comprensión, lectura, etc.). Por tanto, el alumno, en muchas ocasiones y debido a la falta de preparación, se enfrenta a una situación de gran estrés, no sólo en el examen, sino en aquellas situaciones en las que debe hablar en público.

Un problema añadido en la evaluación de la competencia oral, consiste en que los alumnos no disponen de herramientas preparadas para simplificar la coevaluación de la misma. Por ejemplo, en un grupo de 90 alumnos no hay herramientas adecuadas que permitan grabar lo que cada alumno dice para que, desde casa u otro lugar, cada alumno pueda coevaluar el desempeño de los demás en tareas relacionadas con la expresión oral.

El trabajo presentado se ha dividido en dos partes: en la primera se describen las necesidades recogidas a través de las reuniones mantenidas con profesores del departamento de Didáctica de la Lengua para el diseño de la herramienta Babelium. Se muestra a continuación los principales problemas técnicos que se han ido superando en la implementación de los principales módulos de Babelium, una aplicación RIA que atendiendo a las necesidades indicadas, permite la práctica de la expresión oral de segundas lenguas, en tiempo real, mediante el uso de vídeos colaborativos. En la segunda parte se describe una experiencia práctica llevada a cabo con Babelium en el contexto de las asignaturas English II y III - equivalentes a nivel B2 del Marco Europeo- para alumnos del Grado en Educación Infantil y Primaria (90 alumnos en total).

2 Práctica y evaluación oral de segundas lenguas mediante una herramienta web abierta y colaborativa

Tras diversas reuniones con profesores del departamento de Didáctica de la Lengua de la Escuela de Magisterio de Bilbao, profesores e investigadores del departamento de Sistemas Informáticos de la Facultad de Informática (UPV/EHU) y con profesores del departamento de Didáctica de la Lengua de la Universidad Pública de Navarra, se llegó a especificar las necesidades que una herramienta online debería de cubrir para ayudar a los profesores de segundas lenguas a evaluar competencias relacionadas con la expresión oral del alumnado.

En concreto, la aplicación debía permitir :

- posibilidad de practicar la expresión oral (*speaking*): mediante distintos tipos de ejercicios. En concreto, uno de los ejercicios debería de permitir el doblaje de pequeños fragmentos de vídeo, es decir, el usuario debería de poder jugar el rol de un personaje que apareciera en una conversación de un fragmento de vídeo. El doblaje de dicho rol debería de hacerse en tiempo real, es decir, simulando las restricciones de tiempo que acontecen en la vida real. Con vistas a poder evaluar el trabajo de los estudiantes (y verificar que

el orador es realmente quien dice ser), el doblaje debería de poderse hacer con webcam (además de con micrófono)

- la autoevaluación, coevaluación y heteroevaluación de ejercicios de *speaking*. Es decir, permitir que el propio alumno oiga y evalúe sus grabaciones, permitir que otros alumnos, de forma colaborativa, evalúen los trabajos de los demás y permitir así mismo que exista una figura que acredite tener más conocimiento del idioma (generalmente el profesor) de tal forma que pueda realizar evaluaciones formativas y - sobre todo - sumativas
- la práctica de ejercicios de escucha activa (*listening*) y de transcripción de conversaciones en segundas lenguas. En concreto se requería que la aplicación a desarrollar (Babelium) dispusiera de un módulo de edición de subtítulos que los usuarios pudiera usar para subtitular las nuevas secuencias de vídeo que otros usuarios subieran al sistema. Estos subtítulos permitirían que el módulo de práctica oral permitiera saber qué personajes formaban parte de una vídeo-conversación y, por tanto, permitiera el doblaje de uno de ellos
- integración con sistemas de elearning actualmente en funcionamiento. Es muy probable que una entidad dedicada a la formación en segundas lenguas disponga ya de un sistema de elearning implantado. El objetivo a largo plazo de Babelium sería poder ser integrado en forma de módulo o plug-in en los principales sistemas de e-learning (expresamente se requiere la integración con Moodle)
- disponer de un sistema de créditos (o puntos) que motiven la participación. Cuando un usuario se da de alta en Babelium obtendrá un número de créditos que irá en decremento cuando solicite algo al sistema, por ejemplo, cuando solicite ser evaluado por otros usuarios. Estos créditos aumentarán cuando el usuario participe aportando algo al sistema (evaluando las grabaciones de otros usuarios, proponiendo nuevos vídeo ejercicios, subtitulando secuencias de vídeo...)
- publicar el código fuente de la aplicación bajo una licencia abierta. El objetivo es doble: que el proyecto Babelium sea mejorado de forma continua por aportaciones de la comunidad de usuarios y por otro lado, garantizar su futuro independientemente de que el equipo inicial de desarrolladores siga o no con el desarrollo a largo plazo

