Universidad Católica «Nuestra Señora de la Asunción»

Proyecto Final de Ingeniería Informática

Tema: Desarrollo de un sistema de gestión de programas de estudio orientado a resultados pedagógicos.

Alumno: Luis Fernando Villalba Vera

Tutor: Ing. Sebastián Ortíz

Co-Tutor: Ing. Wilfrido Inchaustti

1. Introducción

1.1. Justificación

Existe un consenso, tanto entre los empleadores como los funcionarios de gobierno, de que en los Estados Unidos se necesita un mayor número de estudiantes en nivel técnico y universitario que posean habilidades comprobadas [1]. Esto ha llevado a una gran presión sobre las instituciones educativas para que provean más y mejor evidencia de los logros del proceso educativo [1]. Para intentar llenar este vacío se ha creado la evaluación basada en competencias, la cual se enfoca en valorar las habilidades adquiridas por un estudiante en el proceso de un programa educativo [2].

De esta manera, las aplicaciones de evaluación académica basadas en competencias han adquirido mucha importancia en los últimos años [3]. En dichas aplicaciones se busca conocer información acerca de las fortalezas y debilidades del estudiante de una manera modular, en contraposición a los métodos cuantitativos de evaluación. Dichos aprendizajes y competencias son expresados por segmentos de estudios o actividades, mediante resultados esperados medibles a nivel institucional, de programa, grado, o de curso; expresados en calificaciones asociadas a habilidades específicas y no a módulos o cursos de un programa de estudio [4].

Para poder realizar una evaluación basada en competencias de un programa educativo, el mismo programa tiene que estar diseñado con esta perspectiva educativa: los programas de estudio e incluso los mismos cursos que lo conforman, tienen que estar diseñados con una perspectiva de orientación a competencias [5]. La creación de un programa de estudio es una tarea colaborativa que involucra a un gran número de personas que proponen el diseño curricular, los cursos, los resultados esperados del programa. Cada uno de estos elementos debe pasar por un proceso iterativo de creación, revisión y aprobación que involucra a varios funcionarios, autoridades y profesores de las instituciones involucradas [6].

En el ámbito de las aplicaciones académicas, si bien existen aplicaciones que abarcan el diseño y la publicación de planes de estudio por parte de los profesores o encargados de las universidades, además de su revisión y posterior aprobación por el comité curricular, no se ha encontrado durante el proceso de relevamiento de herramientas existentes una aplicación que integre todos estos elementos en un sistema de gestión de evaluaciones basadas en competencias [7][8][9].

En este caso de estudio [10] planteamos la posibilidad de crear un sistema de gestión de programas de estudio orientado a resultados pedagógicos y exploramos los desafíos tanto técnicos como de interacción humano computador asociados a dicho desarrollo.

1.2. Objetivos

1.2.1. Objetivo general

Diseñar una aplicación que permita integrar y estructurar tareas separadas de un sistema académico para una gestión de programas educativos orientados a resultados pedagógicos, que provea soporte a flujos de trabajo para sus diferentes etapas de aprobación.

1.2.2. Objetivos específicos

- Realizar un relevamiento de los requerimientos y comparación de las herramientas existentes que aborden la problemática planteada.
- Realizar el proyecto en un marco de programación ágil, estimando y desarrollando la aplicación de acuerdo a las directrices brindadas por esta metodología.
- Validar la herramienta desarrollada con expertos del dominio principalmente y de manera preliminar con experiencias limitadas con los usuarios finales.

2. Marco teórico

2.1. Software as a service

SaaS[1], es un paradigma de entrega de software donde la misma se encuentra alojada por lo general en la nube y se entrega como servicio a través de Internet a un gran número de usuarios a través de un modelo de suscripción. Se trata de un modelo de entrega de negocio en el que tanto la aplicación y el alojamiento son gestionados y compartidos con varias empresas, que alquilan y utilizan los servicios de aplicaciones de forma centralizada [11].

2.2. Competencias académicas y su evaluación

La creciente profesionalización trajo al campo educativo elementos evaluativos tales como calidad, equidad, competitividad, eficiencia, y eficacia; junto con ellos surgieron las competencias, que pasaron a jugar papel importante en el contexto educativo. En la formación de profesionales, resalta la necesidad de reflexionar sobre los aprendizajes que se ofrecen en las instituciones educativas, las cuales deben servir al estudiante para ser útil a la sociedad, que es su entorno inmediato [4]. En otras palabras, la competencia es la capacidad de un buen desempeño en contextos complejos y auténticos. Se basa en la integración y activación de conocimientos, habilidades, destrezas, actitudes y valores.

De esa necesidad va surgiendo la idea de las evaluaciones orientadas a competencias. Cabe resaltar cómo se desenvuelve el aprendizaje basado en competencias usando aplicaciones como herramientas para la evaluación de estudiantes, mediante el análisis de los aportes que introduce la tecnología en este campo, que modifican significativamente las prácticas tradicionales [12].

Hoy día existen herramientas que ayudan al alumno a potenciar su aprendizaje y algunas de ellas son los sistemas de gestión de aprendizajes y los sistemas de gestión de evaluaciones basadas en competencias.

2.3. Sistemas de gestión de evaluación

Un AMS es un sistema, generalmente basado en tecnologías web, que permite a la institución la recolección, el manejo, y reporte de datos relacionados a las evaluaciones, por lo general basadas en competencias, del estudiante. Los AMS basadas en competencias permiten a la institución y a los educadores listar sus competencias, guardar, y mantener datos para cada competencia, facilitar conexiones a competencias similares de la institución, y generar reportes [2].

2.4. Sistemas de gestión curricular

Un CMS [2] es un aplicación automatizada que apoya todo el proceso curricular, desde la planificación hasta la implementación y evaluación. Posee una interfaz única y cohesiva en línea que permite proponer, crear, evaluar, revisar, aprobar y aplicar cursos, programas y competencias.

Curriculum es una mezcla sofisticada de estrategias educativas, contenido del curso, resultados de aprendizaje, experiencias educativas y evaluación [13]. Esta visión amplia de un CMS se deriva del ambiente actual de educación elemental y secundaria que es impulsado por los estándares de contenido de cursos obligatorios federales y estatales, y la necesidad de auditorías continuas de currículo [14].

En los enfoques actuales del desarrollo de Curriculum por lo general gira en torno a los comités curriculares. Dichos aspectos, sin embargo, sólo se consideran en un alto nivel de abstracción basado en la comprensión tácita de los miembros del comité sobre la disciplina.

2.5. Metodología Ágil de desarrollo de software

La metodología Ágil envuelve un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto [15]. Los métodos tradicionales, como Waterfall, pretenden ser capaces de modelar completamente el dominio del problema de entrada y luego esperar que se produzcan pequeños cambios (o inclusive ninguno) [15]. Los métodos ágiles asumen que el cambio es inevitable, por lo que abordan el desarrollo de software de tal manera a facilitar la adaptación de los nuevos requisitos mientras vayan surgiendo.

