UNIVERSIDAD CATÓLICA "NUESTRA SEÑORA DE LA ASUNCIÓN"

INGENIERÍA INFORMÁTICA

PROYECTO FINAL DE CARRERA

Desarrollo de un sistema de gestión de programas de estudio orientado a resultados pedagógicos

Autor:
Luis F. VILLALBA V.

Tutores:
Ing. Sebastián ORTÍZ
Ing. Wilfrido INCHAUSTTI

Facultad de Ciencias y Tecnología
Departamento de Electrónica e Informática

2017

Desarrollo de un sistema de gestión de programas de estudio orientado a resultados pedagógicos

---

Autor:
Luis F. VILLALBA V.

Tutores:
Ing. Sebastián ORTÍZ
Ing. Wilfrido INCHAUSTTI

Una tesis presentada en cumplimiento de los requisitos para el título de Ingeniería Informática en la

Universidad Católica "Nuestra Señora de la Asunción"
Facultad de Ciencias y Tecnología
Departamento de Electrónica e Informática

2017

Declaración de autoría

Yo, Luis F. VILLALBA V., declaro que la tesis titulada, «Desarrollo de un sistema de gestión de programas de estudio orientado a resultados pedagógicos» y todo el trabajo presentado son de mi propiedad. Además, confirmo que:

- Este trabajo fue llevado a cabo bajo las normas y regulaciones del Reglamento De Proyecto Final de Carrera (09/2015) del Departamento de Ingeniería Electrónica e Informática.
- Este trabajo fue realizado en su totalidad con fines investigativos y para obtener el título antes mencionado en esta universidad.
- Cuando cualquier parte de esta tesis haya sido previamente sometida a un título en esta universidad o cualquier otra institución, esto se ha manifestado claramente.
- Donde he consultado trabajo publicado por otros autores, eso ha sido siempre claramente atribuido.
- Donde he citado trabajo de otros autores, la fuente siempre se ha dado. Con la excepción de dichas citas, esta tesis es enteramente mi propio trabajo.
- He reconocido todas las fuentes principales de ayuda.
- Donde la tesis está basada en mi trabajo en conjunto con otros autores, he expresado claramente que fue hecho por ellos y en que he contribuido.

Firma:

Fecha:

UNIVERSIDAD CATÓLICA "NUESTRA SEÑORA DE LA ASUNCIÓN"

Abstract

Facultad de Ciencias y Tecnología

Departamento de Electrónica e Informática

Ingeniería Informática

Desarrollo de un sistema de gestión de programas de estudio orientado a resultados pedagógicos

by Luis F. VILLALBA V.

En el proyecto final se toma como caso de estudio con enfoque en interacción humano-computador y en observación participante al diseño e implementación de un módulo de gestión curricular para un sistema de gestión de evaluaciones basadas en competencias académicas, la cual tiene sus cimientos en el mercado y también dispone de clientes utilizando la misma.

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

Estos procesos fueron automatizados por el módulo curricular, además de brindar una forma de notificar a los encargados de revisiones y ediciones que tienen trabajo pendiente o en falta.

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

Agradecimientos

The acknowledgments and the people to thank go here, don't forget to include your project advisor…

Índice General

Lista de Figuras

Lista de Tablas

Lista de Abreviaturas

AMS	**A**ssessment **M**anagement **S**ystem
CMS	**C**urriculum **M**anagement **S**ystem
UX	**U**ser e**X**perience
LMS	**L**earning **M**anagement **S**ystem
SaaS	**S**oftware **a**s **a** **S**ervice
SLO	**S**tudent **L**earning **O**utcomes
PCAH	**P**rogram and **C**ourse **A**pproval **H**andbook
INVEST	**I**ndependent **N**egotiable **V**aluable **E**stimatable **S**mall **T**estable
HCI	**H**uman **C**omputer **I**nteraction
UI	**U**ser **I**nteraction
DECA	**DE**cision a**CA**demic
AWS	**A**mazon **W**eb **S**ervices
VCS	**V**ersion **C**ontrol **S**ystem
MVC	**M**odel **V**iew **C**ontroller
UI	**U**ser **I**nteraction
POJO	**P**lain **O**ld **J**ava **O**bject

Capítulo 1

Introducción

1.1. Justificación

En las últimas dos décadas, de manera a facilitar el acceso a la aplicación por parte de los usuarios, las aplicaciones web se han vuelto populares porque estas se ejecutan en un navegador y no es necesario descargar ningún tipo de software adicional debido a que el peso principal de la aplicación se ejecuta remotamente[1].

Al mismo tiempo, en la educación han surgido avances tecnológicos aplicables donde se aprovechan las TICs para potenciar el aprendizaje adquirido en los estudiantes. Se han buscado idear nuevas técnicas de evaluación para impulsar un aprendizaje significativo en los alumnos, pero en muchos casos nos encontramos que la forma de evaluar los cursos o actividades no reflejan los conocimientos o capacidades de los alumnos. Para intentar llenar este vacío se ha creado la evaluación basada en competencias, la cual se enfoca en evaluar las habilidades adquiridas por un estudiante en el proceso de un programa educativo[2].

Así, las aplicaciones de evaluación académica basadas en competencias han adquirido mucha importancia en los últimos años[2]. En dichas aplicaciones se buscan conocer las fortalezas y debilidades del estudiante de una manera modular, en comparación 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.[3].

Para las personas ajenas al entorno educativo, en específico aquellas que no participan directamente en el diseño de planes y programas, puede resultar un tanto complejo comprender el proceso del diseño curricular y reconocer la importancia de involucrar a todas aquellas autoridades que forman parte del mismo. Por esta razón, definir las bases teóricas que servirán de referencia para el diseño curricular, describir sus conceptos básicos, y distinguir sus elementos, es fundamental para el desarrollo de cada curso[4].

En el ámbito de las aplicaciones académicas basadas en competencias, si bien existen aplicaciones que abarquen 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 a un sistema de gestión de evaluaciones basadas en competencias.

1.2. 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 a realizar 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 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 irán esclareciendo acorde se vayan completando cada valor de negocio, se optó por la metodología ágil como técnica de gestión de desarrollo de software, puesto que es el enfoque más adecuado para poder encarar esta problemática.

La observación participante de parte del investigador es 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 y 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[5].

El primer paso es adaptarse a los cambios y a la metodología de trabajo del equipo de desarrollo. Dicho equipo se constituirá luego de a una encuesta previa de capacidades de todos los desarrolladores de la empresa, donde cada uno colocará 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 procede 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[6]. Para este sistema un equipo en Estados Unidos utilizó la misma para proporcionar una visión general de los conocimientos generales del equipo de desarrollo, más información al respecto en la sección 5.1.

Los casos de estudio con enfoque HCI[1] tienen como meta la comprensión de problemas o situaciones mediante la interacción del ser humano con la computadora. Además, se busca 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[7].

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

1.3. Dominio de la problemática

El proyecto objetivo a desarrollarse utiliza como base una aplicación web AMS[2] que integrará 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[3], es por eso que utilizaremos durante el desarrollo de la aplicación técnicas de HCI[7], 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[4], 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.

[1] de sus siglas en inglés, Human-Computer Interaction, que significa en español interacción humano-computador.

[2] de sus siglas en inglés, Assessment Management System, que significa en español sistema de gestión de evaluaciones.

[3] de sus siglas en inglés, User eXperience, que significa en español experiencia de usuario.

[4] de sus siglas en inglés, Software as a Service, que significa en español software como servicio.

1.4. Objetivos

1.4.1. 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.

1.4.2. 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.

1.5. Organización del libro

El documento se encuentra estructurado en 7 capítulos.

En el capítulo 2, «Marco teórico», el cual presenta el conjunto de escritos académicos tanto técnicos como educativos que constituyen los fundamentos utilizados en este trabajo de tesis.

En el capítulo 3, «Estado del arte», realizamos un relevamiento y análisis de herramientas de informática educativa que intentan resolver problemas relacionados de forma cercana a los problemas planteados en éste trabajo de tesis.

En el capítulo 4, «Propuesta de solución», se presenta en detalle la arquitectura del sistema planteada como solución, con un correspondiente estudio para la determinación de las herramientas para cada paso. Se expone además el flujo de datos y se presenta un diseño del mismo, donde queda gráficamente los pasos a seguir. Se incluyen también las técnicas de validación y evaluación que se deben utilizar.

En el capítulo 5, «Desarrollo de la aplicación», se proveerá una descripción detallada de los modelos obtenidos; en donde, se explicarán las funciones y los principales componentes de cada parte. Se inlcuyen figuras ilustrativas de los resultados de clustering obtenidos, así como la explicación de los parámetros considerados para cada algoritmo.

En el capítulo 6, «Validación del desarrollo», se encuentran los análisis y validaciones de resultados obtenidos. Además, se analiza el flujo de la validación de las historias de usuario como proceso sistemático, consistente

y controlado de parte de los desarrolladores y de los expertos en educación encargados de validar el software desarrollado.

En el capítulo 7, «Conclusión y aportes», se encuentran las conclusiones obtenidas, acompañado de un breve resumen de toda la investigación. Se exponen los aportes a la problemática, asi como también se proponen posibles temas de investigación y trabajos futuros o complementarios, que puedan continuar a partir del presente trabajo.

Capítulo 2

Marco teórico

2.1. 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 [3].

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[8].

Uno de los factores de motivación relevantes para el aprendizaje es la evaluación. Cada actividad ofrece a los estudiantes la oportunidad de conocer cuáles son sus resultados de aprendizaje en lo que se refiere al «qué» se ha aprendido y al «cómo» habría podido hacerse. Cualquier proceso de evaluación debería ser diseñado teniendo en cuenta este principio básico.

En un sistema de gestión de evaluaciones basado en competencias, los encargados hacen evaluaciones según las evidencias obtenidas de diversas actividades de aprendizaje, que definen si un estudiante alcanza o no los requisitos recogidos por un conjunto de indicadores en un determinado grado. Una evaluación por competencias asume que pueden establecerse indicadores posibles de alcanzar por los estudiantes, que diferentes actividades de evaluación pueden reflejar los mismos indicadores[9].

La evaluación por competencias ofrece nuevas oportunidades a los estudiantes al generar entornos significativos de aprendizaje que acercan sus experiencias académicas al mundo profesional, y donde pueden desarrollar una serie de capacidades integradas y orientadas a la acción, con el objetivo de ser capaces de resolver problemas prácticos o enfrentarse a situaciones cotidianas [8].

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, cuyas diferencias se mostrarán a posteriori.

2.2. Sistemas de gestión de aprendizaje

Las plataformas LMS[1] son espacios virtuales de aprendizaje orientados a facilitar la experiencia de aprendizaje a distancia y permite una cómoda interacción de profesores y alumnos. En la misma se pueden hacer evaluaciones, intercambiar archivos y participar en foros y chats, además de otras herramientas de interacción estudiante a profesor.

La centralización y automatización de la gestión del aprendizaje es una de las principales características de los LMS. La plataforma puede ser adaptada tanto a los planes de estudio de la institución como a los contenidos y estilo pedagógico de la misma.

El usuario se convierte en el protagonista de su propio aprendizaje a través del autoservicio y los servicios guiados por los tutores o profesores mediante la herramienta. Además, permite utilizar los cursos desarrollados por terceros, personalizando el contenido y reutilizando el conocimiento adquirido. Además, posee prestaciones y características que hacen que cada plataforma sea adecuada según los requerimientos y necesidades de los usuarios.

Los LMS que almacenan información de programas académicos no involucran el aspecto de resultados pedagógicos en sus estructuras de datos o procesos de aprobación. Sin embargo, no es capaz de soportar competencias de institución de manera completa, aunque puede hacerlo por medio de plugins pero es un soporte superficial y no cubre con todas las necesidades de las universidades comunitarias del estado de California. Por lo que se utilizan los AMS para evaluar a los alumnos.[10].

2.3. Sistemas de gestión de evaluación basados en competencias

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 del estudiante, por lo general basadas en competencias. Los AMS permiten a la institución y a los educadores listar sus competencias, guardar, y mantener datos para cada una, facilitar conexiones a competencias similares de la institución, y generar reportes[11].

Estas aplicaciones permiten que las competencias sean enlazadas a nivel institucional, departamental, de programas, y de división. Esta estructura permite examinar la competencia acorde al nivel que pertenece. Una representación común de este tipo de enlace es el mapa curricular[12]. Algunos

[1] de sus siglas en inglés, Learning Management System, que significa en español sistema de gestión de aprendizajes.

de estos sistemas de manejo de evaluaciones generan sus propios mapas curriculares.

Varios sistemas comerciales existen; incluyendo Blackboard Learn, Campus Lab, eLumen, LiveText, TaskStream, TracDat/Webfolio, Waypoint Outcomes y WEAVEOnline. También existen otras desarrolladas por las propias instituciones para manejar sus datos de evaluaciones.

Mientras que cada AMS tiene un conjunto diferente de capacidades, todas manejan, mantienen, y permiten generar reportes de los datos de las evaluaciones. Generalmente, teniendo como ejemplo los distintos sistemas, los AMS tienen una estructura jerárquica basada en unidades organizacionales (programas, departamentos, escuelas, colegios o la misma institución), por lo tanto, las metas y/o las competencias también se ven adaptadas a esta estructura.

2.3.1. Capacidad de evaluación

La característica más importante de todo AMS es la capacidad de soportar evaluaciones de diferentes tipos[12]. Por ejemplo, algunos AMS se centran en soportar evaluaciones acumulativas; mientras que otros permiten el seguimiento de las evaluaciones formativas. Además, las capacidades de evaluación apoyadas por una AMS pueden residir en una escala de la unidad.

Algunas de las mencionadas anteriormente permiten evaluaciones a nivel de estudiante. Un número cada vez mayor de las AMS apoyan la documentación, desarrollo o aplicación de criterios de evaluación específicos, más comúnmente a las rúbricas que se aplican a los productos creados por los estudiantes.

Como una faceta adicional, muchas de estas herramientas permiten evaluaciones para vincularse con las normas educativas y profesionales, de modo que la información de evaluación de múltiples unidades puede ser adherido como datos de reportes.

2.3.2. Alineación de capacidades

Una importante característica de cualquier AMS es la habilidad de enlazar competencias entre sí.

Primeramente, algunos AMS permiten que las competencias sean enlazadas a nivel de institución, departamento, o programa. Esta estructura permite examinar la competencia acorde al nivel que pertenece. Una representación común de este tipo de enlace es el mapa curricular. Algunos de estos sistemas de manejo de evaluaciones generan sus propios mapas curriculares.

Además, provee soporte a iniciativas de mejora continua educacionales de instituciones y competencias de aprendizaje que necesitan las evaluaciones. Un AMS puede ser levantado y utilizado para mantener y alcanzar altos estándares de calidad y cumpliendo los requisitos de acreditación[3].

Esta evaluación es un proceso complejo que requiere la contribución y la retroalimentación de todo el personal, profesores y alumnado de un centro de educación superior. El sistema AMS facilita la esquematización, la recopilación de pruebas, documentación y presentación de las contribuciones que cada uno de los programas académicos de la institución y los servicios de apoyo hace la consecución de los objetivos de calidad y eficacia institucionales.

Un ciclo de evaluación completo incluye la coordinación, la planificación, la medición, la reflexión y la toma de acción del proceso de evaluación de la institución enteras, programas o cursos.

2.4. Diferencias entre LMS y AMS

Por lo general, en las universidades norteamericanas utilizan los AMS y LMS para mejorar el proceso de evaluación de sus estudiantes. Sin embargo, para las personas que no están familiarizadas con estos sistemas pueden quedar ambiguas las diferencias entre estos sistemas de aprendizaje. Estas diferencias se encuentran sumarizadas y tabuladas en la tabla 2.1.

TABLA 2.1: Comparación de características entre plataformas AMS y LMS.

Características	AMS	LMS
Soporte de evaluaciones a los estudiantes.	✓	✓
Soporte de evaluación colectiva.	✓	
Diferentes tipos de rúbricas para evaluaciones.	✓	
Permite acceder a cursos realizados por terceros (aula virtual para el estudiante).		✓
Permite la evaluación de desempeño estudiantil.	✓	
Permite generar reportes de cursos, progresos, etc.	✓	✓
Soporte de presupuestos (solicitud de profesores, coordinadores, etc.).	✓	
Soporta competencias de aprendizaje del estudiante o Student Learning Outcomes.	✓	
Soporte de alineación de competencias.	✓	
E-portfolio y evaluación de proyectos.	✓	✓
Soporte de comunicación entre estudiantes y profesores (foro).		✓
Soporte de evaluación sumativa y formativa.	✓	
Software distribuido y desarrollado libremente.		✓

Los LMS permiten al estudiante a una capacitación flexible a distancia sin limitaciones de horarios con costos reducidos[13]. Además, como es una plataforma intuitiva, permite a las personas con nivel de conocimiento básico en informática un aprendizaje constante y actualizado a través de la interacción entre alumnos y profesores. En cambio, los AMS son más bien utilizados para evaluaciones de las competencias de los alumnos que como vínculo entre el alumno y el profesor mediante un aula virtual, y permiten la mejora continua del aprendizaje estudiantil.

Los LMS permiten la integración de competencias mediante herramientas externas, pero de una manera superficial en comparación a los AMS.

2.5. Programas de estudio

En términos generales, se puede definir un programa de estudios como una herramienta educativa que regula y ordena el proceso de enseñanza-aprendizaje a desarrollar en una unidad de aprendizaje determinada, orientando las actividades que profesor y alumno han de llevar a cabo para el logro de los objetivos planteados en dicha unidad, en relación con los objetivos del plan de estudios, de tal manera que el egresado concluya su carrera con el perfil deseado. En pocas palabras, es un esquema organizado de los contenidos situados dentro de una determinada unidad de aprendizaje[14].

El termino «unidad de aprendizaje» sustituye al de «asignatura» o «materia» que evocan los tradicionales cursos unidisciplinarios, generalmente teóricos y sobrecargados de información. Un programa resume las características de la unidad de aprendizaje, su contenido mínimo obligatorio, y sus objetivos, principalmente.

En la primera etapa de diseño curricular, se requiere la elaboración de la propuesta de los programas para su aprobación de parte de las autoridades con la colaboración de las academias y, de ser necesario, con la asesoría de externos de la unidad académica.

2.5.1. Proceso de 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 (figura 2.1) que son los encargados de revisar y verificar que los datos sean válidos.

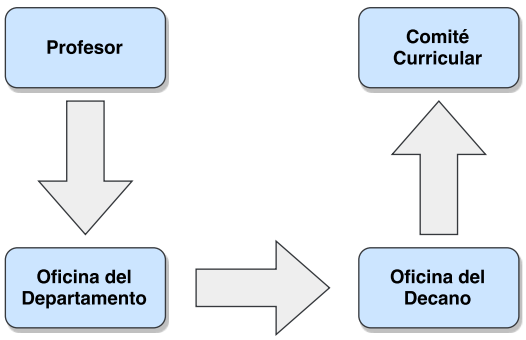


FIGURA 2.1: Flujo de creación y revisión de propuestas.

