UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja

Máster universitario en Ingeniería de Software y Sistemas Informáticos

Desarrollo del módulo de orientación académica, para el software *Open Curricular Management System*-OCMS.

Trabajo Fin de Máster

Presentado por: Gallego Alzate, Duván Alexander

Directores: De Oleo Moreta, Cinthia – Quintero, Juan Bernardo

Ciudad: Medellín
Fecha: 14 de febrero del 2018

Resumen

Buscando subsanar la necesidad que existe actualmente dentro de muchas universidades en cuanto a la administración informática curricular, mediante este trabajo se busca la creación del módulo de orientación académica, para el software *Open Curricular Manager System* (OCMS), el cual es un software open source que busca llenar el vacío existente allí actualmente. El desarrollo de este módulo se realiza aplicando la ingeniería dirigida por modelos, a través de transformaciones M2T con la herramienta MarTE.

Luego para lograr determinar el correcto funcionamiento de este al igual que su nivel de madurez para salir a producción, se realiza una evaluación por medio de un grupo de expertos, quienes nos aportan su experiencia y conocimiento para el mejoramiento del módulo y en general de todo el aplicativo.

Palabras Clave: OCMS, modelos, desarrollo, MarTE

Abstract

Seeking to fill the need that currently exists within many universities in terms of curricular computer management, through this work we are looking for the creation of the academic guidance module for the Open Curricular Manager System (OCMS) software, which is an open source software that seeks to fill the existing vacuum there currently. The development of this module was done by applying model-driven engineering, through M2T transformations with the MarTE tool.

Then to determine the correct functioning of this as well as its level of maturity to go to production, an evaluation is carried out by means of experts group, who provide us with their experience and knowledge for the improvement of the module and in general of the whole application.

Keywords: OCMS, Models, development, MarTE

Índice de contenidos

Índice de tablas

Índice de figuras

DEFINICIONES Y ACRÓNIMOS

OCMS: *Open Curricular Management System*.

UOC: Unidad de Organización curricular.

***Opensource*:** Software de código abierto, es decir, software que se distribuido y desarrollado libremente.

***Framework*:** Marco de trabajo que provee una infraestructura genérica para determinados problemas en un desarrollo de software.

SaaS: *Software As A Service* – Software como servicio.

IDE: Entorno de desarrollo integrado.

GMF: *Graphical Modeling Framework*.

EMF: *Eclipse Modeling Framework*.

GEF: *Graphical Editor framework*.

ATL: *Atlas Transformation Language*.

ADT: *Atlas Development Tools*.

API: *Application programming interface.*

MDD: *Model Driven Architecture*.

MDSD: *Model Driven Software Development.*

MDD: *Model Driven Development* .

UML: Lenguaje Unificado de Modelado.

DSL: *Domain Specific Language.*

GPL: *General Program Language.*

1. Introducción

El objetivo principal de este TFM, es la creación del módulo de orientación académica para el software *Open Curricular Management System* – OCMS, por medio de la herramienta MarTE, a través de transformaciones M2T, como una aplicación práctica de la ingeniería dirigida por modelos.

Debido a que en la actualidad ya existen en el mercado varios sistemas informáticos para la administración curricular, el valor agregado que se pretende dar con este desarrollo es la liberación de todo su código fuente, para un licenciamiento abierto que permita la utilización y modificación de este sin ningún tipo de restricciones.

Dentro de los diferentes capítulos del trabajo, se desarrollan todos los pasos seguidos para lograr la creación de este módulo, hablando inicialmente sobre la justificación, el planteamiento del trabajo y la estructura de la memoria, luego se desarrolla todo el contexto y el estado del arte, donde se tocan temas como la arquitectura dirigida por modelos, el desarrollo de software dirigido por modelos y se habla también un poco sobre diferentes herramientas de gestión curricular, para poder tener un punto de comparación sobre lo que se está construyendo.

Al terminar todo el desarrollo del estado del arte, se trazan los objetivos general y específicos y se inicia con la parte más práctica del trabajo en el desarrollo específico de la contribución, donde se presenta la arquitectura utilizada, los requisitos identificados, los mockups creados con base en los requisitos, el modelo del módulo, la generación del código fuente, los ajustes que se le realizaron a este y finalmente la evaluación por medio del juicio de expertos que se le realizó al módulo.

Si bien el presente TFM es un trabajo práctico, debido a que este se desarrolla por medio de la ingeniería dirigida por modelos, se buscó ahondar un poco más en este tipo de metodología, para lograr entender cuáles son las verdaderas ventajas de trabajar con ella vs las metodologías y formas de trabajo que utilizan actualmente la mayoría de desarrolladores, logrando con esto último plasmar al final varias conclusiones al respecto, antes de las conclusiones generales y las líneas de trabajo futuro.

1.1. Justificación

Los sistemas de información son al día de hoy una pieza fundamental dentro de cualquier organización debido a la complejidad que manejan estas. Las universidades como organizaciones no son ajenas a dichas complejidades que pueden ser incluso mayores, debido a la continua demanda de información e interacción que quieren tener sus alumnos, partiendo del hecho que la mayoría de estos son jóvenes que por lo general están a la vanguardia de la tecnología.

Es así como las tendencias en enseñanza están cambiando, obligando a los modelos educativos a estar cada día más ligados a los avances tecnológicos, para buscar obtener el máximo provecho de estos, de modo que se logre en este sentido un máximo nivel de conocimientos y habilidades en los estudiantes, que es el objetivo final.

Si bien la solución a este problema ya se encuentra en lo que se conoce como software de administración curricular y de estos ya existen varios en el mercado, como se verá en el estado del arte, ninguno de ellos es libre y *open source*, o no se adaptan suficientemente a las necesidades institucionales; lo que obliga a las universidades que quieran tener un sistema de estos, a invertir grandes cantidades de dinero para poder obtenerlo, dinero con el que muchas universidades no siempre cuenta.

La relevancia de este problema surge entonces desde dos necesidades, la primera es la necesidad de las universidades de la administración curricular teniendo en cuenta elementos de diseño curricular, estrategias de trabajo activo, innovación abierta y recursos de apoyo; y la segunda, es la necesidad de los alumnos de tener una mejor interacción con los sistemas de información que disponen las universidades para lograr así potencializar el conocimiento que adquieren allí.

1.2. Planteamiento del trabajo

Al problema con el que cuenta muchas universidades para la administración curricular, se le buscará dar una solución por medio del desarrollo del software *open source* OCMS (*Open Curricular Management System*); el cual una vez terminado permitirá a estas tener un programa completo para la administración curricular, de forma libre y gratuita, de modo que estas se puedan empoderar de el para así lograr una mejor interacción con sus currículos, con los estudiantes y con los profesores, por medio de los siguientes módulos:

- Módulo de gestión curricular.
- Módulo de banco de proyectos.

- Módulo de calendario académico.
- Módulo de orientación académica.
- Módulo de redes sociales.
- Módulo de configuración.
- Módulo de seguridad.

Lo que se propone en este trabajo y considerando el alcance con el que cuenta, es la construcción del módulo de orientación académica por medio de la ingeniería dirigida por modelos, a través de la herramienta MarTE.

El módulo de Orientación Académica se centra en las asesorías que los estudiantes pueden recibir por parte de la universidad, ya sea en el campo académico (como monitorias y asesorías con los profesores u otros estudiantes) o sobre temas como el currículo, elección de materias y organización de tiempo.

Para lograr el objetivo planteando, primero se realizará un análisis general de los programas que existen actualmente para la administración curricular, hecho que se documentará en el estado del arte; luego se buscará comprender correctamente el funcionamiento de la herramienta MarTE, por medio de la cual se creará el modelo del módulo que luego se transformará en el código fuente; posteriormente, se plantearan los requisitos funcionales y no funcionales, se crearán los *mockups* y las historias de usuarios, para usarlos como base del modelo del módulo que se va a desarrollar; con dicho modelo se procederá a generar el código fuente y se le realizarán ajustes de forma manual a este, para que se asemeje a los *mockups* planteados. Finalmente se realizará una evaluación del módulo por medio de validación de juicio de expertos, quienes darán una visión externa y objetiva sobre el funcionamiento de este.

Teniendo en cuenta la experiencia personal como desarrollador usando ingeniería clásica, para finalizar este trabajo, se buscará realizar una breve comparación de la experiencia obtenida desarrollando con ingeniería dirigida por modelos vs lo que normalmente se realiza en cualquier desarrollo de software.

1.3. Estructura de la memoria

La memoria creada durante el presente trabajo, se divide en 5 capítulos principales, por medio de los cuales se sintetizó todo el trabajo realizado durante la construcción del módulo de orientación académica para el software *Open Curricular Management System* - OCMS. A continuación, se realiza una breve descripción de estos junto con su contenido.

Introducción.

Dentro de este capítulo, se discute la justificación y se realiza el planteamiento del trabajo, mostrando el por qué se está realizando el desarrollo del software y cuáles son los problemas que busca solucionar, luego se plantea a modo de resumen todo el desarrollo que se realizó, para lograr cumplir con todos los objetivos de modo que se dé solución a la necesidad que se tienen actualmente.

Contexto y estado del arte.

Dentro del contexto y el estado del arte, se da a conocer inicialmente el contexto del negocio, donde se muestra la necesidad que tienen las universidades en la gestión curricular, luego se habla sobre el contexto del sistema, allí se plantea como el desarrollo que se busca realizar con este trabajo, cubre las necesidades que tienen las universidades. En el estado del arte, se realizó una recopilación del pensamiento de diferentes autores sobre la ingeniería dirigida por modelos, el desarrollo de software dirigido por modelos junto con una breve descripción de la herramienta MarTE y su arquitectura, la cual es utilizada durante el presente trabajo para el modelado y la generación del código fuente del módulo. Allí también se discute sobre diferentes herramientas de gestión curricular que existen actualmente y se realiza una breve descripción de las que demostraron estar muy bien estructuradas y tener un acaparamiento alto del mercado. Finalmente se plantean unas conclusiones del estado del arte, principalmente con respecto al aporte que se busca realizar con el desarrollo versus las características y funcionalidades que tienen las diferentes herramientas analizadas.

Objetivos y metodología de trabajo.

En este capítulo se plantean los objetivos, tanto el general como los específicos y se describe la metodología utilizada, la base del diseño metodológico y las fases que se abordaron dentro de este para el desarrollo de todo el trabajo.

Desarrollo específico de la contribución.

Aquí inicialmente se habla de la arquitectura desarrollada para la elaboración del módulo de orientación académica y en general de todo el software, tocando temas como los estilos arquitectónicos utilizados, la vista conceptual, la vista lógica y la vista física. Luego se identifican los requisitos funcionales y no funcionales, que derivan en los *mockups* e historias de usuarios que también se trabajan en este capítulo. Finalmente, desde un enfoque más práctico se detalla el modelo del módulo construido, la generación del código fuente a partir del modelo y la realización de ajustes manuales a este, para obtener una versión funcional del módulo.

Evaluación.

En este capítulo, inicialmente se habla sobre la evaluación y la necesidad de esta para determinar el nivel de madures del módulo para salir a producción, luego se habla sobre los evaluadores, la cantidad de estos utilizada junto con un breve perfil de los profesionales que la realizarán. Seguidamente se trazan y definen las fases que se seguirán durante todo el proceso de evaluación cuyos resultados se analizan al final del capítulo.

Conclusiones y trabajo futuro.

Dentro de este capítulo, al inicio se habla un poco sobre la experiencia obtenida por medio de la ingeniería dirigida por modelos vs el desarrollo tradicional, luego se encuentran las conclusiones generales derivadas del desarrollo del trabajo y al final de este, se describen las líneas de trabajo futuro que se pueden seguir, si se quiere realizar una continuación de este.

2. Contexto y estado del arte

2.1. Contexto del negocio

En general en todas las universidades existen varios frentes en cuanto a la gestión curricular, en los cuales para lograr un mejor desempeño y eficacia es necesario cambiar de un método tradicional y desarticulado, a un sistema que agrupe y gestione las diferentes asignaturas, haciendo un seguimiento de los proyectos que se realizan en cada una de estas, de modo que se pueda lograr así un seguimiento detallado de lo que los estudiantes aprenden y desarrollan en cada una de las asignaturas, al igual que su grado de participación en estas.

2.2. Contexto del sistema

OCMS es un sistema en línea de gestión curricular asistida por tecnologías de información, teniendo en cuenta elementos de diseño curricular, estrategias de trabajo activo, innovación abierta y recursos de apoyo.

En este contexto los profesores, coordinadores de las diferentes UOC (Unidades de Organización Curricular) y los jefes de programas pueden hacer uso de OCMS para definir el currículo de una asignatura, los horarios y profesores que dictarán las mismas, además de gestionar los proyectos que se llevan a cabo en cada una de las asignaturas, posibilitando de esta manera el desarrollo de estos proyectos de una forma secuencial al integrarlos en varias asignaturas, permitiendo que los estudiantes tengan la posibilidad de desarrollar software más completo y utilizando todos los conocimientos que adquieren en cada una de las asignaturas por las que cursan. Igualmente, mediante OCMS los estudiantes y profesores podrán crear sus respectivos equipos de trabajo del proyecto que estén desarrollando en determinada asignatura. Así mismo, los estudiantes pueden visualizar sus notas de seguimiento y evaluaciones, también como hacer uso de las asesorías para cada asignatura.

2.3. Estado del arte

Para establecer el estado del arte, inicialmente se ha recopilado información de diferentes autores sobre la arquitectura dirigida por modelos (MDA) y el desarrollo de software dirigido por modelos (MDSD) teniendo en cuenta que esos son los pilares sobre los que se sostiene todo el trabajo realizado, seguidamente se explica la arquitectura y en que consiste la

herramienta MarTE, por medio de la cual se realiza la diagramación de los modelos del módulo y la generación del código fuente a través de transformaciones; luego se ha llevado a cabo un estudio de las principales herramientas de gestión curricular existentes en el mercado, lo que ha dado origen a la realización de un análisis detallado de cada una teniendo como referencia los siguientes puntos que se pueden considerar claves: escalabilidad, integración con sistemas legados, costos de adquisición y mantenimiento, soporte, portabilidad, documentación, facilidad de uso, experiencia en el sector, casos de éxito, robustez y/o ligereza.

Finalmente se presenta una tabla de cada software con cada uno de estos aspectos, resumiendo así las principales ventajas y desventajas de estos para posteriormente obtener las conclusiones que se detallan en el apartado 2.4 Conclusiones del estado del arte.

2.3.1 Arquitectura dirigida por modelos (MDA)

La arquitectura dirigida por modelos inicia con la ya muy conocida idea que busca separar las especificaciones funcionales de un sistema, de los detalles y las tecnologías de su implementación. MDA se puede especificar como un *framework* de desarrollo definido por la *Object Management Group* (OMG), que centra sus esfuerzos en darle mayor relevancia a los modelos dentro de todo el proceso de desarrollo, pasando estos de ser modelos estáticos a ser modelos dinámicos que constituyen el centro de todo el desarrollo (Kleppe, Warmer, & Bast, 2003).

Con nuevos métodos y técnicas de desarrollo surgiendo día a día, en MDA podemos encontrar un camino neutral que nos permite un rápido desarrollo e implementación de nuevas especificaciones y tecnologías, que nos brindan una solución estructurada e interoperable de aplicaciones y portabilidad en el futuro (Truyen, 2006).

Así MDA proporciona un enfoque y herramientas para especificar un sistema independiente de la plataforma que lo soporta, luego para una plataforma específica y finalmente transforma estas especificaciones en un sistema para una plataforma particular.

Un aspecto fundamental de MDA es la capacidad que tiene para abordar el ciclo de vida del desarrollo de software, abarcando temas como el análisis y diseño, la programación, las pruebas, la implementación y el mantenimiento (OMG, 2003).

Los tres objetivos principales de MDA son la portabilidad, la interoperabilidad y la reutilización a través de la separación arquitectónica de conceptos.

En la figura 3, la cual es adaptada de (Quintero & Anaya, 2007), se ilustran los 4 tipos de artefactos que se manejan en MDA, identificando la importancia que tienen los modelos y las transformaciones en este tipo de arquitectura.

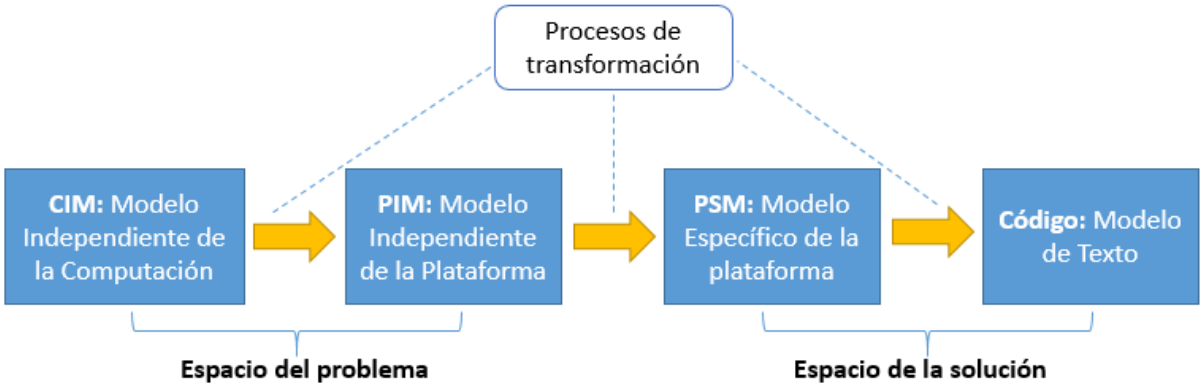

Figura 1. Transformaciones y modelos en MDA. Adaptada de (Quintero & Anaya, 2007)

Finalmente como lo plantea (Kleppe, Warmer, & Bast, 2003), el ciclo de desarrollo utilizado en MDA es muy similar al utilizado dentro del desarrollo tradicional, cambiando solo los artefactos dado que aquí la gran mayoría son modelos que pueden ser entendidos por computadoras.

2.3.2 Desarrollo de software dirigido por modelos (MDSD)

Cuando se habla de desarrollo de software dirigido por modelos (MDSD), el cual también es comúnmente conocido como *Model Driven Development* (MDD), se habla de un enfoque mucho más amplio y flexible al que plantea MDA debido a que este no está sujeto a sus estándares, lo que nos permite por ejemplo utilizar cualquier tipo de meta modelo como DSL y no solo UML y perfiles, ayudándonos de esta manera a definir nuestras propias ideas de PIMs y PDMs, según las necesidades y particularidades de cada proyecto. Para tener un mejor entendimiento de MDSD, debemos dejar claros varios conceptos que nos llevarán a definir los DSL, los cuales constituyen la base de MDSD.

El dominio.

Este puede estar compuesto por subdominios más pequeños y es siempre el punto de partida. (Stahl & Völter, 2006, pág. 56) Lo describe como un campo de interés o conocimiento limitado, que parte en la mayoría de las veces de una ontología con los conceptos del dominio.

El meta modelo.

Es una instancia del meta meta modelo y es el que nos ayuda a definir la estructura del dominio, combinando la sintaxis abstracta y semántica estática de un idioma.

El meta meta modelo.

El término meta es relativo, dado que en palabras simples, es un meta modelo que describe los conceptos que pueden ser utilizados para modelar un modelo. En consecuencia, el meta modelo debe tener un meta modelo que define los conceptos disponibles para el meta modelado (Stahl & Völter, 2006).

Lenguaje específico de dominio - DSL.

Un *Domain Specific Language* o lenguaje específico de dominio, es un idioma pequeño centrado es aspectos particulares de un sistema de software. Aunque no se pude construir un programa completo con solo un DSL, a menudo se juntan varios para formar un lenguaje de uso general (GPL) (Fowler & Parsons, 2011).

En general, como lo describe (Voelter, 2013), los lenguajes de uso específico (DSL) sacrifican la flexibilidad que puede ofrecer cualquier lenguaje de uso general (GPL), para ganar en productividad y en enfoque. Como el mismo autor lo dice, “Los DSLs son tan restrictivos, que solo permiten la creación de programas correctos”.

Un aspecto importante a destacar dentro de los DSLs es que dada la visión que manejan, pueden llegar a ser utilizados por personas que no son necesariamente programadores, sino expertas dentro del dominio; por ejemplo, un DSL de contabilidad podría llegar a ser fácilmente manipulado y operado por un contador, dado que la terminología que se manejará allí, se encuentra dentro de sus conocimientos específicos.

Para dejar clara la diferencia que existe entre los lenguajes de uso general y los lenguajes de uso específico, nos apoyamos en la tabla 6, la cual es tomada y traducida de (Voelter, 2013, pág. 31).

Tabla 1. Lenguajes de uso general vs lenguajes de uso específico

	GPLs	**DSLs**
Dominio	Grande y complejo	Pequeño y bien definido
Tamaño del lenguaje	Grande	Pequeño
Completo	Siempre	No necesariamente
Abstracciones definidas por el usuario	Bastante sofisticadas	Limitadas

Ejecución	Por medio de un intermediario	Nativa
Esperanza de vida	De años a décadas	De meses a años, según el contexto
Diseñado por	Gurús o equipos de trabajo	Unos pocos ingenieros o expertos en el dominio
Uso dentro de las comunidades	Grande, anónimo y generalizado	Pequeño, accesible y local.
Evolución	Lenta y a menudo llega a ser estandarizado	Rápida
Cambios incompatibles	Muy poco probables	Bastante probables

(Voelter, 2013, pág. 31)

2.3.3 MarTE

MarTE también conocido como Quorra-MTE, es un entorno de transformación de modelos construido sobre el IDE eclipse que pretende aumentar la calidad y la productividad del desarrollo de software. Este entorno se centra en la construcción y transformación de modelos, lo que permiten a los desarrolladores automatizar tareas que muchas veces son monótonas, obteniendo así que estos se centren en el negocio en lugar de la tecnología que utiliza el desarrollo (Quintero J. B., 2017).

Partiendo del hecho que eclipse está basado en una arquitectura de *micro-kernels* que permiten ampliar sus funcionalidades a través de *plugins*, se integran allí todas las herramientas de UML2 para cargar y manipular los modelos al igual que algunos otros elementos que se destacan a continuación:

- **GMF** (*Graphical Modeling Framework*), el cual proporciona un conjunto de componentes e infraestructuras para desarrollar editores gráficos basados en EMF (*Eclipse Modeling Framework*) y GEF (*Graphical Editing framework*) (eclipse.org, 2017).

- **ATL** (*Atlas Transformation Language*) el cual hace parte del ADT (*Atlas Development Tools*) que proporciona una forma de producir una serie de modelos de destino a partir de un conjunto de modelos de origen. Un programa de transformación ATL se compone de reglas que definen cómo los elementos del modelo de origen coinciden y navegan para crear e inicializar los elementos del modelo objetivo a través de un

conjunto de reglas de transformación que se les aplican a estos. Este es el enfoque oficial de Eclipse para transformaciones M2M (*model-to-model*) (eclipse.org, 2017).

- **JET** es un componente de EMF para M2T (*model-to-text*) que cuenta con un motor de plantillas genéricas que utiliza un subconjunto de la sintaxis de JSP, para proveer una manera simple de expresar el código que se desea generar. Por medio de JET se pueden generar códigos fuente SQL, XML, Java entre otros (eclipse.org, 2017).

A grandes rasgos la arquitectura de MarTE se puede sintetizar el siguiente diagrama de arquitectura.

