DESARROLLO DE UN SISTEMA DE REGISTRO DE ASISTENCIAS PARA MEJORAR LAS PRÁCTICAS DE TRABAJO DEL CAADI YURIRIA

José Baltazar Ramírez Rodríguez
Universidad de Guanajuato
jb.ramirezrodriguez@ugto.mx

Ignacio Contreras Contreras
Universidad de Guanajuato
i.contrerascontreras@ugto.mx

María Susana Ávila García
Universidad de Guanajuato
susana.avila@ugto.mx

María Isabel Vázquez de la Rosa
Universidad de Guanajuato
mi.vazquez@ugto.mx

Resumen

El Centro de Auto Aprendizaje de Idiomas del Departamento de Estudios Multidisciplinarios de la Universidad de Guanajuato en la Sede de Yuriria es un espacio dirigido a los estudiantes con el fin de promover su autonomía en el aprendizaje del idioma inglés. El registro de asistencia en este centro es necesario para ser reportado a los profesores que incluyen en sus criterios de evaluación la asistencia al mismo. En este trabajo se reporta el desarrollo, puesta en marcha, y mantenimiento de un Sistema de Registro de Asistencia desarrollado bajo algunos de los principios de la metodología de administración de proyectos Scrum y usando herramientas que permiten un desarrollo ágil como Meteor.js y el manejador de base de datos NoSQL Mongo DB. Se discute el impacto que esta

herramienta ha tenido en las prácticas de trabajo de la administradora del centro, y de las oportunidades de crecimiento y trabajo futuro del mismo.

Palabras Claves: Desarrollo de Software, Prácticas de Trabajo, Meteor.js, MongoDB.

Abstract

The Self-Access Language Learning centre of the Department of Multidisciplinary Studies of the University of Guanajuato in Yuriria aims to provide the tools for students to become autonomous learners in English learning. Student attendance reports need to be generated for teachers that include attendance to this centre as part of their evaluation criteria. In this work we report the development, installation and support for an Attendance Management System developed based on the SCRUM methodology and using tools that support agile development such as Meteor.js and the NoSQL database manager system MongoDB. We discuss about the impact that this tool has had in the working practices of the manager and the opportunities and future work for this project.

Keywords: *Software development, Work practices, Meteor.js, MongoDB.*

1. Introducción

El Centro de Auto Aprendizaje de Idiomas (CAADI) del Departamento de Estudios Multidisciplinarios de la Universidad de Guanajuato en la Sede de Yuriria (DEM Yuriria) es un espacio dirigido a los alumnos con el fin de promover su autonomía en el aprendizaje del idioma inglés. El uso y administración de estos centros de autoacceso tienen asociados varios retos que han sido objeto de estudio por la comunidad de investigación (Miller, 2013) (Miller, 2011). Entre ellos se encuentra la manera en que los estudiantes aprecian la efectividad de estos centros en el proceso de aprendizaje del idioma (Koyalan, 2009). Las actividades de los administradores de estos centros incluyen: promover el aprendizaje autónomo, desarrollar material para el centro, y guiar a los estudiantes en el material que pueden usar en el centro para mejorar aspectos de interés, entre otras actividades. El CAADI en la sede de Yuriria atiende a un promedio de 250

alumnos al semestre, y algunos profesores de la asignatura de inglés promueven la asistencia al mismo incluyendo en sus criterios de evaluación la asistencia al centro la cual varía en el número de horas a la semana dependiendo del nivel. Para responder a esta iniciativa, la administradora del centro reporta a los profesores la asistencia de los estudiantes por examen parcial, por semestre, o bien por demanda específica de los profesores. El acceso al CAADI en Yuriria es totalmente independiente del horario de clase, por lo que los alumnos asisten a este centro cuando tienen tiempo disponible, por lo que su asistencia no está regida por un horario pre-definido. Inicialmente el registro de asistencia al CAADI se realizaba usando una bitácora escrita, por lo que la generación de reportes era tardado, tedioso, y propenso a errores. A manera de asisitir este proceso entró en producción una propuesta para automatizar el registro de asistencias por un grupo de estudiantes, sin embargo, aunque el sistema era capaz de registrar información sobre la asistencia de los estudiantes en la base de datos MySQL de manera consistente, el sistema no desempeñaba de manera efectiva la generación de reportes. El sistema propuesto en este trabajo tuvo como objetivo el venir a cubrir esas necesidades de generación de reportes que la primera versión de software no pudo resolver. Es importante mencionar que este sistema fue el resultado de un proyecto académico, cuyo objetivo era el de formar estudiantes de la carrera de Ingeniería en Sistemas Computacionales de la Sede de Yuriria en darle solución a un problema real, con las limitantes y riesgos que esto conlleva. El sistema reportado fue diseñado, desarrollado, y puesto en marcha teniendo un impacto positivo que será discutido en las siguientes secciones.