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[2][15].

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

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 (figura 2.2).

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 ingresar los nuevos datos del sistema de gestión curricular uno a uno como se puede apreciar en la figura 2.3.

2.6. Sistemas de gestión curricular

Un CMS [3] 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[16].

Curriculum es una mezcla sofisticada de estrategias educativas, contenido del curso, resultados de aprendizaje, experiencias educativas y evaluación. 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[17].

En los enfoques actuales del desarrollo de Curriculum por lo general gira en torno a los comités curriculares. Un comité curricular departamental comienza el proceso de desarrollo curricular considerando como entrada cualquiera de los aspectos mostrados en la figura 2.4.

[3] de sus siglas en inglés, Curriculum Management System, que significa sistema de gestión curricular

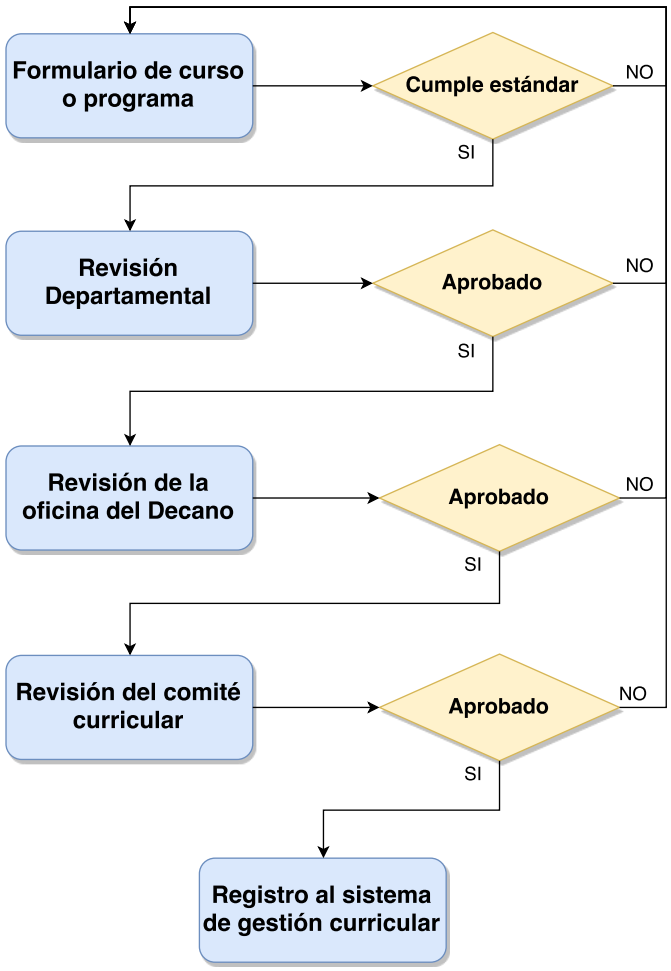


FIGURA 2.2: Flujo actual de diseño de cursos, programas y competencias.

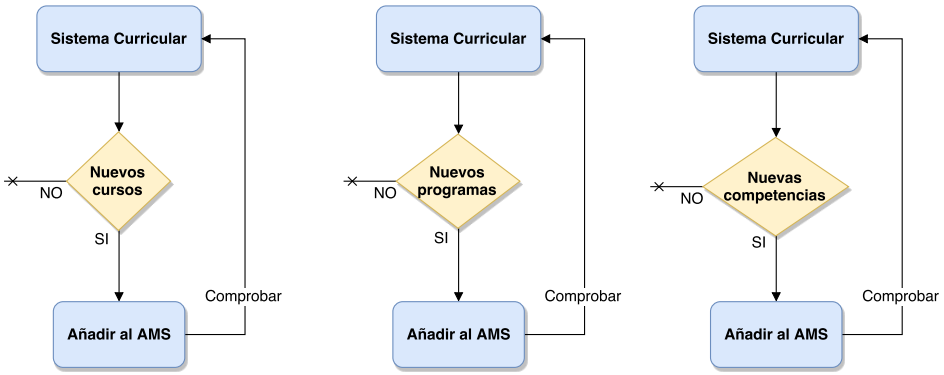


FIGURA 2.3: Esquema de tipos de computación en la nube.

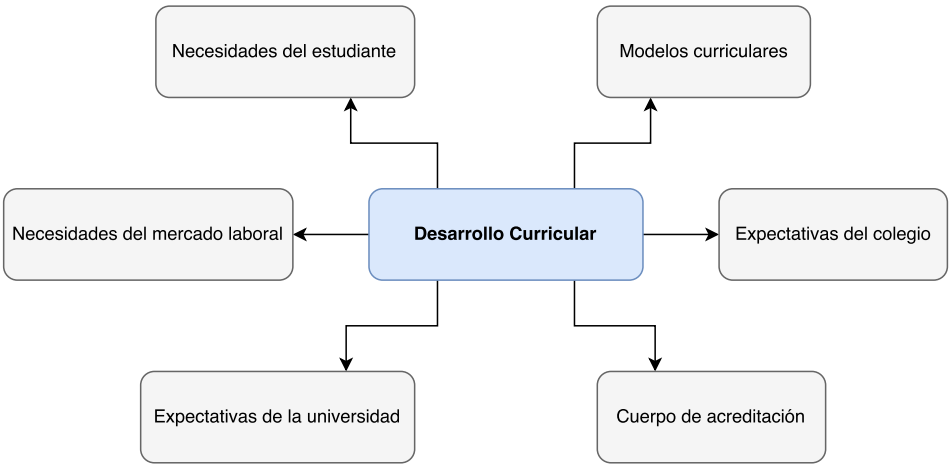


FIGURA 2.4: Esquema de diseño curricular y sus variables determinantes.

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.

2.7. Aplicaciones web

La necesidad de ejecución de operaciones complejas de manera remota y portable, tales como el uso de software sin depender de la potencia del hardware del usuario, la disponibilidad de uso en múltiples plataformas, y muchas otras, han guiado al desarrollo y evolución de las aplicaciones web. En las ciencias de la computación, una aplicación web es un software con arquitectura cliente-servidor donde el cliente (o interfaz de usuario) corre en un navegador web[1].

Los usuarios acceden a la aplicación utilizando un navegador web y no es necesario descargarse ningún tipo de software adicional, ya que la aplicación se ejecuta en el servidor. Para esclarecer el panorama, la lógica de la aplicación web se ejecuta remotamente, y el navegador web sólo se limita a la representación de los datos.

La evolución de las aplicaciones web ha crecido tan vertiginosamente que además de ofrecer una multitud de servicios, mejoran la UX a través de interfaces gráficas e intuitivas para los usuarios[18].

2.8. Cloud computing

La computación en la nube es una metáfora para abastecimiento y consumición de recursos de infraestructura. El nivel de abstracción ofrecida por la nube puede variar de hardware virtual a complejos sistemas distribuidos, debido a que los recursos están disponibles a demanda en enormes cantidades y pagados por uso[19]. Casi toda solución de infraestructura remota es basada hoy día con esta tecnología.

La infraestructura remota o nube puede ser manejada por una organización abierta para uso público, o puede ser privada, donde una nube que virtualiza y comparte la infraestructura con una sola organización o híbrida como son mostrados en la figura 2.5.

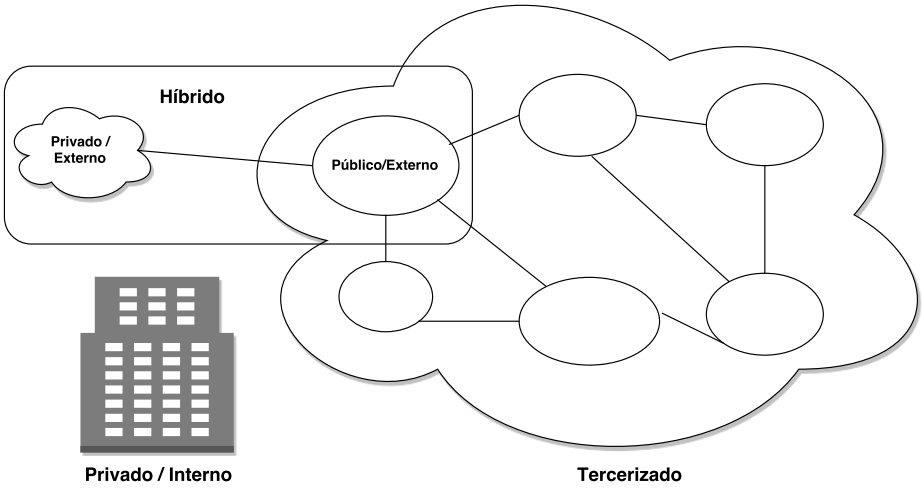


FIGURA 2.5: Esquema de tipos de computación en la nube.

2.9. Software as a service

SaaS[4], 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[20].

El software como servicio es equivalente a servicios de proveedores externos que manejan todo mantenimiento, personalización, actualización y cobro para los servicios que su cliente utiliza de manera mensual o anual. El proveedor se encarga de ofrecer el software basado en un conjunto de códigos y datos definidos junto a las diferentes configuraciones para los diferentes clientes. Los suscriptores al servicio acceden a la aplicación con la sensación de que son los únicos usuarios de la aplicación. Sin embargo, los cambios de configuraciones como cambios de datos, flujo de trabajo, interfaz y el flujo de negocio son realizados de manera masiva y transparente para ellos[21][22].

También existen otras opciones como PaaS[5] e IaaS[6].

[4]de sus siglas en inglés, Software as a Service, que significa en español software como servicio.

[5]de sus siglas en inglés, Platform as a Service, que significa en español plataforma como servicio.

[6]de sus siglas en inglés, Infraestructure as a Service, que significa en español infraestructura como servicio.

2.9.1. Características

Entre las características relevantes de las aplicaciones SaaS, y las que las remarcan como aplicaciones bien construidas, son construidas a la medida, escalables y soportan multitenancy[7]. No todas las aplicaciones SaaS comparten todas las características, pero las más comunes son las siguientes:

Configurabilidad

Las aplicaciones con esta característica poseen el mismo código base y provee a instancias con múltiples opciones de configuración tal que cada cliente pueda tener sus propias configuraciones de software únicas y pueda tener la sensación de que es el único usuario en utilizar la aplicación. Esta es la clave del éxito para las aplicaciones SaaS[23]. Por ejemplo, cada cliente puede tener configurado su sitio para que muestren fondos de pantallas o logos en las páginas de inicio de sesión o páginas principales que ellos especifican. Esta característica también puede ser llamada personalización de la aplicación.

Multitenancy

Con esta característica, una sola instancia de la aplicación corriendo puede servir a una cantidad de clientes. Los diferentes modelos de datos que están disponibles para soportar SaaS son las bases de datos aisladas, arquitectura de bases de datos aisladas compartidas y la construcción de datos[24]. Utilizando la arquitectura multitenant los proveedores de aplicaciones SaaS pueden innovar de forma sencilla y ahorrar tiempo valioso gastado en mantener varias versiones de código deprecado y/o desactualizado[25].

Escalabilidad

Esta característica es la más complicada de agregar a una aplicación SaaS debido a su elevado costo. La escalabilidad es soportada por la virtualización, pero teniendo en cuenta el costo y el problema de complejidad muchas veces el desarrollador de la aplicación no se complica con esta característica[26].

2.9.2. Ventajas

Además, aparte de estas características, algunas ventajas que poseen las aplicaciones con arquitectura SaaS son las siguientes: [26][27][28]

- Las aplicaciones SaaS pueden ser utilizadas por los usuarios por medio de sus navegadores Web. Esto ahorra en costos operacionales para el usuario junto con los requerimientos de hardware mínimo, por lo

[7] En español se conoce como multiples clientes inquilinos, se refiere cuando varios clientes pueden utilizar la misma instancia.

tanto, reduce el costo que el usuario necesita gastar en hardware. Además de los costos de mantenimiento, costos de licencias del software también son minimizados.

- Mejor utilización de recursos, debido a que los recursos requeridos por las aplicaciones con arquitectura SaaS son mínimos.
- El avance de la tecnología Web permite que los proveedores de SaaS se ubiquen en el extranjero y también ofrezcan servicios de alta calidad. De esta manera, permite a los usuarios de la aplicación ahorrar en infraestructura.
- Usualmente, las soluciones SaaS residen en entornos en la nube donde son escalables y poseen integración con otras ventajas SaaS. En comparación al modelo tradicional, los usuarios no tienen que comprar otro servidor o software.

2.10. 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. 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)[29]. 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.

Las metodologías ágiles, en comparación a otras metodologías de desarrollo, ofrecen un modelo de diseño flexible que fomenta al desarrollo evolutivo. Los desarrolladores trabajan en pequeños módulos cada vez y la retroalimentación proveída por el cliente ocurre simultáneamente en el desarrollo. Además, la metodología puede ser bastante útil en situaciones donde los objetivos finales del proyecto no están claramente definidos donde los requisitos del cliente se clarificarán gradualmente a medida que el proyecto avance.

El uso de la metodología ágil como método de entregas de las características del módulo entra como requisito no funcional.

2.10.1. Historias de usuario

Las historias de usuario conforman la parte central de muchas metodologías de desarrollo ágil, tales como XP[8], Scrum, entre otras. Estas definen lo que se debe construir en el proyecto de software, tienen una prioridad

[8] de sus siglas en inglés, eXtreme Programming, que significa en español programación extrema

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.

Por lo general, se espera que una estimación de tiempo de cada historia de usuario se sitúe entre horas y el tiempo máximo de iteración. Estimaciones superiores a este tiempo máximo son indicativas de que la historia es muy compleja y debe ser dividida en varias historias.

Una historia de usuario es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario[29]. Ellas son utilizadas para la especificación de requisitos acompañadas de las discusiones con aquellos y las pruebas de validación.

Cada historia de usuario debe ser limitada. La metodología estipula que las mismas deben ser escritas por los clientes. Son una forma rápida de administrar los requisitos sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos.

Las historias de usuario deben ser:

- **Independientes:** de ser necesario, combinar las historias dependientes o buscar otra forma de dividir las historias de manera que resulten independientes.
- **Negociables:** la historia en sí misma no es lo suficientemente explicita para considerarse un contrato, la discusión con los usuarios debe permitir esclarecerse y éste debe dejarse explicito bajo la forma de pruebas de validación.
- **Valoradas:** Los intereses de los clientes y de los usuarios no siempre coinciden, pero en todo caso, cada historia debe ser más importante para los clientes que para el desarrollador.
- **Pequeñas:** las historias grandes son difíciles de estimar e imponen restricciones sobre la planificación de un desarrollo iterativo. Generalmente se recomienda la consolidación de historias muy cortas en una sola historia.
- **Estimables:** las historias completas en requerimientos de parte del equipo de desarrollo y de parte del cliente son estimables. Y por lo tanto, el equipo debe estar cómodo de puntuar las historias de usuario.
- **Verificables:** las historias de usuario cubren requerimientos funcionales, por lo generalmente son verificables. Cuando sea posible, la verificación debe automatizarse, de manera que pueda ser verificada en cada entrega del proyecto.

Las iniciales de estas características, con sus nombres en inglés, forman la palabra INVEST, que significa “inversión”. Es porque toda historia de usuario, si se construye bien, es una inversión.

Al momento de implementar las historias, los desarrolladores deben tener la posibilidad de discutirlas con los clientes. El estilo sucinto de las historias podría dificultar su interpretación, podría requerir conocimientos de base sobre el modelo, o podría haber cambiado desde que fue escrita.

Cada historia de usuario debe tener en algún momento pruebas de validación asociadas, lo que permitirá al desarrollador, y más tarde al cliente, verificar si la historia ha sido completada. Como no se dispone de una formulación de requisitos precisa, la ausencia de pruebas de validación concertadas abre la posibilidad de discusiones largas y no constructivas al momento de la entrega del producto.

2.10.2. 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[30].

El uso de las épicas en proyectos de gran tamaño ayuda a organizar tareas complejas en un tipo de estructura para que la interrelación de historias de usuario esté bien entendida. Por lo tanto, el diseño de épicas en el proceso de desarrollo es fundamental antes de comenzar cualquier proyecto.

2.11. 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[31]. 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[32].

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[9] 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[33].

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

Capítulo 3

Estado del arte

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[34].

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[35].

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.

El módulo de Curriculum dispone de cuatro componentes:

- Un listado de cursos donde los usuarios pueden encontrar información sobre el curso. También pueden iniciar, editar y presentar propuestas de cursos.
- Una interfaz de los usuarios autorizados a opinar, anotar, rechazar, y aprobar las propuestas.
- Un formulario de curso, un formulario dinámico con campos inteligentes y el formato de respuesta que optimiza la experiencia del usuario.
- Un formulario de los programas, utilizando la misma funcionalidad que los formularios de curso para proponer y mantener la información y los requerimientos para los programas y grados.

3.3. DECA: Curriculum Navigator

DECA ofrece, mediante su módulo de Curriculum Navigator, una solución de desarrollo y administración de Curriculum[36].

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.

Curriculum Navigator ofrece:

- El acceso a un repositorio de datos curricular.
- Editar, guardar y proponer planes de estudio o carreras de grado.
- Historial de cambios y de accesos de revisión.
- Seguimiento en tiempo real de propuestas abiertas.

3.4. Comparación entre plataformas

Una investigación para comprobar otros proyectos o productos con las mismas características propuestas, buscando innovación para el mercado es expuesta en la tabla 3.1.

TABLA 3.1: Relación entre sistemas de gestión curricular.

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.		✓	✓

3.5. Relevancia del módulo curricular

Frecuentemente, descrito como complejo e ineficaz, los métodos tradicionales basados en papel de gestión 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. Además, busca eliminar esta complejidad al acelerar el desarrollo curricular y el proceso de aprobación.

La importancia del módulo reside en la posibilidad de automatizar 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. Sin embargo, hay alternativas que buscan solucionar la misma problemática pero no existe alternativa que pueda soportar el uso de competencias ni que pueda comunicarse con un sistema de gestión de evaluación basadas en competencias.

Como se habló en la sección 2.5.1 de proceso curricular; 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 debe encargarse de cargar uno por uno las nuevas entidades aprobadas y publicadas por el comité curricular.

En la propuesta de solución del capítulo 5 hablaremos de como el proyecto final busca solucionar la problemática y unir los procesos de los cuales hablamos en el párrafo anterior.

Capítulo 4

Propuesta de solución

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 diseñan modelos donde se propone el módulo a ser desarrollado (figura 4.1) y se busca unir procesos separados del diseño curricular (figura 2.2) con el flujo de agregar las competencias, cursos, y programas al AMS (figura 2.3).