2.5.1. Historias de usuario

Las historias de usuario conforman la parte central de muchas metodologías de desarrollo ágil, tales como XP[3], Scrum, entre otras. Estas definen lo que se debe construir en el proyecto de software, tienen

[1] de sus siglas en inglés, Software as a Service, que significa en español software como servicio.
[2] de sus siglas en inglés, Curriculum Management System, que significa sistema de gestión curricular
[3] de sus siglas en inglés, eXtreme Programming, que significa en español programación extrema

una prioridad asociada definida por el cliente de manera a indicar cuales son las más importantes para el resultado final. Son divididas en tareas y su tiempo es estimado por los desarrolladores.

2.5.2. Épicas

Una épica es esencialmente una historia de usuario de un tamaño mucho mayor, siempre superior al tiempo de iteración máximo, y tiene como propósito el de asociar historias de usuario individuales relacionadas con un propósito de más alto nivel que cumplir. La misma es, por lo general, muy grande para que un equipo del proyecto pueda trabajar directamente sin partir en diversas historias de usuario [16].

2.6. Interacción humano-computador

En HCI definen la funcionalidad y la usabilidad de los sistemas que se desarrollan, donde la funcionalidad de un sistema es definida por un conjunto de acciones o servicios que son proveídas a los usuarios, sin embargo, el valor de la funcionalidad es verificada cuando es eficientemente utilizada por el usuario [17]. La usabilidad de un sistema con cierta funcionalidad es el rango y grado por el cual el mismo puede ser utilizada de manera eficiente y adecuada para cumplir ciertas metas para ciertos usuarios. La eficiencia de un sistema es alcanzada cuando se cumple un balance entre la usabilidad y la funcionalidad [18].

HCI es un diseño que debe producir un ajuste entre el usuario, la máquina y los servicios requeridos con el fin de lograr un balance óptimo entre la calidad y la eficiencia de los servicios.

La definición de la estrategia de UI[4] es importante para una mejor usabilidad del sistema, donde este proceso debería comenzar antes que el diseño y desarrollo de las aplicaciones. Es la visión de una solución que necesite ser verificado con potenciales usuarios que prueben que necesite el mercado [19].

3. Estado del arte

Una investigación para comprobar otros proyectos o productos con las mismas características propuestas, buscando innovación para el mercado es expuesta en la tabla 1.

Características	CurricUNET	CourseLeaf	DECA
Creación y versionamiento de competencias.			
Creación y versionamiento de cursos.	✓	✓	✓
Creación y versionamiento de programas de estudio.	✓	✓	
Cumple los Estándares de códigos de California.	✓		
Historial de versiones de competencias.			
Historial de versiones de cursos.	✓	✓	✓
Historial de versiones de programas de estudio.	✓		
Reporte de Comparación entre versiones de cursos.	✓		✓
Soporta competencias de aprendizaje del estudiante.			
Plantilla de flujo de trabajo customizable.	✓		
Permite asignar roles evaluadores en la aplicación.	✓	✓	
Permite asignar usuarios como colaboradores.	✓		
Sistema de alertas para colaboradores y evaluadores.	✓	✓	
Buzón de entrada para colaboradores y autoridades.	✓		✓
Soporte de correlatividades entre cursos.	✓		
Incluye un catálogo de cursos.	✓	✓	✓
Incluye un catálogo de programas de estudio.	✓		
Incluye un catálogo de competencias.			
UX intuitiva y efectiva.		✓	✓

Tabla 1: Relación entre sistemas de gestión curricular.

En la actualidad existen aplicaciones web que son capaces de generar solicitudes de creación de planes de estudio mediante formularios. Entre ellas se encuentra CurricUNET, que es la aplicación que abarca gran parte de las necesidades de las universidades de California pero no cuenta con una manera de gestionar competencias o capacidad de integrarse a un AMS basado en competencias.

El análisis hecho por las herramientas fue una investigación de campo y entrevistas con los usuarios.

[4]de sus siglas en inglés, User Interface, que significa en español experiencia de usuario.

3.1. CurricUNET

CurricUNET es una aplicación web diseñada para automatizar la emisión y aprobación del plan de estudios emitido por profesores y/o encargados de universidades norteamericanas; que incluyen programas, cursos y competencias [7].

En la misma se desarrollan propuestas de cursos y programas de estudio mediante formularios de la aplicación, con el objetivo de reemplazar solicitudes en papel que universidades utilizaban para emitir propuestas. Además, ofrece almacenamiento e información de plan de estudios históricos, activos y propuestos.

Todas las entradas, revisiones y reportes son accedidas por la web desde los navegadores. Posee un sistema de notificaciones integrado que permite al usuario un mejor seguimiento del progreso de las propuestas y cursos en revisión. Además, dispone de un control de versionamiento de cursos, planes de estudio y competencias.

Los usuarios del sistema pueden acceder a los reportes e historial de versiones de sus cursos y planes de estudio por lo que ayuda a una mejora continua del programa universitario.

3.2. CourseLeaf

El módulo de Curriculum de CourseLeaf es una solución de gestión basada en la web, mejorando los procesos de profesores y del comité de Curriculum de al menos 70 instituciones [8].

Cuando el módulo de Curriculum de CourseLeaf se combina con su módulo de catálogo, colegios y universidades son capaces de gestionar y realizar un seguimiento de la información del programa, desde la propuesta hasta publicar en una aplicación integrada con facilidad.

El software proporciona la generación de flujo de trabajo automático, notificaciones automáticas. Además, identifica todos los cursos, programas y departamentos que se ven afectados por los cambios propuestos en el inicio del proceso de propuesta y puede ayudar en la actualización con los cambios completados. El módulo de Curriculum de CourseLeaf se puede implementar con o sin su catálogo de Cursos.

3.3. DECA: Curriculum Navigator

DECA ofrece, mediante su módulo de Curriculum Navigator, una solución de desarrollo y administración de Curriculum [9].

Cuando se utiliza como módulo integrado de su Catálogo, denominado como Catalog Navigator, proporciona una solución que les permite iniciar el camino de aprobación de alguna carrera de grado. Asesores y administradores, por otra parte, utilizan tanto Curriculum Navigator y Catalog Navigator para desarrollar y proporcionar datos públicamente del plan de estudios para sus estudiantes actuales y futuros.

3.4. Relevancia del módulo curricular

Frecuentemente, descrito como complejo e ineficaz, los métodos tradicionales basados en papel de gestión y diseño de programas de estudio proporcionan una visibilidad limitada, lo que resulta en una visión restringida de las etapas implicadas en la creación, modificación y aprobación de planes de estudio. Por lo tanto, se buscó eliminar esta complejidad al acelerar el desarrollo curricular y el proceso de aprobación.

En el flujo actual, una vez finalizado el proceso de diseño y revisión curricular de parte de las oficinas, se procede a publicar la nueva competencia, curso, o programa para que cada universidad tenga la información necesaria para ir cargando la misma en sus correspondientes sistemas.

En el caso de las universidades comunitarias del estado de California, se utilizan los AMS para gestionar y evaluar las competencias de sus estudiantes. El proceso de registro de las nuevas entidades en los AMS es individual; eso quiere decir que un encargado del AMS tiene como trabajo el de cargar uno por uno las nuevas entidades aprobadas y publicadas por el comité curricular.