2. Método

Para el desarrollo de software se siguió el marco de trabajo para la administración de procesos de desarrollo de software Scrum, que se encuentra dentro del marco de las metodologías ágiles que han sido de gran uso en los últimos años (T. Dingsøyr, 2012). Este marco de trabajo es normalmente referenciado como una metodología de desarrollo, la cual se rige por el manifiesto de desarrollo ágil de software (Cunningham, 2001), el cual se basa en valorar:

- Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación excesiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio sobre seguir un plan.

El equipo de desarrollo estaba conformado por seis estudiantes de la carrera de Ingeniería en Sistemas y el proyecto tenía planteada una duración de diez semanas. Dado el tiempo y los recursos disponibles para la realización de este sistema, la metodología Scrum fue la que más se adecuó a nuestras necesidades, sin embargo, dado que el desarrollo de software fue realizado como parte de un proyecto académico, las condiciones de Scrum tendrían que ser adecuadas a las necesidades del equipo. El desarrollo de este proyecto siguió un diseño centrado en el usuario en el que se cuidó que la solución desarrollada no corriera el riesgo de enfocarse sólo en la solución técnica del problema (Eason, K, 1983), sino de ajustarlo a las necesidades específicas de la administradora del centro, quien funge el rol de cliente y usuario, que de aquí en adelante será referenciada como nuestro usuario. Dentro del marco del desarrollo de software, se siguieron los siguientes pasos:

- Elicitación de requerimientos Investigación de las herramientas de desarrollo apropiadas para este desarrollo.
- Definir estrategias de implementación.
- Definición de estrategias de sostenibilidad del proyecto.
- Desarrollo del software.
- Pruebas de implementación.
- Puesta en producción.
- Mantenimiento.

Elicitación de Requerimientos

Esta etapa se realizó usando dos técnicas: la observación de prácticas de trabajo y la entrevista semiestructurada. La observación de prácticas de trabajo y en específico la observación del uso de sistemas heredados puede revelar

información muy importante para la el diseño y desarrollo de nuevas soluciones (Colombi, 2013). Por esto se decidió tomar el uso del sistema heredado como fuente de información para definir requerimientos. La entrevista semi-estructurada a nuestro usuario permitió afinar los detalles e identificar la prioridad de las características del sistema para definir el conjunto de características a implementar.

Se redactó un documento de requerimientos en los que se listaban las características del sistema deseado. Entre estas características destacan:

- Alta, Baja, Consulta, y Edición de Estudiantes.
- Alta, Baja, Consulta, y Edición de Profesores.
- Alta, Baja, Consulta, y Edición de Grupos por un Periodo Escolar.
- Registro de Asistencia.
- Generación de reportes de asistencia por periodo especificado.
- Consultar lista de estudiantes activos en el CAADI.
- Consultar el número de horas que un estudiante ha asistido al CAADI.
- Definir la actividad primordial que el estudiante realiza en una visita al CAADI.
- Proporcionar Acceso remoto al Sistema.
- Proporcionar Acceso a Estudiantes.

Dadas las limitantes de tiempo, se definió como prioridad las primeras cinco características listadas, las siguientes tres características serían el resultado de la retroalimentación del usuario en los diferentes puntos de evaluación del sistema. Las últimas dos características serían implementadas una vez que el equipo destinado para la instalación del software tuviera tarjeta de red y acceso al internet, por lo que quedaron fuera del alcance de este proyecto.