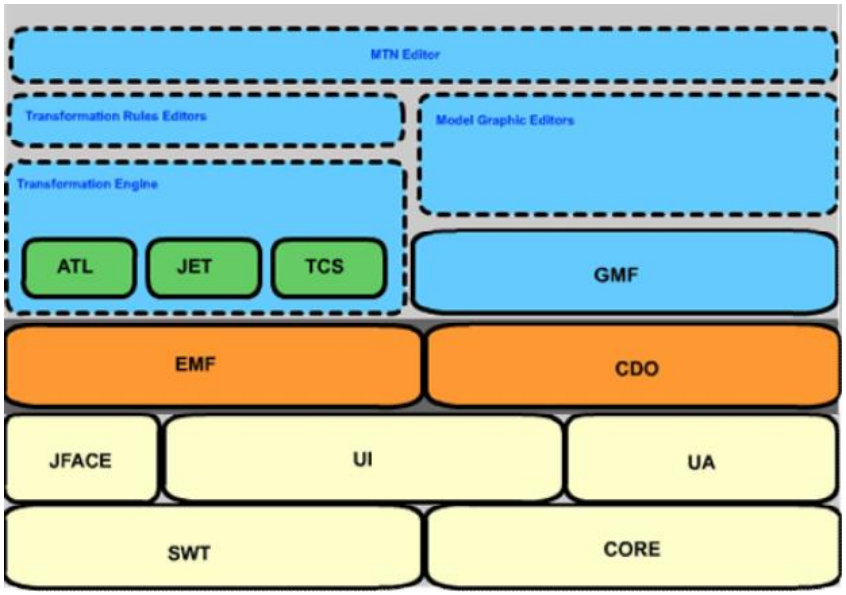

Figura 2. Diagrama de arquitectura de MarTE (Documento de arquitectura de MarTE)

2.3.4 Herramientas de gestión curricular existentes

Para lograr determinar el verdadero aporte que puede llegar a dar el desarrollo del software *Open Curricular Management system* – OCMS y en particular el módulo que se está desarrollando con el presente trabajo, se ha llevado a cabo un estudio que busca identificar las principales herramientas de gestión curricular que existen actualmente en el mercado, para tratar de identificar sus principales características, fortalezas y debilidades, logrando así observar el valor agregado que se le debe impregnar al desarrollo, para tener un factor diferenciador que haga que el aporte realizado sea realmente significativo.

La atención la centramos principalmente en las herramientas *Schoology*, *Kuali Student*, *Curriculog*, *CourseLeaf* que demuestran estar muy bien estructuradas y tener un acaparamiento del mercado bastante alto.

2.3.4.1 Schoology

Schoology, además de ser una plataforma educativa totalmente gratuita y en línea, permite crear y compartir contenido académico, tareas o evaluaciones como una experiencia de aprendizaje colaborativo, está orientada a la gestión curricular centralizada y gestión de contenido. Adicional a esto, Schoology es una plataforma muy fácil de usar en la cual se pueden crear cursos propios, organizar su respectivo contenido, proporcionar notas y tomar asistencia, los docentes pueden invitar a sus estudiantes a unirse al curso y a acceder a diferentes tipos de documentos, mediante un código generado automáticamente en la plataforma pudiendo gestionar así los diferentes currículos. (schoology.com, 2017)

Para tener una mejor visión del software nos registramos en la página con un rol de profesor, por medio del cual se pueden obtener las mayores ventajas de la herramienta.

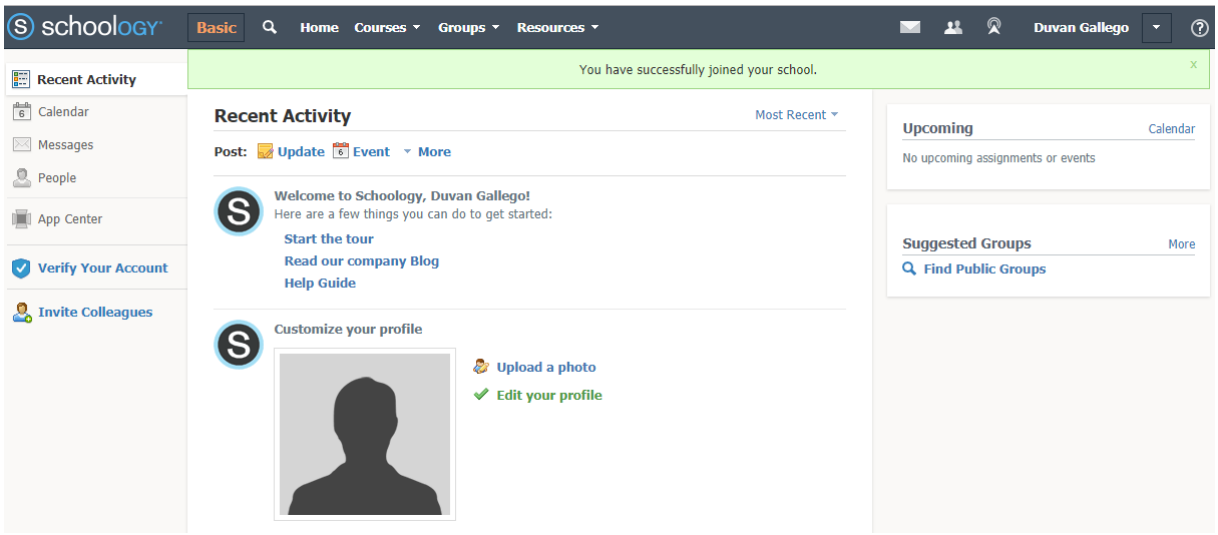

Figura 3. Vista general del software para gestión curricular Schoology

Si bien en la versión gratuita se cuenta con bastantes funcionalidades, para una universidad completa estas no serían suficientes, por lo que allí es donde se encuentra fundamentado su modelo de negocio, ofreciendo soporte, respaldo y espacio de almacenamiento por un costo mensual.

Tabla 2. Análisis detallado del software Schoology

Criterio	Justificación
Escalabilidad	Es una solución que ha demostrado que es escalable y fácil de implementar, ha crecido junto con la cantidad de instituciones que se han unido a ella en los pocos años que llevan.
	La plataforma abierta de integración hace que sea fácil de

Integración con sistemas legados	automatizar la transferencia de datos entre sistemas, por medio de aplicaciones personalizadas con base en su API.
Costos de adquisición y mantenimiento	En su versión básica y sin soporte, la herramienta es totalmente gratuita y se ofrece vía web y móvil. Para obtener soporte directo y mayor capacidad de almacenamiento, si se requiere pagar.
Soporte	Tiene un área de soporte online blog, centro de ayuda, apoyo a la comunidad y documentación de los desarrolladores para construir la integración con la API abierta. El soporte directo solo se obtiene cuando se paga por él.
Portabilidad	Schoology tiene la facilidad para funcionar tanto desde la web como desde móviles y tabletas.
Documentación	Cuenta con buena documentación en línea, tanto del software como tal como del API de integración que tiene. También tiene diseñadas FAQs y una guía de inicio, para que los diferentes usuarios puedan hacer uso de este de una forma más simple.
Facilidad de uso	Bastante fácil de usar, al momento de inscribirse el sistema le permite hacer al usuario un recorrido virtual por toda la aplicación. En general la interfaz de usuario se puede considerar bastante amigable.
Experiencia en el sector	Desde el 2007 entraron al mercado para reinventar la tecnología que se implementan en las aulas y desde entonces han tenido gran acogida en miles de maestros que organizan cursos para sus alumnos.
Casos de éxito	Miles de personas usan la herramienta para sus cursos, sin contar las instituciones reconocidas que lo utilizan como lo son: • Colorado State University-Global Campus • Minnetonka Public Schools • Harlem Academy
Robustez y/o ligereza	La aplicación es bastante robusta y a su vez cuenta con las aplicaciones móviles que compactan de una manera simple su funcionamiento.

(Elaboración propia)

2.3.4.2 Kuali Student

Detrás de Kuali Student se encuentra la fundación Kuali, la cual es una organización sin ánimo de lucro dedicada al desarrollo de soluciones de software administrativo de código abierto para la educación superior, buscando ayudar a las universidades a mantener su dinero en su misión reduciendo significativamente los costos administrativos.

Kuali Student ayuda a los administradores a gestionar todos los aspectos del plan de estudios y la gestión curricular, ya sea que se ofrezcan cursos y programas tradicionales o se forje nuevos caminos para el aprendizaje. (kuali.co, 2017)

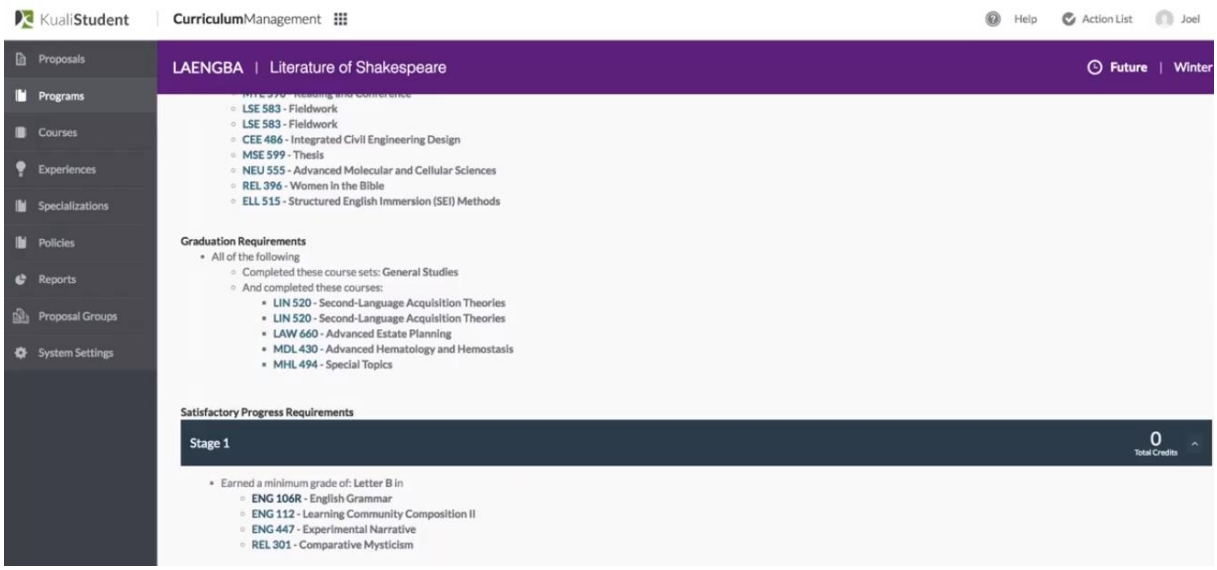

Figura 4. Vista general del software para gestión curricular Kuali Student

El modelo de negocio de Kuali Student se basa en un modelo de membresías anuales, por medio de la cual se tiene acceso a servicios que presta la fundación como lo son el soporte y la asesoría. Si bien este software es *open source*, para usted poder acceder al código y realizar modificaciones, debe contar con la membresía.

Tabla 3. Análisis detallado del software Kuali Student.

Criterio	Justificación
Escalabilidad	Kuali en el servicio de gestión curricular es altamente escalable y cuenta con universidades como el Boston College con 15.000 estudiantes y la universidad de la India con más de 100.000 estudiantes que respaldan esta afirmación.
Integración con sistemas legados	Debido a que las universidades que tienen membresía dentro de la fundación tienen acceso a todo el código fuente, estas pueden realizar las integraciones que requieran modificando

	esto. Adicional a esto, también cuentan con un API de integración para facilitar mucho más este trabajo.
Costos de adquisición y mantenimiento	Para poder acceder a la plataforma se debe contar con una membresía dentro de la fundación, por medio de la cual también se accede a opciones como el código fuente, soporte de la comunidad y la posibilidad de proponer mejoras para el software.
Soporte	Se provee por medio de la fundación, la cual a su vez se apoya en toda la comunidad que ayuda al sostenimiento del software.
Portabilidad	Kuali Student tiene la facilidad para funcionar tanto desde la web como desde móviles y tabletas.
Documentación	Cuenta con buena documentación online tanto en texto como en videos, que ayuda a que su curva de aprendizaje y adaptación sean bastante rápidas.
Facilidad de uso	Kuali Student cuenta con una interfaz intuitiva con estándares de desarrollo, que ayudan a la facilidad de su uso.
Experiencia en el sector	Teniendo en cuenta el listado de universidades que respaldan la fundación que está detrás de Kuali Student, se puede concluir que deben contar con muy buena experiencia.
Casos de éxito	Algunos de los casos de éxito más destacados son los siguientes: • Boston College • Universidad del Estado de Iowa • Universidad de Indiana • Gestión Navigator Partners, LLC • Universidad del Noreste de Sudáfrica • Universidad de Maryland (Collage Park) • Universidad de Toronto • Universidad de Utah • Universidad de Washington
Robustez y/o ligereza	Partiendo del hecho del nivel de transaccionalidad que debe suponer el tener funcionando este software en universidades de renombre con bastante cantidad de alumnos, se pude concluir que el software es bastante robusto y se encuentra en una muy buena etapa de madurez.

(Elaboración propia)

2.3.4.3 Curriculog

Curriculog es una interfaz en línea que permite que programas y cursos fuera del campus sean propuestos, creados, evaluados, revisados, aprobados e implementados. Los profesores y el personal que participan en la revisión a nivel de departamento, escuela y universidad pueden ver el progreso de sus propuestas de principio a fin (usc.edu, 2017).

Aunque de este software no se encontró mucha información debido a que es privado y pertenece a la universidad del sur de california, si se pudo notar que tiene varios de los componentes que estamos buscando dentro de los aplicativos de administración curricular.

Curriculog no cuenta con un modelo de negocio definido, dado que solo funciona dentro de la universidad.

Tabla 4. Análisis detallado del software Curriculog

Criterio	Justificación
Escalabilidad	La escalabilidad es bastante alta, si se tiene en cuenta que esta universidad cuenta con 33.000 estudiantes.
Integración con sistemas legados	Al ser un desarrollo propio, la integración con los sistemas de ellos es completa y transparente.
Costos de adquisición y mantenimiento	Los costos de este software van ligados a los costos del desarrollo como tal.
Soporte	El soporte es prestado directamente por el departamento de IT de la universidad del sur de california.
Portabilidad	El aplicativo solo cuenta con interfaz web.
Documentación	La documentación es bastante básica y solo se limita a temas puntuales.
Facilidad de uso	No se pudo obtener acceso a la plataforma, para evaluar este punto.
Experiencia en el sector	Ninguna, el software solo funciona para la universidad.
Casos de éxito	Al ser un desarrollo propio de la universidad del sur de california, el caso de éxito es solo ellos.
Robustez y/o ligereza	La robustez no es tan alta, debido a que solo cuenta con la funcionalidad de administración y creación de programas.

(Elaboración propia)

2.3.4.4 CourseLeaf

Con CourseLeaf, los profesores y el personal participan más en el proceso del plan de estudios con reuniones más cortas, formularios intuitivos y comentarios más rápidos sobre los cambios. CourseLeaf también ahorra tiempo y reduce la ineficiencia en el proceso de gestión del plan de estudios de principio a fin.

Los principales colegios y universidades que usan el sistema de gestión curricular de CourseLeaf, obtienen como resultado cursos y programas precisos que son consistentes en toda la institución desde el catálogo académico hasta el sistema de interacción estudiantil (courseleaf.com, 2017).

Algunas de las funcionalidades que ofrece este software son la administración de catálogos y workflows, todo el núcleo que es la administración curricular como tal, agendamiento de cursos y búsqueda de cursos.

Su modelo de negocio se basa en SaaS alquilando el software por medio de un costo mensual.

Tabla 5. Análisis detallado del software CourseLeaf

Criterio	Justificación
Escalabilidad	Teniendo en cuenta que lo utilizan más de 200 universidades, la escalabilidad que tiene deber ser muy alta.
Integración con sistemas legados	No se pudo encontrar mucha información al respecto, debido a que la documentación disponible es bastante básica.
Costos de adquisición y mantenimiento	Depende del plan que se contrate según el número de usuarios que van a utilizar el aplicativo y el tamaño de la universidad. Ellos cuentan con planes para universidades pequeñas y de bajo presupuesto, al igual que planes para universidades grandes con presupuestos más elevados.
Soporte	El soporte es directamente con la empresa leepfrog, dueña del software.
Portabilidad	Solo cuenta con plataforma web.
Documentación	No se pudo encontrar información suficiente en línea.
Facilidad de uso	Debido a que no se tuvo acceso a una demo del producto, la facilidad de uso no se pudo juzgar apropiadamente.
Experiencia en el sector	Bastante alta y adicional a esto, cuenta con un gran reconocimiento en el sector.

Casos de éxito	Según la página oficial, este software se encuentra instalado en casi 200 universidades de estados unidos, dentro de las que se destacan algunas como: • Universidad Standford. • Universidad BENTLEY. • Universidad BIOLA. • Universidad de colorado. • Universidad del estado de Georgia.
Robustez y/o ligereza	La robustez es bastante alta, debido a la cantidad de módulos que tiene integrados.

(Elaboración propia)

2.3.4.5 Otros sistemas de administración curricular

Dentro del estudio del estado del arte se realizó la investigación de algunos otros softwares que no se detallaron, dado que esto no contribuía en mucho a mejorar el estado del arte porque tienen características similares a los ya analizados. A modo de resumen los listo a continuación junto con una pequeña descripción y algunos comentarios, para buscar así dar a conocer la gran variedad de software de este tipo que existe en la actualidad.

Tabla 6. Otros sistemas de administración curricular.

Nombre	Descripción	Comentarios
Workday	Workday es un ERP especializado en la administración de las universidades, que ofrece flexibilidad, agilidad y respuesta rápida al cambio. (workday.com, 2017)	Debido a que sus características son más del tipo ERP, no se realizó un análisis profundo de él, a pesar de que cuenta con un módulo especializado en la gestión curricular.
Curriculum management system (CMS) by Ulster	Este software fue desarrollado por la universidad de Ulster y proporciona una fuente única de información relacionada con todos los programas. El personal puede crear, mantener e imprimir especificaciones de los programas, descripciones de módulos y otra documentación	No se realizó un análisis profundo de él, debido a que cuenta con las mismas características que tiene el software curriculog, que fue analizado en el apartado 2.3.3

	para eventos de evaluación y revalidación. El CMS es propiedad y está gestionado por la oficina académica con soporte en aspectos técnicos y de interfaz de usuario por el departamento informático de la propia universidad. (ulster.ac.uk, 2017)	
Software integrado de gestión académica SIGA	Es un sistema modular para la administración académica y curricular, diseñado especialmente para instituciones de educación superior, funciona completamente en Internet e integra tanto datos como procesos en una solución completa, eliminando así barreras de espacio y tiempo. (datasae.co, 2017)	Aunque este software cuenta con un módulo de gestión curricular, está muy enfocado en el registro y control académico, manejando temas como matrículas, planes de estudio y gestión estudiantil.

(Elaboración propia)

2.4 Conclusiones del estado del arte

Luego de analizados varios aplicativos de gestión curricular, se puede notar que la gran mayoría se enfocan principalmente en la gestión de los programas y hacen a un lado todo el módulo de interacción estudiantil, lo que conlleva a tener un software más de uso interno que un software realmente colaborativo que fomente e impulse la educación de los estudiantes.

Dentro de los diferentes aplicativos analizados, Schoology es el que más se acerca a lo que se pretende realizar con el desarrollo de *Open Curricular Management System*; sin embargo, aunque inicialmente se ofrece como gratuito, tiene un costo cuando se trabaja dentro de un ambiente universitario que conlleva el tener gran cantidad de usuarios y cursos.

Si bien algunas universidades han optado por crear su propio sistema de administración curricular, las funcionalidades y módulos con los que estos cuentan solo solucionan problemas específicos de cada una de ellas.

Al analizar los diferentes modelos de negocio se pude concluir que ninguno es completamente libre y de código abierto, el único que se acerca un poco a lo que se busca crear con el desarrollo del software *Open Curricular Management System* en este sentido es Kuali Students, siempre y cuando se cuente con la membresía que se requiere para poder utilizarlo y modificarlo libremente.

En estos ítems es donde se pretende tener el valor agregado del desarrollo que se va a realizar, ofreciendo un software de administración curricular enfocado en la interacción de los alumnos, completamente gratis y de código abierto, de forma que las universidades lo utilicen sin ningún tipo de restricción e incluso lleguen a un punto en el que se empoderen de él para realizar también aportes que ayuden con el crecimiento del mismo.

3. Objetivos y metodología de trabajo

3.1. Objetivo general

Desarrollar el módulo de orientación académica, para el software *Open Curricular Management System* (OCMS) por medio de la ingeniería dirigida por modelos, a través de la herramienta MarTE.

3.2. Objetivos específicos

- Analizar y comprender en buen nivel de detalle, los programas de administración curricular.
- Explorar el funcionamiento de la herramienta MarTE.
- Estructurar los requisitos funcionales y no funcionales con los que cuenta el módulo.
- Crear los *mockups* y las historias de usuario del módulo con base en los requisitos funcionales y no funcionales.
- Establecer el modelo completo para el módulo.
- Obtener del código fuente de todo el módulo, a partir del modelo creado por medio de la herramienta MarTE.
- Realizar ajustes manuales al código fuente obtenido, de modo que se asemeje a los *mockups* creados.
- Indagar sobre la calidad del producto obtenido, por medio de la validación de juicio de expertos.
- Realizar una comparación de la experiencia obtenida con ingeniería clásica, versus el desarrollo por medio de la ingeniería dirigida por modelos.

3.2. Metodología de trabajo

La metodología por medio de la cual se va a realizar el desarrollo del módulo y en general el desarrollo de todo el software, es una metodología que no es muy utilizada al día de hoy por las grandes empresas de desarrollo, pero que es bastante interesante de abordar y trabajar debido a las grandes ventajas que provee.

Es así como se trabajará por medio del desarrollo de software dirigido por modelos (MDSD), tomando algunos lineamientos del *framework* de trabajo definido por la OMG en la arquitectura dirigida por modelos (MDA), levantando todos los requisitos del módulo que se

va a desarrollar, luego realizando mockups del mismo módulo, seguidamente creando los modelos con UML sobre la herramienta MarTE y luego a partir de allí, aplicando transformaciones M2T que darán como resultado una primera versión de los scripts necesarios para la creación de la base de datos sobre MySQL y el código fuente funcional trabajando sobre el *framework* Prado para PHP, el cual se modificará, adaptará y se le aplicarán los estilos necesarios, para que quede muy parecido a los mockups que se diseñaron desde el inicio. Finalmente se realizará una evaluación que ayudará a determinar el nivel de madurez que tiene el módulo para salir a producción. En la figura 5, se puede apreciar un resumen de todos los pasos que se abordarán para la elaboración del módulo, que fueron descritos ligeramente anteriormente y que se irán desarrollando durante el presente trabajo.

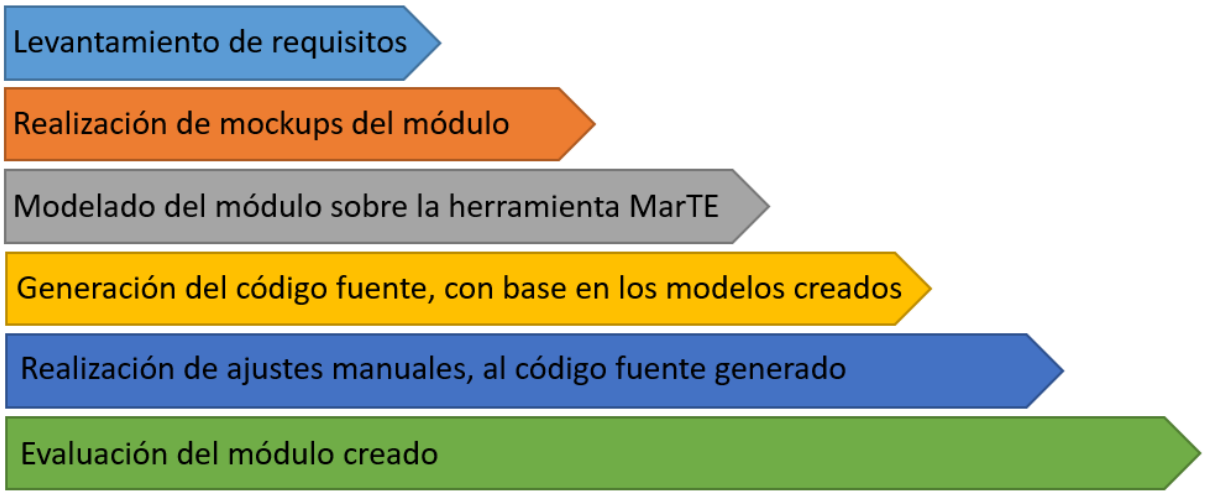

Figura 5. Resumen de todos los pasos que se abordaran para la elaboración del módulo (Elaboración propia)

La base del diseño metodológico utilizado es el propuesto por (Hevner, March, Park, & Ram, 2004), el cual tiene como objetivo proporcionar una comprensión por medio de directrices claras sobre cómo se deben conducir, ejecutar, evaluar y representar las investigaciones y los diseños en los sistemas de información. En la figura 6, se pueden apreciar las 4 fases por medio de las cuales se estructurarán las actividades enmarcadas en los objetivos específicos, que darán como resultado la obtención del objetivo general.