La importancia del módulo reside en la automatización de formularios y procesos que requieren la participación de personas ajenas al flujo de trabajo, para iniciar y validar propuestas de creación o revisión de cursos y programas. Luego, una vez que ha sido aprobada por el comité curricular se procede a la creación de las entidades de manera automática en el AMS.

Actualmente, hay alternativas que buscan solucionar la misma problemática pero no se ha encontrado durante el proceso de relevamiento de herramientas existentes una plataforma que pueda soportar el uso de competencias ni que pueda comunicarse con los sistemas de gestión de evaluaciones basadas en competencias.

4. Caso de estudio

4.1. Diseño de la investigación

El presente trabajo ilustra como caso de estudio la aplicación de la metodología ágil en la gestión de un proyecto de desarrollo de un módulo, integrado a un sistema de gestión y evaluación de competencias de una organización ubicada en los Estados Unidos. El trabajo realizado fue propuesto por la organización que brinda el sistema de gestión de competencias, donde el mismo sirve como base al módulo de gestión curricular a desarrollarse.

El caso refleja de manera práctica cómo se ha desenvuelto un proceso de desarrollo ágil en un diálogo con usuarios ubicados en diferentes localidades como parte del sistema de trabajo. En este proyecto los miembros ubicados en forma remota se encargan del diseño de las tareas, aquí llamadas historias de usuario, a realizarse, y de la validación de las características entregadas.

Debido a que los requisitos de la aplicación a desarrollarse se acordó desde un principio que se irían esclareciendo acorde se vayan completando las funcionalidades, se optó por la metodología ágil como técnica de gestión de desarrollo de software, puesto que era el enfoque más adecuado para poder encarar la problemática.

La observación participante de parte del investigador fue un paso inicial para el desarrollo del sistema en un nuevo equipo, donde se identifican y guían las relaciones con los informantes, lo ayuda a observar de manera embebida la organización y dinámica del equipo, y las prioridades de desarrollo. También, lo permite integrarse con los demás miembros del mismo y de esa manera le facilita el proceso de investigación, además de proveerle una cantidad de interrogantes a ser dilucidadas con los participantes [20].

El primer paso fue adaptarse a los cambios y a la metodología de trabajo del equipo de desarrollo. Dicho equipo se constituyó luego de a una encuesta previa de capacidades de todos los desarrolladores de la empresa, donde cada uno colocó sus conocimientos en herramientas o en partes del sistema de gestión de competencias, para así poder aprovechar las virtudes de cada persona. Acto seguido se procedió al análisis de las herramientas a utilizarse para corroborar que cumplen con las exigencias del sistema a desarrollarse.

Entre algunas técnicas de recolección de información se utilizan las encuestas [21]. Para este sistema un equipo en Estados Unidos utilizó la misma para proporcionar una visión general de los conocimientos generales del equipo de desarrollo.

Como caso de estudio con enfoque HCI[5], tuvo como meta la comprensión de problemas o situaciones mediante la interacción del ser humano con la computadora. Además, se buscó una documentación descriptiva del sistema y de su proceso de desarrollo, que apunta a la evolución del modelo propuesto durante el diseño de la misma, finalmente, se brinda y analiza evidencia de que la herramienta haya sido utilizada de manera exitosa mediante demostraciones a los usuarios o validaciones por parte del mismo [22].

Por lo tanto, el caso de estudio es con enfoque HCI y utiliza la observación participante como método principal de recolección de información.

4.2. Dominio de la problemática

El proyecto objetivo desarrollado utiliza como base una aplicación web AMS[6] que integra un módulo de gestión curricular a la misma. Los AMS son utilizados por las universidades en Estados Unidos para evaluar las competencias adquiridas de los estudiantes durante el proceso de su carrera o grado. También se busca la disponibilidad en dispositivos móviles. Los dispositivos inteligentes que van a poder utilizar la aplicación se definirán durante el proceso de desarrollo de la misma.

En las aplicaciones dirigidas al ambiente educativo, el hecho de utilizar nuevas tecnologías no asegura una mejor UX[7], es por eso que utilizaremos durante el desarrollo de la aplicación técnicas de HCI [22], encargadas de utilizar los patrones de diseño e interacción a seguir a la hora de construir cada uno de los componentes de la aplicación a ser desarrollada, y posteriormente validarlos.

Un requisito no funcional que forma parte de la infraestructura del caso de estudio es la utilización de la arquitectura SaaS[8], ya que provee servicios bajo un modelo de pagos de suscripción por las diferentes características que la misma ofrece.

Durante el proceso de desarrollo, se definirán épicas a desarrollarse durante el periodo de diseño de la aplicación y las historias de usuario que están contenidas en las mismas, para que el equipo de desarrollo pueda entregar valor de negocio del módulo integrado en iteraciones cortas de dos semanas.

[5]de sus siglas en inglés, Human-Computer Interaction, que significa en español interacción humano-computador.
[6]de sus siglas en inglés, Assessment Management System, que significa en español sistema de gestión de evaluaciones.
[7]de sus siglas en inglés, User eXperience, que significa en español experiencia de usuario.
[8]de sus siglas en inglés, Software as a Service, que significa en español software como servicio.

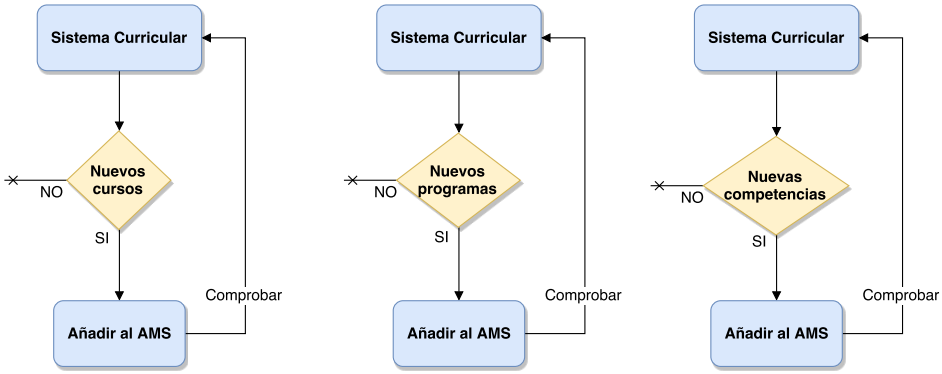

Figura 1: Proceso de cargar datos desde el sistema curricular al AMS.

4.3. Diseño curricular

Para aquellas personas ajenas al entorno educativo, en específico aquellas que no forman parte del proceso de diseño curricular puede ser un tanto complejo el proceso de creación y revisión de material curricular en las instituciones. En la actualidad, un formulario de creación o revisión de competencias, cursos, y programas debe pasar por una serie de evaluadores que son los encargados de revisar y verificar que los datos sean válidos.

Hoy día el estado de California cuenta con un patrón de definición de cursos y programas donde la misma sirve como guía para el desarrollo de propuestas para material académico de las universidades. Dicho estándar además contiene una taxonomía de programas e indica cuál es el flujo para la revisión de las propuestas, donde todo es establecido en el PCAH[9] [23].