Investigación de las Herramientas de Desarrollo de Software

Se realizó un análisis de las herramientas de desarrollo de software que fueran apropiadas para el proyecto considerando el tiempo disponible y la experiencia de los desarrolladores en las mismas. La aplicación a desarrollar sería una aplicación

web, para que en un futuro no sólo la administradora pueda tener acceso a la aplicación, sino que los estudiantes se puedan conectar de manera remota para acceder a recursos en línea y para checar su perfil y su nivel de avance. Dada la experiencia de varios programadores y su interés por seguir aprendiendo la herramienta, se decidió usar Meteor.js, que es un *framework* de código abierto para desarrollo de aplicaciones web, que permite un desarrollo ágil. Meteor.js está basado en JavaScript el cual fue escrito usando Node.js. Además, Meteor.js es una herramienta de software libre, por lo que no sería necesario adquirir licencias para su desarrollo y puesta en producción. Uno de los principios de Meteor.js es que se "adapta al ecosistema", y puede utilizar de manera transparente para el programador las tecnologías: Blaze (que es el *templating* original de Meteor.js), React o AngularJS. AngularJS (Google, 2016) desarrollado por Google, permite extender el vocabulario HTML proporcionando el enlace de datos en dos caminos (o *Two Way Data Binding* término usado en el idioma inglés), permitiendo actualizar los elementos de la página web de manera dinámica sin la necesidad de actualizar de manera explícita la página en el navegador. Node.js (Foundation, 2016) es un entorno de ejecución basada en JavaScript. Node.js está basado en eventos por lo que trabaja de manera asíncrona. Blaze es una poderosa librería para crear interfaces de usuario escribiendo plantillas reactivas de HTML (Blaze, 2016). React es una librería JavaScript para crear interfaces de usuario reactivas desarrollado y distribuido por Facebook Team (React, 2016). El manejador de base de datos utilizado fue MongoDB, por su adaptación a Meteor.js además de que al tiempo de definir el manejador no existían controladores para otras bases de datos. MongoDB es un manejador de base de datos orientado a documentos que se encuentra disponible de manera libre y con código abierto (Mongo, 2016). Este manejador es clasificado como NoSQL y usa documentos en formato BSON para el almacenamiento de los datos. Meteor.js utiliza un Protocolo de Datos Distribuidos (DDP por sus siglas en inglés de *Distribute Data Protocol*) el cual soporta la arquitectura de microservicios, evitando así el desarrollo de soluciones monolíticas, por lo que esto proporcionará una plataforma que permita el crecimiento flexible de este proyecto en el futuro. Dado que estas herramientas en

conjunto fueron utilizadas para desarrollar Meteor.js, este *stack* de tecnologías fue transparente para los desarrolladores quienes se enfocaron en realizar los desarrollos sólo con Meteor.js. La versión de Meteor.js usada en este desarrollo fue la 1.2.0.2.

Diseño y Arquitectura de la Solución

La figura 1 muestra el diagrama de la arquitectura del sistema propuesto, en donde se puede observar la manera en que las tecnologías mencionadas en la sección anterior se conjuntan para dar soporte a un desarrollo de software ágil. Se usó una notación UML para la representación de casos de uso, como se muestra en la figura 2.

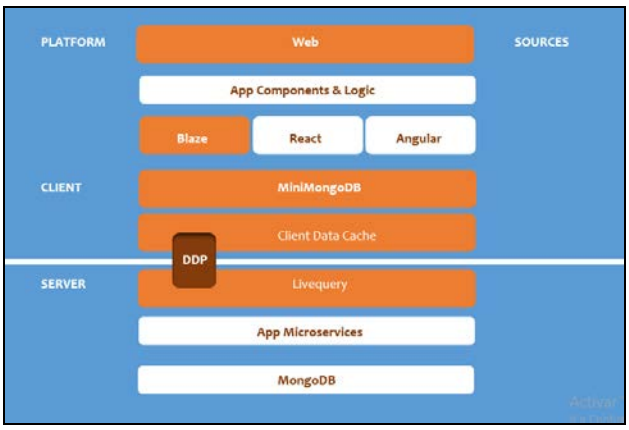

Figura 1 Arquitectura de tecnologías empleadas en el sistema propuesto.