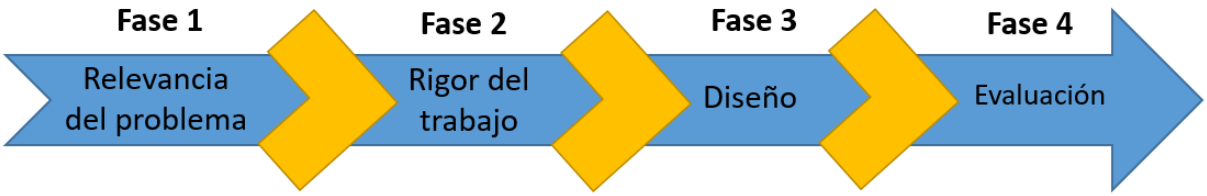

Figura 6. Fases en las cuales se enmarcan los objetivos específicos (Elaboración propia)

A continuación, se realiza una explicación detallada de cada fase, junto con las actividades que se van a trabajar en ellas.

Fase 1: Relevancia del problema.

La relevancia del problema consiste en desarrollar soluciones basadas en tecnología, para problemas de negocios importantes (Hevner, March, Park, & Ram, 2004).

Para lograr determinar esta, se realizará un estudio de los sistemas informáticos de administración curricular que existen actualmente, buscando comprender estos al igual que los problemas que buscan solucionar, determinando así el valor agregado que se puede ofrecer al realizar un desarrollo desde cero.

Fase 2: Rigor del trabajo.

El rigor del trabajo se basa en la aplicación de métodos rigurosos, tanto en la construcción como en la evaluación de todos los artefactos (Hevner, March, Park, & Ram, 2004).

Dentro de la aplicación de la fase del rigor, se buscará inicialmente comprender en un alto nivel el funcionamiento de la herramienta MarTE, luego se estructurarán todos los requisitos funcionales y no funcionales que desencadenarán en el modelo completo del módulo, el cual se realizará también dentro de esta fase.

Fase 3: Diseño.

La fase de diseño debe producir artefactos viables, que constituirán la base para la fase de evaluación (Hevner, March, Park, & Ram, 2004).

En la ejecución de esta fase se crearán los *mockups* del módulo, las historias de usuario, se obtendrá el código fuente con base en el modelo creado dentro de la fase de rigor y se realizarán los ajustes manuales que se deban realizar en este para obtener el módulo final funcionando.

Fase 4: Evaluación.

Debido a que la utilidad, calidad, eficiencia y eficacia de todos los artefactos de software debe ser rigorosamente demostrada a través de métodos bien ejecutados (Hevner, March, Park, & Ram, 2004), luego de tener el módulo funcionando se realizará una evaluación por medio de juicio de expertos, los cuales a través de una encuesta y sus comentarios, nos ayudarán a establecer puntos que se deben mejorar. Finalmente, en las conclusiones, se realizará una evaluación sobre la experiencia del desarrollo dirigido por modelos vs la ingeniería clásica.

4. Desarrollo específico de la contribución

4.1 Arquitectura

La arquitectura planteda para el módulo y en general para todo el software, se basa en 4 vistas que defienen aspectos en los siguientes frentes:

- Vista Conceptual: visión que los usuarios tienen de la aplicación.
- Vista Lógica: visión desde los principales elementos y principios del diseño.
- Vista Física: visión desde la distribución del procesamiento entre los dispositivos.
- Vista de Implementación: visión que muestra cómo serán montados los diferentes componentes de la aplicación y la forma en que interactúan.

4.1.1 Estilos de arquitectura usados

Los estilos de arquitectura que se van a utilizar en el desarrollo son el estilo 3-*Tier* / N-*Tier* y el estilo *layered*, buscando así aislar las funcionalidades en segmentos independientes, que permitan separar de forma clara las responsabilidades de cada *tiers*, pero con la ventaja que cada segmento pueda estar situado en un equipo distinto físicamente. También con estos estilos de arquitectura usados se trata de que cada *tier* de la aplicación, aísle las funcionalidades en segmentos separados que permitan delegar en forma clara las responsabilidades de cada *layer*, de modo que cada segmento tenga un propósito y una estrategia de implementación diferente.

Las implicaciones que tendremos con la implementación de estos estilos de arquitectura, serán la aparición de tres capas:

- *Front-end*: La cual será un explorador de internet y unas hojas de cálculo para la carga masiva de información.
- *Middleware*: La cual será un servidor web con PHP desplegado, utilizando el *framework* prado.
- *Back-end*: La cual será un gestor de bases de datos MySQL, que puede estar ubicado en el mismo servidor donde está el servidor web.

Otra de las implicaciones que se tendrá será la creación de paquetes, clases o utilitarios que mapearán los paquetes lógicos que representan las *layers* a el código de implementación de cada *tier*.

4.1.2 Vista conceptual

El siguiente diagrama, muestra todos los módulos o subsistemas que se identifican en el proyecto OCMS:

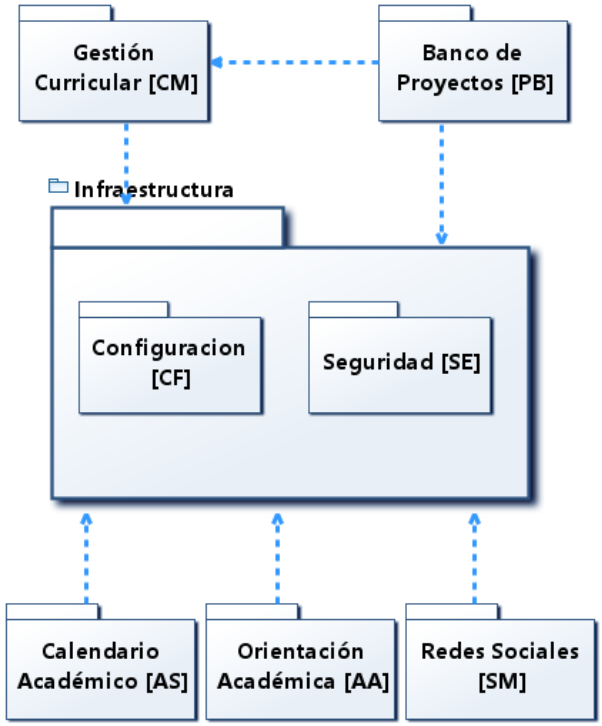

Figura 7. Diagrama de paquetes conceptuales (Documento de arquitectura de OCMS)

Si bien el alcance de este trabajo es solo el módulo de orientación académica, a continuación se realiza una breve descripción de todos los módulos de modo que se pueda tener una visión general del programa completo.

Módulo Gestión Curricular [CM]: Módulo de definición y gestión de referentes curriculares, en términos de planes de formación, UOC, asignaturas y contenidos particulares. A partir de estas definiciones, se permite llevar el registro de los referentes nacionales e internacionales que soportan cada área de conocimiento y los referentes temáticos del plan de formación.

Módulo Banco de Proyectos [PB]: Módulo de registro y seguimiento de proyectos que se desarrollan de manera particular en cada curso o de manera incremental como proyecto integrador a través de las UOC. De cada proyecto es posible definir los entregables o artefactos que evidenciarán el cumplimiento de los objetivos de cada proyecto, por cada estudiante, grupo o curso en general. Estos proyectos a su vez tendrán unas fases de madurez, partiendo desde su idea, análisis, diseño, implementación y pruebas.

Módulo Calendario Académico [AS]: Módulo para la planificación y gestión de cursos, horarios y profesores semestralmente. La planificación apoyará la definición de la programación de cursos por semestre (horarios, aulas, recursos), el plan de profesores, y la programación de exámenes finales y parciales.

Módulo Orientación Académica [AA]: Este módulo es el que se va a desarrollar durante el trabajo y se centra en las asesorías que los estudiantes pueden recibir por parte de la universidad, ya sea en el campo académico (como monitorias y asesorías con los profesores u otros estudiantes) o sobre temas como el currículo, elección de materias y organización de tiempo.

Módulo Redes Sociales [SM]: Módulo de gestión de comunicaciones virtuales y redes sociales con la comunidad académica. Esta gestión puede incluir también publicaciones, divulgación de información académica/profesional, bolsa de empleo, entre otros.

Paquete de infraestructura: Contiene los módulos que se encargan del back office del sistema, es decir, que realizan las actividades que no tienen contacto con el usuario, como por ejemplo la administración (listas personalizadas y tablas maestros) y la seguridad (autenticación y autorización de usuarios). Quedan pendientes la comunicación o mensajería y la auditoria.

- **Módulo Configuración [CF]:** Módulo que se encarga de la gestión de la información general, las páginas de contenido, lista de valores personalizadas, tablas paramétricas o maestras, entre otras.

- **Módulo Seguridad [SE]:** Módulo que se encarga de regular los permisos que posee cada usuario para modificar y visualizar la información contenida en el sistema, teniendo en cuenta los tipos de usuario permitidos para realizar cada acción

4.1.3 Vista lógica

A continuación, se lista la visión que tiene el desarrollador de la aplicación.

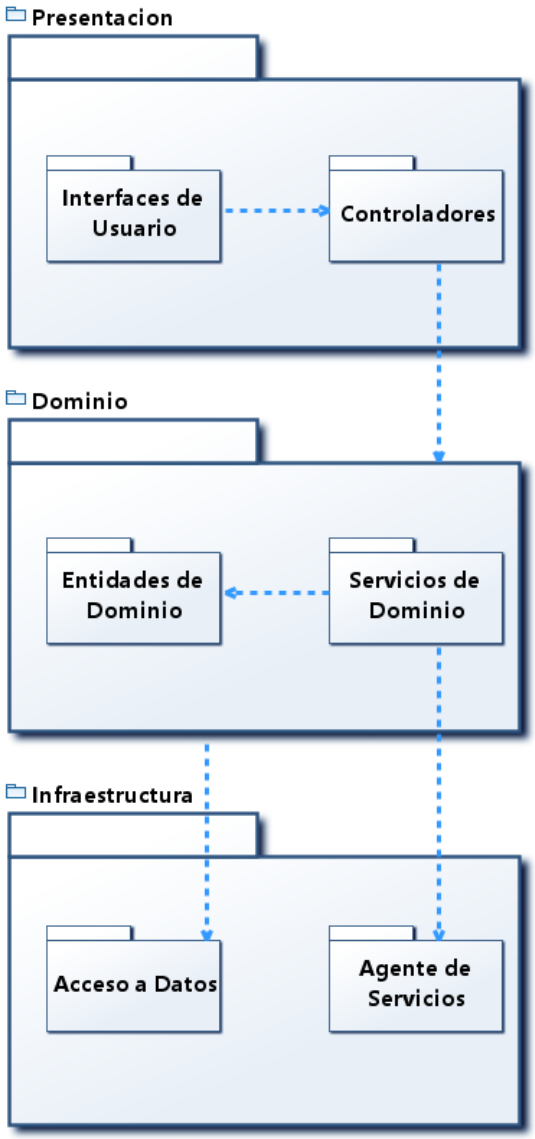

Figura 8. Diagrama de paquetes de desarrollo (Documento de arquitectura de OCMS)

Presentación: Posee toda la lógica de interacción contenida en la interfaz de usuario, está constituida por dos paquetes, el primero son las Vistas UI que son los formularios de la aplicación y el segundo son los controladores que capturan todas las acciones desencadenadas por los controles de las interfaces de usuario.

Dominio: Contienen las clases que constituyen elementos estructurales propios del sistema y está constituido por dos paquetes, el primero son los servicios de dominio, que se encarga de definir las acciones que derivan cada servicio de negocio que se deben prestar y el

segundo son las entidades de dominio que contienen los tipos de datos necesarios para llevar a cabo las acciones para el manejo de las tablas del sistema.

Infraestructura: Contiene las clases que se encargan de almacenar la información y consumir los servicios. Está constituido por dos paquetes que son, acceso a datos que contiene las clases de origen de datos con las labores propias de la conexión a la base de datos y agentes de servicio que se encargará de ser necesario, de llevar a cabo las acciones que involucran servicios web de terceros para integrar el sistema con otros.

4.1.4 Vista física

A continuación, se ilustra la disposición física de los componentes y artefactos de la aplicación web, mostrando el equipo en el que se despliega cada uno.

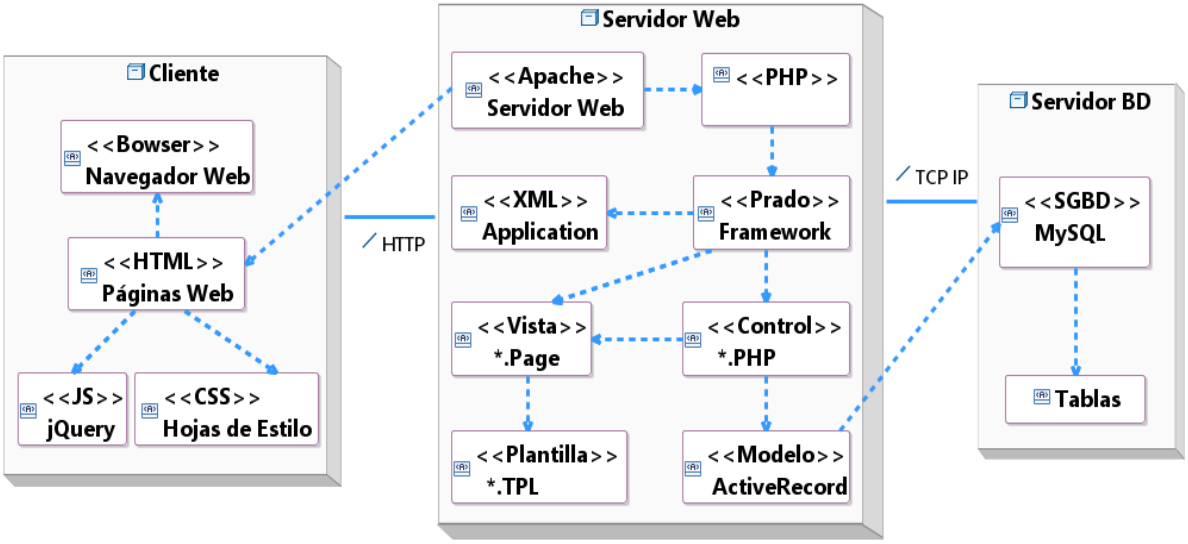

Figura 9. Diagrama de despliegue (Documento de arquitectura de OCMS)

Cliente (*Front-end*): Es un dispositivo con la capacidad de desplegar un navegador web.

Servidor Web (*Middleware*): Servidor ejecutando un servidor de despliegue de contenidos PHP usando el framework prado.

Servidor BD (*Back-end*): Servidor de bases de datos MySQL.

4.2 Identificación de requisitos

Ian sommerville (Sommerville, 2005), describe como los requerimientos de un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas para la solución de un problema puntual que refleja alguna necesidad que se tiene.

Partiendo de esta descripción y con base en entrevistas realizadas con el *product owner*, realicé el levantamiento de los requisitos funcionales, los cuales planteo específicamente para el módulo y los requisitos no funcionales, que teniendo en cuenta que son restricciones y atributos de calidad, los traigo heredados de los requisitos no funcionales de todo el software.

4.2.1 Requisitos funcionales

El módulo de orientación académica, se centra en las asesorías que los estudiantes pueden recibir por parte de la universidad, ya sea en el campo académico (Como monitorias y asesorías con los profesores u otros estudiantes) o sobre temas como el currículo, elecciones de materias y organizaciones de tiempo.

Los requisitos funcionales detectados para este módulo son los siguientes:

Tabla 7. Requisitos funcionales del módulo de orientación académica

Requisito	Descripción
Listar asesorías	Debe existir una pantalla donde se listen todas las asesorías, pudiendo filtrar allí por semestre y por asignatura. Este mismo listado tendrá varias vistas dependiendo del rol que acceda a él. **Rol estudiante:** Verá solo las asesorías en las que ha participado, o tiene programado participar, de modo que este pueda realizar la evaluación de estas. **Rol monitor:** Verá solo las asesorías en las que ha participado y las asesorías que ha dictado o va a dictar, de modo que pueda realizar la planeación de estas, llenar los listados de participantes y evaluar las que así lo requieran. **Rol profesor:** Verá las asesorías de las materias que está dictando o ha dictado. Adicional a esto, tendrá la posibilidad de crear nuevas asesorías, evaluar los monitores por asesoría y podrá editar y eliminar las asesorías existentes.

	Rol administrador: Verá todas las asesorías y tendrá todas las opciones disponibles de todos los roles.
Crear asesoría	Para realizar la programación de una asesoría, se le debe asignar a esta una materia y un monitor, luego un lugar, una fecha y una hora. Finalmente, se debe seleccionar el público objetivo de esta que puede ser un estudiante, un grupo de estudiantes creado previamente, un equipo de trabajo creado también previamente o debe existir la posibilidad de dejarla abierta, de modo que participen todas las personas que deseen asistir.
Planear asesoría	El listado de asesorías tendrá una opción por medio de la cual se realizará la planeación de estas, de modo que allí se puedan seleccionar los temas que se van a desarrollar, según el contenido con el que cuenta la asignatura.
Evaluar monitor	Dentro del listado de asesorías que verán los profesores o coordinadores de unidad de organización curricular UOC, deberá existir una opción por medio de la cual se pueda realizar la evaluación general de esta. Por medio de esta opción se entregará una calificación que va entre 0 y 5 según su criterio y también debe tener un cuadro de texto, donde se ingresarán algunas observaciones si llegasen a existir.
Evaluar asesoría	Dentro del listado que verán los estudiantes que han recibido una asesoría, tendrán una opción por medio de la cual podrán realizar la evaluación de las asesorías. Allí estos podrán seleccionar una calificación entre 0 y 5 según su criterio y podrán también digitar algunas observaciones por medio de un campo de texto, si estas llegasen a existir.
Reportar asistencia	Por medio de esta opción que se tendrá dentro del listado de asesorías, los profesores y monitores podrán reportar la asistencia a la asesoría de los grupos, equipos o estudiantes que se dijo que iban a existir al momento de crear esta.

(Documento de requisitos del software OCMS y elaboración propia)

4.2.2 Requisitos no funcionales

Los requisitos no funcionales que se tendrán en cuenta para el módulo, como lo describí en la introducción de los requisitos, vienen heredados de los requisitos de todo el software y son los siguientes:

Tabla 8. Requisitos no funcionales de todo el software

Requisito	Descripción
Usabilidad	El sistema debe ser fácil de usar e intuitivo por medio de interfaces amigables con la cantidad de opciones estrictamente necesarias.
Seguridad	Para acceder al sistema se debe contar con un usuario y una contraseña, de modo que todos los usuarios sean debidamente identificados y autenticados.
Rendimiento	El sistema debe soportar gran cantidad de información. En las tablas paramétricas de reducido volumen, el tiempo de respuesta no deberá superar los 3 segundos y en las tablas transaccionales de gran volumen, la respuesta no debe superar los 5 segundos.
Mantenibilidad	El sistema debe ser construido con base en estándares y enfoques de desarrollo que faciliten el desarrollo futuro.
Entendibilidad	El sistema deberá usar estándares de codificación y documentación que faciliten el entendimiento del sistema por parte de los desarrolladores para su soporte.
Interoperabilidad	El sistema deberá estar en la capacidad de integrarse con otros sistemas que se definan al momento del diseño y, además, debe permitir futuras integraciones sin que afecten sobremanera el diseño definido.
Portabilidad	Los servicios web deben ser diseñados para que sean fácilmente instalados en diferentes plataformas web que permitan usar IIS u otros como servidor web y que soporten diferentes sistemas operativos.
Disponibilidad	Los servicios web deben tener una disponibilidad de 24 horas diarias durante los 7 días de la semana y los 365 días del año. En caso de una caída los servicios deben estar disponibles nuevamente en un lapso de tiempo no superior a 12 horas.

(Documento de requisitos del software OCMS y elaboración propia)

4.3 *Mockups* del módulo de orientación académica e historias de usuario

Para la creación de los *mockups* se utilizó la herramienta JustInMind, por medio de la cual se plasmó en prototipos la información que se levantó en las entrevistas con el *product owner* que posteriormente se sintetizaron en los requisitos funcionales. Luego de tener estos prototipos terminados, se realizó la respectiva validación y ajustes necesarios de estos, para así obtener los siguientes *mockups* finales que se describen a continuación junto con las historias de usuario de cada uno, para de esta forma tener una mejor claridad de todas las funcionalidades que se implementarán en el módulo.

4.3.1 Listar asesorías

Dentro de esta pantalla lo que se busca es tener un listado completo de todas las asesorías, las cuales se mostrarán según el rol que tenga definido el usuario que ingrese.

En la parte superior se tendrán los filtros que se identificaron dentro de los requisitos funcionales; al lado derecho en la parte de arriba se ubicó el botón para crear nuevas asesorías y en la parte derecha dentro del listado al lado de la descripción de cada asesoría, se situaron los botones con las acciones que se le pueden realizar a cada una de estas.

Estos botones contienen las opciones para planear asesoría ( ), evaluar asesoría ( ), reportar asistencia ( ), evaluar monitor ( ), editar la asesoría ( ) y eliminar una asesoría ( ); la vista o no de estas opciones dependerá del rol que tenga el usuario y del estado en el cual se encuentre la asesoría.

Figura 10. *Mockup* listar asesorías (Elaboración propia)

4.3.2 Crear asesoría

Luego de dar *click* en el botón para crear una asesoría, se desplegará en la misma ventana un formulario con todas las opciones y campos necesarios para crear una nueva asesoría.