El flujo inicia con el profesor o encargado del curso o programa, una vez completado pasa por la mesa de recepción donde se verifica que cumpla con el estándar estatal para luego pasar por la oficina departamental y la oficina del decano para su revisión de contenido. Una vez revisado y con el visto bueno de ambas oficinas pasa por una última revisión por parte de la oficina curricular para ser registrada en los sistemas de gestión curricular.

Es un proceso que se hace con formularios en papel donde el profesor o encargado del curso o programa tiene que completar los campos requeridos para que el estado de California cuente al curso como válido. Dicho proceso tiene varias deficiencias:

- La creación o revisión puede tomar meses debido a los formularios que son completados a mano y requieren de revisión de varias oficinas.
- Es un flujo de una sola dirección, eso quiere decir que si es que una de las oficinas rechaza el formulario debe volver a iniciar el flujo.
- Se puede producir cuellos de botella en los diferentes puntos de revisión.

Una vez ya registrado en los sistemas de gestión curricular es accesible de manera pública para el uso de las universidades del estado de California. De esta manera, si una institución académica posee un sistema de gestión de evaluaciones y quiere incluir los cursos o programas válidos para el estado tiene que ingresar los nuevos datos del sistema de gestión curricular uno a uno como se puede apreciar en la Figura 1.

5. Propuesta de trabajo

Una vez que los requerimientos iniciales han sido fijados y aclarados se busca la manera de automatizar los procesos, investigar tecnologías, y metodologías que ayuden al equipo de desarrollo para entregar funcionalidades de manera iterativa y evolutiva. Durante este proceso se diseñaron modelos donde se propone el módulo a ser desarrollado (Figura 2) y se buscó unir procesos separados del diseño curricular con el flujo de agregar las competencias, cursos, y programas al AMS.

Con el uso de los formularios a papel, el encargado de curso o programa completa su formulario y lo entrega en la mesa de recepción, la misma se encarga de verificar que los datos completados sean válidos y cumpla con el estándar de creación de cursos y programas.

El módulo buscó automatizar dicho proceso sacando la mesa de recepción como iniciador del flujo de validación mediante un formulario web. Además, se agrega un nuevo tipo de formulario para las competencias de las universidades comunitarias de California.

[9]de sus siglas en inglés, Program and Course Approval Handbook, que significa en español manual de aprobación de cursos y programas.

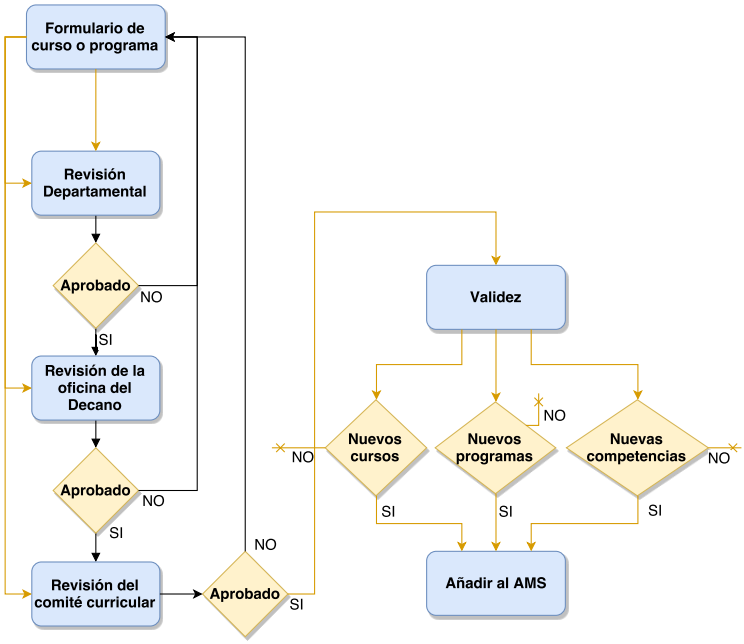

Figura 2: Modelo propuesto del módulo curricular adherido a un sistema de gestión de evaluaciones basadas en competencias.

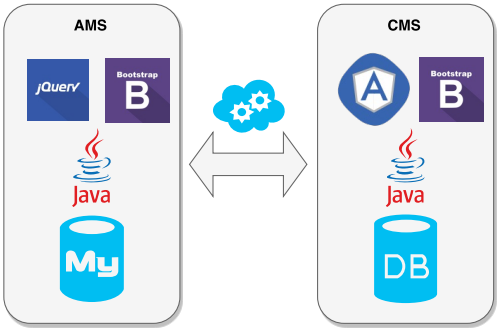

Figura 3: Arquitectura del módulo curricular.

Luego, pasa por las oficinas del departamento, del decano, y del comité curricular para sus correspondientes revisiones. Si es que una de las oficinas rechaza el formulario debe volver al inicio con el encargado del mismo para volver a ser completado, y una vez terminado puede volver a pasar a la oficina que rechazó el formulario sin necesidad de volver a iniciar todo el proceso de corrección.

Y finalmente, una vez que el comité curricular acepta el formulario se procede a generar los nuevos cursos, programas, o competencias en el AMS. También, otro proceso automatizado por el módulo ya que dicha creación se desarrollaba de manera manual.

Además, se añade la funcionalidad de mensajes generados y notificaciones a los integrantes del flujo para evitar de esta manera los cuellos de botella con las revisiones, donde se notifican los pendientes y alertan trabajos en deuda.

5.1. Modelo de arquitectura de módulo

El proyecto final (figura 3) fue diseñado como módulo de un AMS utilizado en universidades del estado de California.

El AMS utilizado como base tiene una trayectoria de largos años de uso en varias universidades. Utiliza MySQL como motor de base de datos, Java como lenguaje de programación para la lógica de la aplicación y como conector a la base de datos se utiliza, y Bootstrap y JQuery para la interfaz de usuario.

El proyecto final ha utilizado la misma base de datos, lenguaje de programación, y Bootstrap como requisitos no funcionales para el módulo curricular. Se optó cambiar JQuery a AngularJS debido a que al contar con un modelo MVC[10] en la capa de presentación desde el comienzo resultó muy atractivo para acelerar el ritmo de trabajo y poder comenzar a implementar interfaces más complejas sin tener que preocuparse por las cuestiones más triviales que Angular maneja con directivas ya definidas como el uso de «data binding», además, cuenta con una cantidad de documentación de parte de la comunidad que lo

[10]de sus siglas en inglés, Model View Controler, que significa en español modelo vista controlador.

hacía aún más atractivo.

6. Proceso de desarrollo

El módulo como proyecto de desarrollo enfocado a la metodología Ágil se encuentra dividido en varias épicas para partir en las funcionalidades.

Una épica se encuentra dividida en varias historias de usuario, donde las historias de usuario tienen el propósito de entregar valores de negocio al cliente en un periodo establecido de 2 semanas como sprint. Estas historias de usuario pueden ser a la vez divididas buscando la simplicidad de las historias donde cada una debe seguir la práctica INVEST de la metodología Ágil.