Salvo el requerimiento de integración con otros sistemas (objetivo a largo plazo que aún no ha sido totalmente desarrollado) el resto de objetivos planteados se han visto cumplidos. En los siguientes apartados se detallan los aspectos más problemáticos con los que los desarrolladores se han encontrado en la implementación de los requerimientos especificados, se muestra la arquitectura final y se plantean algunas líneas de mejora.

3 Principales problemas de implementación abordados

El desarrollo de Babelium ha supuesto un gran reto que ha movilizado a un equipo de 10 desarrolladores a lo largo de todo un año. Generar una aplicación usable, abierta, colaborativa y accesible vía web, con funcionalidades de grabación de vídeo y audio en tiempo real, sincronización de subtítulos y roles, gestión de créditos, etc., no ha sido una tarea fácil entre otras razones porque no se disponía de ninguna aplicación base de código abierto sobre la que construir. El desarrollo ha generado una base de código en ActionScript (17.600 líneas) para la lógica de negocio y control ejecutada en cliente, MXML (18.500 líneas) para la parte visual de la herramienta y PHP (7.200 líneas) para la lógica de negocio en servidor. En total 43.000 líneas de código que el proyecto ha publicado bajo la licencia abierta GPLv3 [1].

Escoger la tecnología a utilizar para el desarrollo fue uno de los primeros escollos. Una vez teniendo claro este apartado, se decidió utilizar un *framework* que permitiera organizar y gestionar de forma razonable la gran cantidad de módulos y código a implementar.

Finalmente, y a pesar de que en la etapa de diseño no surgieron problemas aparentes, un buen número de funcionalidades han requerido una o varias sesiones de refactorización de código, y en algún caso (gestión de subtítulos), la reescritura completa. En los siguientes apartados se describirán los principales retos y las soluciones aportadas a cada uno de ellos.

3.1 Selección de tecnologías

En el momento de comenzar el proyecto Babelium (2009), las principales tecnologías disponibles para el desarrollo de aplicaciones RIA eran Adobe Flex (permite la programación orientada a objetos usando ActionScript y el desarrollo de interfaces gráficas mediante el lenguaje declarativo MXML, exportado el resultado a formato Flash), Microsoft Silverlight y SUN JavaFX (actualmente producto de Oracle).

Aunque las características de estas tecnologías han ido evolucionando hasta la actualidad (Mayo 2011), las razones principales para decantarnos por Adobe Flex frente a las otras alternativas siguen manteniéndose y pueden verse resumidas en la Tabla 1 (Adobe Flex es una tecnología madura, multisistema, soporta RTMP – *Real Time Multimedia Protocol* - y el *framework* está publicado bajo licencia abierta – Mozilla Public License).

1 http://code.google.com/p/babeliumproject

Tabla 1. Comparativa de las principales tecnologías para el desarrollo RIA

	Oracle JavaFX	Microsoft Silverlight	Adobe Flex
Madurez	Abril 2010	Abril 2007	Marzo 2004
Plugin necesario	Java Runtime Environment	Plugin Silverlight	Plugin Flash
Portabilidad	Multisistema	De forma nativa sólo Windows y MacOS	Multisistema
Herramientas RAD	Plugin Netbeans	Plugin Visual Studio	Plugin Eclipse
Permite compilación para móviles	Para móviles con JavaFX Mobile Runtime	Para móviles con Windows Mobile 7	Para cualquier móvil con plugin Flash
Licencia	Runtime privativo	Privativa	Licencia MPL. Player privativo
Soporte RTMP[2]	No	A través de librería externa (FluorineFX)	Sí, de forma nativa

Es necesario hacer una anotación con respecto a la columna “Plugin necesario”. No es sólo una cuestión de gusto o favoritismo por una tecnología u otra, sino que, desde un punto de vista empresarial y de acceso a la mayor cuota de mercado posible, es conveniente evaluar el grado de penetración de cada tecnología. En la Figura 1 se comparan, entre otras alternativas, el grado de penetración del plugin de Flash frente al plugin Java (soporte JRE en el navegador).