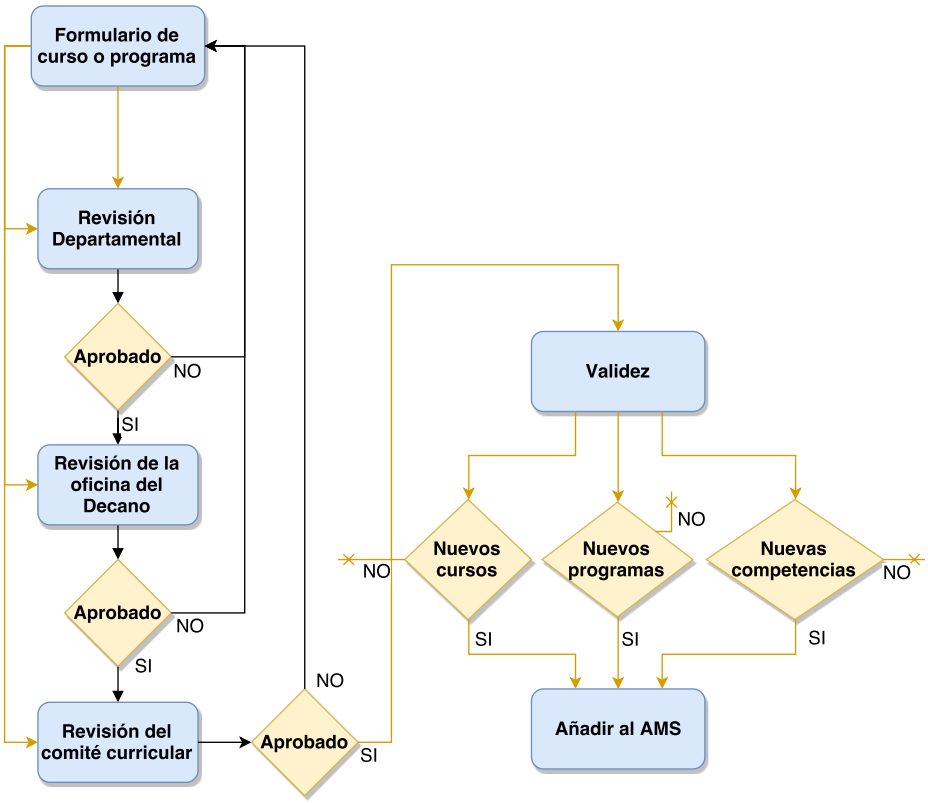


FIGURA 4.1: Modelo propuesto del módulo curricular adherido a un sistema de gestión de evaluaciones basadas en competencias.

Actualmente, cuando 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 propuesto se encargará de 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.

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 a ser automatizado por el módulo ya que hoy día dicha creación se hace de manera manual, como se aprecia en la figura 2.3.

Se agrega también 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.

4.1. Modelo de arquitectura de módulo

El proyecto final (figura 4.2) fue diseñado como módulo de un AMS utilizado en universidades del estado de California.

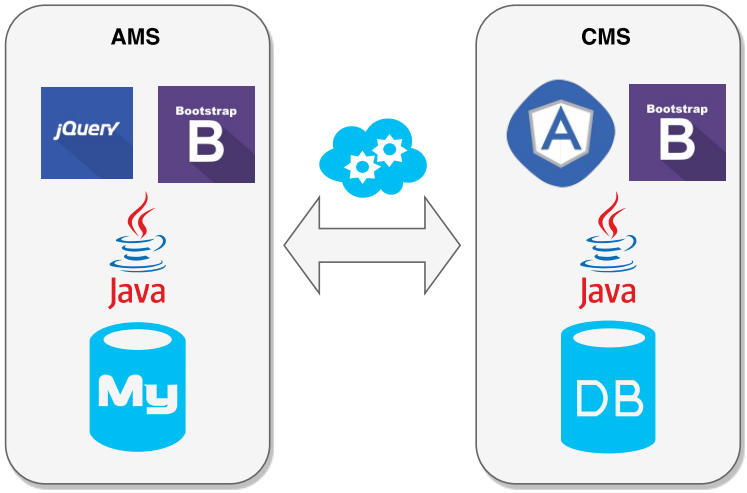


FIGURA 4.2: Arquitectura del módulo curricular.

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 va a utilizar 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[1] 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 hacía aún más atractivo.

4.2. Requerimientos no funcionales

Se ha realizado una breve revisión de las tecnologías y abordajes brindadas como requerimientos funcionales. Sin embargo, al utilizar un abordaje ágil, la validación de las decisiones tecnológicas depende en ultima instancia de la validación del proceso de desarrollo realizada por expertos y usuarios.

Las decisiones de tecnologías para el módulo de gestión curricular fueron basadas en los conocimientos adquiridos de los miembros del equipo de desarrollo, con el fin de optimizar tiempos utilizados en curvas de aprendizaje. Sin embargo, se hicieron análisis previos a su uso para corroborar que dichas tecnologías cumplen con el propósito de desarrollo.

4.2.1. Java

La elección del lenguaje de programación, establecida por la organización, fue utilizada como lenguaje para la lógica del módulo curricular ya que facilita la integración con el código ya existente. El equipo de desarrollo posee conocimiento en este lenguaje de programación o en lenguajes orientados a objetos, por lo que se aprovechó el tiempo que pudo haber sido utilizado en curvas de aprendizaje del lenguaje para investigar buenas prácticas para el proyecto.

Java fue diseñado para alcanzar los desafíos del desarrollo de aplicaciones en el contexto de ambientes heterogéneos y distribuidos en red[37]. Lo más importante de solucionar entre estos desafíos era la entrega segura de aplicaciones que consumen lo mínimo de recursos del sistema, correr en cualquier plataforma de hardware y/o software, y que pueda ser extendido de manera dinámica[38].

Operar en múltiples plataformas con redes heterogéneas invalida la arquitectura tradicional de distribución de binarios, «release», actualización, parcheo, etc[39].

Hoy día, Java es uno de los lenguajes de programación más utilizados a nivel mundial debido a su portabilidad y evolución con el paso del tiempo, como se observa en la figura 4.3. Además, al ser un lenguaje de programación multiplataforma cualquier proyecto se puede desarrollar en cualquier

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

sistema operativo o plataforma para luego ser levantada en el servidor independiente a su plataforma.

El lenguaje es reconocido por ser intuitivo a la hora de programar, altamente portátil y portable[40]. Además, la comunidad es uno de los fuertes de Java. Además, en la Web hay una gran cantidad de foros y librerías de la comunidad para resolver diferentes problemáticas de los desarrolladores.

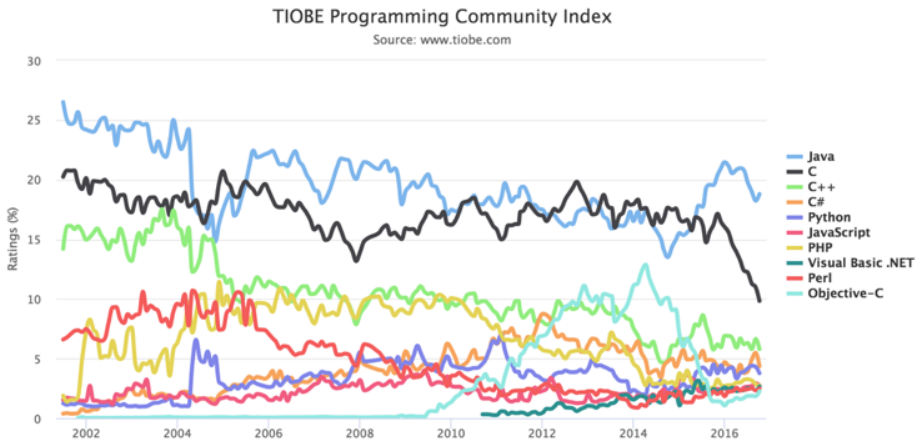


FIGURA 4.3: Gráfico de uso de lenguajes de programación con respecto al tiempo.

El lenguaje Java proporciona un nivel de rendimiento adecuado para la plataforma donde se aloja el AMS.

Dicho uso del lenguaje fue validado durante el proceso de desarrollo al poder resolver los criterios de aceptación de las historias de usuario.

4.2.2. MySQL

Para ayudar a que el AMS sea adaptable y fácil de mantener, se extenderá la base de datos ya utilizada por el AMS. Por lo tanto, la elección de la base de datos del módulo queda a criterio de la organización y como requisito no funcional para el proyecto final. El sistema de base de datos que se utiliza para el sistema de evaluación de competencias es MySQL, por ser un sistema open source y con una de las comunidades más grandes entre las bases de datos existentes.

MySQL es el sistema de base de datos open source más popular disponible. Es particularmente eficaz para sitios web públicos que requieren base de datos rápidas y estables[41]. Para añadir, acceder y procesar datos almacenados en una base de datos informática se necesita de un sistema de administración de Base de Datos como MySQL Server.

Como las computadoras son muy buenas manejando grandes cantidades de datos, los sistemas de administración de base de datos juegan un rol principal en la computación como utilidades autónomas o como partes de otras aplicaciones.

Para representar los datos que se almacenan, utiliza el modelo lógico de datos relacional donde guarda sus datos en tablas separadas, antes que consolidar todos los datos en un solo lugar. La estructura de la base de datos está organizada en archivos físicos optimizados para mayor velocidad[42].

El modelo lógico con objetos tales como bases de datos, tablas, vistas, filas, y columnas, ofrecen un ambiente de programación flexible donde se establecen reglas que gobiernan las relaciones entre las de los distintos tipos de campos, tales como uno a uno, uno a muchos, únicos, requeridos, u opcionales y punteros entre diferentes tablas.

En las siguientes figuras 4.4 y 4.5 podemos observar que MySQL es un sistema de manejo de base de datos muy utilizado por la comunidad y va en aumento de popularidad hasta casi alcanzar a uno de los gigantes que es Oracle.

Rank			DBMS	Database Model	Score		
Jun 2017	May 2017	Jun 2016			Jun 2017	May 2017	Jun 2016
1.	1.	1.	Oracle	Relational DBMS	1351.76	-2.55	-97.49
2.	2.	2.	MySQL	Relational DBMS	1345.31	+5.28	-24.83
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1198.97	-14.84	+33.16
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	368.54	+2.63	+61.94
5.	5.	↓ 4.	MongoDB	Document store	335.00	+3.42	+20.38
6.	6.	6.	DB2	Relational DBMS	187.50	-1.34	-1.07
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	126.55	-3.33	+0.32
8.	8.	↓ 7.	Cassandra	Wide column store	124.12	+1.01	-7.00
9.	9.	↑ 10.	Redis	Key-value store	118.89	+1.44	+14.39
10.	10.	↓ 9.	SQLite	Relational DBMS	116.71	+0.64	+9.92

FIGURA 4.4: Gráfico que muestra el puntaje de uso de motores de base de datos.

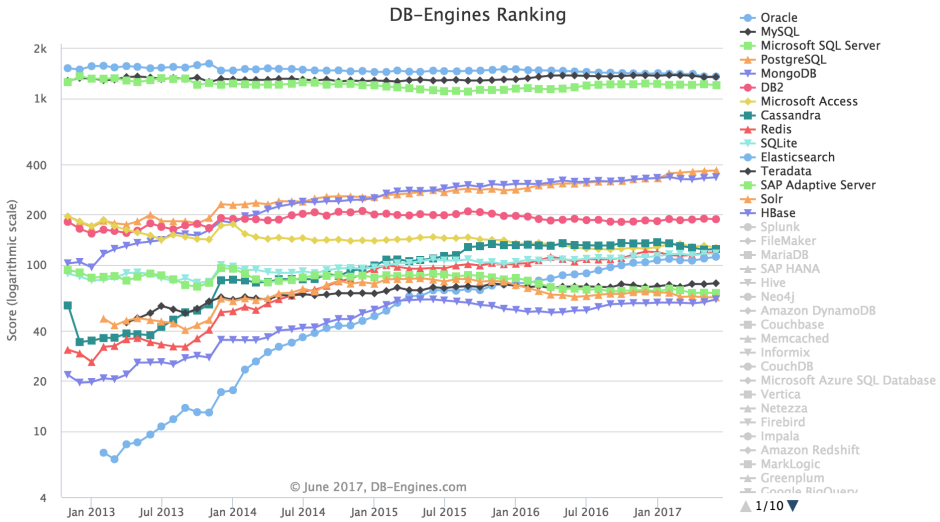


FIGURA 4.5: Gráfico de uso de motores de base de datos con respecto al tiempo.

Como el motor de base de datos es otro requerimiento no funcional, como es otra tecnología utilizada para la aplicación base. Sin embargo, se hizo un breve análisis de vulnerabilidades para verificar que el uso de MySQL

como base de datos relacional podría ser efectivo para el proyecto final, además, los desarrolladores están familiarizados con la misma.

4.2.3. Amazon Web Services

Amazon Web Services, o más conocida como AWS, es una plataforma de servicios web que ofrece soluciones de procesado, almacenamiento y redes en diferentes capas de abstracción. Se pueden usar estos servicios para hospedar sitios web, correr aplicaciones complejas y para minería de grandes cantidades de datos[43].

La interfaz web puede ser manejada por máquinas o por usuarios por medio de una interfaz de usuario gráfica. Los servicios más utilizados son EC2 en la cual ofrece servidores virtuales, y S3 que es utilizada para almacenamiento.

AWS es una nube pública, donde tiene las siguientes clasificaciones como nube:

- **Infraestructura como Servicio:** más conocida como IaaS, ofrece recursos fundamentales como procesamiento, almacenamiento, y capacidades de servicios, utilizando servidores virtuales tales como Amazon EC2, Google Compute Engine y Microsoft Azure en las máquinas virtuales.
- **Plataforma como Servicio:** más conocida como PaaS, provee plataformas para desplegar aplicaciones customizadas a la nube, tales como AWS Elastic Beanstalk, Google App Engine, y Heroku.
- **Software como Servicio:** más conocida como SaaS, combina infraestructura y software corriendo en la nube, incluyendo aplicaciones de oficina como Amazon WorkSpaces, Google Apps for Work, y Microsoft Office 365.

En la figura 4.6 se puede mostrar como AWS lidera entre las alternativas del mercado para soluciones de computación en la nube. Luego, lo sigue Azure de Microsoft como siguiente alternativa más utilizada. AWS y Azure son líderes en opciones de cloud debido a su constante innovación en el mercado como se puede apreciar en la figura 4.7.

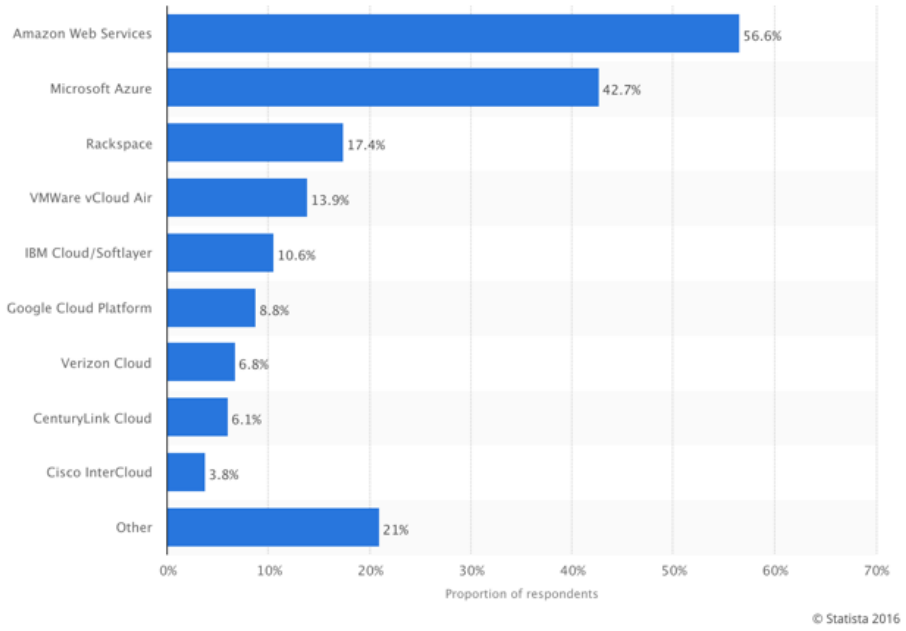


FIGURA 4.6: Ranking de compañias que brindan servicios de cloud computing.

FIGURA 4.7: Figura de Gartner que muestra las alternativas de cloud computing..

El AMS se encuentra alojado en la plataforma de servicios AWS. Por lo tanto, se utilizaron los servidores ya en línea para correr el módulo curricular para la aplicación.

4.2.4. Git

Como un equipo de desarrollo requiere de un sistema de control de versiones eficiente, se utilizó Git por ser el lider en VCS [44]. Para mejorar la integración del código que produce el equipo, la organización utiliza un repositorio en «GitHub»[2]. Además, el uso de un sistema de versionamiento permite minimizar y optimizar el tiempo de unión de módulos que van agregando o actualizando los miembros del equipo de desarrollo.

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de esta forma permite recuperar versiones específicas más adelante[45].

Git modela sus datos más como un conjunto de instantáneas de un pequeño sistema de archivos. Cada vez que se confirma el cambio de un archivo, o se guarda el estado de un proyecto se hace una foto del aspecto de todos los archivos en ese momento y guarda una referencia a esa instantánea. Para ser eficiente, si los archivos no se han modificado, Git no almacena el archivo de nuevo, solo un enlace al archivo idéntico anterior que ya tiene almacenado. Este comportamiento se puede observar en la figura 4.8.

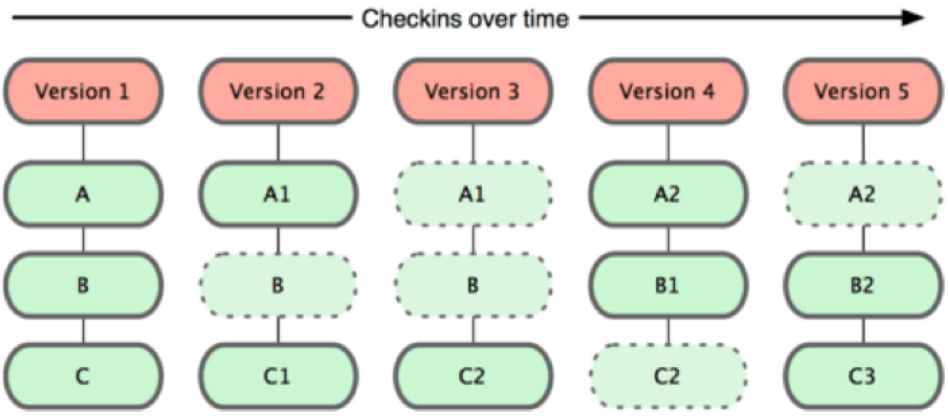


FIGURA 4.8: Flujo de versiones de Git.

Uno de los fuertes importantes de Git como VCS[3] es su integridad, debido a que toda versión es verificada mediante una suma de comprobación[4] antes de ser almacenada, y es identificada a partir de ese momento dicha suma. Esta suma es utilizada para volver a un estado anterior del repositorio en caso de ser necesario o ver cuáles fueron los cambios que se realizaron en ese pedazo de código trabajado o más conocido como commit[45].

[2]Plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.

[3]de sus siglas en inglés, Version Control System, que significa en español sistema de control de versionamiento.