Cada historia puede estar compuesta de tareas que tienen como propósito servir al desarrollador como recordatorio de algunas labores pendientes a la hora de desarrollar la historia. Cada tarea debía tener un encargado, pero no necesariamente una persona trabajando en dicha tarea.

El PO[11] se encarga de la creación de épicas e historias de usuario, en caso de que la historia sea muy grande para terminar en un solo sprint o iteración se vuelve a partir en historias más pequeñas.

Para el caso de estudio, cada sprint consta de 2 semanas de trabajo, donde los desarrolladores como equipo se comprometen a entregar cierto valor de negocio que ellos estiman poder terminar en dicho periodo. Sin embargo, en caso de que el equipo considere que la totalidad de historias no podrán ser entregadas antes de que termine el periodo se pasa al siguiente sprint o se achica la historia minimizando los criterios de aceptación y los restantes se agregan en otra historia de usuario para las siguientes iteraciones.

Cada equipo tiene un líder, donde cada líder tiene como rol ser la brecha que une al PO con los desarrolladores. El PO se reúne con el líder de cada equipo para verificar las prioridades de las historias de usuario que están pendientes en el backlog[12].

Al inicio del diseño de la aplicación se llevará a cabo una serie de diseños de funcionalidad y usabilidad que llevar a la mejor experiencia de uso del módulo de gestión curricular, donde dichos diseños serán validados por el equipo en los Estados Unidos antes de iniciar el desarrollo.

Se ha utilizado la técnica Scrum en la gestión de proyectos ágiles debido a que en la misma se aplican de manera regular un conjunto de prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto, además de que el equipo ya se encontraba familiarizado por la misma. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

Se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al PO. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos donde se necesita obtener resultados con el mínimo esfuerzo y los requisitos son cambiantes o poco definidos. Además, en dichos ambientes la innovación, la competitividad, la flexibilidad, y la productividad son fundamentales.

6.1. Proceso

Como ya se mencionó, el proyecto se ejecutó en bloques temporales cortos y fijos que los conocemos como sprints o iteraciones. Estas iteraciones por lo general duran 2 semanas aunque en algunos equipos son de 3 y hasta 4 semanas, límite máximo de feedback y reflexión [15]. Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El proceso parte de la lista de objetivos o requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en sprints y entregas.

6.2. Planificación de iteraciones

El primer día de la iteración se realiza la reunión de planificación de la iteración y consta de dos partes:

- **Selección de requisitos** (4 horas máximo) – El PO presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo pregunta al PO las dudas que surgen y selecciona los requisitos prioritarios que se compromete a completar en la iteración, de manera que puedan ser entregados en caso de ser solicitados.
- **Planificación de la iteración o sprint** (4 horas máximo) – El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos a que se ha comprometido. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se asignan las tareas.

[11] de sus siglas en inglés, Product Owner, que significa en español dueño del producto.
[12] Bolsa de historias de usuarios pendientes.

6.3. Ejecución del Sprint

El equipo realiza una reunión diaria (15 minutos aproximadamente). Cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido. En la reunión cada miembro del equipo responde a tres preguntas:

- ¿Qué he hecho desde la última reunión diaria?
- ¿Qué voy a hacer a partir de este momento?
- ¿Qué impedimentos tengo o voy a tener?

Durante la iteración el Scrum Master se encarga de que el equipo pueda cumplir con su compromiso y de que no se merme la productividad del equipo. Además, elimina los obstáculos que el equipo no puede resolver por sí mismo.

Durante el sprint, el PO junto con el equipo refinen la lista de requisitos para prepararlos para los siguientes sprints y, si es necesario, cambian o vuelven a planificar los objetivos del proyecto para maximizar la utilidad de lo que se desarrolla y el retorno de inversión.

6.4. Aporte

En la tabla 2 se pueden apreciar los aportes propios en cada historia de usuario. El trabajo consistía en desarrollo de la historia, desarrollo de tests, y el miembro del equipo que no estuvo presente en el desarrollo de la historia es el encargado de hacer la validación de código y funcionalidad.

Historias de usuario	Aporte
Diseño de modelo de versionamiento de entidades	20 %
Versionamiento de competencias	61,5 %
Flujo de trabajo simple	62,5 %
Aprobar pasos completados de flujos de trabajo	40 %
Rechazar pasos completados de flujo de trabajo	60 %
Buzón de entradas de flujos de trabajo	40 %
Notificaciones con soporte a etapas	20 %
Versionamiento de evaluaciones	37,5 %
Versionamiento de cursos	61,5 %
Información básica de curso	20 %
Horas y unidades de evaluación de curso	8 %
Especificaciones de curso	40 %
Requisitos de cursos	60 %
Revisar y aprobar curso	40 %
Competencias de curso	25 %
Esquema de curso	20 %
Códigos de clasificación de curso	60 %
Información básica del programa	50 %
Competencias de programa	60 %
Bloques de curso por programa	20 %
Visualizar cambios en los campos	8 %
Roles de creación y edición para partes	23 %
Diseño e implementación de etapas	38 %
Mejora en comportamientos para las etapas por roles	37,5 %
Composición de etapas por roles	23 %
Etapas y partes opcionales por en la revisión	37,5 %
Notificaciones para las partes de flujos de trabajo	20 %
Reporte de esquemas de curso	25 %
Interfaz de alineación de códigos TOP/CIP	100 %
Reorganización de pestañas del módulo curricular	100 %
Lista mejorada de cursos y programas	100 %
Retoques finales para el flujo de trabajo de curso	37,5 %

Tabla 2: Tabla de historias de usuario y aportes

Dichas historias de usuario eran entregadas para validación de parte del equipo de desarrolladores y por el equipo de expertos en didáctica, una vez que era aprobada se procedía a integrar el nuevo código en el repositorio.

6.5. Inspección y adaptación

El último día de la iteración se realiza la reunión de revisión del sprint la cual consta de dos partes:

- **Demostración** (3 horas aproximadamente) – El equipo presenta al PO los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios ocurridos en el contexto del proyecto, el PO realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, volviendo a planificar el proyecto.
- **Retrospectiva** (1 hora) - El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El Scrum Master se encargará de ir eliminando los obstáculos identificados.

7. Validación del desarrollo

El flujo de validación de las historias de usuario fue un proceso sistemático, consistente y controlado por parte de los desarrolladores y de los expertos en educación encargados de validar el software desarrollado. Las pruebas y otras validaciones abarcan las siguientes dimensiones:

- Entradas, salidas y funciones del módulo curricular.
- Todos los requisitos no funcionales con sus respectivas pruebas. Como la utilización de tecnologías como Java, MySQL, Bootstrap, entre otras.
- Pruebas de rendimiento, fiabilidad, y tiempos de respuesta.
- Pruebas por parte de expertos de dominio.
- Verificación en diferentes clientes con diversos navegadores y sistemas operativos.
- Límites de intervalos, valores por defecto y valores específicos que el módulo acepta.
- Criterios de aceptación, especificaciones de requerimientos, funcionales y no funcionales expresados en las historias de usuario y otros documentos del proyecto.