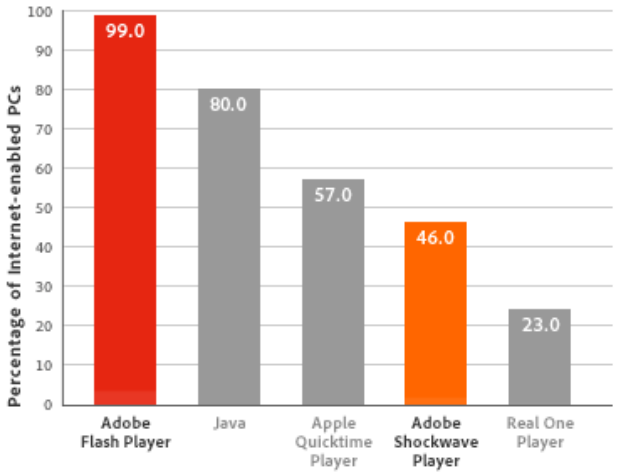

Fig. 1: Encuesta Millward Brown, Diciembre 2010.
Fuente: http://www.adobe.com/products/player_census/flashplayer/

2 Real Time Messaging Protocol, pemite streaming de audio, video y datos

Por otra parte, según las estadísticas de la web RiaStats.com a 20/04/2011, a partir de una muestra de 9 millones de conexiones a 118 webs (con las que RiaStats tiene acuerdos) durante los últimos 30 días, sólo un 2,96% de los usuarios no disponen del plugin de Flash instalado – o dicho de otra forma, más del 97% de los usuarios lo tienen - mientras que en el caso de Microsoft Silverlight la cifra de usuarios detectados sin el plugin necesario se eleva al 29,5% y al 32,5% en el caso de usuarios de JavaFX (esta última tecnología sólo necesita el uso de la máquina virtual Java, no un plugin específico para JavaFX, pero aún así muchos usuarios no tienen activado este soporte).

Finalmente, las estadísticas obtenidas de la web StatOwl[3] (datos obtenidos de servidores web estadounidenses) confirman los porcentajes salvo pequeñas diferencias en el grado de penetración de JavaFX y Silverlight (poniendo por delante a la primera tecnología), pero siempre detrás de Flash.

La conclusión es que, siendo el criterio base el grado de penetración en el mercado, Adobe Flash es desde hace tiempo la opción ideal como tecnología RIA sobre la que desarrollar. Por supuesto no es éste el único criterio a tener en cuenta para una comparativa justa, pero tal y como refleja la Tabla 1, aún sopesando distintos factores, los desarrollos en Flex/Flash son la opción más sólida de cara a obtener un desarrollo rápido y una buena cuota de mercado. En [1] puede verse una comparativa más extensa.

3.1.1 La opción HTML5

HTML5 (*HyperText Markup Language*, versión 5) es la última revisión del estándar HTML. Entre las principales novedades destacan el soporte de las etiquetas <video> , <canvas> (2D y 3D) y <audio> que permiten trabajar con elementos multimedia directamente en HTML5, sin necesidad de usar plugins externos. A primera vista esto podría hacer viable una versión de Babelium usando en exclusiva HTML5, con las consiguientes ventajas de evitar la instalación de plugins al usuario, dar soporte a múltiples navegadores – los principales navegadores trabajan en ofrecer soporte HTML5 – así como permitir el uso de Babelium en plataformas que tienen vetado el uso de Flash – como es el caso de los móviles iPhone y las tabletas iPad de la empresa Apple, que por distintas razones prohibió el uso de Flash en sus dispositivos [2].

Sin embargo, HTML5 aún tiene que mejorar ciertas carencias y desarrollarse mucho más en determinados aspectos para poder ser usado en aplicaciones RIA como Babelium. En concreto, las carencias más importantes de HTML5 en la actualidad – y que afectan todas ellas al desarrollo de una versión HTML5 pura, libre de Flash- son las siguientes:

3 http://www.statowl.com/custom_ria_market_penetration.php

Soporte de cámara web en fase de desarrollo
No es posible acceder desde el navegador al uso de la cámara web. Esta funcionalidad, aunque no es fundamental en Babelium, permite al evaluador asegurar - con una alta probabilidad de acierto - que el alumno que habla es realmente el mismo que se ve en pantalla (opción muy útil para el caso de uso en el que se realizan exámenes de expresión oral). En el futuro se planea añadir soporte de *webcam* a HTML5 y otros dispositivos a través del API conocido como Device API, y en concreto a través del subconjunto System Information API, actualmente un borrador del W3C [3].