[4]También conocida como checksum.

4.2.5. Spring

El AMS cuenta con un framework propio desarrollado internamente, el cual esta deprecado y uno nuevo en el cual se estuvo trabajando para los nuevos módulos que se fueron desarrollando. El mismo es Spring MVC y fue utilizado para la parte lógica de la aplicación.

Spring es un framework que facilita el desarrollo de aplicaciones escritas en Java. El propósito de Spring es manejar la infraestructura de las aplicaciones utilizando el método de inversión de control. Es por ello, que el programador se encargará de programar la lógica de negocio usando objetos simples de Java o POJOs (Plain Old Java Objets) y Spring se encargará de añadir las capacidades de empresa o J2EE a nuestra aplicación[46].

Spring tiene las siguientes características:

- Simplicidad y acoplamiento débil donde permite programar Java de manera sencilla. Busca ser simple y se basa en la inyección de dependencias para obtener un acoplamiento débil.
- Funciona como contenedor ya que gestiona el ciclo de vida de los objetos y como se relacionan entre ellos. Proporciona una gran infraestructura que permite que el programador se dedique a la lógica de la aplicación.
- Ligero porque es muy rápido en tiempo de procesamiento y no es invasivo a la hora de programar.
- Orientado a aspectos, lo que permite facilitar una capa de servicios que son ideales para este tipo de programación como auditoría, o gestión de transacciones.

Spring se utiliza en el proyecto como framework para toda la aplicación, por lo tanto, usar Spring es también un requerimiento no funcional.

4.2.6. AngularJS

Para empezar a trabajar con la interfaz de usuario se hizo un estudio previo de las ventajas entre Frameworks como JQuery y AngularJS, debido a que la organización que brinda el AMS dio total libertad a la hora de elegir cual utilizar. Por lo tanto, la elección del framework Javascript del módulo de Curriculum entra como decisión de diseño para el proyecto final.

Con la llegada de los framework de Javascript, tales como JQuery, las páginas web ya no tenían la necesidad de volver a renderizar las páginas cada vez que se necesitaba información del servidor, ya que con la aparición de las llamadas asíncronas se ha logrado mejorar la experiencia del usuario[47].

JQuery ha hecho un excepcional trabajo al proveer de herramientas que manipulen el DOM de una página, pero no ofrece una guía real de cómo organizar el código en la estructura de la aplicación. Ante la desesperada búsqueda de escribir aplicaciones grandes y fáciles de mantener en Javascript ha dado a luz a un renacimiento de frameworks de Javascript, entre ellos se encuentra el framework de Google más conocida como AngularJS.

AngularJS es un framework de aplicaciones web de código abierto que ofrece a un desarrollador una base estable de código con una comunidad enorme y un entorno rico de librerías hechas por la comunidad[48].

4.3. 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 eso no significaba que esa persona debía hacer sola la implementación.

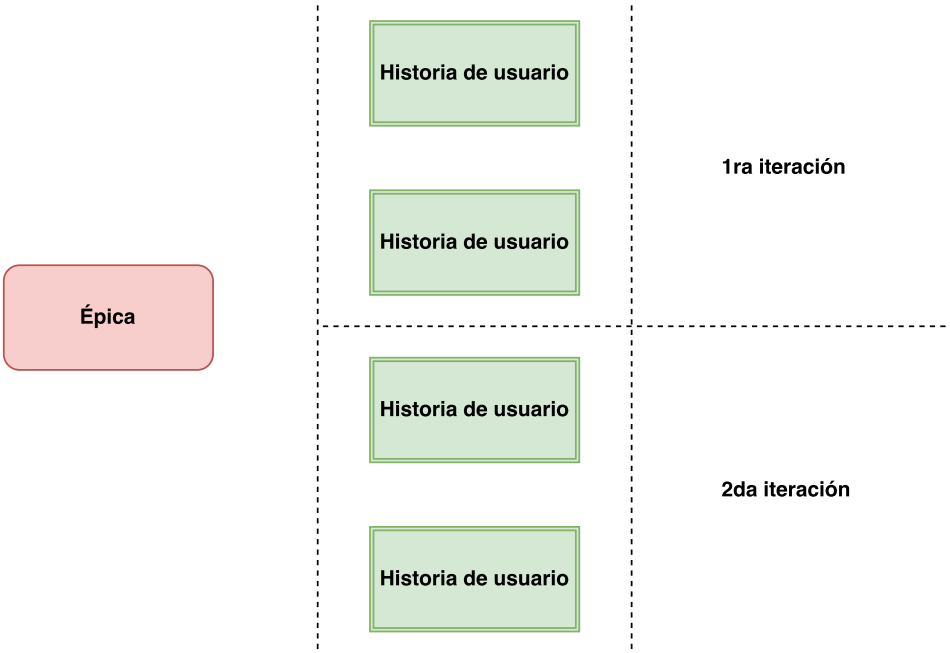


FIGURA 4.9: Diagrama de definición de épicas en la metodología ágil

El PO[5] 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.

En el caso de estudio, cada sprint consta de 2 semanas de trabajo, donde los desarrolladores como equipo se comprometen a entregar cierto valor de

[5]de sus siglas en inglés, Product Owner, que significa en español dueño del producto.

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[6].

En la figura 4.10 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.

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.

4.3.1. SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

[6]Bolsa de historias de usuarios pendientes.

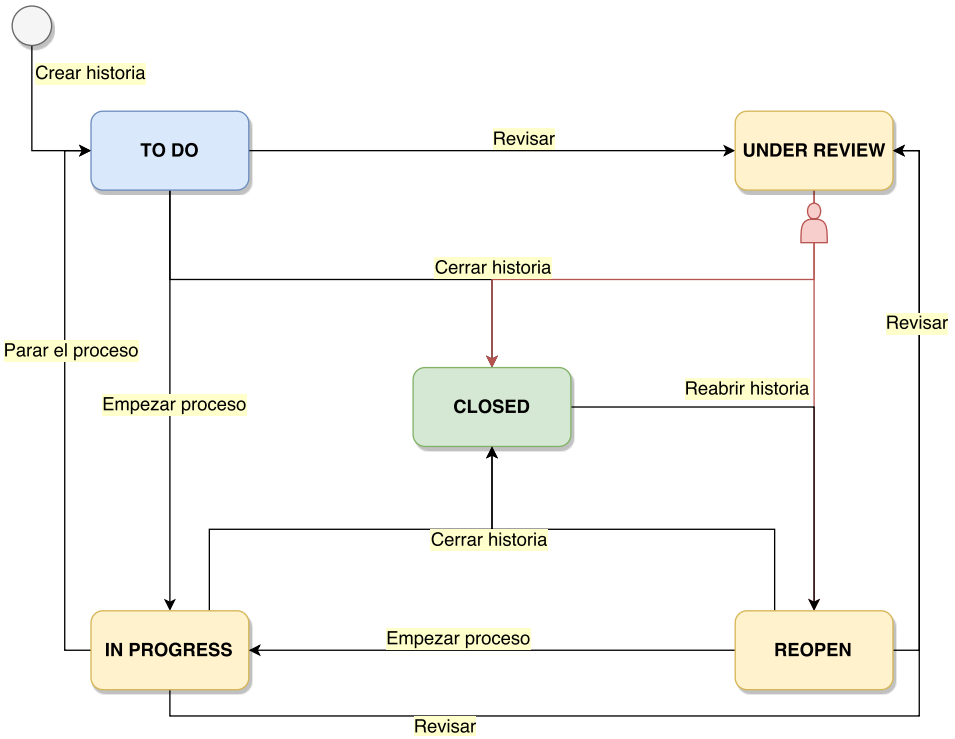


FIGURA 4.10: Flujo de desarrollo de historias de usuario.

En Scrum 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.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

4.3.2. Proceso

En Scrum un proyecto se ejecuta 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[29]. 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.

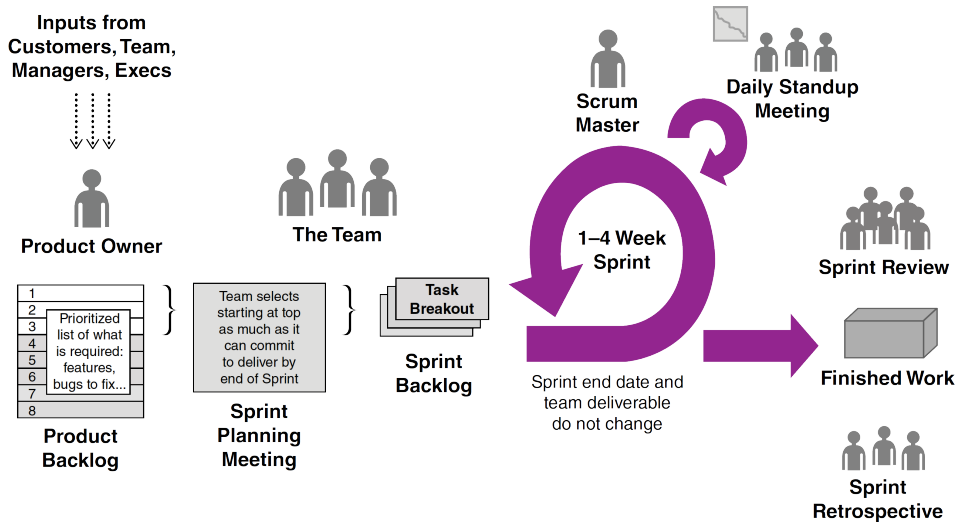


FIGURA 4.11: Flujo de la técnica SCRUM.

4.3.3. 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.

4.3.4. 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.

4.3.5. 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.

Capítulo 5

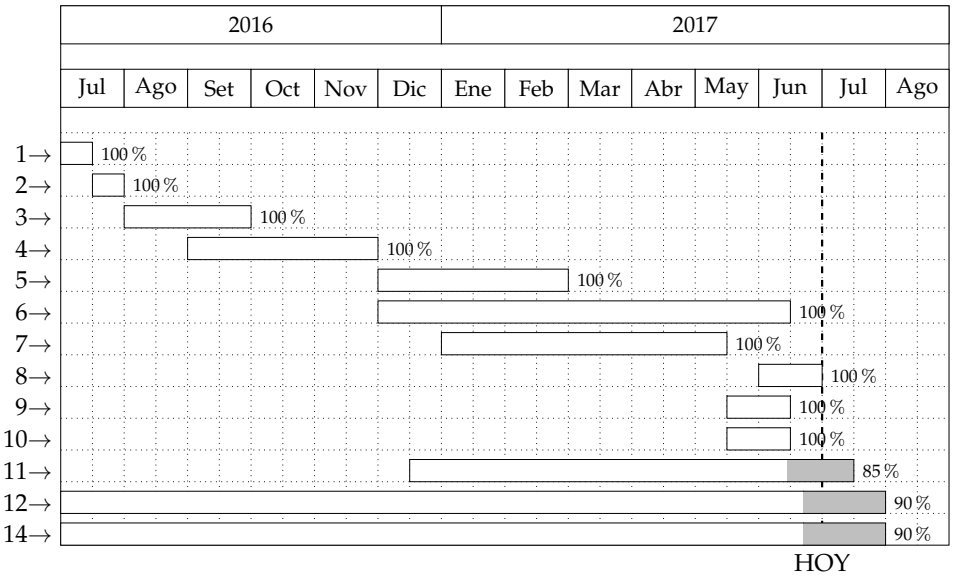
Desarrollo de la aplicación

En éste capítulo presentamos el proceso de desarrollo del módulo de gestión de programas orientado a competencias. Comenzamos listando las épicas y brindando una breve explicación de cada una de ellas.

La validación de las historias de usuaro desarrolladas es de particular importancia. Si bien la validación se realizó de manera iterativa al final de los sprints, las presentamos en este escrito aparte, en el capítulo 7.

Épicas

1. Análisis de herramientas utilizadas y posibles potenciales para el desarrollo del módulo.
2. Diseño de modelo de datos para versionamiento de competencias, cursos y programas.
3. Desarrollo de flujo de trabajo para el versionamiento de competencias.
4. Desarrollo de buzón de entrada para evaluadores y colaboradores del flujo de trabajo.
5. Versionamiento encadenado de evaluaciones debido al versionamiento de competencias.
6. Desarrollo de flujo de trabajo para el versionamiento de cursos.
7. Desarrollo de flujo de trabajo para el versionamiento de programas de estudio.
8. Soporte de etapas en los flujos de trabajo.
9. Desarrollo de reportes de versiones de cursos.
10. Actualizar el AMS para que tome las versiones de competencias cursos y programas correspondientes en los periodos.
11. Retoques finales.



5.1. Conformación del equipo de trabajo

La primera etapa del proyecto fue realizar una encuesta a los posibles integrantes del nuevo equipo de desarrollo donde se relevaron las capacidades adquiridas en cuanto a lenguajes de programación y tecnologías utilizadas, como así también de los conocimientos de dominio de la aplicación.

Dicha encuesta tiene como propósito permitir una mejor organización de los miembros de equipos y de esta manera una distribución eficaz de conocimientos y dominio de la aplicación para resolver las diferentes problemáticas que podría afectar al módulo currilar.

Una vez formado lo que sería el equipo de desarrollo se procedió a hacer análisis de las herramientas que podrían resolver la problemática entre ellas las que eran tomadas como requerimientos no funcionales para el módulo de gestión curricular del capítulo 4.2.

5.2. Diseño de modelo de datos para versionamiento

Al iniciar el proceso de desarrollo se debía iniciar un «spike» para buscar la manera de modelar los datos y tablas ya existentes en el sistema de competencias, cursos, y programas.

Un «spike» es un término que se utiliza en el desarrollo Ágil para incluir una tarea en un sprint que no pertenece necesariamente a una historia de usuario. Es necesario ya que sirve para incluir tareas que ayudan de alguna manera en el futuro desarrollo de las historias de usuario, pero que no implica directamente un incremento al producto que se está desarrollando ni un valor de negocio para el cliente[49].

Se diseñó de una manera que fuera lo más general posible en caso de que cualquier otra entidad se decida versionar en el futuro. En futuras iteraciones se llegó a la conclusión que la idea fué acertada, debido a que se abarcaría el versionamiento de evaluaciones de manera automática por el sistema (sección 5.5.1) y la misma contiene una de las entidades versionables que es la de competencias.

TABLA 5.1: Historias de usuario para el diseño de modelo de datos para versionamiento

Historias de usuario	HE	HC	PH	Sprints
Diseño del modelo de versionamiento para competencias, cursos, y programas.	61	61	5	1

5.2.1. Diseño del modelo de versionamiento para competencias, cursos y programas

Al iniciar con las historias de versionamiento, se definió una tarea que tenía como propósito principal el diseño de una lógica de negocios que permita adaptar las tablas existentes de las competencias, evaluaciones, cursos, y programas para que soporten una revisión o versionamiento de sus registros.

Como resúmen de actividades del modelo desarrollado (figura 5.1) se puede resaltar lo siguiente:

- Cada tabla de cualquier entidad posee un identificador único. Las tablas entidades versionables son las de competencias, cursos, programas, y evaluaciones.
- Se decidió agregar una nueva columna «entity_atid» que tiene como propósito el de apuntar al origen de la versión. Por ejemplo; si el usuario crea un nuevo curso para el año lectivo, este curso tiene su identificador «course_id» y su «course_atid» apuntando a su mismo identificador por ser el origen para las versiones posteriores. Luego, se crea una nueva versión para el año posterior, esta nueva versión tiene su propio identificador pero su campo denominado como «course_atid» que apunta al primer curso creado u origen.
- Para hacer más sencilla la búsqueda de competencias, cursos, programas o evaluaciones actuales se agregó un campo a cada tabla identificando los actuales. Este campo denominado «is_current» o "es actual" es una bandera que indica la validez del registro.
- Además de registrar el origen, se registra la versión previa o de donde parte el registro con el campo «previous_entity_id».
- Como cada registro de cualquier tabla ahora tiene un periodo de validez, se diseñaron tablas de relación entre cada tabla y la tabla de periodos lectivos denominada como «calendar». Por ejemplo; «slo_term_rel» para las competencias, «new_course_term_rel» para los cursos, «asmt_term_rel» para las evaluaciones y «credential_term_rel» para los programas.

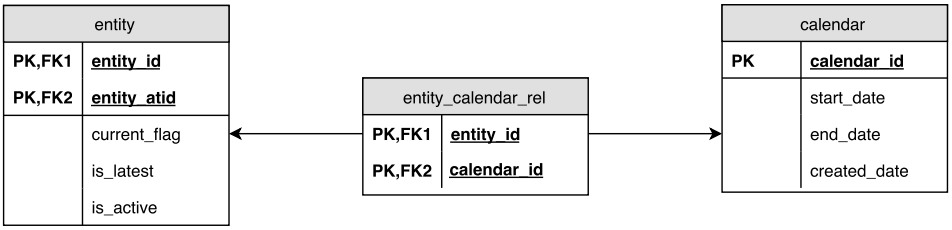


FIGURA 5.1: Modelo de datos para el versionamiento de competencias, cursos y programas.

5.3. Flujo de trabajo para el versionamiento de competencias

TABLA 5.2: Historias de usuario para el flujo de trabajo para el versionamiento de competencias

Historias de usuario	HE	HC	PH	Sprints
Versionamiento de competencias	148	170	13	3
Flujo de trabajo simple	78	78	8	1
Aprobar pasos del flujo de trabajo	44	44	5	1
Rechazar pasos del flujo de trabajo	52	53	5	1

5.3.1. Versionamiento de competencias

Esta historia de usuario tenía como descripción: «*Como encargado del sistema de gestión de competencias, me gustaría ser capaz de versionar competencias con la finalidad de que se puedan redefinir competencias con el paso del tiempo, sin perder datos de corrección de las mismas*».

Algunas tareas que se definieron en la historia de usuario son las siguientes:

- Investigar y diseñar el versionamiento de las competencias. La misma fue desarrollada de manera en que toda competencia versionada apunta al origen y el origen se apunta a sí mismo, de esta manera se puede saber la familia de versiones de una competencia.
- Actualizar todos esos lugares de la aplicación que listan las competencias, donde solamente deberían traer las competencias actuales.
- Manejar la distribución de competencias a periodos futuros, de manera que una competencia no pueda ser distribuida a periodos en las que no tiene validez.
- Diseñar y mantener pruebas automatizadas.

Fue desarrollado durante tres iteraciones con un total de 170 horas cargadas en el sistema, debido a la complejidad a la hora de migrar los datos ya existentes de todas las universidades y por la cantidad de servicios que debían ser modificados.

5.3.2. Flujo de trabajo simple

En la siguiente historia de usuario inicia el proceso de creación de flujos de trabajo donde las plantillas de los mismos pueden ser creados, editados, y eliminados por el administrador encargado de la aplicación de cada universidad. Para esta historia se debe diseñar y desarrollar las plantillas de manera que el administrador pueda agregar los diferentes pasos del flujo de trabajo si así lo decide en el futuro. Inicialmente se considera un solo paso para la iteración inicial de desarrollo.