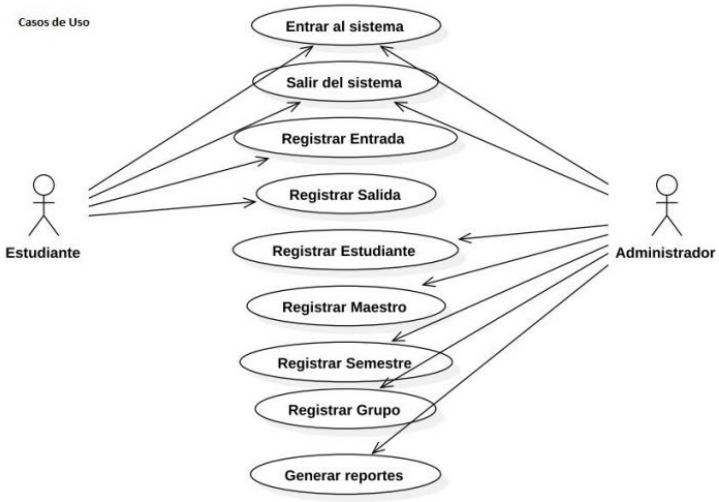

Figura 2 Casos de Uso.

Como puede notarse en esta figura 2, se definen dos tipos de usuarios: el estudiante y el administrador. Sin embargo, el sistema desarrollado no tiene contemplado en esta etapa el realizar una administración de usuarios, por lo que para que un estudiante tenga acceso a la funcionalidad de registro de entrada y salida, el administrador tendrá que dejar la estación de trabajo lista para que los estudiantes puedan utilizarlas.

De esta manera el estudiante tendrá acceso en el sistema para el registro de entrada y salida al CAADI. Por otro lado, el administrador puede registrar y administrar: Estudiantes, Maestros, Semestres y Grupos, así como generar reportes de asistencia. Los diagramas de secuencia para el registro de entrada y salida al CAADI son presentados en las figuras 3 y 4 respectivamente.

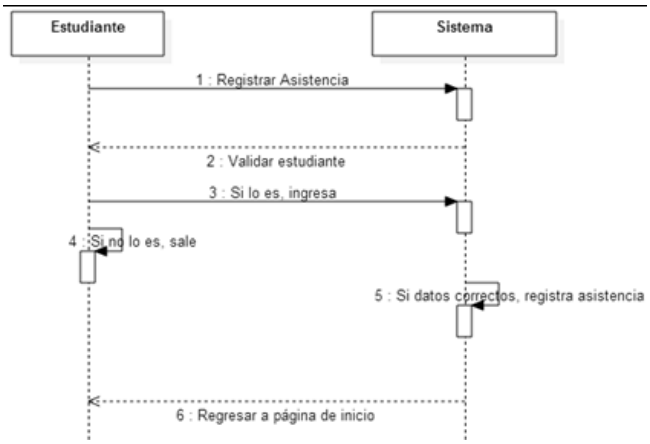

Figura 3 Diagrama de Secuencia de Registro de Entrada de Asistencia.

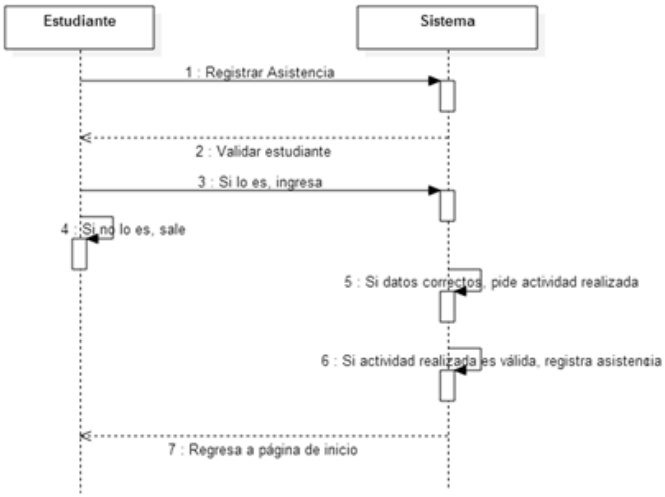

Figura 4 Diagrama de Secuencia de Registro de Salida de Asistencia.