No es posible la reproducción de audio en formato mp3
El estándar sólo admite .ogg y .wav. Babelium trabaja con el formato FLV (Flash Video), independientemente de que haya vídeo y audio o sólo audio. El audio está codificado en formato MP3. Aunque en el futuro hubiera una migración completa a HTML5, de forma transitoria convendría poder usar los formatos ya disponibles en Babelium.

Incompatibilidad entre navegadores
Babelium tiene como objetivo llegar al mayor número de navegadores y usuarios posible, al menor coste de desarrollo razonable. No es viable hacer una versión por cada navegador que no soporte el estándar HTML5 al completo.

La tabla comparativa más completa sobre el grado de soporte de las etiquetas <audio> y <video> en HTML5 entre los principales motores de los navegadores más usados que se ha encontrado está disponible en la versión inglesa de la Wikipedia[4].

A pesar de los inconvenientes de HTML5 detallados en los párrafos anteriores, el proyecto Babelium se está portando actualmente a una versión lo más compatible posible con HTML5. Las razones de esta migración son fundamentalmente tres: rapidez de desarrollo, ampliación de mercado y compatibilidad con los principales indexadores (SEO, Search Engine Optimization).

3.2 Implementación del módulo de subtítulos

En un primer momento, para la gestión de subtítulos se pensó en utilizar la funcionalidad de cuepoints que ofrece el formato de vídeo FLV.

Un cuepoint en FLV puede verse como un punto singular del vídeo, es decir, un punto concreto del vídeo que queremos marcar. Por ejemplo, podríamos pensar en usar un cuepoint para indicar que a partir de cierto segundo se quiere visualizar un anuncio publicitario o banner en pantalla. Los cuepoints se incrustan dentro del

4 http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(HTML5_Media) tabla comparativa de soporte de etiquetas multimedia HTML5 en los principales motores (Gecko/Firefox, Trident/IE, WebKit/Chrome)

fichero binario FLV mediante herramientas ad-hoc para el tratamiento de vídeo (en concreto mediante herramientas que permiten la inserción de metadatos en un FLV).

Inicialmente se hizo uso de los cuepoints de FLV para marcar los subtítulos que éste contenía. Se incluía el tiempo de inicio, texto del subtítulo y nombre del rol implicado (qué personaje de los que entran en juego en el vídeo está hablando en ese preciso instante).

La idea es que el servidor multimedia (como se discute en el apartado 3.5, se usará el servidor Red5[5]) emite por un canal de streaming el vídeo propiamente dicho y - tras analizar su contenido – el conjunto de *cuepoints*. Bastaría por tanto con programar un gestor que capture este flujo de cuepoints y en función de los mismos visualice por pantalla el contenido de cada uno de ellos (cuya carga es el subtítulo y rol en cuestión). En ActionScript el código sería similar al del listado 1.

Listado 1. Código para abrir un stream de datos y conectarse al evento onMetadata

```
var ns:NetStream = new NetStream(netConnection);

ns.client = this; // el objeto actual -this- contiene

       // un listener que trata el evento onMetadata
```

Sin embargo, surgieron problemas que hicieron inviable esta opción:

a) como se ha dicho, es necesario incrustar en el fichero de vídeo FLV cada uno de los cuepoints cada vez que éstos cambien. Esto requiere recodificar el fichero FLV cada vez que haya un pequeño cambio (lo cual carga el servidor de forma innecesaria)
b) aunque existen herramientas para incrustar cuepoints en un FLV, es necesario disponer de una que permita trabajar desde la línea de comandos (con el objetivo de que esta labor se automatice en el servidor, que en el caso de Babelium es un servidor Linux). En este sentido, hay una enorme carencia de herramientas como las descritas para el sistema indicado. La más usada, flvtool2[6] no ha recibido ninguna actualización de código desde 2009.
c) los cuepoints se idearon para marcar un punto concreto del vídeo, pero no una zona o un subtítulo. En concreto, para trabajar con subtítulos, hay que indicar cuándo comienza y cuándo termina cada subtítulo (generando al menos 2 cuepoints por cada subtítulo y programando la lógica de mostrar/ocultar en la parte del cliente). No basta con ocultar un subtítulo cuando comienza el siguiente, dado que en ocasiones hay silencios en los fragmentos de vídeo que también se quieren marcar.
d) El servidor multimedia utilizado (Red5) tiene problemas documentados cuando el FLV contiene cuepoints (en concreto, en ocasiones algunos cuepoints no se generan)[7]