Esta historia de usuario tenía como descripción: «*Como coordinador del AMS, me gustaría ser capaz de crear flujos de trabajo simples para administrar la aprobación de revisiones de competencias y que se pueda tener un mejor manejo de las creaciones y aprobaciones de las mismas en el campus*».

La historia tiene los siguientes criterios de aceptación:

- Diseñar e implementar plantillas de flujo de trabajos simples para creación y revisión de todos los niveles de competencias.
- Diseñar e implementar un flujo de trabajo simple sin aprobación por parte de evaluadores, donde el iniciador del flujo puede revisar y aprobar su propio formulario.
- La plantilla de flujo de trabajo simple debe soportar el uso de pasos personalizados.
- El que inició el flujo es el único que puede llenar los campos del formulario.

El usuario debe ser capaz de agregar pasos personalizados para la plantilla de flujo de trabajo de la institución. Estos pasos personalizados son pasos que puede diseñar el usuario, donde puede colocar una pregunta como título y por cada título tiene un campo que puede llenar el usuario. Por lo general, un paso personalizado puede tener una o más preguntas definida por el usuario.

En los mockups entregados para el desarrollo se contemplan trabajos futuros donde cada paso tiene que ser aprobado por un rol del AMS, donde cualquier usuario con dicho rol puede aprobar o rechazar el flujo de trabajo con solo rechazar uno de los pasos.

Además, se da inicio al desarrollo de plantillas de flujos de trabajo con las competencias, donde se podía asignar un tipo de flujo para cada plantilla ya sea de creación o versionamiento de los diferentes niveles de competencias.

Como las plantillas era una funcionalidad conocida y utilizada en otra parte de la aplicación, se imitó el comportamiento de la misma utilizando las mismas tablas para el almacenamiento de los datos en la base de datos relacional como requerimiento no funcional de la organización. Se diseñaron las nuevas pantallas con la definición de las plantillas de flujo de trabajo y también la pantalla para listar las mismas. En la misma el usuario administrador puede crear, editar si aún no ha sido usada, eliminar, y clonar plantillas.

El flujo simple de competencias consta con campos para agregar nombre, descripción, y periodo donde empieza a ser válido. Se empezó con el desarrollo del flujo para competencias como era el proceso de desarrollo inicial más simple, ya que la información para la creación de competencias en el sistema sin el módulo curricular solo requería de un nombre para la competencia en el nivel en la que se está creando. Además, los flujos simple también pueden ser personalizados, la única diferencia que adquiría es que se agregan campos personalizados para la definición del formulario institucional, para que puedan responder los que proponen la nueva competencia.

La historia de usuario fue desarrollada durante una iteración con un total de 78 horas cargadas en el sistema.

5.3.3. Aprobación de pasos completados de flujos de trabajo

Luego de la historia en la que se diseñaron las plantillas y fue desarrollado un flujo de trabajo simple inicial para creación o versionamiento de competencias, el siguiente paso es que un usuario designado desde la plantilla pueda iniciar y otro pueda aprobar el proceso de creación o revisión de competencias de cualquier nivel.

Esta historia de usuario tenía como descripción: «*Como evaluador de un flujo de trabajo, me gustaría un simple proceso paso por paso en el que pueda revisar y/o aprobar competencias de manera sencilla e intuitiva*».

La historia de usuario tiene los siguientes criterios de aceptación:

- Soporte de asignaciones de tareas de creación y revisión por roles del AMS en las plantillas de flujos de trabajo.
- Diseño e implementación de vista de revisión para el flujo de trabajo.

Cada paso del flujo de trabajo debe estar terminado para que pase a la etapa de revisión por parte de los encargados. Luego de enviar el formulario, cada rol debe hacer su revisión para que el sistema pueda agregar la nueva competencia.

Como en las reuniones de demostración de cada sprint se notaban ciertos aspectos de las historias de usuario que no llenaban las expectativas de los clientes, los desarrolladores decidieron diseñar maquetas de pantallas que mostraban el posible diseño de la página. Luego de recibir feedback de parte de los clientes, se empezaba a desarrollar las nuevas pantallas. Finalizando la historia de usuario con pruebas automatizadas.

La historia de usuario fue desarrollada durante una iteración en un periodo de tiempo de 44 horas cargadas en el sistema.

5.3.4. Rechazar pasos completados del flujo de trabajo

Esta historia de usuario tenía como descripción: «*Como evaluador de flujos de trabajo, me gustaría ser capaz de rechazar partes de los mismos y poder dar feedback a partes que no cumplen con nuestros estándares*».

– Diseño e implementación de funcionalidad de rechazo de pasos en los flujos de trabajo.

– Diseño e implementación de funcionalidad de retroalimentación de parte de los evaluadores y encargados.

Esta funcionalidad tiene como propósito permitir a la persona que hace la revisión de los pasos rechazar y dejar feedback para que se puedan hacer los cambios correspondientes. Cuando se rechaza un paso, se rechaza el flujo de trabajo, y por lo tanto vuelven a estar activos los campos para que se hagan los cambios correspondientes.

El trabajo se inició la actualización del modelo de base de datos actual, luego de crear las clases correspondientes en el código para su utilización. Luego, se actualizaron las páginas donde el usuario puede aprobar los pasos para que soporte rechazar pasos y poder así dejar algunos comentarios.

La historia de usuario fue desarrollada durante una iteración en un periodo de tiempo de 53 horas cargadas en el sistema.

5.4. Buzón de entrada para evaluadores y colaboradores del flujo de trabajo

TABLA 5.3: Historias de usuario para el buzón de entrada para evaluadores y colaboradores del flujo de trabajo

Historias de usuario	HE	HC	PH	Sprints
Buzón de entrada de flujos de trabajo	48	56	5	2
Notificaciones con soporte a etapas	32	28	5	1

5.4.1. Buzón de entradas de flujos de trabajo

La siguiente historia tiene como propósito mostrar a cada usuario la lista de workflows pendientes que requiere de su aporte. Además de adaptar el nuevo buzón de entrada para otros rasgos de la aplicación como son las evaluaciones, los planes de acción y preguntas de parte del usuario a profesores.

Tiene como descripción lo siguiente «*Como aprobador de eLumen, me gustaría una vista unificada de los workflows que tengo que revisar – además de mis evaluaciones, planes de acción y mis preguntas a profesores – para que no vaya cazando workflows por la aplicación*»

Como criterios de aceptación se encuentran los siguientes:

– Diseño e implementación de buzón de entrada para flujos de trabajo.

– Diseño e implementación de buzón de entrada para planes de acción.

– Diseño e implementación de buzón de entrada para pedidos de profesores.

La historia fue finalizada en dos iteraciones con una cantidad de 56 horas cargadas en el sistema.

5.4.2. Notificaciones con soporte a etapas

La historia de usuario tiene como descripción lo siguiente «*Como presidente curricular, me gustaría que el equipo de diseño y revisión curricular reciban notificaciones cuando tengan alertas de deuda de trabajo (y alertas cuando pase el tiempo), para que se puedan manejar mejor de esa manera los procesos curriculares*».

Como criterios de aceptación se encuentran los siguientes:

- Establecer notificaciones cuando las partes del flujo de trabajo son asignadas a los roles de las personas.
- Establecer notificaciones de alerta a asignaciones de partes y etapas. Por ejemplo, 5 días después de su asignación.
- Mandar notificaciones por mail.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Diseñar e implementar nuevos modelos de datos que permitan soportar el uso de roles para creadores y editores de partes.
- Actualizar el sistema de notificaciones del AMS.
- Diseñar e implementar la página de configuración de notificaciones.

La historia fue finalizada en tres iteraciones con una cantidad de 60 horas cargadas en el sistema.

5.5. Versionamiento de evaluaciones

La épica fue creada para abarcar todos los cambios que requerían las evaluaciones en sus diferentes formas de versionamiento. Entre ellas, se encuentra el versionamiento encadenado de las evaluaciones debido al flujo terminado de versionamiento de competencias.

En caso de que existan evaluaciones ya creadas para las secciones de los cursos y en dicho curso se generaba una nueva versión de alguna de sus competencias, el sistema debía versionar las evaluaciones que utilizaba la versión anterior de dicha competencia con la versión actual para el periodo.

Como trabajo futuro se cuenta con el desarrollo de flujos de trabajo para las evaluaciones, ya que el versionamiento que ahora cuenta el sistema es automático y no a través de formularios web.

TABLA 5.4: Historias de usuario para el versionamiento encadenado de evaluaciones debido al versionamiento de competencias

Historias de usuario	HE	HC	PH	Sprints
Versionamiento de evaluaciones	102	87	8	2

5.5.1. Versionamiento de evaluaciones

La historia de versionamiento de evaluaciones tiene como propósito permitir el versionamiento automático de evaluaciones existentes que utilizar competencias del sistema. Por ejemplo, en caso de que una evaluación hecha por un profesor tenga una nueva versión en el nuevo periodo de su sección, el sistema versiona la evaluación para ese periodo obteniendo las competencias actuales.

La historia de usuario tiene como descripción: «*Como usuario del módulo curricular, me gustaría ser capaz de versionar mis evaluaciones, para que se observen los cambios a través del tiempo. Y que la interfaz y los reportes sigan presentando datos para los diseños históricos*».

Algunas de las tareas fueron las siguientes:

- Adaptar versionamiento para el modelo de datos de las evaluaciones.
- Migrar datos de los usuarios para que soporten versionamiento de las mismas.
- Actualizar el selector de evaluaciones de los profesores para que puedan seleccionar evaluaciones actuales.
- Actualizar el widget de profesores que utilizan las evaluaciones como datos.

Esta historia de usuario fue realizada en una iteración con un total de 87 horas cargadas en el sistema.

5.6. Flujo de trabajo para el versionamiento de cursos

TABLA 5.5: Historias de usuario para el flujo de trabajo para el versionamiento de cursos

Historias de usuario	HE	HC	PH	Sprints
Versionamiento de cursos	84	96	13	3
Información básica de curso	46	53	5	1
Horas y unidades de evaluación	48	66	5	1
Especificaciones de curso	40	40	5	1
Requisitos de curso	44	40	5	1
Revisar y aprobar curso	48	68	5	1
Competencias de curso	80	76	8	1
Esquema de curso	40	48	5	1
Códigos de clasificación de curso	56	80	5	1

5.6.1. Versionamiento de cursos

Esta historia de usuario fue realizada en tres iteraciones con un total de 96 horas cargadas, debido a que los cambios que suponía conlleva a una migración importante de datos de los usuarios.

Tenía como descripción: «*Como coordinador del AMS, me gustaría poder hacer una versión de mi plan de curso para que pueda realizar un seguimiento de los cambios para cosas como la revisión de programas y los acuerdos de articulación y transferencia en la aplicación*».

Algunas de las tareas realizadas en la historia fueron las siguientes:

- Buscar técnicas y herramientas de versionamiento parecidas para implementar.
- Diseñar una posible solución a la problemática.
- Implementar cambios en la base de datos mediante scripts en el proyecto.
- Actualizar clases Java existentes en el proyecto de cursos.
- Implementar la solución para el flujo de trabajo.
- Actualizar la creación de cursos sin el módulo curricular con los nuevos campos.
- Adaptar la relación de cursos y competencias para que soporte el versionamiento de los mismos.
- Actualizar la lista de competencias por cursos.

5.6.2. Información básica de curso

Esta historia de usuario tiene como propósito de diseñar páginas que permitan al usuario completar la información básica de curso que buscan diseñar, así también fueron proporcionados mockups para la pantalla (figura 5.2).

Tiene la siguiente descripción: «*Como miembro del comité de Curriculum, me gustaría ser capaz de administrar la página de información básica de cursos, para que no tenga que buscar por documentos a la hora de crear o versionar cursos*».

- Diseñar un modelo de datos que soporte el nuevo formato de información de curso.
- Adaptar tablas existentes y crear clases nuevas para las nuevas entidades de base de datos.
- Actualizar la plantilla de creación de Workflow para que soporte el nuevo paso.
- Actualizar el visualizador de Workflow.
- Diseño de pruebas automatizadas.

La historia se terminó en una iteración con un total de 53 horas de desarrollo.

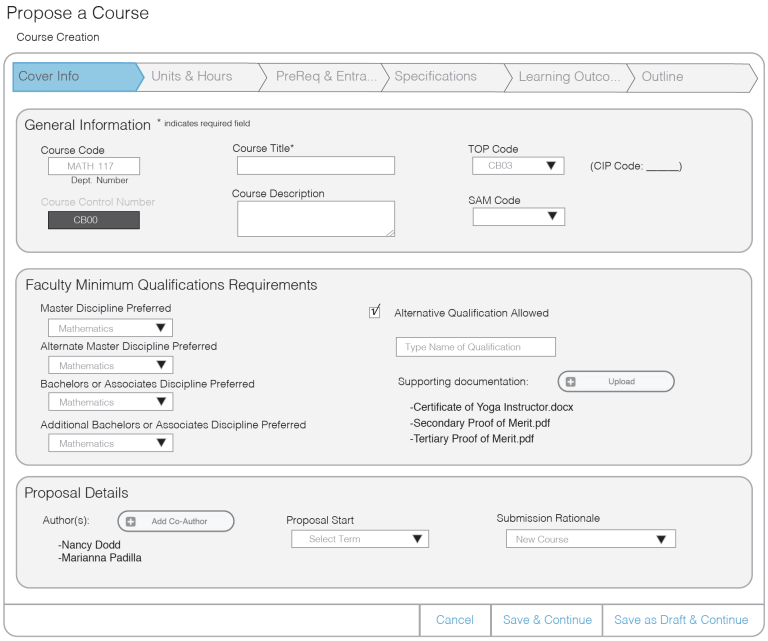


FIGURA 5.2: Mockup de la pantalla de información básica de curso.

5.6.3. Horas y unidades de evaluación

En esta historia se desarrolló un nuevo paso para el desarrollo de flujo de trabajo, en la cual el encargo del mismo va a poder detallar las horas y unidades que requiere el curso o que va a requerir.

La organización ha proveído mockups (figura 5.3) para la página como criterio de aceptación de la historia es que siga el modelo de la misma.

La historia tenía la siguiente descripción: «*Como profesor encargado del curso, me gustaría tener una página de horas y métricas para que pueda conseguir información básica sobre mi curso en eLumen*».

La historia a desarrollar se dividió entre miembros del equipo de desarrollo en las siguientes tareas:

- Modificar el modelo de datos para que soporte los nuevos campos de curso.
- Crear y/o editar las clases Java.
- Actualizar la plantilla de flujo de trabajos.
- Actualizar el visualizador de flujo de trabajos.
- Pruebas de funcionalidad.

La historia ha sido terminada en una iteración con un total de 66 horas de desarrollo.

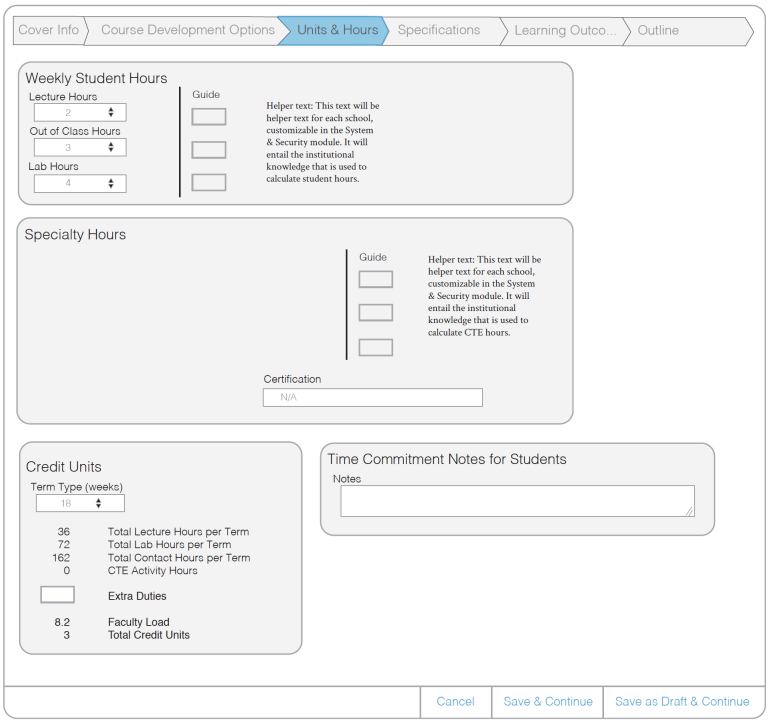


FIGURA 5.3: Mockup de la pantalla de horas y unidades de evaluación de curso.

5.6.4. Especificaciones de curso

En la siguiente historia de usuario ha desarrollado un nuevo paso para el flujo de trabajo en la cual el encargado del flujo de trabajo puede agregar objetivos, información acerca de los métodos de evaluación de la materia, algunos equipos requeridos y libros que se necesitará en el curso.

Se han proporcionado de mockups (figura 5.4) para el nuevo paso y era un criterio de aceptación de la historia de usuario seguir el mismo formato para el desarrollo de la misma.

La historia de usuario proporciona la siguiente descripción: «*Como profesor encargado de curso, me gustaría ser capaz de agregar o editar especificaciones de curso como parte del flujo de trabajo de creación y/o versionamiento de mi curso para no tener que hacerlo en papel*».

Las tareas fueron separadas y desarrolladas por los desarrolladores y eran las siguientes:

- Migrar los datos para que soporte el nuevo formato de cursos.
- Crear y/o modificar clases de Java para el nuevo modelo de datos.
- Actualizar la página de plantillas de flujo de trabajo para que soporte el nuevo paso.
- Actualizar el visualizador de flujo de trabajo.
- Actualizar los servicios de guardado para creación y versionamiento de cursos y flujo de trabajos.
- Actualizar el servicio de aprobación de flujo de trabajo.

La historia fue terminada en una iteración con 40 horas de desarrollo cargadas en el sistema

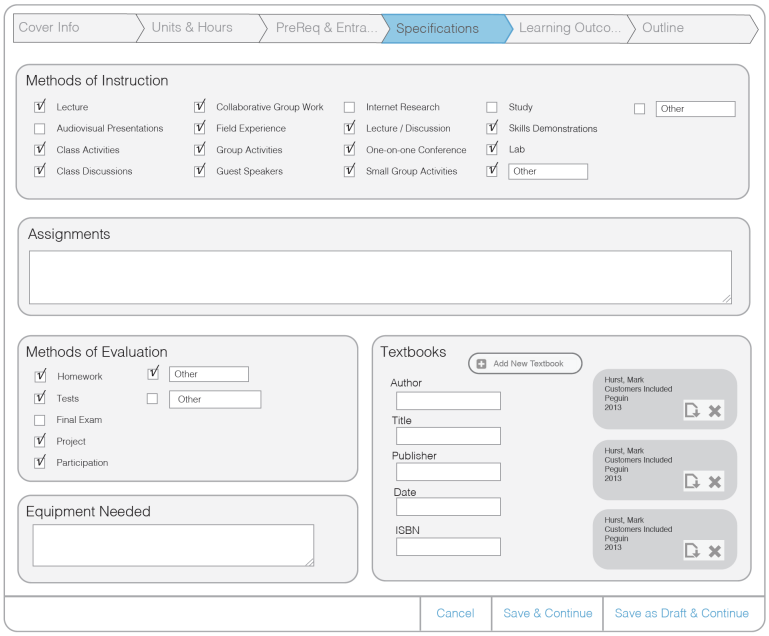


FIGURA 5.4: Mockup de la pantalla de especificaciones de curso.