Como puede observarse en el Registro de Entrada el sistema necesita checar si el número único de alumno (NUA) es un número válido y si es así registra su asistencia. Para el Registro de Salida, se necesita verificar si el estudiante se encuentra con el estatus de activo en el CAADI para entonces registrar la salida, además se registra la actividad primordial que ha realizado durante su visita al CAADI, la cual puede tomar los valores de: "Reading", "Writing", "Vocabulary", "Grammar" y "Listening". Esto permitirá al estudiante, administrador, y profesores, monitorear el tipo de actividades que los alumnos realizan. Considerando que el manejador de base de datos utilizado es NoSQL y aunque MongoDB es considerado un manejador sin esquemas, en la figura 5 se presentan las colecciones usadas para el almacenamiento de los datos.

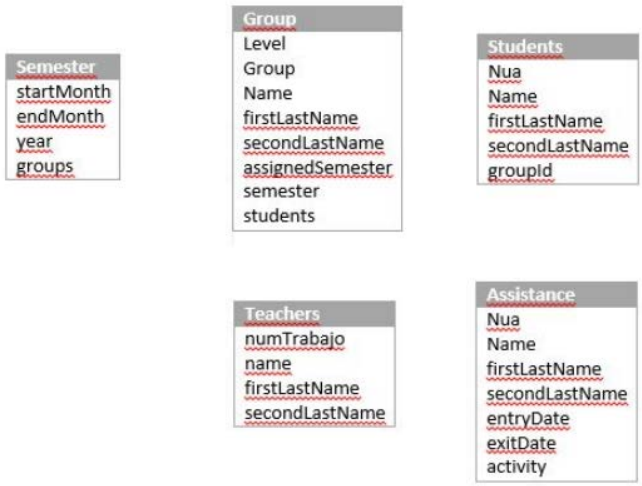

Figura 5 Colecciones definidas en MongoDB

Definición de estrategias de implementación

En esta etapa se definieron los pasos a seguir para que la implementación de la solución propuesta no impactara de manera negativa en las prácticas de trabajo. Dado que el sistema heredado que se encontraba en producción realizaba el registro correcto de los datos en la base de datos de MySQL este siguió en producción sin cambios. Entonces, se decidió que las soluciones derivadas de los *sprints* fueran puestas a prueba corriendo en paralelo con el sistema heredado, pero solo para registrar la asistencia de los estudiantes de posgrado, para poder

comparar los resultados obtenidos sin causar estrés o frustración a toda la comunidad estudiantil por tener que realizar el registro de entrada y salida en dos sistemas en cada visita al centro. Con esto se corroboraría el buen funcionamiento de la solución propuesta.

Definición de estrategias de sostenibilidad del proyecto

Una de las características de desarrollo de un proyecto académico, es el de definir los mecanismos y pasos a seguir para asegurar que la aplicación sea sostenible y siga proporcionando los servicios para la cual fue desarrollada. Aunque Meteor.js es una herramienta de desarrollo ágil, ésta no es una herramienta que sea enseñada de manera formal en un curso en el DEM Yuriria, por lo que el número de estudiantes que conocen la herramienta y pueden hacer uso de ella es reducido. Otro de los factores a considerar es que, al término del curso, los estudiantes involucrados en el proyecto tendrán una nueva carga de materias y de compromisos que no necesariamente les permitirá continuar con los desarrollos o darle mantenimiento al sistema. Además, algunos de ellos pueden perder el interés en el proyecto, por lo que el seguimiento y la sostenibilidad del proyecto pueden verse seriamente afectados. Para ello, el grupo de trabajo diseñó e impartió un pequeño taller para los estudiantes del DEM interesados en conocer esta herramienta.