5 http://code.google.com/p/red5/
6 https://github.com/unnu/flvtool2
7 http://code.google.com/p/red5/issues/detail?id=18

Por las razones indicadas, se propuso otra fórmula para la gestión de subtítulos en tiempo real: usar una tabla en una base de datos relacional (MySQL en concreto). Mediante esta opción no es necesario embeber en el propio fichero FLV los metadatos de los subtítulos bastando con insertar en la tabla MySQL correspondiente los datos relacionados. En el módulo de reproducción, cuando un usuario visualiza un vídeo, el servidor envía al cliente todo el conjunto de subtítulos de dicho vídeo, con tiempo inicial, final, texto del subtítulo y rol. En la parte cliente, un método ActionScript se encarga de mostrar y ocultar correctamente cada subtítulo.

3.3 Subtítulos colaborativos

Crear los subtítulos de un vídeo puede ser una ardua tarea para una sola persona. Es necesario indicar cuándo comienza a hablar cada personaje, cuándo termina cada frase, y qué es lo que dice. Los ajustes de tiempo deben ser precisos o se notará una desincronización entre lo que se oye y lo que se lee. Para aliviar esta tarea de creación de subtítulos se ha diseñado un módulo que permite guardar en cualquier momento el trabajo de subtitulado realizado hasta ese mismo instante, de tal forma que otro usuario (o el mismo) pueda retomar el trabajo de subtitulado allá donde se dejó. Por otra parte, esta funcionalidad permite ver, a lo largo del tiempo, cómo han ido evolucionando los subtítulos asociados a un vídeo, conociendo además el nombre del responsable de cada cambio y la fecha en la que lo hizo, a través de un histórico de versiones de subtítulos, que pueden recorrerse por medio de una simple lista desplegable (Figura 2).

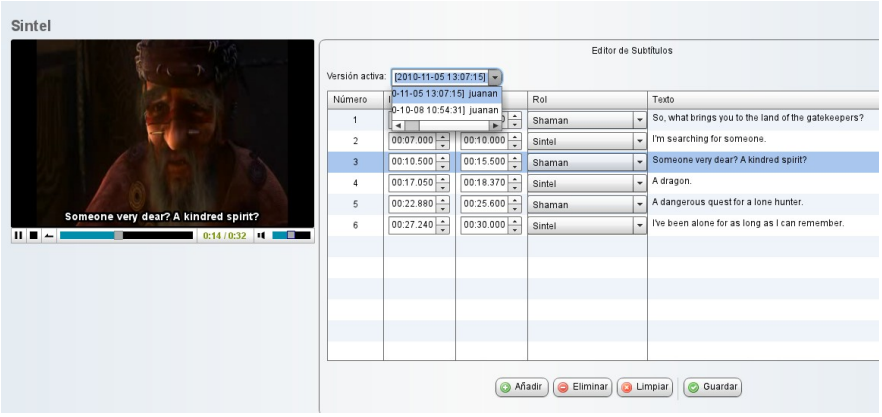

Fig. 2: Subtítulos colaborativos. Existe un control de versiones sobre los subtítulos

La funcionalidad descrita también es útil en el trabajo de recuperación ante sabotajes o despistes. Supongamos que un usuario con aviesas intenciones edita incorrectamente un conjunto de subtítulos para un mismo vídeo. Otro usuario, al ver el desajuste, podría seleccionar del desplegable con el histórico de versiones aquella que sea más reciente y correcta, fijándola como la versión actual de subtítulos. Por

otra parte, una actualización hacia atrás (un cambio de versiones hacia atrás) generará automáticamente un email de revisión a los administradores del sistema, para que en aquel caso en el que se aprecie algún tipo de sabotaje (en la versión actual o anterior de los subtítulos), se actúe en consecuencia, mediante la sustracción de créditos al usuario en cuestión.

3.4 Modularización de Babelium

Internamente, a nivel de código, Babelium se subdivide en distintos módulos atendiendo a la funcionalidad implementada por cada uno de ellos y las recomendaciones del *framework Cairngorm*[8] - que sugiere dividir el proyecto en carpetas y subcarpetas basándose en el patrón arquitectural MVC –*Model View Controller* [4]- , tal y como muestra la Figura 3.