5.6.5. Requisitos de curso

Esta historia tiene como propósito el de adaptar el modelo de datos para que una lista de cursos como pre-requisitos, co-requisitos, anti-requisitos, y

recomendaciones para su nuevo curso. Además, de ciertas capacidades que el alumno debe tener como requisito para tomar el curso.

Para entrar un poco en contexto de la historia vamos a definir cuáles son los tipos de requisitos que puede tener un curso:

- **Pre-requisito:** es un tipo de requisito que impide al usuario tomar o cursar un curso sin haber aprobado antes del curso que está como pre-requisito.
- **Co-requisito:** es un tipo de requisito que impide al usuario tomar un curso si no cursa también el curso que tiene como co-requisito.
- **Anti-requisito:** es un requisito que impide al usuario tomar un curso si ya aprobó o va a tomar un curso que tiene como anti-requisito.
- **Recomendación:** es una recomendación por parte del sistema que materia tomar para aprovechar mejor la malla. Es opcional.

La historia de usuario tiene como descripción: «*Como persona encargada de un curso, me gustaría ser capaz de introducir requisitos para cursos y ciertas competencias adquiridas en la creación o revisión de flujo de trabajo, para que podamos seguir durante su desarrollo y aprobación*» y la figura 5.5 muestra mockups para la pantalla.

La historia fue dividida en partes para que los desarrolladores puedan trabajar en partes independientes durante el proceso de la misma, y eran las siguientes:

- Diseñar y actualizar el modelo de datos atual.
- Generar y actualizar clases Java para la lógica.
- Actualizar la plantilla de flujo de trabajo para que soporte un nuevo paso.
- Actualizar el visualizador de flujo de trabajo.
- Actualizar los servicios de guardado y aprobación.
- Pruebas de funcionalidad.

La historia de usuario ha sido terminada en una iteración con 44 horas de desarrollo cargadas en el sistema.

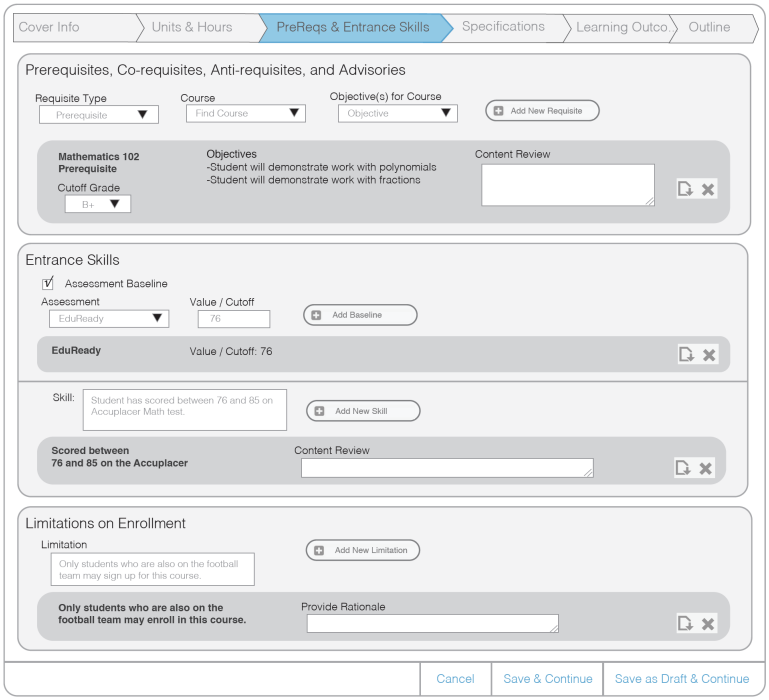


FIGURA 5.5: Mockup de la pantalla de requisitos de curso.

5.6.6. Revisar y aprobar curso

La historia de usuario tiene como criterios de aceptación los siguientes puntos:

- Las páginas para revisar los flujo de trabajos tienen una región de retroalimentación o feedback debajo de cada paso, con la opción de ocultar y mostrar para que el usuario que está revisando el flujo de trabajo en desarrollo pueda dejar comentarios al encargado del formulario del curso.
- La interfaz tiene elementos de estado que indican que cierta parte es nueva, aprobada, y rechazada.
- Los pasos tienen regiones que permiten aceptar o rechazar los campos propuestos por los desarrolladores del curso. Por lo tanto, deben tener elementos de interfaz que indiquen al usuario que puede aprobar o rechazar cada parte.

Además de los criterios de aceptación, había que volver a actualizar el buzón de entrada para que acepten los cambios que tiene la historia de usuario. Debido a que más de una persona puede revisar el flujo de trabajo y podría trancar el proceso si es que no se le notifica debidamente que hay nuevos cambios que revisar.

Algunas de las tareas descompuestas de la historia de usuario son las siguientes:

- Actualizar el visualizador de flujo de trabajo para que pueda soportar la nueva característica de aprobación o rechazo de cada parte.

- Actualizar el buzón de entrada de Cursos.
- Pruebas de funcionamiento.

La historia se ha terminado en una iteración con 68 horas cargadas de desarrollo

5.6.7. Competencias de curso

Esta historia tiene como propósito de crear o versionar competencias para el curso a ser creado o versionado.

La organización ha proporcionado mockups (figura 5.6) para el paso a desarrollarse y era un criterio de aceptación de parte del ticket que siga el mismo formato.

La historia de usuario tiene como descripción: «*Como encargado del formulario de curso, me gustaría ser capaz de articular las competencias de mi nuevo curso, para de esa manera de tener que estar añadiendo una a una después de completar el proceso de creación de cursos con el flujo*».

Como criterio de aceptación de la historia fue la de agregar el flujo de trabajo de competencias en el flujo de trabajo de cursos. Algunas de las tareas de la historia fueron:

- Modificar la base de datos para que soporte el nuevo modelo de datos de las competencias dentro de flujo de trabajo de curso.
- Modificar o agregar clases de las entidades que van a ser usadas durante la historia.
- Actualizar la plantilla de flujo de trabajo para que soporte el nuevo paso para la creación o versionamiento de competencias.
- Actualizar el visualizador de flujo de trabajo para que soporte el nuevo paso de competencias.
- Actualizar los servicios de guardado y de versionamiento de cursos y competencias.
- Pruebas de nuevas funcionalidades.

La historia ha sido terminada en dos iteraciones con un total de 76 horas cargadas en el sistema.

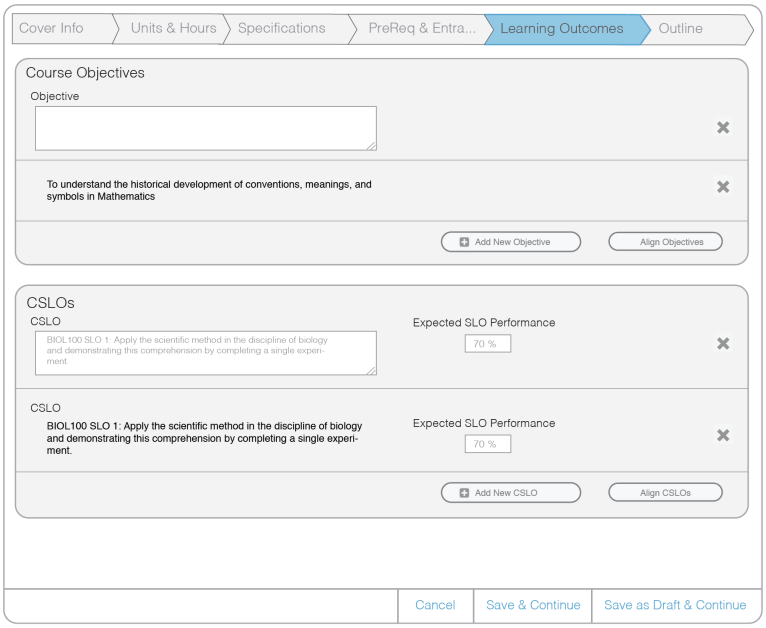


FIGURA 5.6: Mockup de la pantalla de competencias de curso.

5.6.8. Esquema de curso

Esta historia tiene como propósito de diseñar la pantalla para un nuevo paso para el flujo de trabajo.

La organización ha proporcionado mockups (figura 5.7) para el paso a desarrollarse y era un criterio de aceptación de parte del ticket que siga el mismo formato.

La historia de usuario tiene como descripción: «*Como encargado del formulario de curso, me gustaría ser capaz de agregar el esquema de un curso, para de esta manera dar un resúmen de curso para el que esté revisando mi flujo y para que los estudiantes puedan tener una idea de se trata una vez que se curse*».

Como criterio de aceptación de la historia fue la de agregar el flujo de trabajo de competencias en el flujo de trabajo de cursos. Algunas de las tareas de la historia fueron:

- Modificar la base de datos donde se tendría que almacenar los nuevos campos de esquema ya sea para el curso y para el flujo que se desarrolla.
- Modificar o agregar clases de las entidades que van a ser usadas durante la historia.
- Actualizar la plantilla de flujo de trabajo para que soporte el nuevo paso de esquema de cursos.
- Actualizar el visualizador de flujo de trabajo para que soporte el nuevo paso de competencias.
- Actualizar los servicios de guardado y de revisión para que soporte nuevo paso.

La historia ha sido terminada en dos iteraciones con un total de 48 horas cargadas en el sistema.

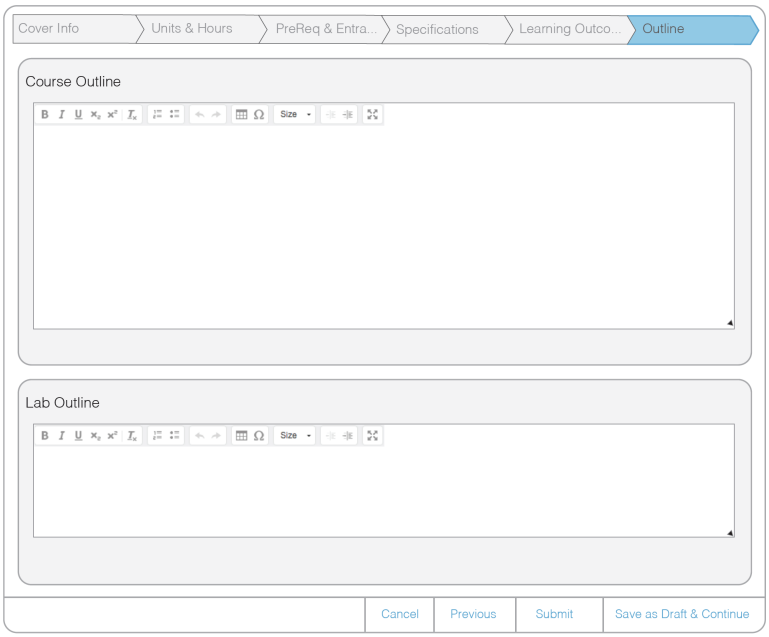


FIGURA 5.7: Mockup de la pantalla de esquema de curso.

5.6.9. Codigos de clasificación de curso

Esta historia tiene como propósito la de asignar códigos de clasificación a los cursos.

Para entrar en contexto, habría que definir primero que es TOP[1].

TOP es un sistema numérico de códigos usados a nivel de Estado para recolectar y reportar información en cursos y programas, en diferentes instituciones educativas [15] por todo el Estado.

Ha sido diseñado para agregar información acerca de los programas. Sin embargo, un código TOP debe ser asignado a cada curso del sistema.

Aunque no contiene tantas opciones específicas como lo haría un sistema diseñado para cursos, a cada curso se le debe dar el código que se aproxima a describir el contenido del curso.

Algunos usos a los códigos:

– En el inventario de programas aprobados y rechazados, para tener información que tipos de cursos y programas son ofrecidas por el estado.

– En bases de datos de administración de información, para recolectar y reportar información en logros estudiantiles (licenciaturas y certificados) en ciertos programas.

[1] de sus siglas en inglés, Taxonomy Of Programs, que significa en español taxonomía de programas.

– En contabilidad vocacional estudiantil, para reportes de compleción de programas y cursos de ciertos programas vocacionales.

La historia de usuario tiene como descripción: «*Como miembro del comité curricular, me gustaría ser capaz de asignar a mis cursos de códigos de clasificación como parte de la aprobación de mis flujo de trabajos para asegurar que estén correctos, como esto es motivo de rechazo en la oficina del canciller del Estado*».

Algunas de las tareas fueron las siguientes:

– Diseño del nuevo modelo, donde se debían generar tablas para cada nueva entidad del modelo de datos ajustado para las taxonomías de programas. Además, cargar todos los datos de códigos de cursos existentes para el estado de California.

– Creación de clases Java.

– Diseño e implementación páginas CRUD para disciplina, sub-disciplina, y campo.

– Diseño e implementación de la nueva página de asignación de códigos de clasificación para los cursos en proceso de diseño.

– Hacer servicios para cada una de las nuevas páginas.

– Pruebas de funcionalidad.

La historia de usuario ha sido terminada en una iteración con 80 horas cargadas en el sistema.

5.7. Flujo de trabajo para el versionamiento de programas de estudio

TABLA 5.6: Historias de usuario para flujo de trabajo para el versionamiento de programas de estudio

Historias de usuario	HE	HC	PH	Sprints
Información básica del programa	54	56	8	1
Competencias de carrera o programa	48	68	5	1
Bloques de curso	58	60	5	1
Visualizar cambios en los campos	180	210	13	4

5.7.1. Información básica del programa

La historia de usuario tiene como descripción lo siguiente «*Como coordinador del departamento o encargado del AMS, me gustaría ser capaz de agregar o revisar programas en el módulo de gestión curricular, para que de esta forma pueda manejar mejor mis registros de la institución en el sistema*». Y los criterios de aceptación consistían en el desarrollo de la pantalla que se puede apreciar en la figura 5.8.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Adaptar la base de datos para soportar los nuevos campos a ser guardados por el flujo de trabajo.
- Luego de hacer los cambios en la base de datos, actualizar o agregar nuevas clases de Java para su posterior uso.
- Desarrollar la página de información básica del programa.
- Actualizar la plantilla de flujos de trabajo institucional para que soporte la creación y revisión de programas.
- Diseñar servicios para guardar los registros de la nueva página.
- Diseñar servicios de aprobación de flujo de trabajo de programas.

La historia fue finalizada en una iteración con una cantidad de 56 horas cargadas en el sistema.

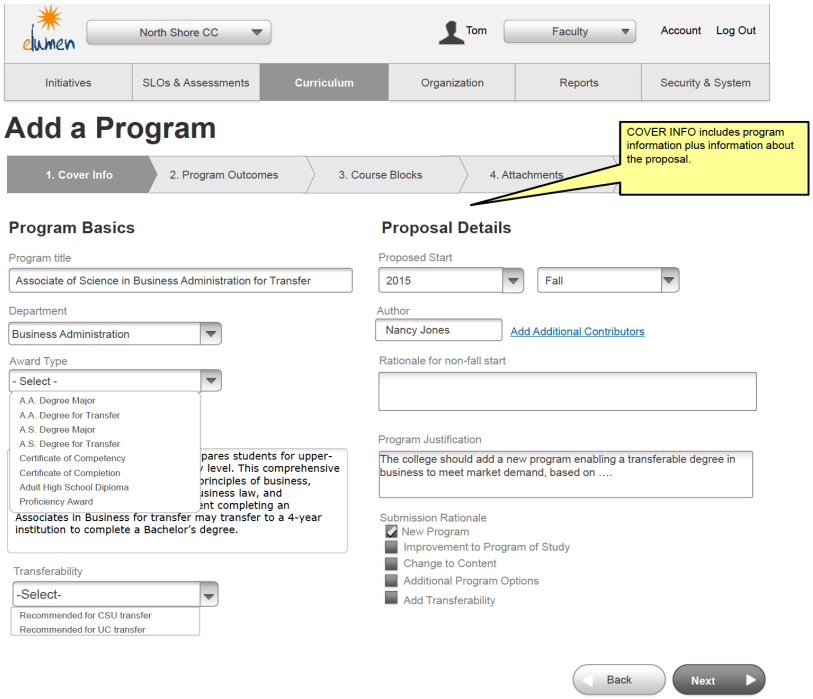


FIGURA 5.8: Mockup de la pantalla de información básica del programa.

5.7.2. Competencias de carrera o programa

La historia de usuario tiene como descripción lo siguiente «*Como coordinador, me gustaría ser capaz de administrar las competencias asociadas a mi programas durante el flujo de creación y revisión del mismo, para que pueda hacer una revisión comprensiva de los programas que tiene el AMS*». Y los criterios de aceptación consistían en el desarrollo de la pantalla que se puede apreciar en la figura 5.9.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Adaptar la base de datos para soportar los nuevos campos a ser guardados por el flujo de trabajo.
- Luego de hacer los cambios en la base de datos, actualizar o agregar nuevas clases de Java para su posterior uso.
- Desarrollar la página de información básica del programa.
- Actualizar los servicios de guardado de campos para el flujo de trabajo.
- Actualizar los servicios de aprobación de flujo de trabajo de programas.

La historia fue finalizada en una iteración con una cantidad de 68 horas cargadas en el sistema.

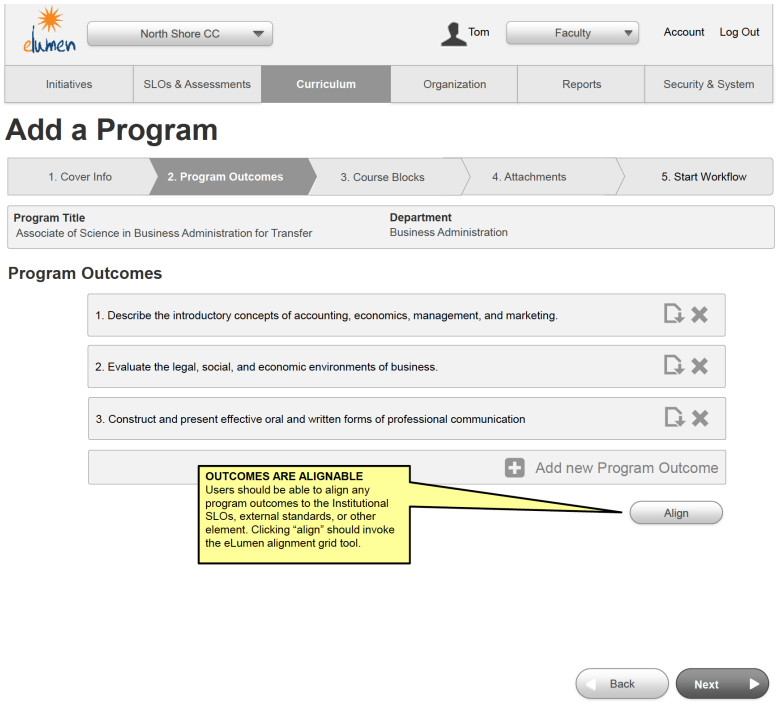


FIGURA 5.9: Mockup de la pantalla de competencias del programa.