Desarrollo de Software

El desarrollo de software siguió, en la medida de lo posible, los principios fundamentales de la metodología Scrum. Un profesor tomó el rol de Scrum master. Se seleccionó el conjunto de características a desarrollar, se definió tiempo de entrega y periodos de pruebas con corridas en paralelo, y se obtuvo retroalimentación constante de nuestro usuario. Las reuniones diarias de Scrum tomaban la forma de correos matutinos al Scrum master para indicar:

- Lo que se había hecho el día anterior.
- Lo que se iba a hacer el día presente.
- Así como los obstáculos o dificultades que anticipaban tener.

En el entendido de que nuestro equipo de desarrolladores no estaba dedicado al 100% al desarrollo de este proyecto, por tener que cumplir con otras clases, se podía anticipar que, en algunos de estos reportes, los desarrolladores indicaran que ese día no dedicarían tiempo al proyecto.

Pruebas de Implementación

El sistema fue puesto a prueba corriendo a la par con el sistema heredado. En estas pruebas e realizó el diagnóstico de los elementos a considerar para esta prueba:

- Equipo a usar. Inicialmente se consideró un equipo de cómputo que estaba en desuso, pero no fue posible seguir adelante ya que no contaba con características básicas suficientes. Entonces se realizó la instalación en una máquina con 2 Gb en RAM, procesador Intel i3 2.4 GHz, 500 Gb de almacenamiento y sistema operativo Windows 7.
- Ubicación del equipo en el CAADI. El sistema fue ubicado en el escritorio de la entrada al centro.
- Aspectos de seguridad, incluyendo acceso a la aplicación y a la base de datos.
- Entrada al sistema por parte del usuario desde el sistema operativo. Se definió una secuencia de comandos para que el usuario pudiera abrir la aplicación, dado instrucciones desde el encendido del equipo hasta el acceso a la aplicación.

Como se mencionó anteriormente, solo se les pidió a los estudiantes de maestría el realizar su registro de entrada y salida en ambos sistemas. Esto nos permitió probar la siguiente funcionalidad con número reducido de la población de estudiantes:

- Alta de estudiantes, maestros, grupos, y semestres.
- Registro de Entrada y salida de estudiantes.
- Generación de reportes.

Además, esto permitió a la administradora familiarizarse con la nueva herramienta de manera pausada y controlada.

Puesta en producción

El Sistema de Registro de Asistencias del CAADI entró en producción por primera vez en enero del 2016. Se realizó el alta de todos los estudiantes y los profesores, se crearon los grupos y se realizó el alta del semestre (o periodo) en cuestión. Los alumnos al ingresar al centro registran su NUA en la computadora instalada a la entrada del centro. Ingresan al CAADI y realizan la actividad deseada. Al finalizar su sesión introducen su NUA en el sistema para registrar su salida, así como la actividad que realizaron.

Mantenimiento

Dos de los seis estudiantes inicialmente involucrados en el proyecto decidieron continuar proporcionando mantenimiento al sistema y resolviendo las necesidades de información del usuario basadas en las características planteadas inicialmente para este proyecto. El sistema que fue desarrollado en un periodo de diez semanas, de tiempo parcial, tiene sin lugar a duda oportunidades de mejora y de crecimiento.

3. Resultados

El sistema actualmente se encuentra en producción en el CAADI de Yuriria. El sistema ha venido a cubrir las necesidades de información de la administradora del sistema en lo referente al registro de asistencia de los alumnos.

La figura 6 muestra las pantallas del sistema con la funcionalidad de registro de entrada con el NUA del estudiante, registro de salida, especificando NUA y la habilidad trabajada en esa sesión. La figura 7 muestra el perfil de asistencia de un estudiante específico, mostrando con gráficas el trabajo realizado en cada habilidad. El impacto que el sistema ha tenido sobre las prácticas de trabajo ha sido positivo en varios aspectos. La efectividad y eficiencia en la generación de los reportes de asistencia a los profesores ha mejorado de manera sustancial.