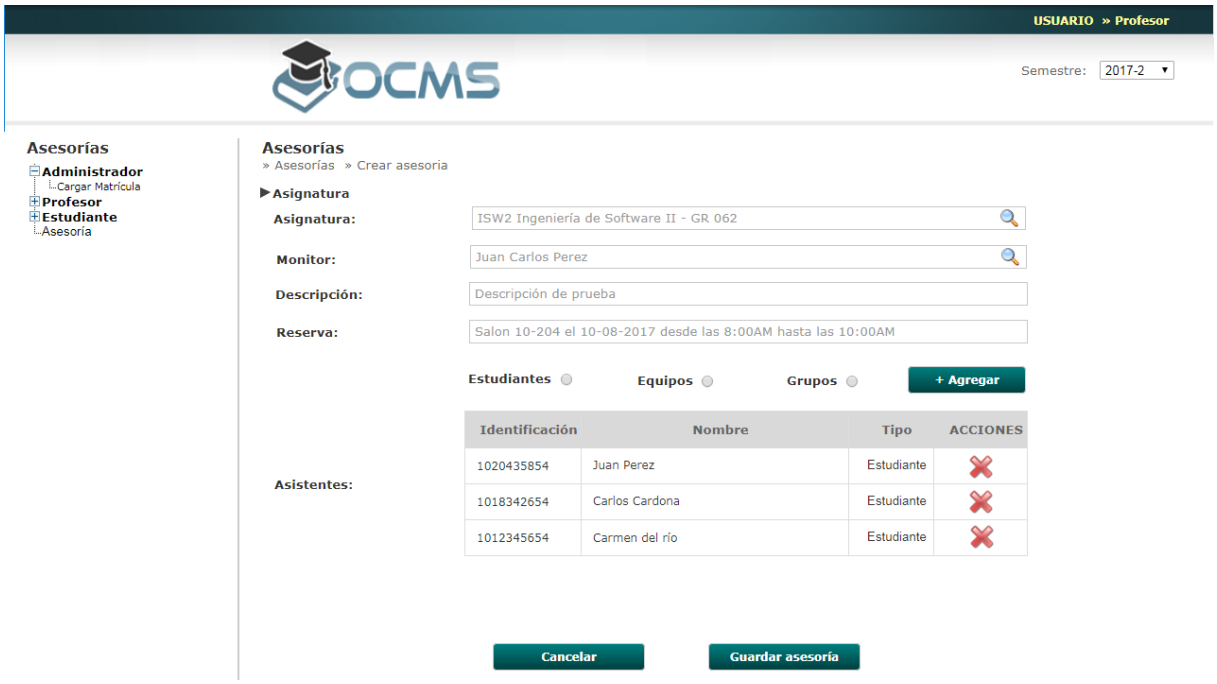

Figura 11. *Mockup* crear asesoría (Elaboración propia)

El primer campo es el selector de asignatura; este campo es de solo lectura y tendrá una lupa que al darle *click* mostrará una ventana modal por medio de la cual se podrá realizar la selección de la asignatura a la que se le desea crear la asesoría.

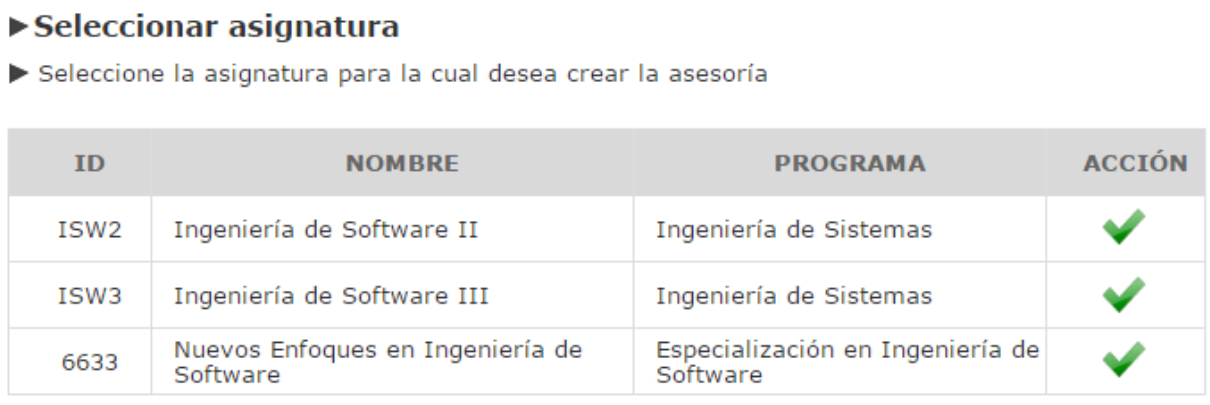

Figura 12. *Mockup* del modal para la selección de la asignatura (Elaboración propia)

Las asignaturas que se mostrarán dentro del *mockup* para la selección de asignaturas, son solo las asignaturas relacionadas con el usuario que se encuentra logueado en el sistema o

todas en caso de que el usuario sea un administrador o un coordinador. Estas en su parte derecha tendrán un *check* verde por medio del cual se realizará la selección de la asignatura deseada.

El campo para seleccionar el monitor tendrá un comportamiento similar al selector de asignatura, será también de solo lectura y al dar *click* sobre la lupa, se mostrará un modal para seleccionar al monitor que va a impartir esta asesoría.

Figura 13. *Mockup* del modal para seleccionar al monitor (Elaboración propia)

Los usuarios que se mostrarán en este modal son los estudiantes de la asignatura seleccionada previamente.

El campo posterior es el de lugar, fecha y hora de reserva; este campo será de tipo texto y allí se permitirá el ingreso de la información pedida.

Lego se tendrán 3 radio *buttons* para seleccionar a qué tipo de usuarios se les va a impartir la asesoría, estos pueden ser un grupo de usuarios que se debió crear previamente, un equipo que también se debió crear previamente o un estudiante; luego de la selección se habilitará el botón agregar, el cual creará una nueva fila dentro del listado de asistentes, donde se ingresará el código del grupo, del equipo o la identificación del estudiante el cual al coincidir con alguno de la base de datos, cargará el nombre y el tipo de forma automática. Al lado derecho de cada fila de los asistentes a la asesoría, se tendrá un botón para eliminar el estudiante, equipo o grupo ingresado.

Finalmente, en la parte inferior se tendrá un botón cancelar, que hará que el software regrese a la ventana con el listado de todas las asesorías y un botón de guardar, el cual guardará toda la información diligenciada en el formulario.

4.3.3 Planear asesoría

Por medio de esta opción se realiza la planeación de las asesorías, seleccionando los temas de la asignatura que se van a tratar allí. Al momento de dar *click* sobre el botón planear asesoría ( ), se desplegará un modal que lucirá de la siguiente manera:

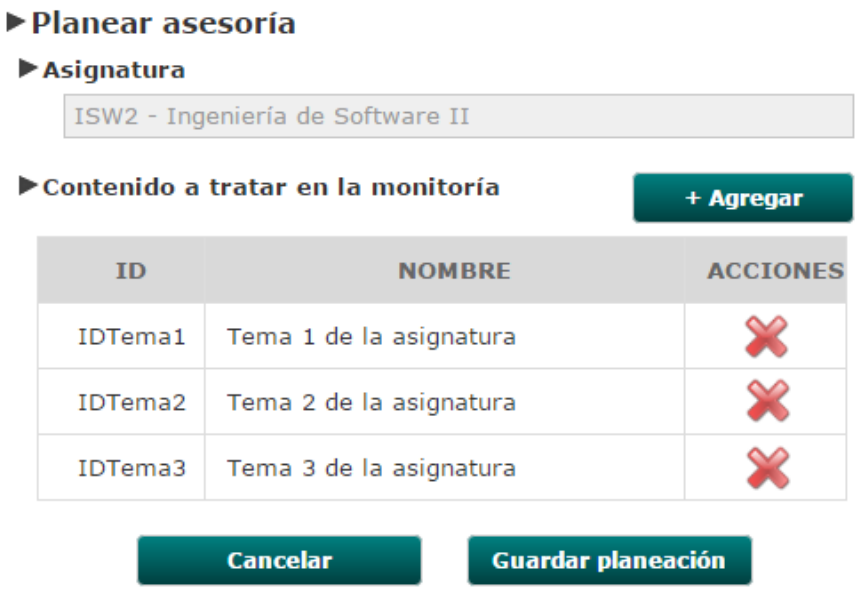

Figura 14. *Mockup* de modal para la planeación de la asesoría (Elaboración propia)

En la parte superior se traerá precargada la asignatura que ya se encuentra guardada dentro de la asesoría y seguidamente se encontrará un botón, por medio del cual se agregarán filas a la tabla, donde se irán seleccionando los temas de la materia que se trabajarán en esa asesoría. Al lado derecho de cada tema, se encontrará una equis roja que permitirá la eliminación de las filas que no se deseen.

Finalmente, en la parte inferior se encontrarán dos botones, uno que será cancelar, que cerrará el modal y otro con la opción guardar planeación, que guardará toda la información ingresada en el formulario.

4.3.4 Evaluar monitor

Por medio de esta opción se realizará la evaluación del monitor que dictó la asesoría, esta evaluación la realizarán los profesores o los coordinadores de las unidades organizativas. Al momento de dar *click* sobre el botón de evaluar monitor ( ), se desplegará una ventana modal con las siguientes opciones:

Figura 15. *Mockup* de modal para evaluar monitor (Elaboración propia)

Inicialmente se contará con una lista que permitirá seleccionar la calificación que se le quiere otorgar al monitor, luego un campo de texto donde se podrán ingresar las observaciones que se tengan y finalmente dos botones, uno botón para cancelar la operación que simplemente cerrará el modal y otro con la opción guardar evaluación, que guardará la información ingresada en el formulario.

4.3.5 Evaluar asesoría

Con esta opción se busca que los usuarios que asistan a una asesoría puedan realizar la evaluación de esta. Luego de dar *click* sobre la opción evaluar asesoría ( ), se desplegará el siguiente modal:

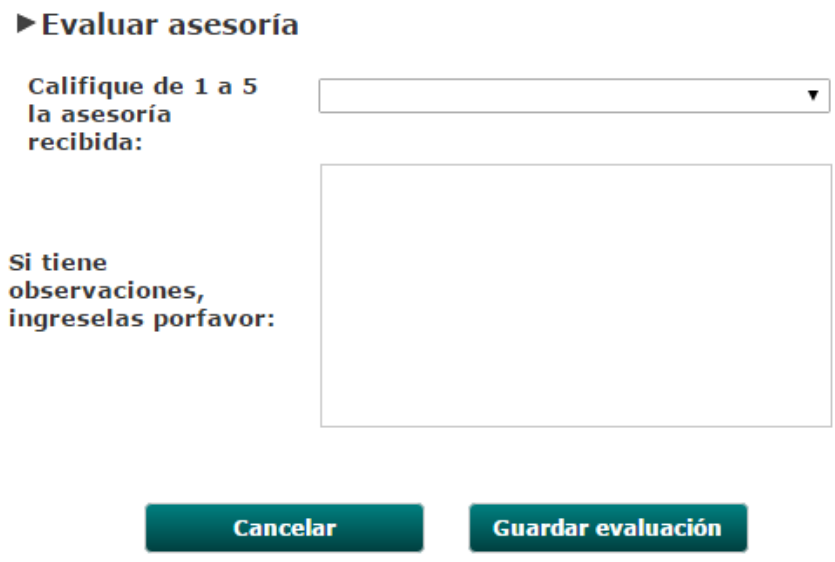

Figura 16. *Mockup* de modal para evaluar asesoría (Elaboración propia)

Las opciones que se tendrán dentro de este modal son muy similares a las que se tienen dentro de la opción evaluar monitor. Inicialmente se tendrá una lista donde se podrá seleccionar la nota que se le asignará a la asesoría y luego se tendrá un campo de texto donde se podrá ingresar la información de las observaciones si se tienen; finalmente se tendrán dos botones, uno con la opción cancelar que cerrará el modal sin hacer nada y otro con la opción guardar evaluación que guardará toda la información ingresada en el formulario.

4.3.6 Reportar asistencia

Por medio de esta opción el monitor que dicta la asesoría, pude reportar la asistencia de los usuarios a la asesoría. Luego de dar *click* en la opción reportar asistencia ( ), se abrirá la siguiente ventana modal:

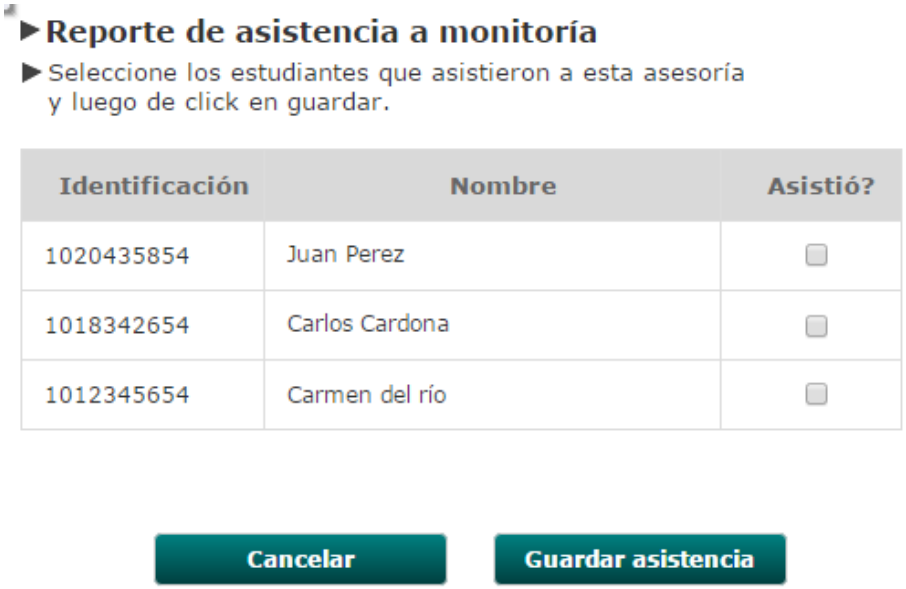

Figura 17. *Mockup* de modal para reportar asistencia (Elaboración propia)

Allí se listarán todos los usuarios que se encontraban registrados como asistentes a la asesoría, por lo que el monitor solo debe seleccionar los usuarios que asistieron y luego presionar guardar asistencia. El botón cancelar cerrará el modal sin realizar ninguna acción.

4.3.7 Editar asesoría

Cuando se de *click* sobre la opción editar asesoría ( ), se mostrará la misma pantalla para la creación de la asesoría de la que se habla en el apartado 4.3.2, con la diferencia que se precargará toda la información que tiene la asesoría guardada. Todo el comportamiento de este formulario será el mismo que al momento de crear una asesoría.

4.3.8 Eliminar asesoría

Luego de dar *click* sobre la opción eliminar asesoría ( ), se mostrará el siguiente cuadro de diálogo de confirmación:

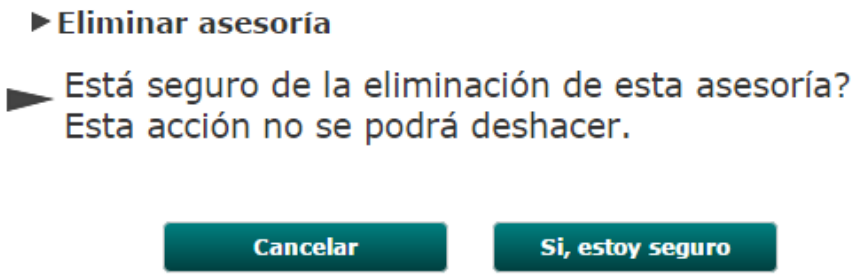

Figura 18. *Mockup* de modal para confirmar la eliminación de una asesoría (Elaboración propia)

Si se opta por dar click en la opción cancelar, se cerrará el cuadro de dialogo y no pasara nada, pero si da click en la opción sí, estoy seguro, se eliminará por completo la asesoría.

4.4 Modelo del módulo de orientación académica

El modelo final del módulo de orientación académica, luce de la siguiente manera.

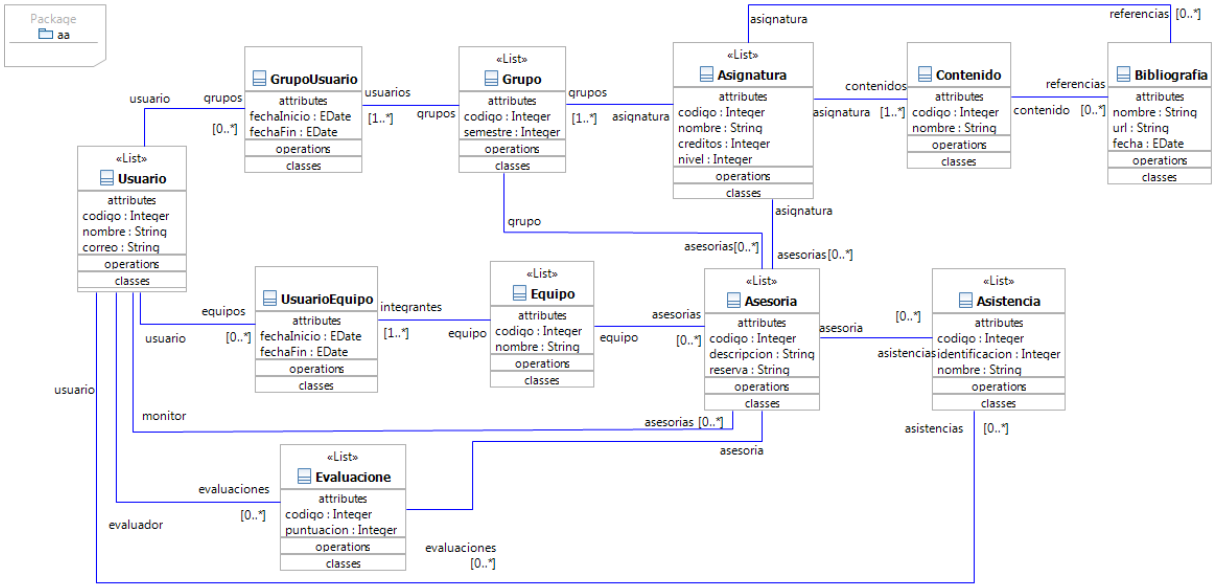

Figura 19. Modelo general del módulo de orientación académica (Elaboración propia)

Si se puede detallar dentro del modelo adicional a la clase asesoría, la cual es la clase principal del módulo, se modelaron algunas otras como lo son asignatura, contenido,

bibliografía, grupo, usuario, entre otras, las cuales no hacen necesariamente parte directamente del módulo, pero si se necesitan para que este funcione correctamente; esto se realizó así debido a que cada módulo que se modeló, se debía entender como un programa autónomo, que luego se unirá con todos los otros módulos, ubicando cada cosa en su respectivo lugar, para lograr de esta forma tener la correcta relación necesaria de todos los campos.

A modo de ilustración, la imagen a continuación hace zoom en el módulo de asesoría, para mostrar este en un mejor nivel de detalle.

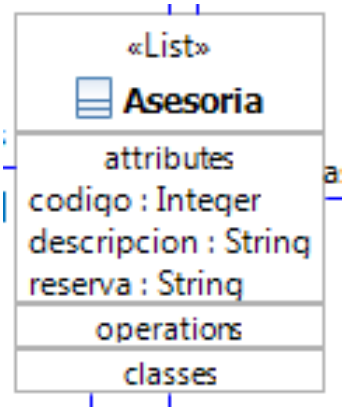

Figura 20. Modelo de la clase asesoría del módulo de orientación académica (Elaboración propia)

Como se puede observar en la figura 20, la clase asesoría tiene aplicado el estereotipo *List*, el cual nos ayudará a que una vez el código sea generado, se cree también un listado con todas las asesorías. Esta clase cuenta también con los campos código, descripción, tipo y reserva y los campos encargado, asignatura, equipo, grupo_codigo, grupo_asignatura, heredados de las relaciones.

Adicional a esto, se marcaron como *primary key* los campos código, encargado y asignatura, para garantizar así la integridad de la información, como se puede observar en la figura 21.

En la figura 22 también se puede notar como se le aplicó el estereotipo *lookup* a los campos asignatura, grupo, equipo, y encargado o monitor, para que al momento de la generación del código, la herramienta MarTE entienda que nos debe generar una ventana modal dentro del formulario para la creación de asesoría, por medio de la cual se podrá seleccionar la información que se desea incluir allí de estos campos.

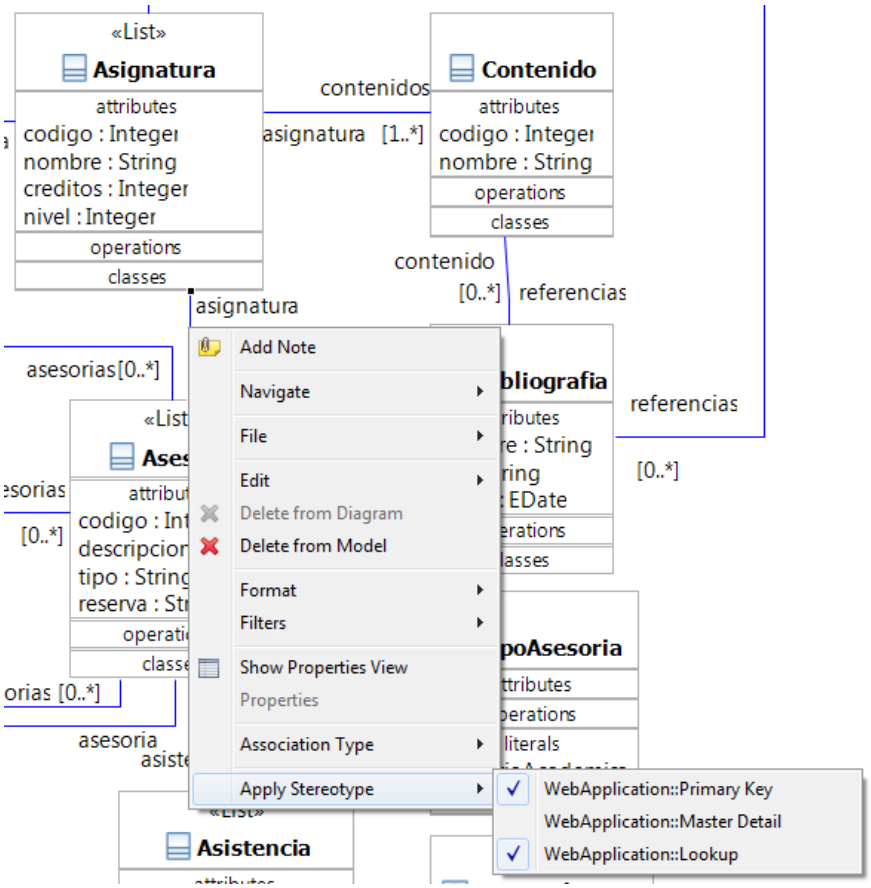

Figura 21. Marcación como *primary key* al campo asignatura, generado desde la clase asignatura (Elaboración propia)

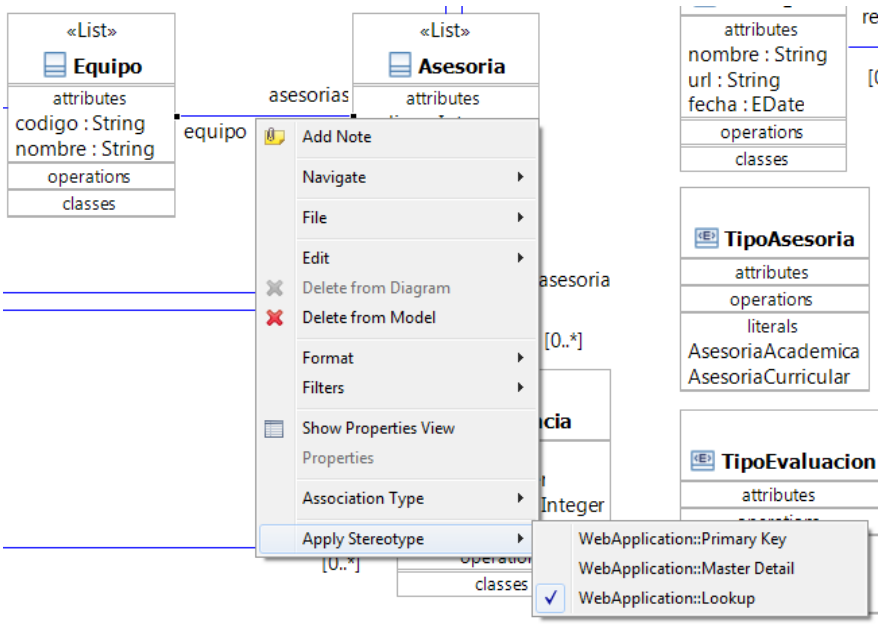

Figura 22. Aplicación del estereotipo *lookup* a la clase Equipo (Elaboración propia)

Finalmente, otro aspecto importante a destacar dentro del modelado es la aplicación del estereotipo maestro detalle a la clase evaluación y asistencia, para que de este modo MarTE entienda la relación que existe entre estas clases y la clase asignatura, permitiendo así la correcta generación de esté código.