5.7.3. Bloques de cursos

La historia de usuario tiene como descripción lo siguiente «*Como coordinador de departamento, me gustaría ser capaz de diseñar bloques de cursos para mis programas, para que de esta manera pueda diseñar la malla para mis programas de estudio*». Y los criterios de aceptación consistían en el desarrollo de la pantalla que se puede apreciar en la figura 5.10.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

– Diseño e implementación del modelo de datos.

– Diseño e implementación de clases Java.

– Desarrollar la página de paso para creación de bloques de cursos en los diferentes flujos de trabajo.

– Desarrollar servicios de guardado y aprobación de la funcionalidad.

– Actualizar la plantilla de flujos de trabajo.

La historia fue finalizada en una iteración con una cantidad de 60 horas cargadas en el sistema.

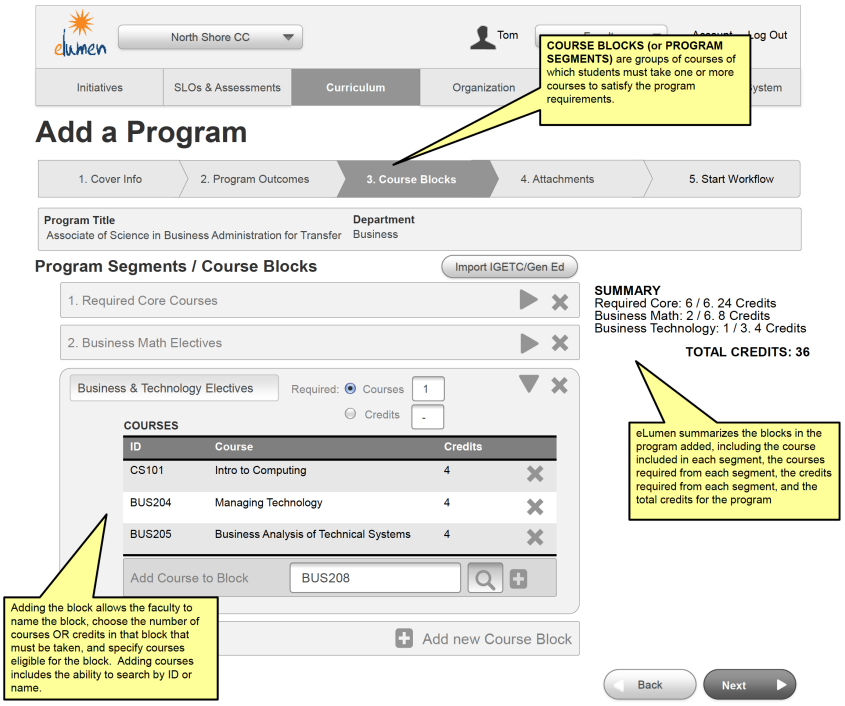


FIGURA 5.10: Mockup de la pantalla de bloques de cursos de programa.

5.7.4. Visualizar cambios en los campos

La historia de usuario tiene como descripción lo siguiente «*Como coordinador de departamento, me gustaría ser capaz de diseñar bloques de cursos para mis programas, para que de esta manera pueda diseñar la malla para mis programas de estudio*». Y los criterios de aceptación consistían en el desarrollo de la pantalla que se puede apreciar en la figura 5.11.

Como criterios de aceptación se encuentran los siguientes:

– Los campos borrados se deben marcar en rojo.

– Los nuevos campos se deben marcar en verde.

– Se debe visualizar el estado anterior y el nuevo con una forma de identificar con el usuario que hizo la modificación.

– Limitado para los cambios del programa.

– Diseño de interfaz aprobada por el equipo de validación.

– Las diferencias limitada a dos versiones.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

– Diseño e implementación del modelo de datos.

– Diseño e implementación de clases Java.

– Desarrollar la página de paso para creación de bloques de cursos en los diferentes flujos de trabajo.

– Desarrollar servicios de guardado y aprobación de la funcionalidad.

– Actualizar la plantilla de flujos de trabajo.

La historia fue finalizada en una iteración con una cantidad de 60 horas cargadas en el sistema.

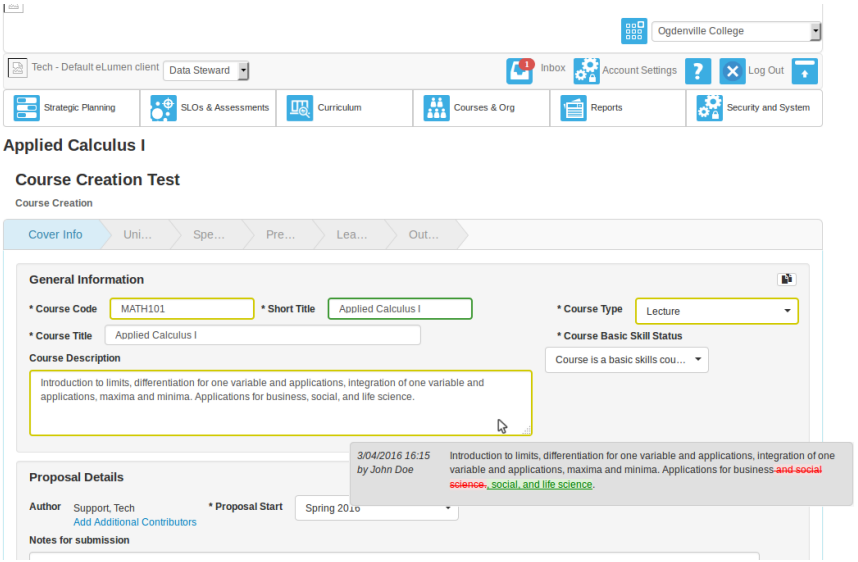


FIGURA 5.11: Mockup de la pantalla de la funcionalidad de visualización de cambios.

5.8. Soporte de etapas en los flujos de trabajo

TABLA 5.7: Historias de usuario para soporte de etapas en los flujos de trabajo

Historias de usuario	HE	HC	PH	Sprints
Roles de creación y edición para las partes de flujos de trabajo	102	112	13	1
Diseño e implementación de etapas	288	505	21	3
Mejora en comportamientos para las etapas por roles	84	108	8	2
Composición de etapas y partes	216	391	13	3
Etapas y partes opcionales en la revisión del flujo	64	76	8	1

5.8.1. Roles de creación y edición para las partes de flujos de trabajo

La historia de usuario tiene como descripción lo siguiente «*Como participante en el proceso curricular, podría no solo revisar nuevos cursos y programas, sino que también realizar pequeñas revisiones como parte del proceso o participe también en el paso de compleción del formulario*».

Como criterios de aceptación se encuentran los siguientes:

- Diseñar el privilegio de creador que permitan a un iniciador de flujo o un diseñador de flujo que designe a ciertos roles para escribir o llenar cada paso.
- Diseñar el privilegio de editor que permita editar partes a los que revisan el flujo de trabajo antes de aceptar o rechazar.
- Configurar flujos de trabajo para que en cada parte o etapa de un flujo de trabajo puedan ser asignados por roles para la tarea de creador o evaluador, o ambos.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Diseñar e implementar nuevos modelos de datos que permitan soportar el uso de roles para creadores y editores de partes.
- Diseñar e implementar servicios de visualización para las diferentes secciones de flujos de trabajo.
- Actualizar las plantillas de flujo de trabajo para agregar el soporte de roles de creación y edición.
- Actualizar el flujo de trabajo para soportar la compleción de secciones por paso.
- Implementar la funcionalidad de edición.

La historia fue finalizada en tres iteraciones con una cantidad de 112 horas cargadas en el sistema.

5.8.2. Diseño e implementación de Etapas

La historia de usuario tiene como descripción lo siguiente «*Como administrador curricular quiero ser capaz de configurar mi plantilla de flujo de trabajo para que pueda dividir en etapas donde se especifiquen que roles pueden completar que funciona en una o múltiples secciones o partes de mi programa o curso*». Y los criterios de aceptación consistían en el desarrollo de la pantalla que se puede apreciar en la figura 5.11.

Como criterios de aceptación se encuentran los siguientes:

- En un diseño de flujo de trabajo se debe elegir un rol, sección o parte y la acción (completar, revisar, aprobar).
- Para la primera etapa solo la acción de completar debe estar disponible.

- Las acciones de revisar y aprobar deben estar disponibles si una etapa anterior tiene las mismas secciones o partes con la acción de completar.
- Para la primera etapa todas las secciones o partes deben ser completadas en orden secuencial.
- Cualquier etapa después de la primera puede tener la opción de mostrar comentarios para la persona que completó los datos antes de transicionar a la siguiente etapa.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Analizar las zonas posibles a ser afectadas por la nueva funcionalidad.
- Actualizar la plantilla de flujos de trabajo.
- Actualizar el flujo de trabajo de cursos.
- Actualizar el flujo de trabajo de programas.
- Diseñar e implementar un modelo de datos que soporte la nueva funcionalidad.
- Actualizar el buzón de entrada.
- Actualizar las notificaciones a colaboradores.
- Diseñar e implementar migraciones de datos.

La historia fue finalizada en tres iteraciones con una cantidad de 505 horas cargadas en el sistema.

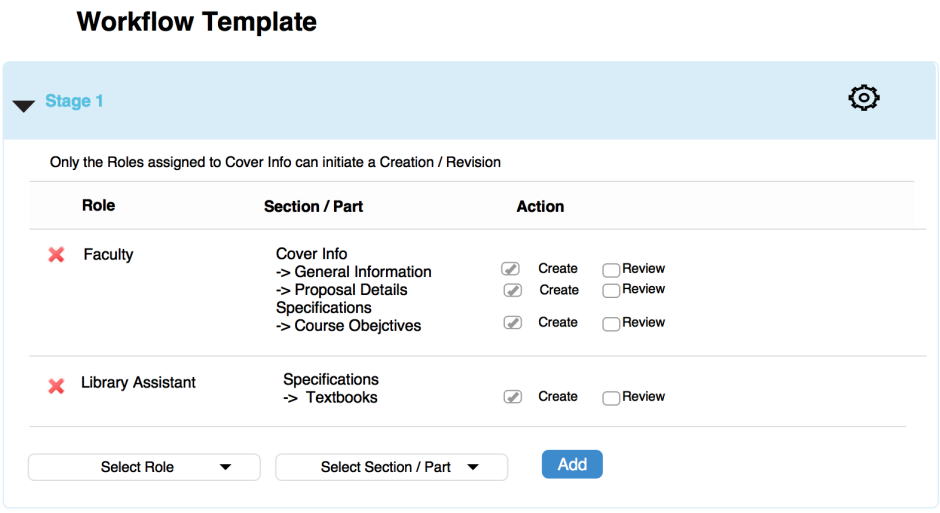


FIGURA 5.12: Mockup de la pantalla de plantillas soportando las etapas.

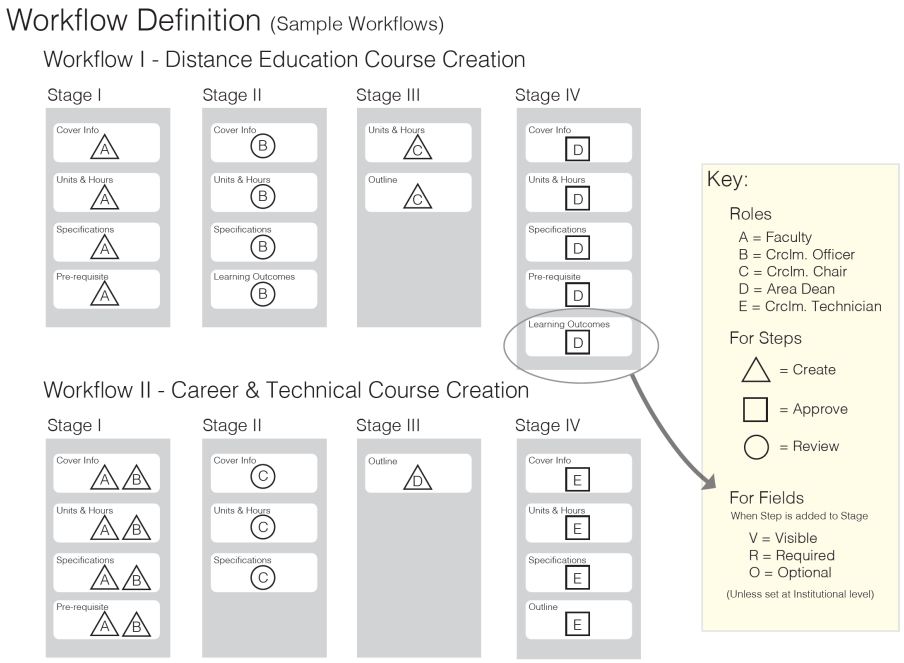


FIGURA 5.13: Mockup de la pantalla de plantillas con etapas por flujo.

5.8.3. Mejora en comportamientos para las etapas por roles

La historia de usuario tiene como descripción lo siguiente «*Como coordinador de educación a distancia, yo solo necesito revisar el esquema del curso para aquellos cursos diseñados para educación a distancia*».

El criterio de aceptación de la historia consistía en permitir que una etapa que no tiene roles asignados sea opcional.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Diseñar un plan de pruebas, en estas se identifican las posibles zonas afectadas por la nueva funcionalidad.
- Actualizar la pantalla de creación de flujos de trabajo.
- Actualizar el mecanismo de transición de etapas.
- Mejorar la UI de la vista de etapas de flujos.

La historia fue finalizada en tres iteraciones con una cantidad de 108 horas cargadas en el sistema.

5.8.4. Composición de etapas y partes

La historia de usuario tiene como descripción lo siguiente «*Como presidente curricular, me gustaría ser capaz de componer flujos de trabajo de cursos o programas y etapas (incluyendo actores en cada etapa y sus acciones), para que de esa manera se pueda modelar el proceso curricular en la aplicación*».

Como criterios de aceptación se encuentran los siguientes:

- Cada etapa puede contener una o más partes.
- Los actores de cada etapa pueden asignarse actiones a cada parte, o a todas.
- Etapas equivalentes no pueden bloquearse entre sí. Por ejemplo, si Suzy y Joe son evaluadores de tres partes, Joe no tiene que esperar que Suzy revise la parte 1 antes de que el pueda evaluar la parte 2, es decir, ambos pueden evaluar sus partes en simultáneo.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Diseño de mockups para su aprobación previa al desarrollo.
- Diseño e implementación del modelo de datos que soporte la nueva funcionalidad.
- Actualizar las plantillas de flujos de trabajo.
- Actualizar los flujos de trabajo de cursos y programas.
- Actualizar el sistema de notificaciones.

La historia fue finalizada en tres iteraciones con una cantidad de 391 horas cargadas en el sistema.

5.8.5. Etapas y partes opcionales en la revisión del flujo

La historia de usuario tiene como descripción lo siguiente «*Como administrador curricular quiero ser capaz de configurar mi plantilla de flujo de trabajo para que pueda dividir en etapas donde se especifiquen que roles pueden completar que funciona en una o múltiples secciones o partes de mi programa o curso*».

Como criterios de aceptación se encuentran los siguientes:

- Los roles pueden ser configurados para la acción de creación o revisar, o ambos.
- El rol de creación lleva el nombre de creador y para la revisión lleva el nombre de evaluador.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Diseñar un plan de pruebas, en estas se identifican las posibles zonas afectadas por la nueva funcionalidad.
- Actualizar la pantalla de creación de flujos de trabajo.
- Actualizar el mecanismo de transición de etapas.

La historia fue finalizada en tres iteraciones con una cantidad de 76 horas cargadas en el sistema.

5.9. Reportes y notificaciones para flujos de trabajo

TABLA 5.8: Historias de usuario para los reportes y notificaciones de versiones de cursos

Historias de usuario	HE	HC	PH	Sprints
Notificaciones para las partes del flujo de trabajo	58	60	5	1
Reporte de esquemas de curso	88	91	8	2

5.9.1. Notificaciones para las partes del flujo de trabajo

La historia de usuario tiene como descripción lo siguiente «*Como especialista curricular, me gustaría que mi equipo de diseño y revisión de flujos de trabajo reciban las notificaciones cuando tengan trabajos pendientes (y alertas cuando este retrasado), para que pueda manejar mejor mis procesos curriculares*».

Como criterios de aceptación se encuentran los siguientes:

- Establecer notificaciones cuando las partes del flujo de trabajo son asignadas a los roles de las personas.
- Establecer notificaciones de alerta a asignaciones de partes y etapas. Por ejemplo, 5 días después de su asignación.
- Mandar notificaciones por mail.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Diseñar e implementar nuevos modelos de datos que permitan soportar el uso de roles para creadores y editores de partes.
- Actualizar el sistema de notificaciones del AMS.
- Diseñar e implementar la página de configuración de notificaciones.

La historia fue finalizada en tres iteraciones con una cantidad de 60 horas cargadas en el sistema.

5.9.2. Reporte de esquemas de curso

La historia de usuario tiene como descripción lo siguiente «*Como especialista curricular, me gustaría ser capaz de hacer reportes de registro de esquemas de curso, para que pueda de esta manera ser compartidas por los diferentes colaboradores*».

Como criterios de aceptación se encuentran los siguientes:

- Reporte diseñado por el equipo de diseño curricular.
- Incluir competencias es opcional para el reporte.
- Incluir alineación de competencias es opcional para el reporte.
- Formatos en DOC, PDF o HTML (no Excel).

- Se puede ejecutar desde la lista de cursos o de la lista de reportes.
- Cuando se corre desde la lista de reportes se pueden elegir uno o más cursos.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Diseñar e implementar el reporte.
- Implementar los métodos que acceden a la base de datos.
- Diseñar e implementar la página de generación de reportes.

La historia fue finalizada en tres iteraciones con una cantidad de 111 horas cargadas en el sistema.

5.10. Soporte de versionamiento en el AMS

TABLA 5.9: Historias de usuario para soporte de versionamiento en el AMS

Historias de usuario	HE	HC	PH	Sprints
Interfaz de alineación de códigos TOP/CIP	32	36	3	1
Renombrar/Reorganizar pestañas para una mejor apariencia del módulo curricular	32	44	3	1
Lista curricular mejorada para cursos y programas	132	135	8	2

5.10.1. Interfaz de alineación de códigos TOP/CIP

La historia de usuario tiene como descripción lo siguiente «*Como encargado del sistema de gestión de evaluaciones, me gustaría ser capaz de mantener alineados los codigos TOP a los códigos federales CIP y de esta manera no tener que depender de los administradores para manejar o interpretar estos datos*».