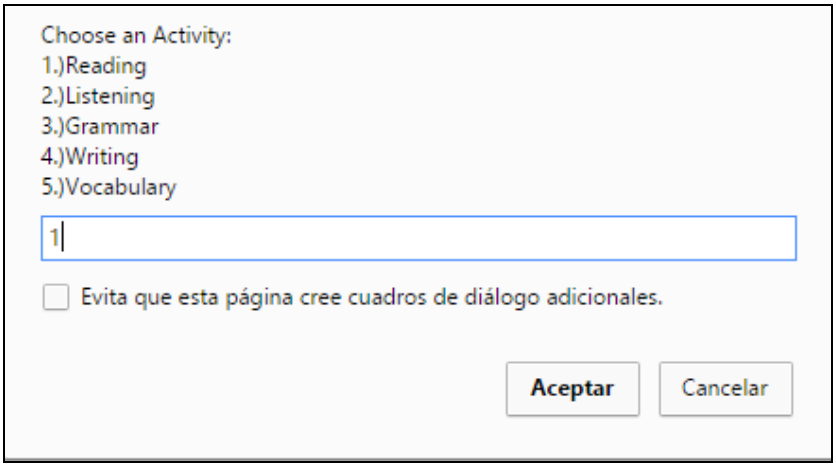

Figura 6 Pantalla.

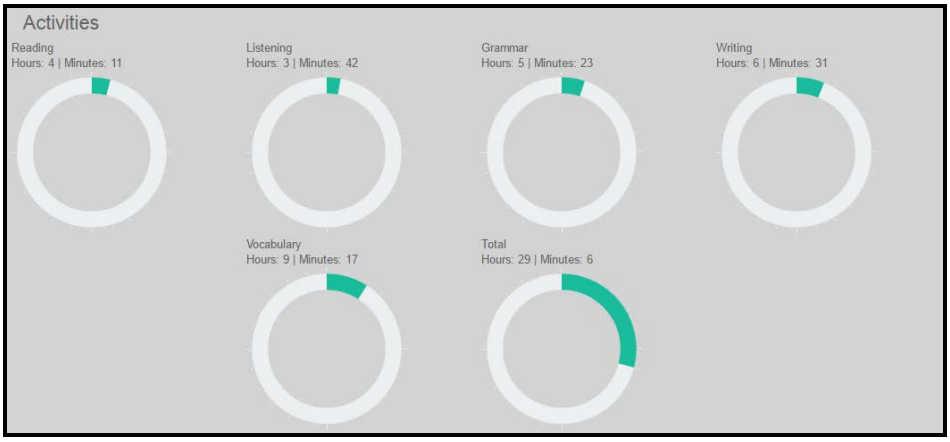

Figura 7 Perfil con las actividades del alumno.

Además, la administradora tiene la posibilidad de dar rápida respuesta a la solicitud de reportes de periodos específicos y personalizados. Se reporta también una mejora en la imagen que el CAADI Yuriria ha presentado no sólo a usuarios del DEM Yuriria, sino también, a nivel de División de Ingenierías ya que el administrador del centro está en la posibilidad de ofrecer reportes de asistencia más específicos al Director de la División. El proyecto tiene también la posibilidad de tener también un impacto a nivel del Campus Irapuato-Salamanca.

En un taller organizado en el DEM Yuriria sobre las "Prácticas de Trabajo y Necesidades de Información del CAADI" a nivel Campus permitió conocer las prácticas de trabajo de los CAADI en las Sedes de Salamanca e Irapuato. A partir de este taller, el equipo del CAADI Irapuato ha mostrado interés en adoptar la

solución propuesta en este trabajo, lo que coloca este desarrollo en la posición de extenderse y tener un impacto que trasciende las fronteras del departamento. Esto trae un nuevo panorama para la evaluación y planeación de trabajos futuros para este creciente proyecto en el que se incluye la posible integración con los sistemas de información de la Institución.

4. Discusión

En este trabajo se presenta el trabajo que inició como un proyecto de clase que ha tenido un impacto importante en las prácticas de trabajo de la administradora del CAADI y en la imagen del centro a varios niveles de manera interna y externa a la institución. Sin embargo, el desarrollo de proyectos de clase enfrenta muchos retos en el camino entre los que se encuentran:

- El tiempo de desarrollo es limitado.
- El conocimiento no uniforme de herramientas y lenguajes de programación de los miembros del equipo y la limitación en el tiempo de entrenamiento de los mismos en nuevas tecnologías.
- La falta de experiencia de los miembros del equipo en el desarrollo formal de software.
- El seguimiento al proyecto al finalizar el curso, dada la falta de interés, o de tiempo para invertir en el proyecto.