Cabe resaltar que la documentación de los casos de prueba en Scrum no involucra necesariamente una secuencia paso a paso a ser utilizada en las pruebas y otros controles de calidad. Al mismo tiempo, los usuarios que validan las historias (expertos en educación en el marco de este proyecto), tienen amplia libertad para aceptar o rechazar las historias de usuarios entregadas por el equipo de desarrollo.

Así también, los expertos de dominio tienen la potestad de pedir cambios a partir de la experiencia de utilización de las herramientas desarrolladas. Todos estos preceptos fueron observados en la realización de este trabajo.

En la figura 4 se puede apreciar el ciclo de vida de las historias de usuario, donde una vez que es creada pasa al estado de «TODO», que quiere decir que está pendiente a ser desarrollada. Una vez que un miembro del equipo de desarrollo comienza una historia o tarea pasa al estado de «IN PROGRESS» y cuando termina pasa al estado de «UNDER REVIEW».

En dicho estado se revisa la funcionalidad mediante validaciones de parte de los miembros del equipo de desarrollo y de parte del equipo de expertos en dominios de didáctica en universidades norteamericanas incluyendo a un PhD en educación, donde se deben cumplir los criterios de aceptación para que pase al estado de «CLOSED» que quiere decir que se terminó y que la historia fué aprobada.

En caso de que la historia no consiga cumplir los criterios de aceptación correspondientes durante la validación se considera que la historia no está terminada y que debe pasar al estado de «REOPEN», en este estado se puede pasar ya sea desde el estado «UNDER REVIEW» o si ya está en el estado «CLOSED».

Cualquier otro problema o error de código que tenga la nueva funcionalidad se debe crear un ticket de error o bug especificando como reproducir el problema y el comportamiento esperado. En caso de no poder reproducir este comportamiento se pide más información al respecto o pasa al estado de «CLOSED» en caso de que el comportamiento ya no se pueda reproducir.

7.1. Revisión por pares

Revisión por pares significa que los miembros del equipo de desarrollo revisa el código del otro miembro.

El equipo de desarrollo puede realizar evaluaciones por pares durante el desarrollo. La forma en que están sentados puede facilitar este proceso ya que puede dirigirse a la persona que está a su lado y pedirle que revise su trabajo. El equipo de desarrollo también puede reservar tiempo durante el día

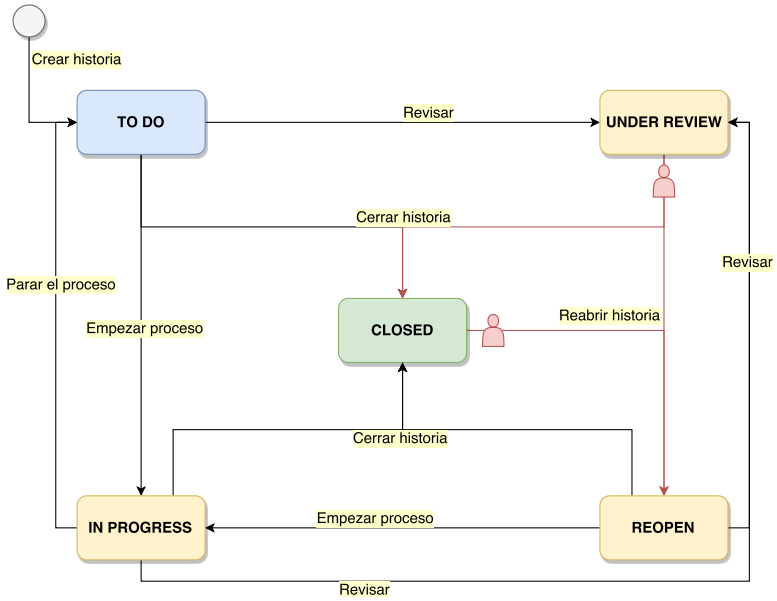

Figura 4: Flujo de desarrollo de historias de usuario.

específicamente para revisar el código. Los equipos autogestionarios deben decidir qué es lo que funciona mejor para su equipo ya sea al inicio de cada iteración o luego de cierto periodo de pruebas.

Por cada funcionalidad terminada y aún en estado «In Progress» se procede a crear un PR[13] desde la página de «GitHub».

La plantilla de los PR del repositorio tienen los siguientes campos:

- **Descripción:** se describen los cambios en detalle, ya sea una nueva funcionalidad para el módulo o el problema, causa y cómo se arreglo una falla.
- **Pasos de prueba:** se describe de manera detallada los pasos que se siguieron para probar la rama. Por ejemplo, se coloca el usuario con el rol y cuáles fue la interacción del mismo con la aplicación.
- **Tipo de cambio:** si es una nueva funcionalidad, falla o cambio urgente al sistema.
- **Aceptación de términos y condiciones:** afirmando que se siguió la guía de buenas prácticas de código y que se leyó el documento con las reglas internas de escritura de código. Además, afirmando que las pruebas automatizadas nuevas y existentes pasan.

Y deben seguir los siguientes requisitos:

- La rama debe estar probada desde una de las instancias del equipo en AWS. En caso de que las pruebas pasan, el usuario que no trabajó en la rama y que se encarga de probar la misma debe aprobar el PR desde la interfaz de revisión de «GitHub».
- Se deben cumplir los criterios de aceptación y no debe agregar fallas al proyecto. En caso de encontrar fallas se debe notificar al propietario de la rama mediante comentarios desde la interfaz.
- Por cada rama en PR se ejecutan las pruebas automatizadas de karma.
- Debe pasar las reglas de código estático de la herramienta «Sonarqube».

En caso de que uno de estos requisitos no se cumplan, «GitHub» no permite a los usuarios hacer la integración de la rama en el repositorio.

7.2. Pruebas automatizadas

Consiste en la automatización de pruebas por medio de código ejecutable, que permite controlar la ejecución de pruebas y la comparación entre los resultados obtenidos y esperados. Las pruebas automatizadas permiten incluir pruebas repetitivas y necesarias dentro de un ciclo de desarrollo para optimizar tiempo utilizado en pruebas manuales [24].

A menudo, el equipo de desarrollo desarrolla el código durante el día y corren las pruebas automatizadas durante la noche para tener los resultados a la mañana. Por la mañana, el equipo del proyecto

[13]de sus siglas en inglés, Pull Request, que es una petición que el propietario de una rama del repositorio hace al propietario del repositorio original para que incorpore los commits que están en la rama.

puede revisar el informe de errores que generó el programa de pruebas, informar sobre cualquier problema durante el informe diario de cada desarrollador y buscar corregir esos problemas inmediatamente durante el día.

Algunas de las pruebas utilizadas son:

- **Pruebas de integración:** 264 pruebas automatizadas de karma escritas para el módulo curricular, en la cual debían pasar las que hay en el sistema con las nuevas que se agregan.
- **Pruebas de exploración:** con el fin de asegurar la funcionalidad y la estabilidad del producto haciendo pruebas manuales buscando un comportamiento semejante a lo que haría un usuario normal.
- **Análisis de código estático:** 1221 reglas de Java de «Sonarqube» donde se informa sobre código duplicado, estándares de codificación, potenciales errores, comentarios y diseño del software.