Esta clasificación por carpetas permite una rápida aproximación al diseño modular real que buscamos para permitir integrar Babelium en aplicaciones externas, a través de un API de extensión.

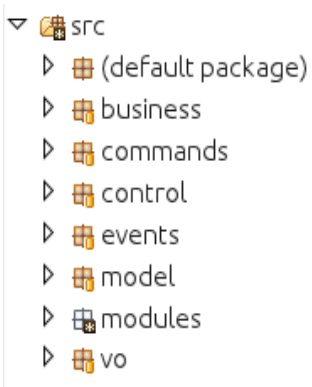

Fig. 3: Organización del código según la especificación del framework Cairngorm

Pueden distinguirse las carpetas principales *business (*contiene referencias a los objetos remotos – scripts PHP – y una colección de clases delegadas encargadas de llamar a cada objeto remoto, delegando en las *clases comando* el tratamiento de las respuestas), *commands (*recogen las respuestas de las llamadas a los objetos remotos y actualizan el modelo de datos) , *control (*la capa de Control en el patrón MVC, aquí se especifica qué comando responderá a cada tipo de evento*), events*, *model* (la capa Modelo), *modules* (la Vista, contiene la definición de las interfaces gráficas en formato MXML) y *vo (ValueObjects*, clases con atributos - sin métodos – que sirven para encapsular y transmitir datos relacionados con objetos). Las carpetas *commands* y *modules* contienen, a su vez, una misma subclasificación, donde cada funcionalidad principal de Babelium genera una subcarpeta: *autoevaluation*, *configuration*,

8 http://opensource.adobe.com/wiki/display/cairngorm/Cairngorm

evaluation, exercises, home, main, search, subtitles, userManagement, videoPlayer y *videoUpload.*

3.5 Streaming y grabación de vídeo-respuestas sincronizadas

Como se menciona en los apartados descritos hasta el momento, para la implementación de Babelium se ha hecho uso, entre otras, de las tecnologías Flex y el *framework* Cairngorm (en el lado del cliente). Pero hasta el momento no se ha descrito qué tecnologías se han usado en el lado del servidor para permitir la grabación de vídeo-respuestas. Recordemos que el usuario, una vez visualizado un vídeo, podrá tomar el rol de alguno de los personajes que en él aparecen y doblarlo en tiempo real (tal y como se ve en la Figura 4). Esto requiere que el servidor sea capaz de recibir secuencias de audio y vídeo sincronizados de tal forma que cuando otro usuario quiera evaluarlo, el mismo servidor sea capaz de retransmitir el vídeo original junto con la vídeo-respuesta sincronizada en el tiempo.

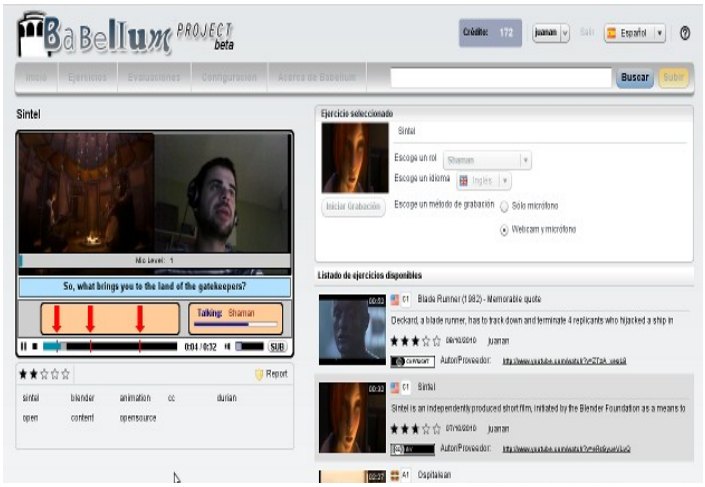

Fig. 4. Práctica oral. Módulo de grabación

Fundamentalmente existen tres tipos de servidor multimedia maduros capaz de ejecutar las funcionalidades descritas en el párrafo anterior: Flash Media Server, Wowza y Red5. De todos ellos, el único servidor de código abierto y gratuito es Red5. Debido a los requerimientos elaborados en un apartado anterior, Babelium optó por tanto por ser implementado bajo Red5.