El uso del conjunto de tecnologías utilizado jugó un papel importante en el desarrollo y entrega del sistema en tiempo y forma. Y aunque la falta de experiencia en el desarrollo formal de software es listada como un reto, es esta la motivación principal de este trabajo por parte de los profesores involucrados.

Uno de los factores que podemos marcar como determinante para que el sistema siga en producción a la fecha de publicación de este artículo es sin duda el haber seguido un diseño centrado en el usuario en el que se enfocó a resolver una necesidad del usuario. Otro de los factores que ha permitido al sistema seguir vigente es el interés por las diferentes partes involucradas en el proyecto. La División de Ingenierías ha facilitado los mecanismos para evaluar la manera en

que el sistema pudiera integrarse con otros sistemas del CAADI, y actualmente se está evaluando la posibilidad de que el sistema puede interactuar con los sistemas de información de la Universidad para contar por ejemplo con catálogos actualizados de profesores y estudiantes.

Siendo este un proyecto desarrollado por estudiantes, el reto vigente para la administración del mismo es el de reclutar estudiantes interesados en el sistema y en darle seguimiento. El cambio de integrantes en el equipo de desarrollo es otro de los retos en la administración de proyectos de software y se espera que los estudiantes también sean capaces de reconocer estos retos que no solo se encuentran en este escenario, sino que es un riesgo vigente en cualquier desarrollo formal de software tanto en la academia como en la industria.

El poner en producción trabajos realizados por estudiantes y que estos ganen la experiencia de cómo se echa a andar un proyecto en un escenario real, dando soluciones a necesidades reales, y teniendo que responder en tiempo real a los problemas reportados por el usuario, es uno de los objetivos primordiales de este trabajo, abriendo la posibilidad de que al final de su carrera pueden reportar estas experiencias a las diferentes instancias donde estén interesados en integrarse como profesionistas.

Agradecimientos

Los autores de este artículo agradecen de manera especial a los compañeros Iván Núñez García, Edwin Miguel Lara Espinoza, Víctor Figueroa Chávez por su valiosa participación en el desarrollo del proyecto inicial, y a Uriel Calderón Uribe por apoyar a la administradora del CAADI en la administración de la base de datos y en la generación de reportes del sistema heredado. Al Dr. Roberto Rojas Laguna, Director de la División al tiempo de desarrollo de este proyecto, por todo el apoyo brindado para que el proyecto siga creciendo. A la Mtra. Gloria Josefina Ronzón Montiel, de la Universidad Veracruzana, por la retroalimentación proporcionada y compartir con nosotros las experiencias del Centro de Auto Acceso del Centro de Idiomas Córdoba de la Universidad Veracruzana.

5. Bibliografía y Referencias

[1] AngularJS Superheroic JavaScript MVW Framework. https://angularjs.org/.

[2] Blaze (2016). Meteor Guide. Obtenido de https://guide.meteor.com/blaze.

[3] Cunningham, W. (2001). Manifesto for Agile Software Development. http://agilemanifesto.org/.

[4] Eason, K. (1983). User centred design for information technology systems. Phys. Technol., 219-224.

[5] Foundation, N. (2016). Node.js. https://nodejs.org/en/ Google. (2016).

[6] Koyalan, A. (2009). The evaluation of a self-access centre: A useful addition to class-based teaching System, 731-740.

[7] Miller, D. G. (2011). Managing self-access language learning: Principles and practice. System, 78-89.

[8] Miller, D. G. (2013). Self-access managers: An emerging community of practice. System, 817-828.

[9] Mongo Inc. (2016). MongoDB for Giant Ideas | MongoDB. https://www.mongodb.com/.

[10] React. (2016). Meteor Guide. Obtenido de https://guide.meteor.com/react.