Esto permitió comprobar que la aplicación funciona sobre los motores de javascript que la van a ejecutar, lo que nos ayudó a detectar posibles problemas por incompatibilidades de javascript entre ellos.

Con karma para las pruebas de interfaz, se decidió también el uso de la herramienta «Sonarqube» para evaluar el código fuente que se escribe para el módulo curricular. Donde dicha plataforma utiliza diversas herramientas de análisis estático de código funte para obtener métricas que pueden ayudar a mejorar la calidad del código del programa. Como ya se comentó, uno de los requisitos para que la funcionalidad pueda ser agregada al repositorio es que el PR pase las reglas que la plataforma impone.

7.3. Revisión por parte del equipo de validación

Una vez finalizado el proceso de desarrollo y verificación de las historias de usuario, el equipo de validación se encarga de revisar la funcionalidad y verifica que cumple con los criterios de aceptación. El equipo de validación forma parte del ciclo de vida de cada historia de usuario y hace las verificaciones acorde se agreguen a su lista de trabajos pendientes.

Finalmente, el propietario del producto debe comprobar y verificar que la historia de usuario en cuestión cumple con la definición de «DONE». Cuando una historia de usuario cumple la definición de «DONE», el propietario del producto actualiza la tabla de tareas moviendo la historia de usuario de la columna «UNDER REVIEW» a la columna «DONE».

Por lo general, una vez que el sprint termina, se hace una revisión de las funcionalidades agregadas mediante una reunión utilizando la herramienta «GoToMeeting», la cual sirve para reuniones desde cualquier parte del mundo con la capacidad de interactuar y compartir pantallas. El equipo de desarrollo muestra las nuevas funcionalidades y como se acceden a las mismas para facilitar la validación de parte del equipo experto en didáctica. Todos los errores o cambios pequeños de las historias se arreglan luego de la reunión o en caso de que sea muy grande y tome más de un día en arreglarse se procede a la creación de tickets de fallas.

Una vez finalizada la reunión, el equipo de validación y el equipo de desarrollo se comunican a través de la plataforma «Slack», donde la misma permite compartir cualquier tipo de archivos, pantallas, y mantener una conversación accediendo desde cualquier plataforma. En la misma plataforma el equipo de validación hace sus consultas y reporta los errores antes de que cree tickets de falla en la plataforma «JIRA».

8. Conclusión y aportes

8.1. Conclusiones generales

En el proyecto final se planteó como un caso de estudio con enfoque en la interacción humano-computador a través de una observación participante en el diseño e implementación de un módulo de gestión curricular para un sistema de gestión de evaluaciones basadas en competencias académicas. Se contó con un grupo de expertos en educación así como un grupo de usuarios que acompañaron todo el proceso de desarrollo y validación.

Se diseñó la manera de integrar y estructurar procesos separados de validación de cursos y programas para el estado de California con un sistema de gestión de evaluaciones basadas en competencias. Dicho proceso se realizaba en el pasado en papel y tenía sus falencias debido a que el mismo requería mucho tiempo en revisar y aprobar los formularios pertinentes. Asimismo, la complejidad del flujo aumentaba cuando habían más personas colaboradoras o evaluadoras en el proceso.

Debido a que los requerimientos eran cambiantes y el equipo que diseñaba no disponía de un panorama completo de las funcionalidades del módulo curricular, la elección de la metodología ágil para el desarrollo del proyecto fue acertada debido a que la misma permitía el desarrollo iterativo e incremental del software con validaciones del cliente como proceso de desarrollo, que en este caso el equipo de validación tomaba el rol de cliente debido al conocimiento y experiencia en didáctica de sus miembros.

Por lo tanto, algunas de las experiencias adquiridas en este caso de estudio al utilizar la metodología ágil fueron las siguientes:

- Se pudo mejorar rápidamente la elección de que construir, pero tuvimos que tener en cuenta las revisiones de los expertos y usuarios que sugirieron cambios para luego realizarlos.
- Se realizó un importante esfuerzo en modular las unidades de trabajo en pequeños incrementos.
- Con la experiencia que se fue acumulando, el equipo fue progresivamente mejorando la calidad de sus estimaciones respecto al trabajo que se podía realizar en cada sprint.

El diseñar los flujos de trabajo para el diseño y revisión de formularios de competencias, cursos, y programas permitió a los profesores encargados y a los evaluadores de dichos formularios seguir el proceso de una manera intuitiva logrando una mejor experiencia de usuario. Esto se puede apreciar en la aceptación de las historias de usuario y la creación de nuevas historias de mejora, propuestas por los usuarios, que a su vez fueron aprobadas. Un ejemplo de estas mejoras fue la generación de mensajes en cada paso para que el usuario pueda acceder en su buzón de entrada en caso de que tuviera trabajo pendiente.

En el desarrollo del módulo se utilizaron muchas de las tecnologías y herramientas que disponía el sistema como requerimiento no funcional de parte de la organización. El uso de estas tablas en común para la funcionalidad de plantillas de flujos de trabajo trajo consigo muchos problemas técnicos, debido a que agregaba complejidad a las mismas y además las pruebas de componentes se convertían en pruebas de regresión debido a la complejidad de la estructura.

La decisión de utilizar MySQL para guardar el flujo de trabajo y todos sus datos temporales también acarreó muchos problemas técnicos pues la funcionalidad y el estándar de estructuras de datos de cursos tienen cambios constantes. Sin embargo, la naturaleza iterativa del proceso de desarrollo ágil nos permitió identificar estas falencias y corregirlas en el curso del proyecto.

Debido a la capa adicional de comunicaciones con el usuario final personificada por el equipo de Estados Unidos (que en nuestro caso actúa como cliente), la realimentación de valor real o valor aún necesario provisto al usuario final es lenta e implica grandes cambios luego de varios sprints. Sin embargo, dado el gran número de usuarios finales involucrados, el grupo de expertos también ayudó a clasificar y organizar la comunicación con los mismos.

A pesar de todas las falencias de desarrollo, el módulo tuvo resultados positivos por parte de los usuarios finales ya que la herramienta construida automatiza trabajos de validación curricular para las instituciones. Además, al tener comentarios acerca de qué habría que mejorar en la aplicación y con la utilización de la metodología ágil se permitió que se creen nuevas historias de usuario para algunos retoques futuros en el módulo curricular.

8.2. Aportes del proyecto

La capacidad otorgada por este sistema para gestionar el diseño y validación de material curricular en el estado de California, y la comunicación que ofrece entre los encargados del diseño de los formularios y los evaluadores de los mismos constituye el principal aporte de este trabajo.

Además, podemos citar otros aportes:

- El módulo desarrollado reemplaza la preparación manual de formularios, los procesos de revisión y aprobación curricular de colegios ya sea para universidades, cursos, y programas.
- El módulo constituye un ambiente que puede ser accedido desde cualquier navegador y la capacidad de compartir comentarios, permite a las personas trabajar en conjunto sin necesidad de agendar reuniones para desarrollar el formulario o las revisiones.
- El módulo desarrollado permite almacenar los datos nuevos, históricos, propuestos y activos de competencias, cursos, y programas.
- Al proveer notificaciones automatizadas cuando hay cambios de estado en los flujos de trabajo.
- En la opinión de expertos y usuarios, la utilización del módulo desarrollado reduciría el tiempo promedio de formulación y revisión de cada flujo de diseño e implementación de flujos de trabajo, debido a que el mismo facilita la labor de llenar formularios utilizando información ya existente en el AMS.