4 Arquitectura de Babelium

Se han descrito algunas necesidades tecnológicas tanto en el cliente (Flex, Cairngorm) como en el servidor (Red5). La arquitectura de Babelium se completa con la capa de persistencia, fundamentada en el uso de la base de datos de código abierto MySQL.

No obstante, ha de tenerse en cuenta que un cliente Flash no puede comunicarse directamente con un servidor MySQL. Para hacerlo se ha empleado un intermediario que convierte los mensajes que llegan del cliente Flash (en formato AMF, Action Message Format) al lenguaje de scripting PHP. Este conversor se llama AMFPHP y se instala en el servidor HTTP Apache, quedando a la espera de la recepción de mensajes. Los scripts PHP se encargan de hacer solicitudes a la base de datos. Cuando los scripts devuelven sus resultados, AMFPHP los captura y los convierte al formato AMF para posteriormente enviarlos de vuelta al cliente, de forma asíncrona, en un formato que éste entiende. Toda la arquitectura explicada hasta el momento puede verse de forma gráfica en la Figura 5.

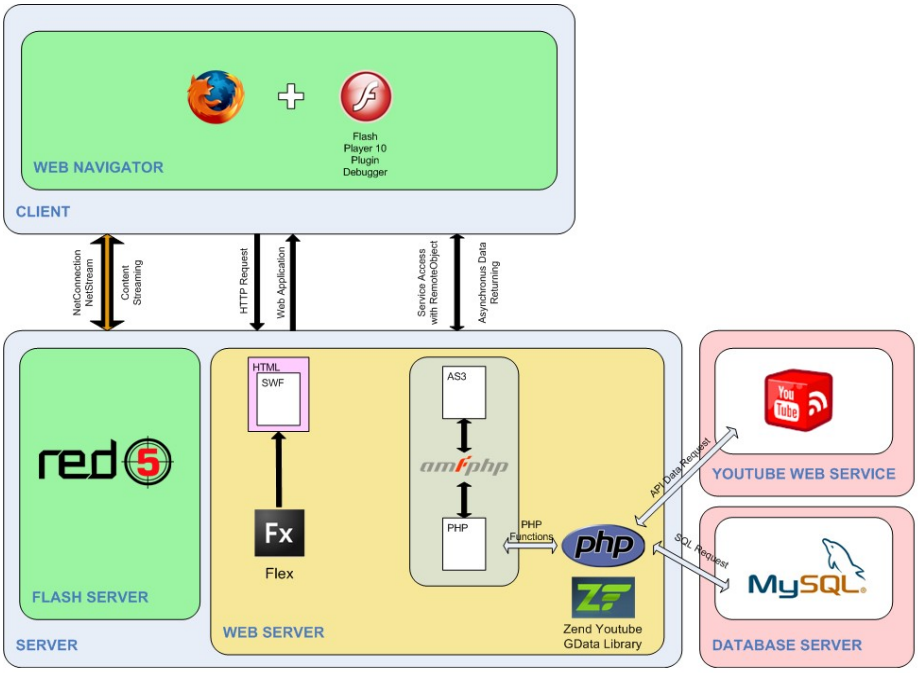

Fig. 5: Arquitectura del proyecto Babelium

5 Una experiencia práctica en el aula de inglés

La aplicación RIA que presentamos dispone de una web demostrativa en la dirección http://babeliumproject.com. Sin embargo, para probar las bondades de la herramienta, se ha generado una nueva instancia particular de Babelium en la URL http://46.51.178.168 para dar servicio a las asignaturas English II y English III del Grado en Educación Infantil y Primaria de la Universidad Pública de Navarra.

Decimos instancia particular debido a que incluye algunas características solicitadas ad-hoc por el profesor de esta asignatura. En concreto se requirieron los siguientes cambios con respecto a la versión original: sólo los profesores – en nuestro caso, sólo el profesor – pueden evaluar los trabajos de los alumnos y éstos no pueden verse entre sí. Las evaluaciones han de poderse realizar en una escala de 0 a 10, con un gránulo de 0,5 puntos. Finalmente los criterios de evaluación de Babelium (fluidez, pronunciación, espontaneidad, ritmo, evaluación general) han de ser todos optativos (en la versión matriz de Babelium son todos obligatorios).

Respecto a las asignaturas de nuestro experimento, la asignatura English II es cursada por un total de 98 alumnos, de los cuales 30 usan Babelium – grupo experimental. El nivel de inglés exigible es el equivalente al B2.1 del Marco Europeo. Por otro lado, la asignatura English III tiene 240 alumnos, de los cuales 60 alumnos usan Babelium. El nivel de inglés exigible es el B2.2.