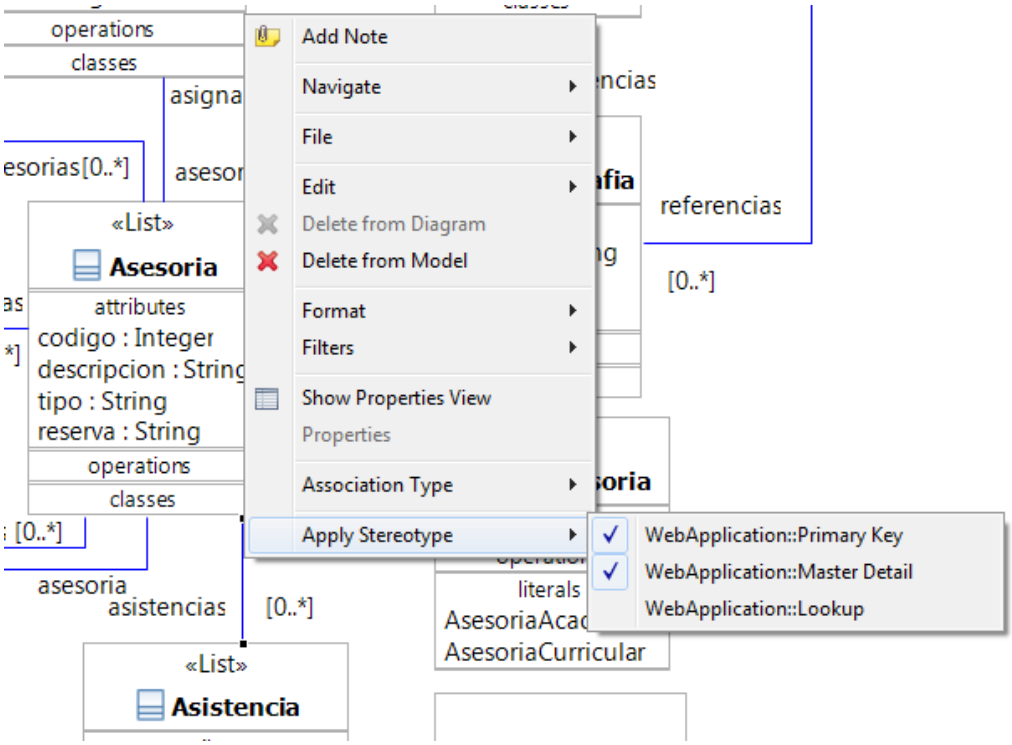

Figura 23. Aplicación del estereotipo maestro detalle a la clase asistencia (Elaboración propia)

4.5 Generación del código fuente con base en el modelo del módulo

Luego de generado el código fuente de todo el módulo, se obtuvo una aplicación funcional que luce de la siguiente manera.

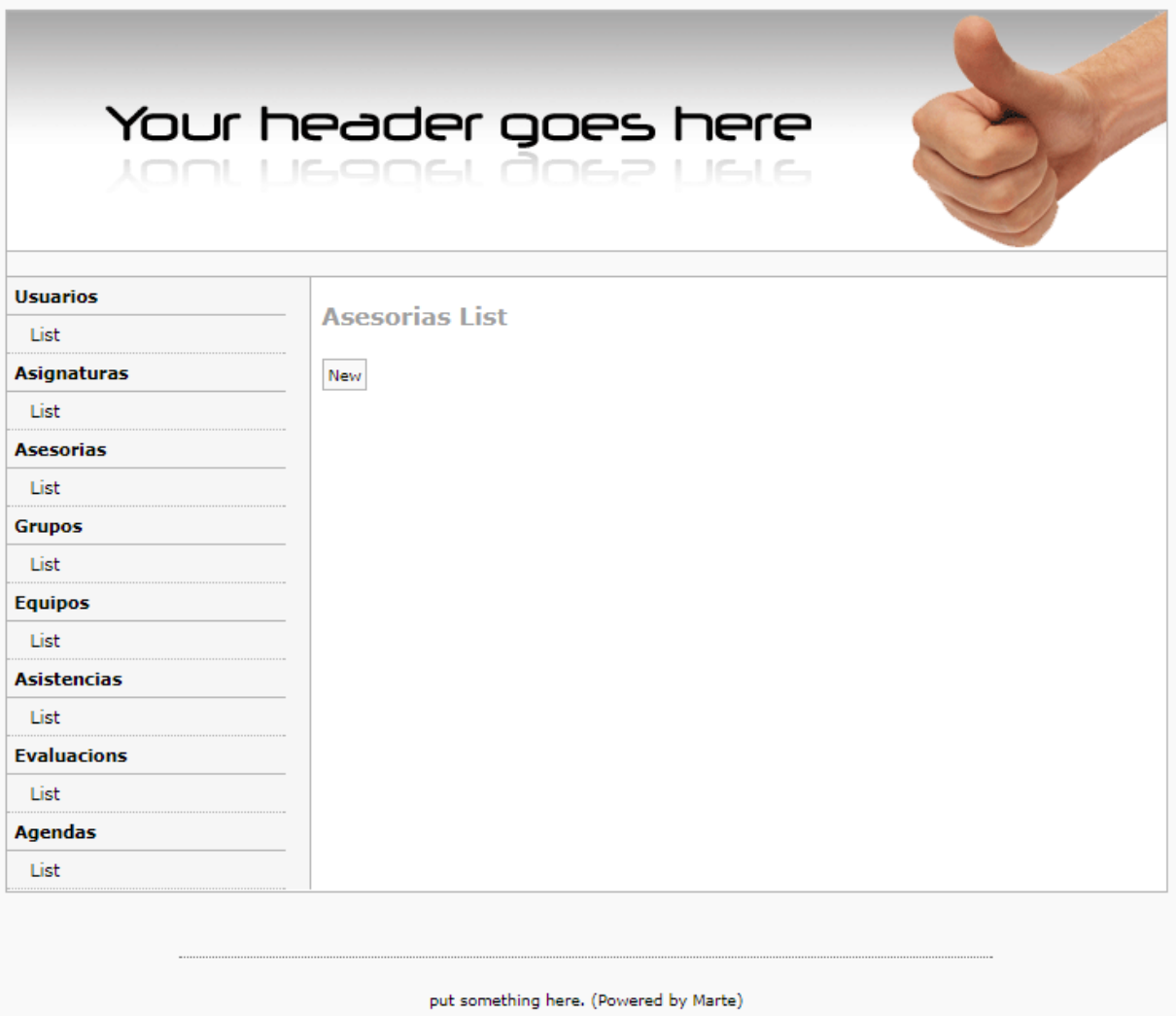

Figura 24. Vista general del módulo de orientación académica, una vez generado el código a partir del modelo (Elaboración propia)

Como se puede observar, como ya se había planteado en el modelo del módulo, en la generación se obtuvieron también las funcionalidades de usuarios, asignaturas, grupos, equipos, asistencias, evaluaciones y agendas, debido a que si bien varias de estas funcionalidades están por fuera del alcance del desarrollo del módulo, son necesarias para que este funcione y por esto se debían incluir allí, así más adelante en los ajustes manuales al código fuente se retiren y se pongan en los módulos que realmente pertenecen.

La estructura global del código generado, luce de la siguiente manera.

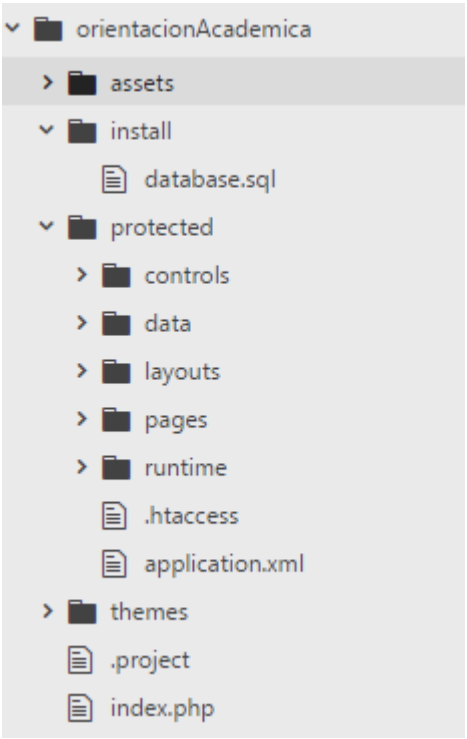

Figura 25. Estructura general de carpetas del módulo de orientación académica, generada por la herramienta MarTE (Elaboración propia)

Teniendo en cuenta que la clase principal de todo el módulo que se está desarrollando es la de asesoría; a continuación, se realiza un análisis detallado del código de esta, que se podrá extender a cualquiera de las otras clases, debido a que a grandes rasgos solo cambian los campos que se definen.

Este análisis se inicia con el código para la creación de la tabla y sus columnas, seguido por el detalle de la separación del patrón MVC que como ya se había dado a conocer desde la arquitectura, es el utilizado dentro del desarrollo de todo el aplicativo.

4.5.1 Código generado para la base de datos

El código generado para la tabla asesorías, cuenta con todas las columnas necesarias para su funcionamiento y como se puede observar en la figura 26, incluso tiene las *primary key* necesarias que buscan garantizar la integridad de la información.

```
-- table asesorias
CREATE TABLE  asesorias (
  encargado INT (30) NOT NULL   ,
  asignatura INT (30) NOT NULL  ,
  equipo VARCHAR (250) NOT NULL   ,
  grupo_codigo VARCHAR (250) NOT NULL   ,
  grupo_asignatura INT (30) NOT NULL  ,
  agenda INT (30) NOT NULL  ,
  codigo INT (30) NOT NULL  ,
  descripcion VARCHAR (250) NOT NULL  ,
  tipo VARCHAR (250) NOT NULL   ,
  reserva VARCHAR (250) NOT NULL  ,
  CONSTRAINT  pk_Asesoria
  PRIMARY KEY( codigo , encargado , asignatura )
);
```

Figura 26. Código generado por la herramienta MarTE, para la creación de la tabla asesorías en la base de datos (Elaboración propia)

A este código junto con el resto del código generado para la base de datos, solo se le requiere realizar ajustes menores para ejecutarlo directamente sobre MySQL.

4.5.2 Código generado para el Modelo

Como se puede observar en la figura 27, el código generado para el modelo queda dentro de la carpeta data, organizado con los mismos nombres que se definieron en las clases.

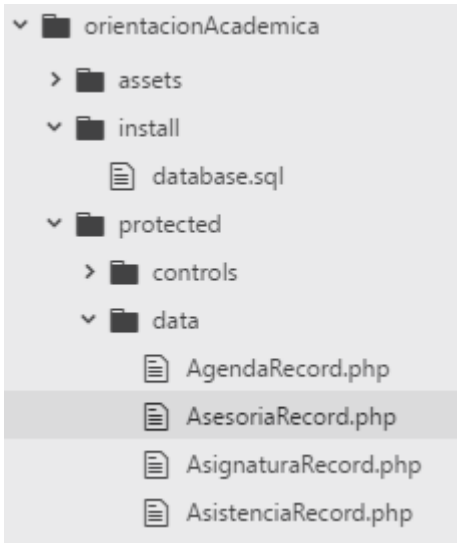

Figura 27. Carpeta data con los archivos donde se encuentra definido el modelo (Elaboración propia)

A continuación, en la figura 28 se puede observar la estructura de la clase asesorías, destacando allí la utilización del patrón de persistencia *active record*, el cual se encarga de implementar todas las consultas y modificaciones necesarias de la tabla en la base de datos.

```php
<?php

class AsesoriaRecord extends TActiveRecord
{
  const TABLE="asesorias";

  public $encargado;
  public $asignatura;
  public $equipo;
  public $grupo_codigo;
  public $grupo_asignatura;
  public $agenda;
  public $codigo;
  public $descripcion;
  public $tipo;
  public $reserva;

  public static $RELATIONS=array
  (
    'Usuario' => array(self::BELONGS_TO, 'UsuarioRecord', 'encargado'),
    'Asignatura' => array(self::BELONGS_TO, 'AsignaturaRecord', 'asignatura'),
    'Equipo' => array(self::BELONGS_TO, 'EquipoRecord', 'equipo'),
    'Grupo' => array(self::BELONGS_TO, 'GrupoRecord', 'grupo_codigo,grupo_asignatura'),
    'Agenda' => array(self::BELONGS_TO, 'AgendaRecord', 'agenda'),
    'asistencias' => array(self::HAS_MANY, 'AsistenciaRecord', 'asesoria_codigo,asesoria_encargado,asesoria_asignatura'),
    'evaluacions' => array(self::HAS_MANY, 'EvaluacionRecord', 'asesoria_codigo,asesoria_encargado,asesoria_asignatura'),
  );

  public static function finder($className=__CLASS__)
  {
    return parent::finder($className);
  }
}
?>
```

Figura 28. Clase asesorías (Elaboración propia)

4.5.3 Código generado para la Vista

El código de la vista y el controlador quedan dentro de la carpeta llamada *pages*.

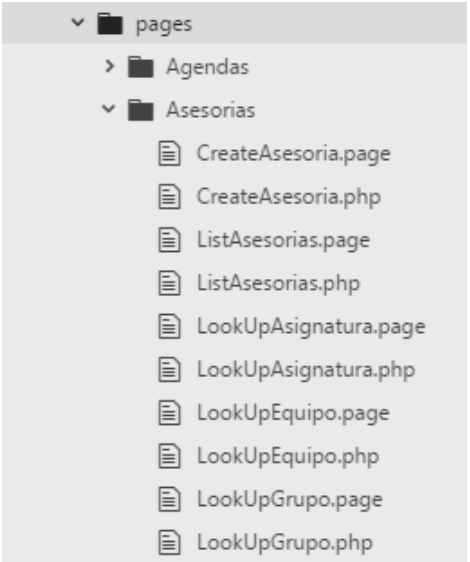

Figura 29. Carpeta con los códigos de la vista y el controlador (Elaboración propia)

Como se puede observar en la figura 29, aunque estos códigos quedan juntos, los relacionados con la vista se encuentran con extensión ".page" y los relacionados con el controlador, con extensión .php.

A continuación, en la figura 30 se muestra un fragmento del código del archivo CreateAsesoria.page, el cual es un XML con los elementos que contiene la página que por medio del *template engine* del *framework* Prado, se transforma este en un archivo html con todas las acciones y validaciones que se hayan especificado desde el modelo, para la creación de una nueva asesoría.

```
<com:TContent id="body">
  <h2>Crear Asesoria </h2>
  <label for="<%= $this->encargado->ClientID %>">Encargado: </label>
    <com:TTextBox ID="encargado" Columns="40" />
      <com:THyperLink
        CssClass="lookup-button"
        ToolTip="Select a Encargado"
        Text="LookUp"
        NavigateUrl="#"
        Attributes.onclick="window.open('?page=Asesorias.LookUpUsuario&fieldId=<%= $this->encargado->ClientID %>',
                           '',
                           'toolbar=0, width=500, height=400, location=0, status=0, menubar=0, scrollbars=1, resizable=0');
                           return false;"
      />
      <com:TRequiredFieldValidator
        Display="Dynamic"
        ControlToValidate="encargado"
        ErrorMessage="...is required"
        ValidationGroup="form" />
      <br /><label for="<%= $this->asignatura->ClientID %>">Asignatura: </label>
      <com:TTextBox ID="asignatura" Columns="40" />
      <com:THyperLink
        CssClass="lookup-button"
        ToolTip="Select a Asignatura"
        Text="LookUp"
        NavigateUrl="#"
        Attributes.onclick="window.open('?page=Asesorias.LookUpAsignatura&fieldId=<%= $this->asignatura->ClientID %>',
                           '',
                           'toolbar=0, width=500, height=400, location=0, status=0, menubar=0, scrollbars=1, resizable=0');
                           return false;"
        />
        <com:TRequiredFieldValidator
          Display="Dynamic"
          ControlToValidate="asignatura"
          ErrorMessage="...is required"
          ValidationGroup="form" />
        <br /><label for="<%= $this->equipo->ClientID %>">Equipo: </label>
        <com:TTextBox ID="equipo" Columns="40" />
        <com:THyperLink
        CssClass="lookup-button"
        ToolTip="Select a Equipo"
        Text="LookUp"
        NavigateUrl="#"
        Attributes.onclick="window.open('?page=Asesorias.LookUpEquipo&fieldId=<%= $this->equipo->ClientID %>',
```

Figura 30. Fragmento de código XML de la vista para crear una asesoría (Elaboración propia)

4.5.4 Código generado para el Controlador

Partiendo de la idea de que cada método localizado dentro del controlador es una acción desencadenada por un control de la vista, en la figura 31 se puede observar el código encargado de guardar una nueva asesoría dentro de la base de datos, la cual realiza las validaciones configuradas, crea el nuevo objeto y luego lo guarda.

```php
<?php

class CreateAsesoria extends TPage
{
  public function onInit($param)
  {
    parent::onInit($param);

  }

  public function onLoad($param)
  {
    parent::onLoad($param);
  }

  public function saveAsesoria($sender, $params)
  {
    if( $this->IsValid)
    {
      $asesoriaRecord = new AsesoriaRecord();
      $asesoriaRecord->encargado = $this->encargado->Text;
      $asesoriaRecord->asignatura = $this->asignatura->Text;
      $asesoriaRecord->equipo = $this->equipo->Text;
      $asesoriaRecord->grupo_codigo = $this->grupo_codigo->Text;
      $asesoriaRecord->grupo_asignatura = $this->grupo_asignatura->Text;
      $asesoriaRecord->codigo = $this->codigo->Text;
      $asesoriaRecord->descripcion = $this->descripcion->Text;
      $asesoriaRecord->tipo = $this->tipo->Text;
      $asesoriaRecord->reserva = $this->reserva->Text;
      $asesoriaRecord->save();
      $this->Response->redirect("?page=Asesorias.ListAsesorias");
    }
  }

  public function cancelAsesoria($sender, $params)
  {
    $this->Response->redirect("?page=Asesorias.ListAsesorias");
  }

}
?>
```

Figura 31. Código del controlador para crear una nueva asesoría (Elaboración propia)

4.5.4 Otros códigos generados

Otros códigos que la herramienta MarTE generó son los localizados dentro de la carpeta themes, los cuales serán los que tendremos que intervenir mayormente, debido a que allí es donde se localizan todos los estilos del módulo.

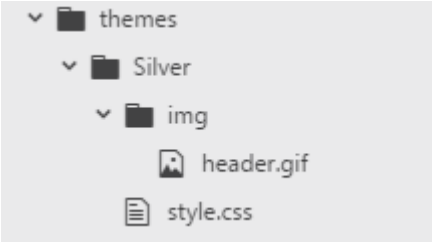

Figura 32. Código generado dentro de la carpeta *themes* (Elaboración propia)

4.6 Realización de ajustes manuales al código fuente

Luego de generado todo el código fuente del módulo con la herramienta MarTE, el paso siguiente es la integración de este con todo el aplicativo y la realización de forma manual de algunos ajustes, con el fin de adaptarlo de una forma más específica a los requerimientos que se tienen planteados en el desarrollo. Estos ajustes solo se realizaron en la base datos, para integrar el script de creación de las tablas del módulo en el script de creación de las tablas de todo el aplicativo y en la parte visual, donde sí se realizó una gran cantidad de ajustes, para lograr obtener la vista que se había planteado desde los *mockups*.

4.6.1 Ajustes al código generado para la base de datos

Como se mencionó anteriormente, el código final obtenido para la base de datos, viene concebido teniendo en cuenta el módulo como un software por sí solo, por lo que trae consigo las relaciones y la creación de otras tablas, que si bien son necesarias para el funcionamiento del módulo, no hacen parte de este, por lo que dadas las características con las que se está desarrollando el módulo, el primer ajuste que se realizó aquí fue la eliminación de todas estas tablas, dejando solo las propias del módulo, las cuales posteriormente se incluyeron dentro del script general que crea la base de datos de todo el aplicativo, para de este modo ir consolidando en un solo archivo toda esta información.

4.6.2 *Merge* del código del módulo con el código de todo el software

Debido a que el software en la actualidad ya cuenta con una estructura global de menús, estilos y colores, como se puede apreciar en la figura 33, lo más simple es integrar todo el código generado para el modelo, la vista y el controlador allí, para de este modo traer hereda toda la estructura general necesaria y así realizar solo los ajustes específicos en las pantallas del módulo.

Figura 33. Menú principal del software OCMS (Elaboración propia)

Luego de tener integrada la vista y el controlador, la estructura de carpetas del módulo quedó de la siguiente manera.

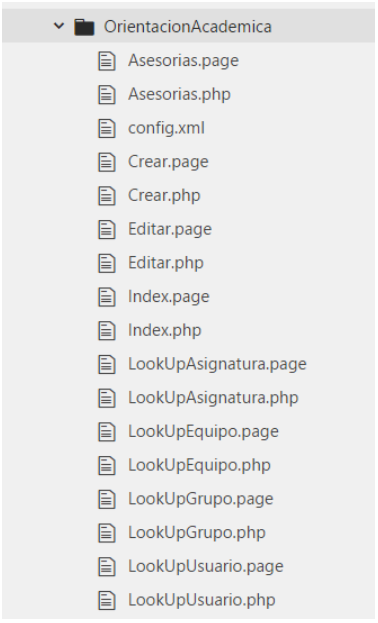

Figura 34. Estructura de carpetas de la vista y el controlador del módulo de orientación académica (Elaboración propia)

Finalmente dentro de la carpeta data, copiamos el archivo AsesoríaRecord.php, que es donde se encuentra definida toda la estructura que necesitamos para el funcionamiento del módulo y realizamos los ajustes necesarios en la estructura de navegación del software, de modo que ya respondan los archivos agregados, a las diferentes opciones del menú.

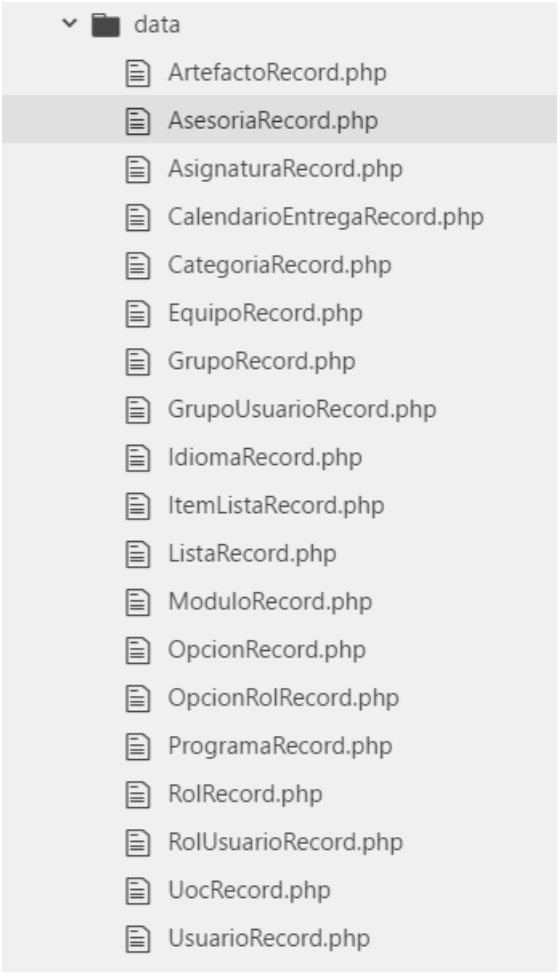

Figura 35. Estructura de carpetas del modelo del software OCMS (Elaboración propia)

4.6.3 Ajustes realizados al código de la vista

Al ya tener integrada toda la estructura del módulo con el software, se creó un archivo CSS llamado orientacionAcademica.css, donde se incluyeron todos los estilos particulares del módulo, para lograr así una mejor apariencia de este.