Como criterios de aceptación se encuentran los siguientes:

- Diseñar e implementar una página CRUD para códigos TOP/CIP.
- Permitir alinear los códigos.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Diseñar mockups para la nueva página.
- Diseñar e implementar el modelo de datos que soporte la funcionalidad.
- Desarrollar la página y los métodos de guardado.

La historia fue finalizada en una iteración con una cantidad de 36 horas cargadas en el sistema.

5.10.2. Reorganización de pestañas para una mejor apariencia del módulo curricular

La historia de usuario tiene como descripción lo siguiente «*Como presidente del comité curricular, me gustaría ver las pestañas que están enfocadas a mi trabajo para que pueda navegar y manejar mi tiempo en la aplicación*».

Como criterios de aceptación se encuentran los siguientes:

- Crear pestaña curricular.
- Identificar y esconder pestañas no relevantes para el rol.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Diseño de pestañas.
- Implementación de pestañas y modificaciones de espacio.

La historia fue finalizada en una iteración con una cantidad de 44 horas cargadas en el sistema.

5.10.3. Lista curricular mejorada para cursos y programas

La historia de usuario tiene como descripción lo siguiente «*Como presidente curricular o miembro del plantel de profesores, me gustaría ser capaz de ordenar/filtrar/visualizar mis cursos y programas, para que de esta manera pueda ver que es importante para mi para luego tomar la acción correspondiente desde esta vista y mi trabajo pueda ser realizado de manera eficiente y con buena visibilidad*».

Como criterios de aceptación se encuentran los siguientes:

- Filtrar por estado de flujo de trabajo, fecha de inicio, fecha de fin y otros a ser designados por el equipo de validación.
- Ser capaz de tomar acciones desde la lista (mirar reporte de esquema, empezar una revisión, iniciar el flujo, otros).
- Columnas configurables para la vista (resultados de curso, semestre inicial, estado de flujo de trabajo, perfomance de competencia, etc.).

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Diseñar mockups para la nueva página.
- Diseñar e implementar el modelo de datos que soporte la funcionalidad.
- Implementar los filtros.
- Implementar página para las columnas configurables.
- Actualizar lista.
- Actualizar el reporte de esquema de cursos para que soporte múltiples cursos.

La historia fue finalizada en dos iteraciones con una cantidad de 135 horas cargadas en el sistema.

5.11. Retoques finales

TABLA 5.10: Historias de usuario para los retoques finales

Historias de usuario	HE	HC	PH	Sprints
Retoques finales para el flujo de trabajo de cursos	86	162	8	1

5.11.1. Retoques finales para el flujo de trabajo de cursos

La historia de usuario tiene como descripción lo siguiente «*Como especialista curricular, me gustaría poder visualizar y editar todos los campos requeridos por el PCAH*».

Como criterios de aceptación se encuentran los siguientes:

- El flujo de trabajo soporta todos los campos del PCAH.
- El flujo de trabajo soporta los campos almacenados adicionales.

Algunas de las tareas identificadas en la planificación de las iteraciones eran los siguientes:

- Diseñar mockups para todas las partes del flujo de trabajo.
- Actualizar las pantallas de las partes del flujo de trabajo.
- Actualizar los procedimientos de guardado y versionamiento de entidades.

La historia fue finalizada en tres iteraciones con una cantidad de 162 horas cargadas en el sistema.

5.12. Aporte

En la tabla 5.11 se puede apreciar cuáles son los aportes en cada historia de usuario utilizada para desarrollar de manera iterativa el módulo curricular del AMS, los datos fueron calculados desde la herramienta «JIRA» en la que se cargaban las horas, historias de usuario y fallas del sistema.

En la cual el trabajo consistía en desarrollo de la historia, desarrollo de tests, y la persona que no desarrolló la historia es la encargada de hacer la validación de código y funcionalidad.

TABLA 5.11: Tabla de historias de usuario y aportes

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 %

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 con el del AMS. Si en la aplicación o en el módulo se encontraban fallas, se procedían a crear tickets de fallas. Además, si habían mejoras que hacerse se creaban tickets de mejoras. En la figura 5.14 podemos ver la distribución de tickets aportados por tipo.

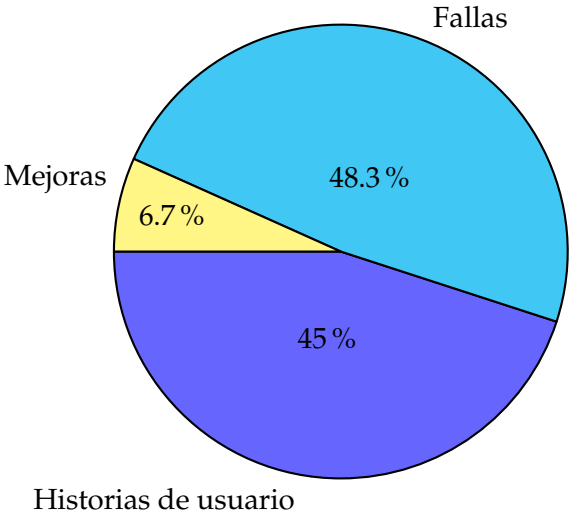


FIGURA 5.14: Distribución de tickets por tipo.

Las historias de usuario que no cumplieron con todos los criterios de aceptación vuelve a ser abierta para que se continue su desarrollo. El porcentaje de historias que no cumplieron los criterios de aceptación se puede apreciar en la figura 5.15.

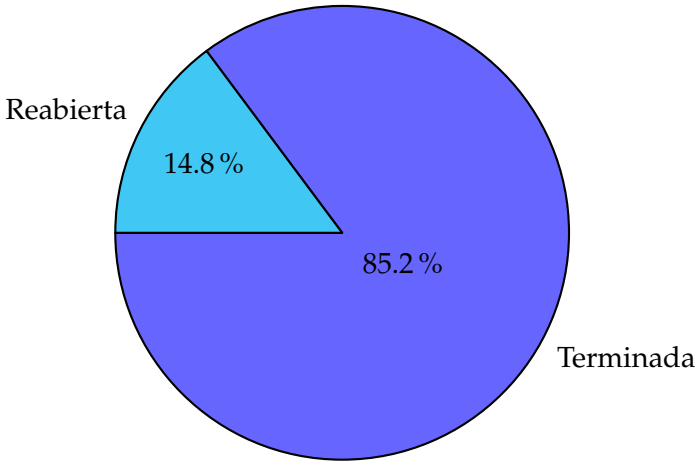


FIGURA 5.15: Porcentajes de historias de usuario reabiertas.

Los tickets de fallas que se trabajaron fueron fallas o errores de dominio encontrados en la funcionalidad del módulo curricular, cuando el equipo de desarrollo solucionaba dichas fallas debía seguir el mismo proceso de validación. En caso de que la falla mediante pruebas seguía se volvía a abrir para que se continúe trabajando hasta que cumpla con el criterio de aceptación que era solucionar la falla que se reportaba. En la figura 5.16 se puede ver cuántas fallas fueron cerradas y reabiertas en el periodo de desarrollo.

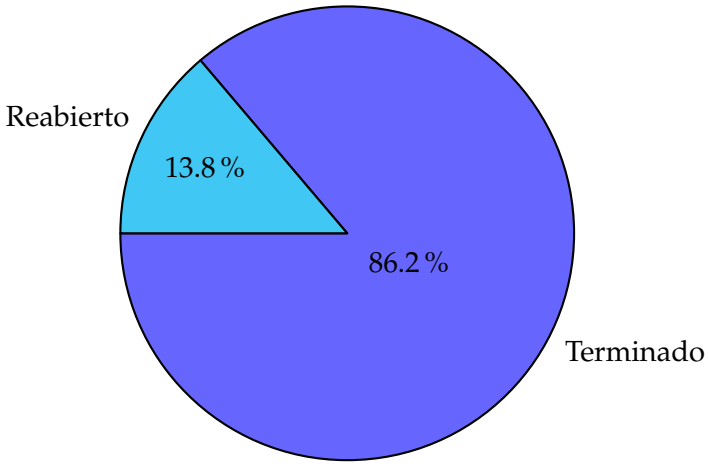


FIGURA 5.16: Porcentajes de tickets de fallas que fueron reabiertos.

Los tickets de fallas por lo general eran tickets de dificultad pequeña que un programador podía terminar en uno o dos días aproximadamente, es decir en un sprint. En cambio, las historias de usuario eran de mayor dificultad y podía tomar a varios desarrolladores hasta más tiempo que sprint podía brindar, en esos casos se continuaba en el siguiente sprint o se separaba la funcionalidad en tickets más pequeños, como se puede apreciar en la figura 5.17.

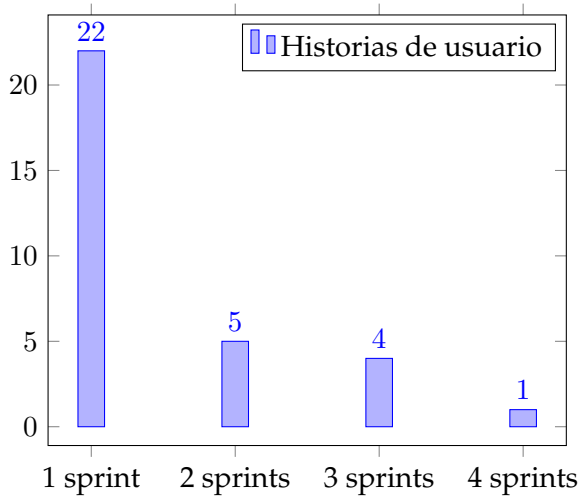


FIGURA 5.17: Cantidad de historias de usuarios terminadas en cantidad de sprints.

Capítulo 6

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.

6.1. 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 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[1] 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.
- 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 del PR en «GitHub».
- Por cada rama en PR se ejecutan las pruebas automatizadas de karma, debe pasar las nuevas pruebas y las ya existentes.
- Debe pasar las reglas de código estático de la herramienta «Sonarqube».
- No deben de haber conflictos de código de la rama de la funcionalidad con el del repositorio.

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.

[1] 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.

6.2. 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[50].

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 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 automatizadas 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:**
- **Análisis de código estático:**

Los sistemas de pruebas automatizadas empleados en la organización, tanto unitarios como extremo a extremo, se basan en el uso de Karma, un lanzador de tests que se integra con distintos frameworks (Jasmine, QUnit, Mocha, etc.) para ejecutar tests en navegadores reales (Firefox, Chrome, Phantom, e Internet Explorer).

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

Con karma para las pruebas de interfaz, se decidió también el uso de la plataforma «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.

6.3. 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 criterias 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».

Capítulo 7

Conclusión y aportes

7.1. Conclusiones generales

En el proyecto final se tomó como caso de estudio con enfoque en la interacción humano-computador y con observación participante el diseño e implementación de un módulo de gestión curricular para un sistema de gestión de evaluaciones basadas en competencias académicas, la cual tiene sus cimientos en el mercado y también dispone de clientes utilizando la misma.

Se diseñó la manera de integrar y estructurar procesos separados de validación de competencias, cursos, y programas para el estado de California en un sistema de gestión de evaluaciones basadas en competencias. Dicho proceso era un proceso que se hacía en papel y tenía sus falencias debido a que el proceso requería mucho tiempo en revisar y aprobar el formulario como se muestra en la figura 2.2, y 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 conclusiones al utilizar la metodología ágil son las siguientes:

- Posee el potencial para mejorar rápidamente la elección de que construir, pero sólo si se tiene en cuenta las señales del mercado que sugieren el cambio, y sólo si luego se toman medidas de cambios.
- Posee el potencial para entregar de manera más eficiente, pero sólo si se invierte tiempo en modular en unidades de trabajo más pequeñas.
- Posee el potencial para proporcionar mejores predicciones, pero sólo si usted invierte en habilidades de estimación, y sólo si se hace una distinción entre la predicción y el compromiso.
- Posee el potencial para ofrecer productos de mayor calidad, pero sólo si se recopilan los comentarios de los clientes sobre el enfoque de la solución y sólo si se realizan los cambios apropiados.

Al diseñar los flujos de trabajo para el diseño y revisión de formularios de competencias, cursos, y programas permitió a los profesores encargados de los mismos y evaluadores de dichos formularios, seguir el proceso de una manera intuitiva buscando la mejor experiencia de usuario y con menos cuellos de botella. Como ayuda, que cada paso genera un mensaje que el usuario puede 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 fue una decisión errónea, 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.

Sin embargo, al utilizar MySQL para guardar el flujo de trabajo y todos sus datos temporales no fue la mejor decisión debido a que la funcionalidad y el estándar tienen cambios constantes en cuanto a datos que tendrían que guardarse, y migrar los datos y columnas de los usuarios aumenta siempre la complejidad de la historia de usuario.

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.

A pesar de todas las falencias de desarrollo, el módulo tuvo resultados positivos por parte de los usuarios finales ya que es una herramienta que 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.

7.2. Aportes del proyecto

Se puede afirmar que la llegada del módulo curricular representa la tecnología al servicio de la educación, en específico para automatizar el proceso de diseño curricular.

Una buena parte de su importancia radica en las amplias posibilidades que ofrece, entre las cuáles las más sobresalientes son 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.

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.

- Al implementar el módulo en 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.
- Al permitir 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.
- Se redujo el tiempo promedio de formulación y revisión de cada flujo de diseño e implementación de flujos de trabajo, debido a que se facilitó llenar formularios utilizando información ya existente en el AMS.

7.3. Proyectos 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[1] 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 2.3. Por lo tanto, como trabajo futuro para el módulo curricular se busca catalogar los cursos de manera pública desde una interfaz web.

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

- **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.

Bibliografía

[1] *NET application architecture guide*. 2nd ed. Patterns & practices. OCLC: ocn320803280. Redmond, Wash.: Microsoft, 2009. ISBN: 978-0-7356-2710-9.

[2] George D. Kuh et al. *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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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. URL: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0470723378.

[8] 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.

[9] 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.

[10] Wil van der Aalst y Kees van Hee. *Workflow management: models, methods and systems*. eng. 1. MIT Press paperback ed. Cooperative information systems. OCLC: 255171394. Cambridge, Mass.: MIT Press, 2004. ISBN: 978-0-262-72046-5 978-0-262-01189-1.

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

[12] Megan Oakleaf, Jackie Belanger y Carlie Graham. «Choosing and Using Assessment Management Systems: What Librarians Need to Know». En: *ACRL* (abr. de 2013). URL: http://www.ala.org/acrl/

sites/ala.org.acrl/files/content/conferences/confsandpreconfs/2013/papers/OakleafBelangerGraham_Choosing.pdf.

[13] Cindy De Smet et al. «The design and implementation of learning paths in a learning management system». En: *Interactive Learning Environments* 24.6 (2016), págs. 1076-1096.

[14] 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.

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

[16] 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.

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

[18] Brad A. Myers y Andrew J. Ko. *The Past, Present and Future of Programming in HCI*. Feb. de 2009. URL: http://www.academia.edu/2669908/The_past_present_and_future_of_programming_in_HCI.

[19] Michael Wittig y Andreas Wittig. *Amazon Web Services in action*. OCLC: ocn934475856. Shelter Island, NY: Manning, 2016. ISBN: 978-1-61729-288-0.

[20] 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.

[21] Santosh Kumar y R. H. Goudar. «Cloud Computing – Research Issues, Challenges, Architecture, Platforms and Applications: A Survey». English. En: *International Journal of Future Computer and Communication* 1.4 (dic. de 2012), pág. 356.

[22] Jihyun Kang. «Web based development Framework for Customizing Java-based Business Logic of SaaS Application». En: *ICACT* (2012).

[23] Yashpalsinh Jadeja y Kirit Modi. «Cloud computing-concepts, architecture and challenges». En: *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*. IEEE. 2012, págs. 877-880.

[24] Jihyun Lee, Sungju Kang y Sung Jin Hur. «Web-based development framework for customizing Java-based business logic of SaaS application». En: *Advanced Communication Technology (ICACT), 2012 14th International Conference on*. IEEE. 2012, págs. 1310-1313.

[25] Sungjoo Kang, Sungwon Kang y Sungjin Hur. «A design of the conceptual architecture for a multitenant saas application platform». En: *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on*. IEEE. 2011, págs. 462-467.

[26] Qian Li, Shijun Liu y Ying Pan. «A cooperative construction approach for SaaS applications». En: *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*. IEEE. 2012, págs. 398-403.

[27] Dan Ma. «The business model of"software-as-a-service"». En: *Services Computing, 2007. SCC 2007. IEEE International Conference on*. IEEE. 2007, págs. 701-702.

[28] David C Chou y Amy Y Chou. «Software as a Service (SaaS) as an outsourcing model: An economic analysis». En: *Proc. SWDSI'08* (2008), págs. 386-391.
[29] 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.
[30] Charles G Cobb. *The project manager's guide to mastering Agile: Principles and practices for an adaptive approach*. John Wiley & Sons, 2015.
[31] 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.
[32] Jakob Nielsen. *Usability engineering*. eng. Nachdr. OCLC: 760142137. Amsterdam: Kaufmann, 2010. ISBN: 978-0-12-518406-9.
[33] 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.
[34] Governet. *CurricUNET*. http://curricunet.com. 2013.
[35] Leepfrog Technologies. *Courseleaf*. https://www.leepfrog.com/courseleaf/. 2017.
[36] CSDC Systems. *DECA: Curriculum Navigator*. http://www.decisionacademic.com/products/curriculum-navigator/. 2017.
[37] Bruce Eckel. *Thinking in Java (4th Edition)*. 4.ª ed. Published: Paperback. Prentice Hall, feb. de 2006. ISBN: 0-13-187248-6. URL: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0131872486.
[38] Kathy Sierra y Bert Bates. *Head first java*. .ºReilly Media, Inc.", 2005.
[39] Ken Arnold, James Gosling y David Holmes. *The Java programming language*. Addison Wesley Professional, 2005.
[40] Charlie Hunt y Binu John. *Java performance*. Prentice Hall Press, 2011.
[41] Russell JT Dyer. *Learning MySQL and MariaDB: Heading in the Right Direction with MySQL and MariaDB*. .ºReilly Media, Inc.", 2015.
[42] Mikael Ronstrom y Lars Thalmann. «MySQL cluster architecture overview». En: *MySQL Technical White Paper* (2004).
[43] Andreas Wittig y Michael Wittig. *Amazon Web Services in Action*. Manning Publications Co., 2015.
[44] Jon Loeliger y Matthew McCullough. *Version Control with Git: Powerful tools and techniques for collaborative software development*. .ºReilly Media, Inc.", 2012.
[45] Scott Chacon y Ben Straub. *Pro git*. Apress, 2014.
[46] Christian Bauer y Gavin King. «Hibernate in action». En: (2005).
[47] Lukas Ruebbelke y Brian Ford. *AngularJS in action*. Manning, 2015.
[48] Peter Bacon Darwin y Pawel Kozlowski. *AngularJS web application development*. Packt Publ., 2013.
[49] Dean Leffingwell. *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional, 2010.
[50] Lisa Crispin y Janet Gregory. *Agile testing: A practical guide for testers and agile teams*. Pearson Education, 2009.