8.3. Trabajos futuros

En consenso con los usuarios finales de la aplicación y los que diseñan las historias de usuarios se observaron ciertas características que podrían dar una mayor utilidad al proyecto, donde citaremos algunos de los trabajos futuros ya creadas como épicas del proyecto:

- **Importador de cursos:** muchos de los cursos que ya fueron agregados al sistema de gestión curricular del estado de California existen y como trabajo futuro para el módulo curricular es la forma de importar todos estos cursos al AMS sin necesidad de hacer todo el flujo de trabajo para las competencias, cursos y programas válidos actualmente.
- **Migración de motor de base de datos de los flujos de trabajo:** como ya comentamos, la decisión de tecnología en cuanto al motor de base de datos fue una decisión errónea, debido a que el estándar de California para diseño y revisión de cursos y programas puede variar con el tiempo y la utilización de MySQL como motor de base de datos dificulta el desarrollo en sí debido al constante cambio del modelo de datos, por lo que se considera un motor de base de datos no relacional basada en documentos, e.g. «MongoDB» como una opción a futuro.
- **Acceso de información a través de API[14] pública:** es una práctica que exige el estado de California que todos los datos en cuanto a cursos y programas de las universidades puedan ser accedidas desde una API pública. Por lo tanto, se debe diseñar e implementar una interfaz pública que permita comunicarse y acceder a los datos del módulo curricular.
- **Catálogo de cursos:** los CMS del estado tienen la opción de mostrar sus cursos válidos de manera pública para que las universidades puedan cargar en sus sistemas académicos como se aprecia en la figura 1. Por lo tanto, como trabajo futuro para el módulo curricular se busca catalogar los cursos de manera pública desde una interfaz web.
- **Flujo de trabajo para evaluaciones:** el mismo flujo de trabajo que se diseño e implementó para las competencias, cursos, y programas, se debe implementar para las evaluaciones.

Desde el punto de vista académico sería interesante realizar una experiencia controlada de usabilidad del sistema para medir con mayor precisión las ganancias en materia de productividad que conlleva la utilización de este módulo.

Bibliografía

[1] George D. Kuh y col. *Knowing What Students Know and Can Do: The Current State of Student Learning Outcomes Assessment in U.S. Colleges and Universities*. English. Inf. téc. National Institute for Learning Outcomes Assessment, ene. de 2014.

[2] Rebecca Cartwright, Ken Weiner y Samantha Streamer-Veneruso. «Student learning outcomes assessment handbook». En: *Montgomery County, MD: Montgomery College* (2009).

[3] Casey A. Barrio Minton, Donna M. Gibson y Carrie A. Wachter Morris. *Evaluating student learning outcomes in counselor education*. Alexandria, VA: American Counseling Association, 2016. ISBN: 978-1-55620-337-4.

[4] George D. Kuh, ed. *Using evidence of student learning to improve higher education*. First edition. San Francisco, CA: Jossey-Bass, 2015. ISBN: 978-1-118-90339-1.

[5] Angela Di Michele Lalor. *Ensuring high-quality curriculum: how to design, revise, or adopt curriculum aligned to student success*. Alexandria, VA: ASCD, 2017. ISBN: 978-1-4166-2279-6.

[6] Bill Boyle y Marie Charles. *Curriculum development: a guide for educators*. OCLC: ocn919483148. Los Angeles: SAGE, 2016. ISBN: 978-1-4462-7330-2 978-1-4462-7329-6.

[7] Governet. *CurricUNET*. `http://curricunet.com`. 2013.

[8] Leepfrog Technologies. *Courseleaf*. `https://www.leepfrog.com/courseleaf/`. 2017.

[9] CSDC Systems. *DECA*. `http://www.decisionacademic.com`. 2017.

[10] Per Runeson y col. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons, 2012.

[11] Nitu Gupta y Varshapriya JN. «Software as a Service». English. En: *International Journal of Innovative Research in Advanced Engineering (IJIRAE)* 1.6 (2014). ISSN: 2349-2163.

[12] Ronald S. Carriveau. *Connecting the dots: developing student learning outcomes and outcomes-based assessments*. Second edition. Sterling, Virginia: Stylus Publishing, LLC, 2016. ISBN: 978-1-62036-479-6 978-1-62036-480-2.

[13] RM Harden. «AMEE Guide No. 21: Curriculum mapping: a tool for transparent and authentic teaching and learning». En: *Medical teacher* 23.2 (2001), págs. 123-137.

[14] de sus siglas en inglés, Application Programming Interface, que sirve como un conjunto de reglas para que las aplicaciones puedan comunicarse entre ellas.

[14] Gary West. «Technology tools to make educational accountability work». En: *THE Journal (Technological Horizons In Education)* 28.5 (2000), pág. 60.

[15] Christopher W. H. Davis. *Agile metrics in action: how to measure and improve team performance*. OCLC: ocn907160912. Shelter Island, NY: Manning, 2015. ISBN: 978-1-61729-248-4.

[16] Charles G Cobb. *The project manager's guide to mastering Agile: Principles and practices for an adaptive approach*. John Wiley & Sons, 2015.

[17] Ben Shneiderman y Catherine Plaisant. *Designing the user interface: strategies for effective human-computer interaction*. 5th ed. OCLC: ocn244066651. Boston: Addison-Wesley, 2010. ISBN: 978-0-321-53735-5.

[18] Jakob Nielsen. *Usability engineering*. eng. Nachdr. OCLC: 760142137. Amsterdam: Kaufmann, 2010. ISBN: 978-0-12-518406-9.

[19] Jaime Levy. *UX strategy: how to devise innovative Digital products that people want*. First edition. Beijing ; Sebastopol: O'Reilly Media, 2015. ISBN: 978-1-4493-7286-6.

[20] David A. Erlandson, ed. *Doing naturalistic inquiry: a guide to methods*. Newbury Park, Calif: Sage, 1993. ISBN: 978-0-8039-4937-9 978-0-8039-4938-6.

[21] Colin Robson. *Real world research: a resource for users of social research methods in applied settings*. eng. 3. ed. OCLC: 729956086. Chichester: Wiley, 2011. ISBN: 978-1-4051-8240-9 978-1-4051-8241-6.

[22] Jonathan Lazar, Jinjuan H. Feng y Harry Hochheiser. *Research Methods in Human-Computer Interaction*. 1.ª ed. Published: Paperback. Wiley, feb. de 2010. ISBN: 0-470-72337-8.

[23] Brice W. Harris. *Program and Course Approval Handbook*. English. Fifth Edition. Sep. de 2013.

[24] Lisa Crispin y Janet Gregory. *Agile testing: A practical guide for testers and agile teams*. Pearson Education, 2009.