En una refactorización de código posterior y una vez se tengan definidos todos módulos, se podría pensar en incluir estos estilos dentro de los globales, para no tener código repetido, debido a que varias de las medidas y colores definidos aquí, son estándares para todos.

```
.generalWrapper {
  width: 980px;
  margin-bottom:5px;
  margin-right:20px;
  text-align:left;
}

.subWrapper {
  width: 940px;
}

.title {
  margin-bottom: -7px;
}

.title > span {
  style="color:#888888;
}

.filters {
  display: flex;
  align-items: center;
}

.item {
  width:350px;
}

.subTitle {
  margin-bottom: 15px;
  margin-top: 5px;
}

.subTitle > span {
  style="color:#888888;
}

.searchButton {
  width: 581px; background: white url(webroot/img/icons/search.png) right  no-repeat;
}
```

Figura 36. Porción de código del archivo orientacionAcademica.css (Elaboración propia)

Las modificaciones siguientes realizadas, se efectuaron dentro de las plantillas; a modo de ilustración en la figura 37 se muestra un trozo del código final de la pantalla que contiene el listado de las asesorías.

Estos mismos ajustes de divs, estilos y posicionamientos, se realizaron en los *lookups*, que son los modales y la pantalla para la creación y edición de asesorías.

```
<h2>Asesor&#xED;as <p> &#xBB; Asesor&#xED;as</p></h2>

<h4 class="title"><span>&#9658; </span> Asignatura</h4>

<div class="filters">

<div class="item">
  <com:TDropDownList ID="facultades" CssClass="text-select" AutoPostBack="false">
  </com:TDropDownList>
</div>

<div class="item">
  <com:TDropDownList ID="programas" CssClass="text-select" AutoPostBack="false">
  </com:TDropDownList>
</div>

<com:TLinkButton
  Text="<%[+ Crear Asesoria]%>"
  CssClass="button"
  OnClick="newAsesoria"
/>

</div>

<h4 class="subTitle"><span>&#9658; </span> Lista de asesor&#xED;as </h4>

<com:TDataGrid id="AsesosriasGrid"
  AutoGenerateColumns="false"
  ItemStyle.CssClass="odd"
  AlternatingItemStyle.CssClass="even"
  AllowPaging="true"
  PageSize="10"
  PagerStyle.CssClass="pagenavi"
  PagerStyle.Mode="Numeric"
  OnPageIndexChanged="changePage"
  OnPagerCreated="pagerCreated"
  OnItemCommand="processCommand"
  AllowSorting="true"
  OnSortCommand="sortList"
  >

  <prop:EmptyTemplate>
    <div class="info-box"><strong><%[Informacion]%>:  </strong><%[No se encontraron asesorías]%>.</div>
  </prop:EmptyTemplate>
```

Figura 37. Porción de código final de la pantalla del listado de las asesorías (Elaboración propia)

Al ya tener todo el diseño listo, debido a que uno de los requerimientos era que se mostraran las acciones del listado de asesorías según el rol y el estado de esta, se procedió a realizar estos filtros de forma manual.

Al final y luego de todas las modificaciones efectuadas, como se puede apreciar en las figuras 38, 39 y 40 se obtuvo el resultado buscado, el cual era tener las pantallas funcionales, pero con el mismo diseño que se había planteado desde los *mockups*.

Aunque se realizó la modificación en todas las pantallas, solo se adjuntas estas tres a modo de ilustración, debido a que son las principales.

Figura 38. Vista final de la pantalla con el listado de asesorías (Elaboración propia)

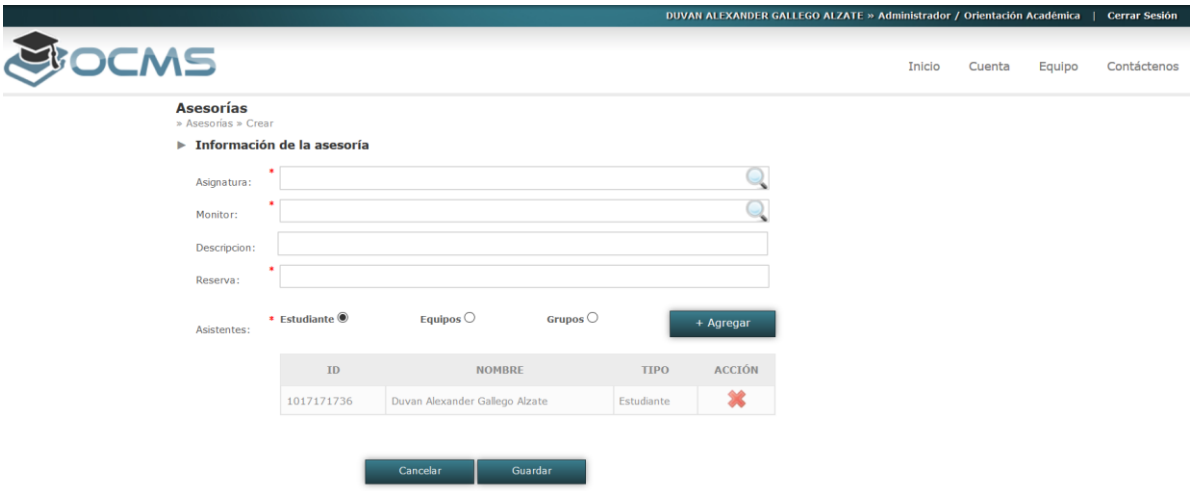

Figura 39. Vista general de la pantalla para la creación de las asesorías (Elaboración propia)

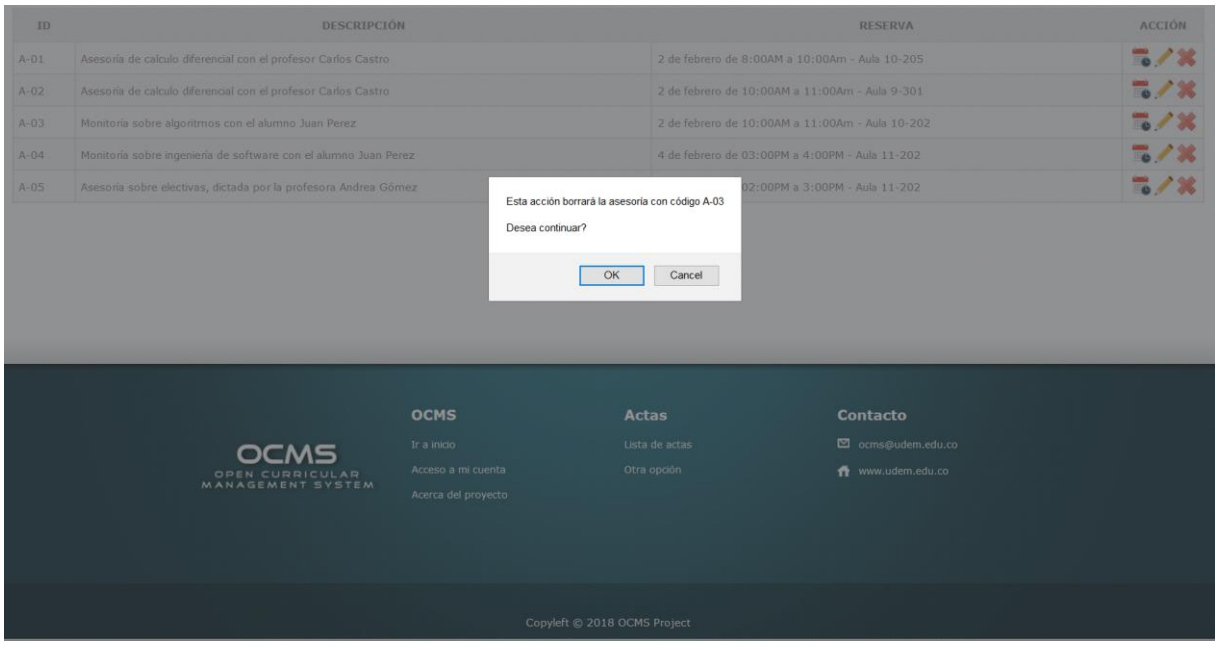

Figura 40. Ventana de confirmación al querer eliminar una asesoría (Elaboración propia)

4.7 Evaluación

La evaluación como ayuda para el desarrollo de software se ha aplicado desde la última década, cuando la comprensión del papel de la evaluación dentro de la interacción humano-computadora ha cambiado, obligando a complementar todo el ciclo de desarrollo de software con esta tarea, que siempre ayuda a decidir el progreso de este y los próximos pasos que se deben dar (Gediga, Hamborg, & Düntsch, 2001).

La norma ISO/IEC 25000:2014, nos ayuda a definir con respecto a que aspectos se puede realizar la evaluación de un desarrollo de software; estos aspectos son funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad y calidad en uso.
La evaluación que se le realizará al módulo será por medio de un grupo de expertos, quienes analizarán de forma global y sin ninguna restricción el módulo desarrollado, para finalmente dar un juicio con puntos que se pueden mejorar o cambiar, logrando así determinar el nivel de madurez con el que cuenta este para ser desplegado en un ambiente de producción.

Aunque a los evaluadores se les plantearán todos los aspectos que se deben tener en cuenta, la primera sección de la evaluación será centrada solo en la usabilidad y la calidad de uso, partiendo del hecho de que estos aspectos tal y como lo plantea (Gediga, Hamborg, & Düntsch, 2001) han ido adquiriendo una importancia particular durante los últimos años con el uso creciente de software interactivo.

4.7.1 Evaluadores

Para la realización de la evaluación, se seleccionaron 4 expertos de diferentes áreas como lo son *user experience designer*, *Web UI developers* y *PHP developers*, buscando incluir así todas las verticales involucradas dentro del desarrollo del módulo, para lograr tener un contexto y visión más amplia de los posibles problemas que pueda tener este.

Como lo plantea (Nielsen J. , 1993) con esta cantidad de expertos evaluadores, deberíamos de ser capaces de encontrar aproximadamente un 70% solo de los problemas de usabilidad, como se puede observar en la figura 41 tomada del mismo autor, donde se representa un modelo base de predicción de problemas encontrados por medio de evaluación heurística versus la cantidad de evaluadores involucrados.

Si bien un 70% podría ser entendido como un porcentaje bajo, para el alcance que debe tener en esta evaluación se puede considerar como un porcentaje aceptable.

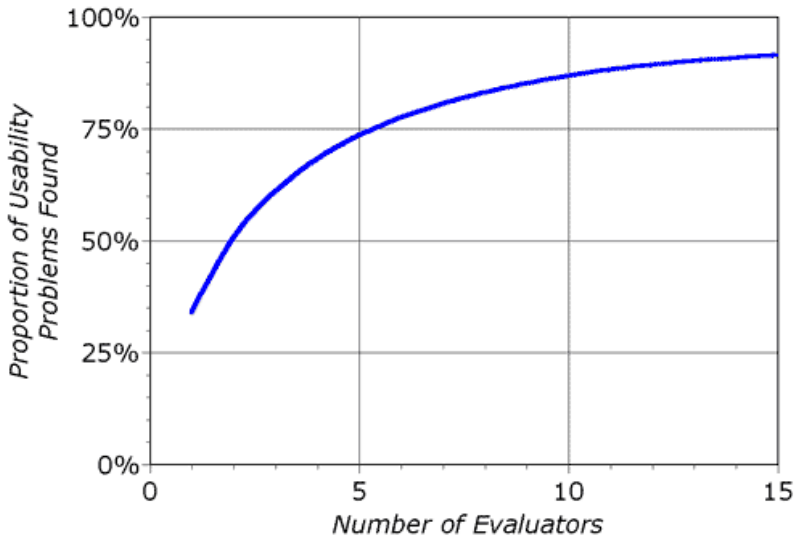

Figura 41. Curva de número de evaluadores vs la proporción de problemas de usabilidad encontrados (Nielsen J. , 1993).

Los perfiles de los 4 expertos seleccionados para realizar la evaluación son los siguientes:

Tabla 9. Perfiles de los evaluadores

Nombre	**Cargo**	**Empresa**	**Experiencia total acumulada**
Héctor Adrián Serna Gómez	*Web UI Developer*	Globant	5 Años
Mauricio de Jesús Arroyabe Alzate	*Web UI Developer*	Globant	7 Años
Juan Esteban Calle Arroyave	*UX Designer*	Globant	7 Años
Daniel Alberto Montoya Londoño	*PHP Developer*	Globant	10 Años

(Elaboración propia)

4.7.2 Fases y procedimiento para la evaluación

El procedimiento para la realización de la evaluación, se ejecutará siguiendo 6 fases, que se listan en la siguiente figura a modo de resumen.

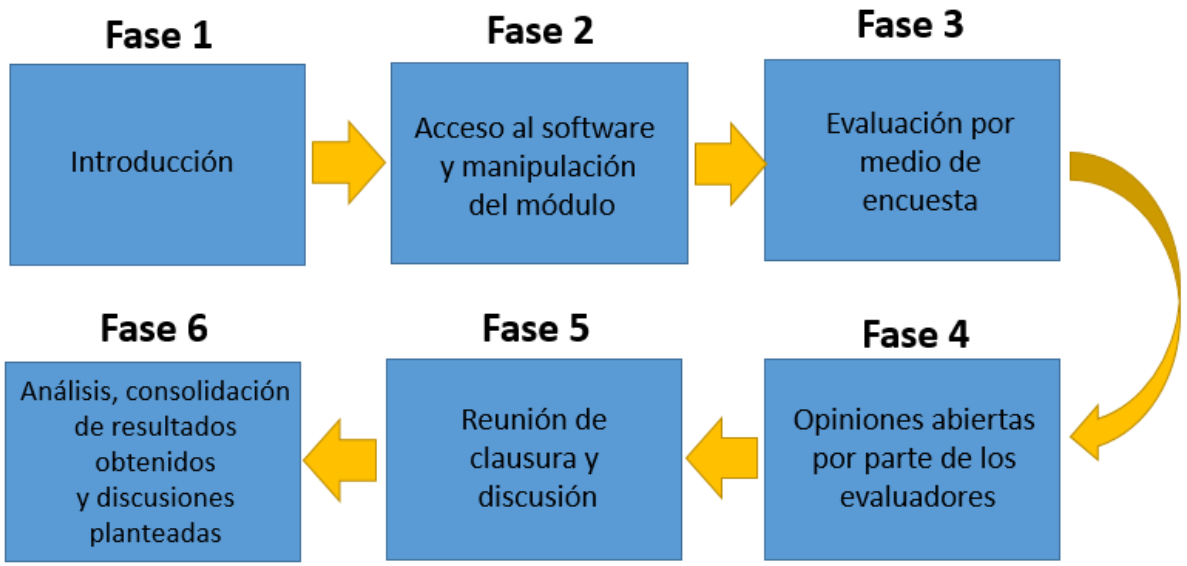

Figura 42. Fases del proceso de evaluación (Elaboración propia).

A continuación, se describen cada una de estas fases.

Fase 1- Introducción: Durante esta fase se realizará una reunión introductoria con todos los evaluadores, donde se les explicará el trabajo que se está realizando, como se está realizando, los modelos arquitectónicos y las tecnologías que se están utilizando, en que consiste el software y el módulo, las fases que contiene la evaluación, como se va a efectuar esta y se les hará firmar el consentimiento adjunto en el anexo 3, con el fin de obtener el permiso para el uso de sus datos personales y la grabación de su voz durante el presente trabajo. La idea con esta introducción es que los evaluadores tengan una visión global de lo que se va a realizar y que se espera de ellos.

Fase 2 – Acceso al software y manipulación del módulo: En esta fase se les dará acceso a todos los evaluadores al software por medio de una dirección online, un usuario y una contraseña. Lo que se busca es que ellos entren, exploren y manipulen todo el módulo a nivel general sin ningún tipo de restricciones, con el objetivo de que puedan tener una visión clara y concisa que los llevará luego a realizar una buena evaluación.

Fase 3 – Evaluación por medio de encuesta: El objetivo de esta fase es que una vez los expertos evaluadores terminen la exploración del módulo, se les realice una encuesta la cual nos ayudará a medir de una forma concisa diferentes aspectos de usabilidad y calidad de

uso. La encuesta realizada se encuentra en el anexo 1 y para su elaboración se tomó como base la guía de evaluación heurística de sitios web de (Hassan Montero & Martín Fernández, 2003), la evaluación heurística de interfaces de usuario de (Nielsen & Molich, 1990), la norma ISOIEC 25000:2014, los principios heurísticos de usabilidad, recopilados por (Baena, 2018) y el libro evaluación heurística de (Gonzáles, Pascual, & Lorés, 2001).

Fase 4 – Opiniones abiertas por parte de los evaluadores: Durante esta fase se les proporcionará a los evaluadores un documento de Word, donde podrán expresar de manera abierta y escrita sus opiniones sobre el módulo, sin ningún tipo de limitaciones ni preguntas.

Fase 5 – Reunión de clausura y discusión: Aquí se buscará realizar una última reunión, donde los evaluadores podrán compartir de manera oral las conclusiones a las que han llegado. También se realizará una pequeña discusión sobre el porqué de las conclusiones, buscando llegar a consensos sobre que se debe quitar, poner o mejorar dentro del módulo y su nivel de madurez. Aunque la idea es que sea una conversación abierta, se preparó un guion que se encuentra en el anexo 2, con los temas y preguntas mínimos que se deben tocar en esta reunión.

Fase 6 – Análisis, consolidación de resultados obtenidos y discusiones planteadas: En esta última fase, se buscará consolidar todos los resultados obtenidos de las encuestas que contestaron los evaluadores y se recopilarán las principales discusiones y sugerencias que surgieron de las fases 5 y 6 de la evaluación. Los datos finales de estas recopilaciones, se encuentra documentados dentro del apartado 4.7.3 Resultados obtenidos y discusiones.

4.7.3 Resultados obtenidos y discusiones

El primer aspecto a destacar dentro de este apartado, es que todos los expertos concluyeron que efectivamente el módulo se encuentra funcionando correctamente en términos generales, pero como todo, tiene varios puntos que se pueden mejorar, para darle mayor calidad y presentación a este y al software en general.

A continuación, se describen todos los apartados tratados en la encuesta, junto con las discusiones que se fueron efectuando allí.

Visibilidad del estado del módulo

En este aspecto se puede decir que el módulo está cumpliendo a cabalidad con lo que se había planteado, teniendo urls amigables, buenos menús gráficos y una estructura organizada de navegación.

Lenguaje de los usuarios

Dentro de este punto si se vio una marcación clara de algunos aspectos a mejorar como lo son el ingreso automático de comas o puntos en los valores, los colores utilizados que si bien son estándares en todo el software, no son tan llamativos y modernos al igual que los íconos que aunque funcionan, se considera que podrían tener un diseño más llamativo y ser más acordes con su acción, para obtener de este modo una mejor claridad al momento de utilizarlos. Dentro de las palabras y las terminologías manejadas, no se encontraron mayores aspectos a mejorar, más allá de la sugerencia de que si se está planteando un software de uso masivo, se debería mirar la posibilidad de soportar al menos los idiomas español e inglés en toda su estructura.

Control y libertad para el usuario

Dentro del control y la libertad, se puede destacar que hay que mejorar un poco las acciones de confirmación, para tratar de incluir estas no solo en la eliminación de una asesoría, sino también en algunos eventos como el cierre de la página cuando se está editando un formulario y no se ha guardado y al momento de guardar información como las evaluaciones y asistencias, para estar seguros de que la información que va a almacenar, es la que realmente se quiere guardar. Otra discusión importante que se dio, fue la importancia de que los usuarios puedan tener control sobre la apariencia de este, de modo que pueda ser mucho más accesible para por ejemplo usuarios daltónicos.

El resto de puntos planteados en este aspecto, en general se cumplen correctamente.

Consistencia y estándares

Dentro de este ítem, en general todas las calificaciones fueron buenas, pero los expertos consideran que el *look & feel* se puede mejorar en diferentes aspectos puntuales como lo son por ejemplo el cambio del listado de asesorías, por cuadros que hagan más fácil la adaptación a móviles, tener priorización en los botones, de modo que los que tienen acciones principales, sean más llamativos que los que tienen acciones secundarias y finalmente otro aspecto a mejorar, sería tratar de resaltar mucho más los modales, poniendo un fondo opaco detrás de estos.

En la adaptabilidad de la pantalla a móviles, se pudieron notar también algunos problemas y aquí los expertos consideran que se debe dar soporte a una resolución mínima de 320px, rediseñando todo lo que esto conlleva como los botones, los menús y el encabezado principal que se debería de tener.

Ayuda a los usuarios, reconocimiento, diagnóstico y recuperación de errores

Aunque los mensajes y las validaciones respectivas se tienen y los expertos concuerdan en que estos están bien redactados y ayudan al usuario en la solución del problema, si se planteó la necesidad de tener validaciones más amigables, mostrando estas en tiempo real, de modo que no se permita el guardado de los formularios o las acciones, si todos los datos no se encuentran correctos.

Reconocimiento antes que cancelación

Dentro de este apartado, los expertos consideran que se tiene una buena distribución y diseño de los espacios, las etiquetas, los campos, los botones, los títulos y en general todos los formularios son acordes al diseño del software. El único aspecto a mejorar aquí, es la especificación o ejemplo que se debe tener en los campos como fechas, debido a que en la actualidad no se tiene.

Flexibilidad y eficiencia de uso

Un factor muy importante a mejorar dentro de este apartado, es la inclusión de atajos de teclado para de esta forma, lograr un completo uso del módulo sin la necesidad del mouse, buscando trabajar la accesibilidad de este.

Estética de diálogos y diseño minimalista

Aquí los expertos destacaron el diseño minimalista del módulo en todas sus ventanas, al igual que el uso de títulos y etiquetas cortas y descriptivas, pero reiteraron el mejoramiento que consideran que se le puede dar a los íconos, sugiriendo por ejemplo unos íconos con diseño plano, que ayuden a la mnemotecnia de estos.

Ayuda general y documentación

Este apartado fue el más mal calificado de todos, debido a que en la actualidad no se tiene ningún tipo de ayuda y la documentación con la que se cuenta, solo es técnica y no es enfocada hacia el usuario final. Para los expertos, esto si es un aspecto que se debe mejorar bastante antes de salir a producción.

Habilidades

Dentro del punto de habilidades, los expertos consideran que se está cumpliendo con la mayoría de ítems, pero reiteraron la necesidad de controlar la tabulación y los atajos de teclado, para mejorar la accesibilidad del módulo.

Interacción con el usuario placentera y respetuosa

En este enunciado, los expertos reiteraron varios de los temas mencionados anteriormente sobre la posibilidad que se debería tener para el cambio de los colores para los usuarios daltónicos, y plantearon también lo que se ha venido discutiendo sobre los íconos.

Otro aspecto sobre el que se habló bastante, fue sobre los autocompletados en los campos de texto, debido que actualmente no se utiliza esta técnica dentro del software, e incluyéndola, podría llegar a reemplazar varios de los modales que se tienen actualmente, con el fin de evitar la sobre carga de estos dentro del aplicativo.

Privacidad.

En este punto los expertos concluyeron que al menos dentro de un uso que se podría denominar normal, las diferentes opciones solo son accesibles por los usuarios que tienen permiso para realizarlas, aunque más adelante en las discusiones se planteó la necesidad de que una vez esté terminado todo el software, se le realice a este un test de intrusión, para de este modo lograr tener mucha más confianza en la seguridad y privacidad de este.

Finalmente, luego de analizados todos los puntos de la encuesta y desarrollados allí los principales hallazgos discutidos con los expertos, se describen a continuación, algunas otras mejoras que ellos consideran que se pueden realizar y que no están categorizadas dentro de los temas tratados; sino que por el contrario, fueron planteados dentro de las opciones abiertas que tenían la posibilidad de expresar.

Opiniones abiertas por pate de los expertos.