La labor del grupo de English II fue responder oralmente – usando obligatoriamente micrófono y webcam - a 2 ejercicios para la práctica de la expresión oral. El objetivo del primero de ellos (*"reading aloud"*) era evaluar la calidad fonética empleada por los alumnos al pronunciar un conjunto de palabras sueltas (44 palabras, donde el alumno veía cada palabra escrita durante 3 o 4 segundos, en los que debía practicar la pronunciación de cada fonema). El segundo ejercicio consistía en leer un pasaje de texto diseñado para la pronunciación específica de ciertos sonidos fonéticos trabajadas en clase previamente. Igualmente, el alumno visualizaba cada párrafo del texto durante 3 o 4 segundos, franja de tiempo destinada a su vez para leer en voz alta cada uno de esos párrafos.

El experimento realizado buscaba por un lado encontrar variables predictoras de la puntuación que cada alumno tendría en la evaluación final de la expresión oral y por otro, ayudar a los alumnos a practicar ejercicios de *speaking*. El estudio de variables predictoras aún no ha terminado (cuando termine el cuatrimestre dispondremos de esa información) pero sí que se dispone de información al respecto de las horas de práctica oral que han invertido los alumnos. En concreto, la Figura 6, muestra un histograma de tiempos invertidos en labores de speaking a mitad del experimento – 19 alumnos habían terminado los ejercicios. La media es de 141 minutos en 1 semana.

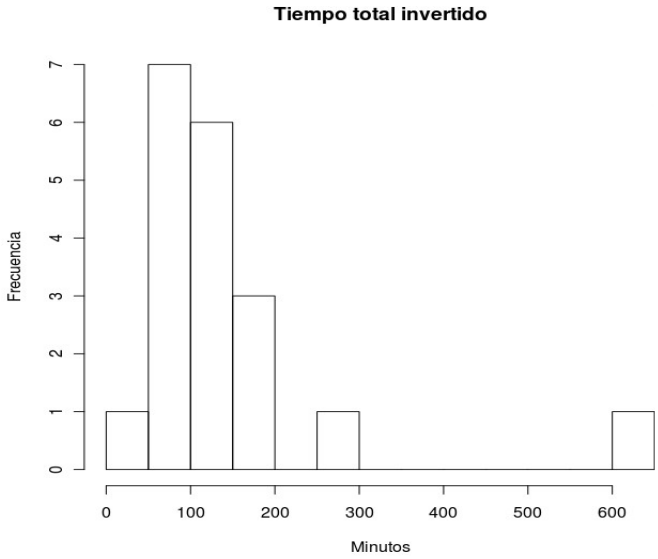

Fig. 6: Tiempo invertido en speaking en un intervalo de 1 semana

Al comienzo del curso los alumnos del grupo II y III cumplimentaron una encuesta en la que se les inquiría acerca del número de horas semanales que dedicaban en general a la práctica de la expresión oral del inglés. La respuesta más repetida fue 0 horas, es decir nada o casi nada. El uso de Babelium en clase, ha incrementado el número de horas dedicadas a la práctica oral de inglés de forma ostensible.

En el futuro, con la introducción del módulo de evaluación colaborativa de Babelium como parte de la asignatura de inglés II y III, se pretende que los alumnos practiquen entre ellos (mediante la creación, resolución y evaluación de ejercicios de "speaking") aún más horas y de forma más constante la expresión oral de esta segunda lengua, pretendiendo de esta forma conseguir mejorar (en media) las evaluaciones globales recibidas en este aspecto.

Referencias

1. Granbäck, Carl-David, Rich Internet Applications: A Comparison Between Adobe Flex, JavaFX and Microsoft Silverlight (2009)
2. Jobs, S. http://www.apple.com/hotnews/thoughts-on-flash/ Carta abierta del CEO de Apple Steve Jobs, sobre las razones de Apple para vetar el uso de Flash en los dispositivos iPhone y iPad (2010)
3. http://dev.w3.org/2009/dap/system-info/ Especificación W3C que define un API para ofrecer a las aplicaciones web acceso a dispositivos del sistema en el que se ejecutan [2009]
4. Fowler, M. http://www.martinfowler.com/eaaDev/uiArchs.html, Model View Controller [2006]