Uno de los temas tratados fue la arquitectura, en la cual todos concuerdan que aunque es una arquitectura buena, rápida y funcional, si se deberían mejorar aspectos como la inclusión de gestores de paquetes como *npm* y *composer*, para facilitar y optimizar mucho más el tiempo que se requiere invertir durante los ajustes manuales que se le deban realizar al código; otro aspecto a destacar aquí fue la utilización de *JQuery* en el lado del *front-end*, la cual para los expertos en *web-ui*, es una librería que ya va saliendo del mercado, debido a que van entrando formas más optimizadas de trabajar todo el *front-end*.

Desde el lado del *back-end*, también se discutió bastante sobre el uso del *framework* prado, el cual para el experto en php en un futuro se podría pensar en cambiar por uno que tenga mayor versatilidad y una comunidad más activa en su desarrollo y soporte, como lo son por ejemplo *laravel*, *symfony* y *yii*, enfatizando que el *framework* prado no es que sea malo, sino que a juicio de él, existen mejores herramientas para encarar el problema.

Otro aspecto que también se destacó bastante, es la coherencia que existe entre las herramientas que se están utilizando para el desarrollo y la finalidad del software, debido a que no tendría sentido realizar un software que va a hacer open *source*, con herramientas comerciales que no sean de fácil adquisición, por lo que en este punto en particular el desarrollo se encuentra bastante bien.

Además de los temas tratados anteriormente, los expertos piensan que se debería de tratar de orientar la arquitectura un poco más hacia los servicios, para de esta forma lograr la integración de una forma más simple en el proceso de desarrollo de aplicaciones móviles.

5. Conclusiones y trabajo futuro

5.1. Experiencia de la ingeniería dirigida por modelos vs desarrollo tradicional.

A lo largo de la historia de la ingeniería de software, se han planteado varias problemáticas que aquejan siempre a esta industria, entre las que se destaca la eficiencia de los desarrolladores en la creación de nuevas aplicaciones, buscando por todos los medios posibles la optimización del tiempo de todas las personas involucradas en este proceso, tratando de automatizar las tareas genéricas, que aunque son necesarias, quitan mucho tiempo y hacen que los desarrolladores pierdan el foco de los verdaderos análisis estructurales necesarios. Dentro de este proceso de automatización de tareas genéricas se encuentra la ingeniería dirigida por modelos, la cual es uno de los conceptos que promete dar un paso adelante en la optimización de los tiempos de desarrollo.

Luego de trabajar día a día con lo que se podría describir como desarrollo tradicional y con las pruebas y actividades desarrolladas durante la ejecución de este trabajo, se realizó una comparación destacando los principales aspectos, obstáculos y beneficios que puede aportar una u otra tecnología durante cualquier desarrollo.

Ingeniería dirigida por modelos.

Iniciando a trabajar con esta metodología, una de las primeras cosas interesantes por destacar es que el ciclo de desarrollo es muy similar al ciclo utilizado en las metodologías tradicionales, permitiendo esto llegar incluso a la utilización de buenas prácticas de desarrollo como las planteadas en Scrum y XP, entendiendo por supuesto que aquí la mayoría de artefactos son modelos que pueden entender las computadoras.

Uno de los primeros obstáculos que se puede encontrar un desarrollador que quiera iniciar a trabajar con esta metodología, es la necesidad de comprender a fondo todos los conceptos que se requieren para lograr la modelación de un aplicativo de una forma correcta y estructurada, debido a que aquí estos modelos constituyen la base sobre la que se construye todo, entendiendo que cualquier error que se tenga en este punto, va a afectar el correcto funcionamiento de todo el aplicativo, debido a que el código no va a quedar correctamente generado. El segundo obstáculo que se puede destacar es la selección del *framework*, los DSL y las herramientas con las cuales se trabajará; si bien durante el presente trabajo esta no fue una dificultad porque desde un inicio ya se encontraban definidos todos estos aspectos, en un desarrollo normal esto si puede constituir un

problema, debido a que se deben realizar varias evaluaciones de herramientas y conceptos que no son muy comúnmente utilizados por la mayoría de personas y de los cuales aún existen muchos con un nivel de madurez bastante bajo.

Luego de sorteados estos dos obstáculos y tener definidos los modelos dentro de la herramienta que se haya seleccionado vienen las transformaciones, las cuales deberían de ser la parte más simple, sino fuera por el perfeccionamiento que deben tener los modelos para que todo funcione bien y la poca ayuda que proporcionan las herramientas para detectar los problemas que tenga este.

Finalmente, de una forma realmente mágica, se obtiene el código fuente del aplicativo y según la herramienta seleccionada, también el script para la creación de la base de datos. Este último punto es bastante interesante, debido a que luego de tener un buen diagrama, no se requiere ninguna preocupación en cuando a la creación de la base de datos, porque el generador ya nos entrega un script que con solo ejecutarlo, crea toda la estructura completa con las tablas, las columnas, las relaciones y las llaves primarias y foráneas necesarias para el funcionamiento del aplicativo; en cuanto al código fuente, si bien es un código funcional y hace lo que se diagramó dentro del modelo, la poca flexibilidad que posee en cuando a plantillas y estilos (al menos en cuanto a desarrollo web), hacen que a este código se le deba trabajar bastante para lograr obtener una mejor apariencia y usabilidad; un aspecto a destacar aquí es que aunque toca realizar ajustes, el código que se generó no se desperdicia dentro de estos ajustes, sino que por el contrario se complementa.

Para terminar este punto, en general se pudo notar en todo lo que está involucrado dentro de la ingeniería dirigida por modelos, que el trabajo que se viene realizando es muy grande y alentador para lograr llegar a lo que los grandes impulsores de esta metodología prometen.

Luego del trabajo que se realizó con esta metodología, el único aspecto bastante desalentador que se pudo detectar, es la poca flexibilidad al cambio que tiene luego de generado y ajustado el código, debido a que cualquier nuevo requerimiento, obliga la generación completa de todo este, que si bien puede ser un trabajo menor por la estructura modular que se maneja en los desarrollos modernos, no deja de ser molesto el tener que realizar nuevamente ajustes al código que ya se habían realizado; aunque los nuevos requerimientos que se planteen si no son estructurales se podrían efectuar directo en el código para no tener que generar este nuevamente a partir del modelo, siendo puristas esto no estaría bien hecho porque se pierde el concepto y se terminaría trabajando con una metodología tradicional.

Desarrollo tradicional.

Dentro del presente trabajo, se llama desarrollo tradicional al trabajo que realizan la mayoría de desarrolladores a diario en sus jornadas laborales, utilizando *frameworks*, herramientas y lenguajes de programación ampliamente conocidos.

Si bien sobre el desarrollo tradicional en realidad no hay mucho que decir debido al amplio prestigio y conocimiento que se tiene sobre él, es importante mencionar algunos aspectos que se podría destacar dentro de esta metodología, como lo es la variedad y flexibilidad de herramientas y *frameworks* de trabajo con los que cuenta, trayendo esto también consigo un problema en la especificidad de estos, debido a que cada vez son más genéricos, pesados y cuentan con menos expertos centrados solo en ellos, por la abrumadora velocidad de su evolución y el surgimiento de nuevos cada día; otro punto importante es que al ser lo más utilizado en el mercado, la cantidad de ayudas y tutoriales que se pueden encontrar en línea, facilitan mucho más el trabajo que se realiza día a día.

Conclusiones de la experiencia

Si bien dentro de la programación tradicional se cuenta con muchos *frameworks* que ya automatizan tareas genéricas, todos se basan solo en la generación de CRUDS (*Create, Read, Update and Delete*) y no van más allá de eso, mientras que con la MDSD se pueden implementar estos mismos CRUDS y adicional a esto, reglas de negocio que ayudan a una generación de código mucho más específica.

En la actualidad con la llegada de los dispositivos móviles y toda la tecnología *loT*, se está realizando una marcada diferencia entre lo que es el *back-end* y el *front-end*, realizando una completa separación de estas dos tecnologías y comunicándolas a través de A*pis*. Lo que se ha podido notar con experiencias recopiladas, es que el desarrollo del *back-end* tiende a ser mucho más estable en cuanto a tecnologías que el *front-end*, al menos durante los últimos años, por lo que se podría llegar a considerar que el MDSD podría llegar a tener mucha más fuerza, centrándose solo en el *back-end* en un inicio, que es donde se pudo notar su punto fuerte o en la creación de mínimos productos viables, que sirvan como base para la obtención de aplicaciones rápidamente.

Para finalizar como se mencionó en algunos apartados anteriores, la MDSD tiene un gran potencial de uso, pero aún le falta evolución para llegar a ser una alternativa práctica, debido a que en el punto en el que está, se podría considerar más como un complemento de las metodologías tradicionales, que como una metodología completa en sí misma.

5.2. Conclusiones generales

El principal resultado, el cual era la construcción del módulo de orientación académica para el software OCMS se logró y el módulo se encuentra actualmente funcionando, a la espera de la terminación del resto de módulos del software y de pruebas de usuarios, para terminar de determinar su correcto funcionamiento. Todos los objetivos específicos, los cuales fueron los pasos para llegar al objetivo general, también se lograron a satisfacción, consiguiendo comprender en un buen nivel de detalle los software de administración curricular, explorando el funcionamiento de la herramienta MarTE, estructurando los requisitos funcionales y no funcionales, creando los *mockups* y las historias de usuario del módulo, estableciendo el modelo completo para el módulo, obteniendo el código fuente a partir del modelo creado por medio de la herramienta MarTE, realizando ajustes manuales al código fuente obtenido, indagando sobre la calidad del producto por medio de validación de juicio de expertos y finalmente realizando una comparación de la experiencia obtenida con ingeniería clásica, versus el desarrollo por medio de la ingeniería dirigida por modelos.

Luego del juicio de expertos realizado a este módulo, se puede concluir que este funciona correctamente y cumple con todos los requisitos, logrando ser un módulo usable en todos los aspectos.

Las sugerencias aportadas por los expertos, fueron en la gran mayoría estéticas, de usabilidad y accesibilidad, a excepción de las sugerencias arquitectónicas y las ayudas, las cuales a juicio de ellos si se deben entrar a mejorar bastante antes de sacar el módulo a producción.

Una vez el módulo se encuentre en producción, este permitirá de una forma sistematizada administrar y evaluar las asesorías que los estudiantes pueden recibir por parte de la universidad, ya sea en el campo académico (como monitorias y asesorías con los profesores u otros estudiantes) o sobre temas como el currículo, elección de materias y organización de tiempo.

Una vez este módulo se encuentre completamente integrado con el resto del software, se logrará obtener un programa completo para la administración curricular de forma libre y gratuita.

5.3. Líneas de trabajo futuro

La primera línea de trabajo futuro que se podría seguir, es la complementación de la evaluación con pruebas de usuarios, para obtener de este modo un panorama mucho más amplio del correcto funcionamiento del módulo.

Una segunda línea es la aplicación de varias de las mejoras sugeridas por los expertos en la evaluación, inicialmente al módulo construido durante el presente trabajo y luego a todo el aplicativo que se está desarrollando.

Otra línea de trabajo futuro, es la continuación de todos los módulos que aún se tienen pendientes por construir; si bien estos módulos ya cuentan con modelos estructurados, aún se tiene pendiente la validación de estos y la generación de todo el código fuente, que posteriormente se deberá añadir a la parte que se tiene funcionando actualmente del software.

Luego de completadas las líneas anteriores de trabajo, no queda más que poner en producción el módulo por medio de una prueba piloto. Para esta prueba piloto ya se cuenta con el apoyo de la Universidad de Medellín, la cual una vez terminado todo el desarrollo, permitirá el montaje allí de algunos cursos, por medio de los cuales se evaluará el funcionamiento de todo el software y las mejoras que se le deben realizar a este, antes de salir completamente a producción en una versión estable.

Finalmente si se quisiera ahondar mucho más y realizar aportes a la herramienta MarTE, allí también existen líneas de trabajo interesantes como lo son el ampliar el espectro de elementos en los meta-modelos, para posibilitar nuevas funcionalidades en las transformaciones, agregando descriptores para generar controles avanzados en la interfaz de usuario como formularios con pestañas, barras de progreso, acordeones y asistentes (Quintero J. B., Redes de transformación de modelos: Una estrategia para especificar detalles del proceso en el desarrollo de software centrado en modelos, 2014).

6. Bibliografía

Baena, L. R. (05 de 01 de 2018). *colimbo.net.* Obtenido de colimbo.net: http://www.colimbo.net/documentos/documentacion/fipo/Principios_heurisiticos_20151120.pdf

Burriel, D. T. (05 de 01 de 2018). *torresburriel.com.* Obtenido de torresburriel.com: http://www.torresburriel.com/weblog/2008/11/28/plantilla-para-hacer-analisis-heuristicos-de-usabilidad/

courseleaf.com. (06 de 12 de 2017). Obtenido de courseleaf.com: https://www.courseleaf.com/curriculum/

datasae.co. (06 de 12 de 2017). Obtenido de datasae.co: http://www.datasae.co/siga/

eclipse.org. (09 de 12 de 2017). Obtenido de eclipse.org: http://www.eclipse.org/modeling/gmp/?project=gmf-tooling#gmf-tooling

eclipse.org. (10 de 12 de 2017). Obtenido de eclipse.org: http://www.eclipse.org/atl/

eclipse.org. (10 de 12 de 2017). Obtenido de eclipse.org: http://help.eclipse.org/kepler/index.jsp?topic=/org.eclipse.emf.doc/tutorials/jet1/jet_tutorial1.html

Fowler, M., & Parsons, R. (2011). *Domain Specific Languages.* United States: Addison-Wesley.

Gediga, G., Hamborg, K.-C., & Düntsch, I. (2001). Evaluation of Software Systems. *Encyclopedia of Computer Science and Technology*.

Gonzáles, M. P., Pascual, A., & Lorés, J. (2001). Evaluación Heurística. En G. M. Paula, P. Afra, & L. Jesús, *Evaluación Heurística* (págs. 9-22). Lérida: Universitat de Lleida.

Hassan Montero, Y., & Martín Fernández, F. J. (2003). Guía de Evaluación Heurística de Sitios Web. *No Solo Usabilidad*.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research.

Kleppe, A., Warmer, J., & Bast, W. (2003). *MDA Explained: The Model Driven Architecture : Practice and.* Addison Wesley.

kuali.co. (5 de 12 de 2017). Obtenido de kuali.co: https://www.kuali.co/products/student/

Nielsen, J. (1993). How to Conduct a Heuristic Evaluation. *Nielsen Norman Group*.

Nielsen, j., & Molich, R. (1990). Heuristic evaluation of user interfaces.

OMG. (2003). *MDA Guide Version 1.0.1.* Joaquin Miller and Jishnu Mukerji.

Quintero, J. B. (2014). Redes de transformación de modelos: Una estrategia para especificar detalles del proceso en el desarrollo de software centrado en modelos. Medellín, Colombia.

Quintero, J. B. (09 de 12 de 2017). *metafora.abcflex.net.* Obtenido de metafora.abcflex.net: http://metafora.abcflex.net/mde/reports/07.QM/Quorra-MTE%20Report.pdf

Quintero, J., & Anaya, R. (2007). Marco de Referencia para la Evaluación de Herramientas Basadas en MDA. *Memorias de la X Conferencia Iberoamericana de Software Engineering (CIbSE 2007)*, (págs. 225-238). Isla de Margarita, Venezuela.

schoology.com. (4 de 12 de 2017). Obtenido de schoology.com: https://www.schoology.com/about

Sommerville, I. (2005). Ingeniería del software Séptima edición. En I. Sommerville, *Ingeniería del software Séptima edición* (pág. 108). Madrid: Pearson Educación SA.

Stahl, T., & Völter, M. (2006). Model-Driven Software. Technology, Engineering, Management. England: John Wiley & Sons Ltd.

Truyen, F. (2006). *The Fast Guide to Model Driven.* Cephas Consulting Corp.

ulster.ac.uk. (06 de 12 de 2017). Obtenido de ulster.ac.uk: https://www.ulster.ac.uk/isd/services/business-applications/curriculum-management-system-cms

usc.edu. (06 de 12 de 2017). Obtenido de usc.edu: https://arr.usc.edu/services/curriculum/cms.html

Voelter, M. (2013). DSL Engineering. En M. Voelter, *DSL Engineering.*

workday.com. (06 de 12 de 2017). Obtenido de workday.com: https://www.workday.com/en-us/homepage.html

Anexos

Anexo 1. Encuesta realizada a los expertos para la evaluación del módulo.

Según los análisis realizados dentro de la fase 2 de la evaluación, a cada una de las siguientes preguntas, responda con uno de estos valores:

1 – Si la funcionalidad no es encontrada.
2 – Si se encuentra la funcionalidad pero esta tiene errores.
3 – Si se encuentra la funcionalidad sin errores, pero pobremente implementada.
4 – Si se encuentra la funcionalidad sin errores, pero medianamente implementada.
5 – Si se encuentra la funcionalidad sin errores y muy bien implementada.

Esta evaluación se encuentra dividida en los temas de visibilidad del estado del módulo, lenguaje de los usuarios, control y libertad para el usuario, consistencia y estándares, ayuda a los usuarios, reconocimiento, diagnóstico y recuperación de errores, reconocimiento antes que cancelación, flexibilidad y eficiencia de uso, estética de diálogos y diseño minimalista, ayuda general y documentación, habilidades, interacción con el usuario placentera y respetuosa y privacidad.

A continuación, se listas las preguntas agrupadas por cada tema.

Tabla 10. Encuesta realizada a los expertos

Visibilidad del estado del módulo
¿El módulo tiene una URL correcta, clara y fácil de recordar?
¿Las URL de sus páginas internas, son claras y permanentes?
¿El esquema de diseño de íconos y su estética es consistente en todo el módulo?
Los menús de instrucciones, puntos de entrada de datos y mensajes de error ¿Aparecen siempre en el mismo lugar de la pantalla o en el mismo menú?
En pantallas múltiples para entrada de datos, ¿Cada página esta etiquetada para mostrar su relación con las otras?
¿El sistema posee algún tipo de *feedback* visual en menús o cajas de dialogo que indiquen las opciones que pueden seleccionarse?
¿Se ha controlado el tiempo de respuesta en las opciones del módulo?
¿Es predecible la respuesta del módulo antes de hacer clic sobre cualquier enlace?
¿Los menús gráficos, muestran de manera obvia cual es el ítem que ha sido seleccionado?
¿Los menús gráficos, muestran de manera clara las opciones que pueden ser seleccionadas?
La estructura de organización y navegación del módulo, ¿Es la más adecuada?

En menús de navegación, ¿Se ha controlado el número de elementos y de términos por elemento para no producir sobrecarga memorística?
¿Existen elementos de navegación que orienten al usuario acerca de dónde está y cómo deshacer su navegación?
Lenguaje de los usuarios
¿El módulo es amigable, familiar y cercano?
¿Los íconos son concretos y familiares para el usuario?
¿Las opciones de los menús están ordenadas en la manera más lógica para el usuario?
¿Los colores utilizados dentro del módulo, son acordes a los utilizados en todo el software?
Cuando se ingresan datos en la pantalla ¿la terminología utilizada para describir la tarea es familiar para los usuarios?
Las opciones de los menús ¿Se corresponden lógicamente con categorías que tengan un significado unívoco?
Los títulos de los menús ¿Siguen un mismo estilo gramatical?
¿La página usa el lenguaje del usuario con palabras, frases y conceptos que le son familiares, no utiliza jergas ni tecnicismos si no son absolutamente necesarios?
¿El sistema ingresa automáticamente comas o puntos en valores numéricos grandes?
Control y libertad para el usuario
¿Se pregunta al usuario que confirme acciones que tendrán consecuencias drásticas, negativas o destructivas?
¿Existe una función para "deshacer" al nivel de cada acción simple, cada entrada de datos y cada grupo de acciones completadas?
¿Los usuarios pueden cancelar acciones en progreso?
¿Los usuarios pueden reducir tiempo de entrada de datos copiando y modificando datos existentes?
¿Los menús son anchos (muchos ítems) antes que profundos (muchos niveles)?
¿Los usuarios pueden revertir sus acciones de manera sencilla?
¿Los usuarios pueden configurar la apariencia de su propio sistema, sesión, archivo y valores por defecto para la pantalla?
¿Se informa constantemente al usuario acerca de lo que está pasando?
¿Se informa al usuario de lo que ha pasado?
¿Hay algún tipo de feedback para cada acción u operación?
Luego de que un usuario completa una acción, ¿El *feedback* del sistema indica el siguiente grupo de acciones que puede comenzar?
Consistencia y estándares
¿Es coherente el diseño general del sitio web y más específicamente del módulo?
¿El *look & feel* general se corresponde con los objetivos, características, contenidos y servicios del módulo y del sitio web?
¿El abuso de letras mayúsculas en las pantallas ha sido evitado?
¿Los íconos poseen etiqueta?
¿Es posible utilizar las barras de desplazamiento vertical y horizontal en cada ventana?
¿Los títulos de los menús están centrados o justificados a la izquierda?
¿Las etiquetas de campos y los campos se distinguen tipográficamente entre sí?

¿Las etiquetas de los campos mantienen una forma consistente entre una pantalla y otra?
¿Las técnicas para atraer la atención del usuario están utilizadas solamente en condiciones excepcionales o para tareas dependientes del tiempo?
¿Se utiliza correctamente la jerarquía visual para expresar las relaciones del tipo "parte de" entre los elementos de la página?
¿Las fuentes son legibles y tienen un tamaño adecuado?
¿La página utiliza los estándares (HTML, XHTML, CSS, etc.) de forma correcta?
¿Se utilizan los colores estándares para los vínculos visitado y no visitados?
¿Se utiliza un diseño que se adapte a las diferentes resoluciones posibles que pueda tener un usuario?
Ayuda a los usuarios, reconocimiento, diagnóstico y recuperación de errores
Cuando se produce un error, ¿se informa de forma clara y no alarmista al usuario de lo ocurrido y de cómo solucionar el problema?
¿Los mensajes de error están expresados de manera tal que es el sistema, y no el usuario, quien se hace cargo de los errores?
¿Los mensajes de error, son gramaticalmente correctos?
¿Los mensajes de error, evitan el uso del signo de admiración?
¿Los mensajes colocan al sistema bajo el control del usuario?
Si se detecta un error en un campo de entrada de datos, ¿El sistema posiciona el cursor en ese campo o lo resalta de alguna manera?
¿Los mensajes de error indican que acción debe realizar el usuario para corregir el error correspondiente?
¿Los mensajes de error dan soluciones o sugerencias para solucionar el presente error?
¿La situación de error permite, de una forma evidente, volver a la situación anterior al error?
Reconocimiento antes que cancelación
¿El despliegue de datos comienza en la parte superior izquierda de la pantalla?
¿Todos los datos que el usuario necesita se muestran en cada paso de una transacción?
¿La información está estructurada con títulos, negritas, indentados y viñetas?
¿El lenguaje y la disposición de la información son asequibles y de lectura rápida para el usuario?
¿La estructura y presentación de la información no necesita explicaciones o información adicional para su comprensión?
¿Existen zonas en "blanco" entre los objetos informativos de la página para poder descansar la vista?
¿La interfaz del módulo es limpia y sin ruido visual?
¿Se hace un uso correcto del espacio visual de la página?
¿Se ha evitado la sobrecarga informativa?
¿La página especifica o da un ejemplo sobre cómo debe introducirse la información en campos problemáticos? Eje. Fechas (dd/mm/AAAA)
¿Las etiquetas de los campos están cercanas a estos pero separadas por al menos un espacio en blanco?
¿Se utilizan los bordes para identificar grupos significativos?
¿Los mismos elementos son iguales en todo el módulo?
¿Las fuentes utilizan colores con suficiente contraste con el fondo?
¿Las pantallas de entrada de datos y las cajas de diálogo indican donde los campos son opcionales?

Flexibilidad y eficiencia de uso
Para pantallas de entrada de datos con muchos campos ¿Tienen los usuarios la posibilidad de grabar una pantalla parcialmente completada?
En las pantallas de entrada de datos, ¿Los usuarios tienen la opción de hacer "click" directamente sobre un campo o utilizar un atajo de teclado?
En los menús, ¿Los usuarios tienen la opción o bien de hacer "clic" directamente en un ítem del menú o utilizar un atajo de teclado?
Estética de diálogos y diseño minimalista
¿Los íconos son visualmente distinguibles de acuerdo a su significado conceptual?
¿Cada ícono está resaltado con respecto a su fondo?
¿Cada pantalla de entrada de datos incluye un título simple, corto, claro y suficientemente distintivo?
¿Las etiquetas de los campos son familiares y descriptivas?
¿Los títulos de los menús son breves pero suficientemente largos como para comunicar su contenido?
Ayuda general y documentación
¿La función de ayuda del menú es visible?
¿Existe ayuda sensible al contexto?
¿Es fácil acceder y regresar al sistema de ayuda?
Tras haber accedido a la ayuda ¿Pueden los usuarios continuar con su trabajo desde donde lo dejaron interrumpido?
¿La información encontrada en la ayuda es exacta, completa, comprensible y relevante?
Habilidades
¿Las operaciones para ventanas son fáciles de aprender y usar?
¿Son los usuarios iniciadores de las acciones antes que ser quienes deben responder?
¿En los valores para campos se evita mezclar caracteres numéricos y alfabéticos siempre que sea posible?
Cuando el usuario accede a una pantalla o una caja de diálogo, ¿el cursor está posicionado en el campo que más probablemente el usuario vaya a necesitar?
¿Se ha evitado la auto-tabulación excepto cuando los campos tienen longitudes fijas o los usuarios son experimentados?
Interacción con el usuario placentera y respetuosa
¿Es cada icono individual un miembro armonioso dentro de una familia de íconos?
¿Se ha evitado el detalle excesivo en el diseño de iconos?
¿Se ha usado el color con discreción?
¿El color se ha usado específicamente para llamar la atención, comunicar la organización, indicar cambios de status y establecer relaciones?
¿Los usuarios pueden desactivar la codificación automática de color si fuese necesario?
¿El sistema completa entradas parciales inequívocas en un campo de entrada de datos?
Privacidad
¿Las áreas protegidas son completamente inaccesibles?

(Elaboración propia, tomando como base la guía de evaluación heurística de sitios web de (Hassan Montero & Martín Fernández, 2003), la evaluación heurística de interfaces de usuario de (Nielsen & Molich, 1990), la norma ISOIEC 25000:2014, los principios heurísticos de usabilidad, recopilados por (Baena, 2018) y el libro evaluación heurística de (Gonzáles, Pascual, & Lorés, 2001)).

Anexo 2. Preguntas utilizadas como libreto para la reunión de clausura y discusión en la fase 5 de la evaluación.

A continuación, se listan las principales preguntas que se utilizaron como libreto, para la reunión de clausura y discusión en la fase 5 de la evaluación que realizaron los expertos.

1. ¿En términos generales, que tal está funcionando el módulo de orientación académica?
2. ¿Cómo les fue encontrando las diferentes opciones que posee y utilizándolas?
3. ¿Qué aspectos destacarían del módulo?
4. ¿Qué aspectos consideran ustedes que son los más importantes a mejorar?
5. ¿Les fue sencillo utilizar el módulo, o que dificultades tuvieron para entenderlo?
6. ¿La vista y ubicación de elementos se pude mejorar en algo? ¿Si la respuesta es afirmativa, en qué?
7. Sobre la idea general del software y en especial del módulo, ¿Logran detectar un factor innovador allí?
8. ¿La arquitectura utilizada es la adecuada o que aspectos consideran que se podrían mejorar?
9. ¿Qué otras herramientas se podrían utilizar, durante el proceso de desarrollo para mejorar el producto final?
10. ¿Conocían sobre la ingeniería dirigida por modelos?
11. ¿Han utilizado alguna vez la ingeniería dirigida por modelos?
12. Basados su experiencia y en lo expuesto durante la introducción, ¿Qué apreciación tienen sobre este tipo de ingeniería?

Anexo 3. Autorización para la grabación de voz y uso del nombre y los datos personales, durante el presente trabajo.

AUTORIZACIÓN PARA LA GRABACIÓN DE VOZ Y USO DE NOMBRE Y DATOS PERSONALES.

Yo _____ con cédula de ciudadanía número _____, doy mi consentimiento a **Duvan Alexander Gallego Alzate** con cédula de ciudadanía número **1.017.171.736**, para la creación, uso y reproducción de grabaciones de voz de mi persona al igual que la utilización de mis datos personales.

Entiendo que el uso de las grabaciones y los datos personales, serán utilizados solo con fines de enseñanza y estos solo se podrán utilizar dentro del TFM “Desarrollo del módulo de orientación académica, para el software *Open Curricular Management System*-OCMS” y sus actividades conexas como lo son artículos y presentaciones que se desprendan del mismo trabajo.

Para el uso de las grabaciones de voz y mis datos personales para cualquier otro fin no mencionado anteriormente, se me informará para yo aceptar o rechazar este uso.

No existe ningún límite de tiempo en cuanto a la vigencia de esta autorización; ni tampoco existe ninguna especificación geográfica en cuanto a dónde se puede distribuir este material.

Esta autorización podrá ser rectificada o cancelada a petición del interesado en cualquier momento que lo desee.

Firma: _____
Cédula: _____

Firmado en Medellín, Colombia el 22 de Enero del 2018.

Artículo

Desarrollo del módulo de orientación académica, para el software *Open Curricular Management System* - OCMS.

Duvan A. Gallego.

Resumen

Buscando subsanar la necesidad que existe actualmente dentro de muchas universidades en cuanto a la administración informática curricular, mediante este trabajo se busca la creación del módulo de orientación académica, para el software *Open Curricular Manager System* (OCMS), el cual es un software open *source* que busca llenar el vacío existente allí actualmente. El desarrollo de este módulo se realiza aplicando la ingeniería dirigida por modelos, a través de transformaciones M2T con la herramienta MarTE.

Luego para lograr determinar el correcto funcionamiento de este al igual que su nivel de madurez para salir a producción, se realiza una evaluación por medio de un grupo de expertos, quienes nos aportan su experiencia y conocimiento para el mejoramiento del módulo y en general de todo el aplicativo.

Palabras Clave: OCMS, modelos, desarrollo, MarTE

Introducción

Los sistemas de información son al día de hoy la pieza fundamental dentro de cualquier universidad que está ayudando a cambiar las tendencias en enseñanza, obligando a los modelos educativos a estar cada día más ligados a estos, para buscar obtener su máximo provecho.

Al problema con el que cuenta muchas universidades para la administración curricular, se le buscará dar una solución por medio del desarrollo del software *open source* OCMS (*Open Curricular Management System*); el cual una vez terminado permitirá a estas tener un *software* completo para la administración curricular, de forma libre y gratuita por medio de los módulos de gestión curricular, banco de proyectos, calendario académico, orientación académica, redes sociales, configuración y seguridad.

Lo que se propuso en este trabajo y considerando el alcance con el que cuenta, es la construcción del módulo de orientación académica por medio de la ingeniería dirigida por

modelos, a través de la herramienta MarTE, el cual se centra en las asesorías que los estudiantes pueden recibir por parte de la universidad, ya sea en el campo académico (como monitorias y asesorías con los profesores u otros estudiantes) o sobre temas como el currículo, elección de materias y organización de tiempo.

Contexto y estado del arte

Contexto

En general en todas las universidades existen varios frentes en cuanto a la gestión curricular, en los cuales para lograr un mejor desempeño y eficacia es necesario cambiar de un método tradicional y desarticulado, a un sistema que agrupe y gestione las diferentes asignaturas. OCMS es un sistema en línea de gestión curricular asistida por tecnologías de información, teniendo en cuenta elementos de diseño curricular, estrategias de trabajo activo, innovación abierta y recursos de apoyo, por medio del cual los profesores, coordinadores y jefes de programas, pueden definir currículos de las asignaturas, horarios y profesores que dictan las mismas además de gestionar los proyectos que se llevan a cabo en cada una de estas.

Estado del arte

Arquitectura dirigida por modelos (MDA)

MDA se puede especificar como un *framework* de desarrollo definido por la *Object Management Group* (OMG), que centra sus esfuerzos en darle mayor relevancia a los modelos dentro de todo el proceso de desarrollo, pasando estos de ser modelos estáticos a ser modelos dinámicos que constituyen el centro de todo el desarrollo (Kleppe, Warmer, & Bast, 2003).

Con nuevos métodos y técnicas de desarrollo surgiendo día a día, en MDA podemos encontrar un camino neutral que nos permite un rápido desarrollo e implementación de nuevas especificaciones y tecnologías, que nos brindan una solución estructurada e interoperable de aplicaciones y portabilidad en el futuro (Truyen, 2006).

Un aspecto fundamental de MDA es la capacidad que tiene para abordar el ciclo de vida del desarrollo de software, abarcando temas como el análisis y diseño, la programación, las pruebas, la implementación y el mantenimiento (OMG, 2003).

En la figura 1, la cual es adaptada de (Quintero & Anaya, 2007), se ilustran los 4 tipos de artefactos que se manejan en MDA, logrando allí identificando la importancia que tienen los modelos y las transformaciones en este tipo de arquitectura.

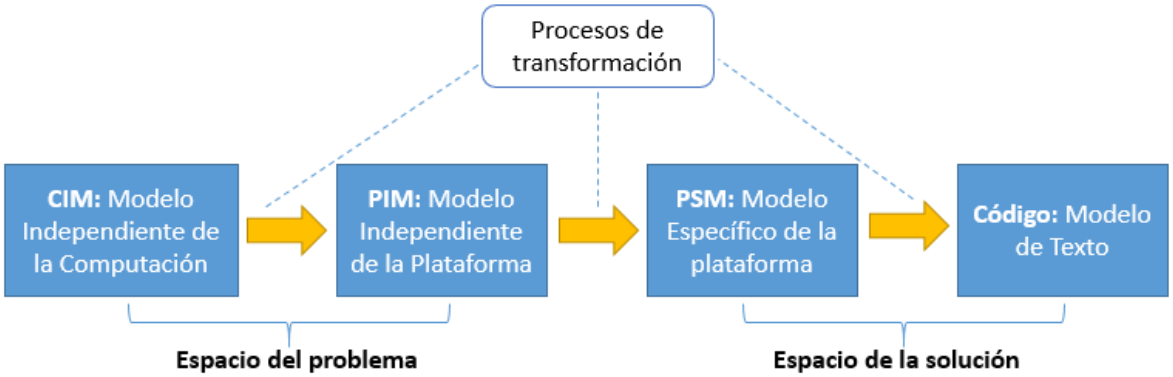

Figura 1. Transformaciones y modelos en MDA. Adaptada de (Quintero & Anaya, 2007)

Desarrollo de software dirigido por modelos (MDSD)

Cuando se habla de desarrollo de software dirigido por modelos - MDSD, el cual también es comúnmente conocido como *Model Driven Development* (MDD), se habla de un enfoque mucho más amplio y flexible al que plantea MDA debido a que este no está sujeto a sus estándares, lo que nos permite por ejemplo utilizar cualquier tipo de meta modelo como DSL y no solo UML y perfiles, ayudándonos de esta manera a definir nuestras propias ideas de PIMs y PDMs, según las necesidades y particularidades de cada proyecto. Para tener un mejor entendimiento de MDSD, debemos dejar claros varios conceptos que nos llevarán a definir los DSL, los cuales constituyen la base de MDSD.

El dominio

(Stahl & Völter, 2006, pág. 56) Lo describe como un campo de interés o conocimiento limitado, que parte en la mayoría de las veces de una ontología con los conceptos del dominio.

El meta modelo

Es una instancia del meta meta modelo y es el que nos ayuda a definir la estructura del dominio, combinando la sintaxis abstracta y semántica estática de un idioma.

El meta meta modelo

El término meta es relativo, en palabras simples es un meta modelo que describe los conceptos que pueden ser utilizados para modelar un modelo. En consecuencia, el meta

modelo debe tener un meta modelo que define los conceptos disponibles para el meta modelado (Stahl & Völter, 2006).

MarTE

MarTE también conocido como Quorra-MTE, es un entorno de transformación de modelos construido sobre el IDE eclipse que pretende aumentar la calidad y la productividad del desarrollo de software. Este entorno se centra en la construcción y transformación de modelos, lo que permiten a los desarrolladores automatizar tareas que muchas veces son monótonas, obteniendo así que estos se centren en el negocio en lugar de la tecnología que utiliza el desarrollo (Quintero J. B., 2017).

Herramientas de gestión curricular existentes

Schoology

Schoology, además de ser una plataforma educativa totalmente gratuita y en línea, permite crear y compartir contenido académico, tareas o evaluaciones como una experiencia de aprendizaje colaborativo, está orientada a la gestión curricular centralizada y gestión de contenido (schoology.com, 2017).

Kuali Student

Detrás de Kuali Student se encuentra la fundación Kuali, la cual es una organización sin ánimo de lucro dedicada al desarrollo de soluciones de software administrativo de código abierto para la educación superior, buscando ayudar a las universidades a mantener su dinero en su misión reduciendo significativamente los costos administrativos.

Kuali Student ayuda a los administradores a gestionar todos los aspectos del plan de estudios y la gestión curricular, ya sea que se ofrezcan cursos y programas tradicionales o se forje nuevos caminos para el aprendizaje. (kuali.co, 2017)

Curriculog

Curriculog es una interfaz en línea que permite que programas y cursos fuera del campus sean propuestos, creados, evaluados, revisados, aprobados e implementados. Los profesores y el personal que participan en la revisión a nivel de departamento, escuela y universidad pueden ver el progreso de sus propuestas de principio a fin (usc.edu, 2017).

CourseLeaf

Con CourseLeaf, los profesores y el personal participan más en el proceso del plan de estudios con reuniones más cortas, formularios intuitivos y comentarios más rápidos sobre

los cambios. CourseLeaf también ahorra tiempo y reduce la ineficiencia en el proceso de gestión del plan de estudios de principio a fin (courseleaf.com, 2017).

Conclusiones del estado del arte

Dentro de los diferentes aplicativos analizados, Schoology es el que más se acerca a lo que se pretende realizar con el desarrollo de *Open Curricular Management System*; sin embargo, aunque inicialmente se ofrece como gratuito, tiene un costo cuando se trabaja dentro de un ambiente universitario que conlleva el tener gran cantidad de usuarios y cursos.

Al analizar los diferentes modelos de negocio se pude concluir que ninguno es completamente libre y de código abierto, el único que se acerca un poco a lo que se busca crear con el desarrollo del software *Open Curricular Management System* en este sentido es Kuali Students, siempre y cuando se cuente con la membresía que se requiere para poder utilizarlo y modificarlo libremente.

Metodología

La metodología por medio de la cual se realizó el desarrollo del módulo y en general el desarrollo de todo el software, es la de desarrollo de software dirigido por modelos (MDSD), tomando algunos lineamientos del *framework* de trabajo definido por la OMG en la arquitectura dirigida por modelos (MDA), abordando los pasos que se detallan en la figura 2.

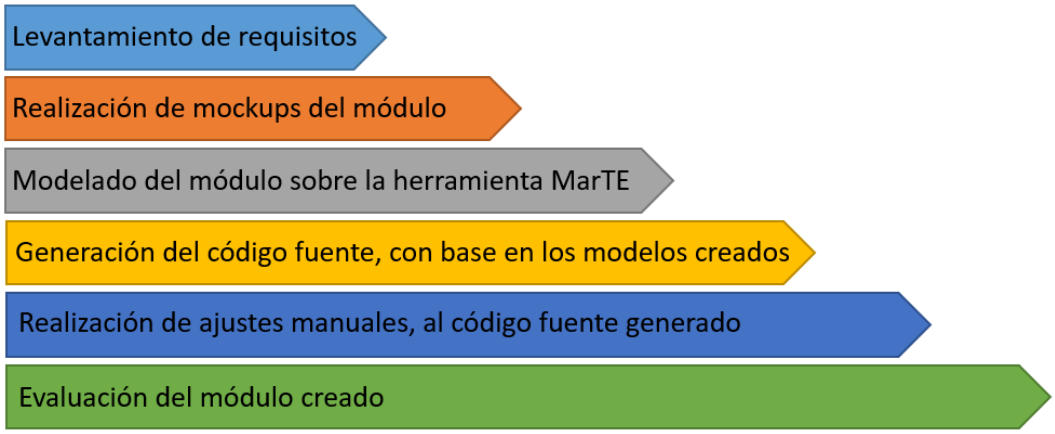

Figura 2. Resumen de todos los pasos que se abordaron para la elaboración del módulo (Elaboración propia)

La base del diseño metodológico utilizado es el propuesto por (Hevner, March, Park, & Ram, 2004), el cual tiene como objetivo proporcionar una comprensión por medio de directrices claras sobre cómo se deben conducir, ejecutar, evaluar y representar las investigaciones y los diseños en los sistemas de información. En la figura 3, se pueden apreciar las 4 fases

por medio de las cuales se estructurarán las actividades enmarcadas en los objetivos específicos, que darán como resultado la obtención del objetivo general.

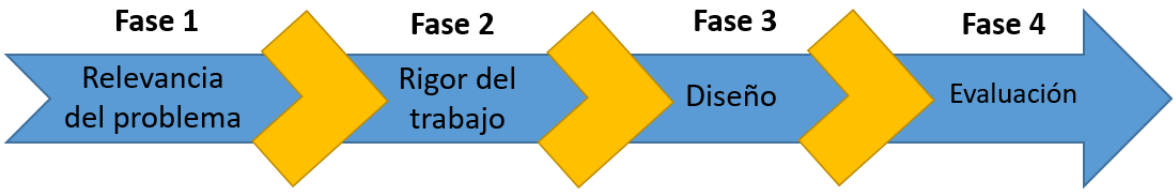

Figura 3. Fases en las cuales se enmarcan los objetivos específicos (Elaboración propia)

Resultados y discusiones

La arquitectura planteda para este trabajo se basa en 4 vistas que defienen aspectos en los siguientes frentes:

- Vista Conceptual: visión que los usuarios tienen de la aplicación.
- Vista Lógica: visión desde los principales elementos y principios del diseño.
- Vista Física: visión desde la distribución del procesamiento entre los dispositivos.
- Vista de Implementación: visión que muestra cómo serán montados los diferentes componentes de la aplicación y la forma en que interactúan.

Los estilos de arquitectura que se utilizaron en el desarrollo son el estilo 3-*Tier* / N-*Tier* y el estilo *layered*, buscando así aislar las funcionalidades en segmentos independientes, que permitan separar de forma clara las responsabilidades de cada *tiers*, pero con la ventaja que cada segmento pueda estar situado en un equipo distinto físicamente.

Las implicaciones que se tuvo con la implementación de estos estilos de arquitectura, fueron la aparición de tres capas:

- *Front-end*: La cual será un explorador de internet y unas hojas de cálculo para la carga masiva de información.
- *Middleware*: La cual será un servidor web con PHP desplegado, utilizando el framework prado.
- *Back-end*: La cual será un gestor de bases de datos MySQL, que puede estar ubicado en el mismo servidor donde está el servidor web.

A continuación, en la figura 4 se ilustra la disposición física de los componentes y artefactos de la aplicación web, mostrando el equipo en el que se despliega cada uno.

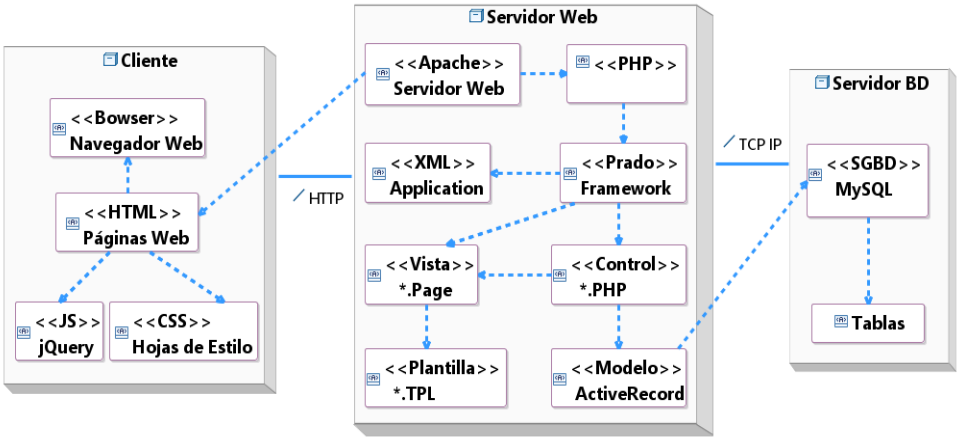

Figura 4. Diagrama de despliegue (Documento de arquitectura de OCMS)

Para la creación de los *mockups* se utilizó la herramienta JustInMind, por medio de la cual se plasmó en prototipos la información que se levantó en las entrevistas con el *product owner,* que posteriormente se sintetizaron en los requisitos funcionales. Luego de tener estos prototipos terminados, se realizó la respectiva validación y ajustes necesarios de estos para así obtener los *mockups* finales. A continuación, en la figura 5 se puede apreciar la vista general del módulo, en la pantalla para listar y crear las asesorías.

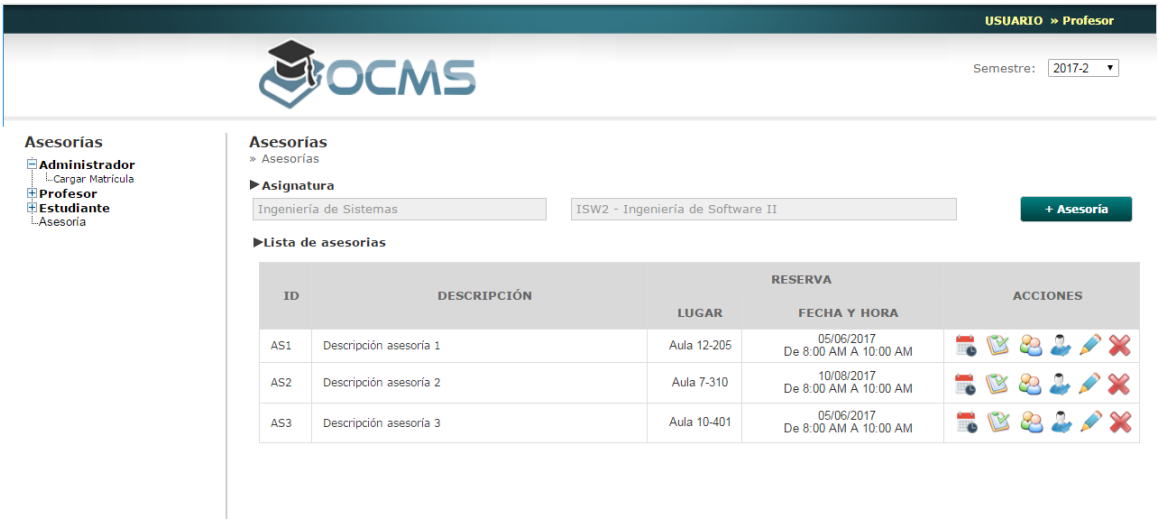

Figura 5. *Mockup* para listar y crear asesorías (Elaboración propia)

Luego de tener todos los *mockups* del módulo se creó el modelo de este, el cual como se puede apreciar en la figura 6 adicional a la clase asesoría, la cual es la clase principal del módulo, se modelaron algunas otras como lo son asignatura, contenido, grupo, usuario, entre otras, las cuales no hacen parte directa del módulo, pero si se necesitan para que este funcione correctamente; esto se realizó así debido a que cada módulo que se modele, se

debe entender como un programa autónomo, que luego se unirá con todos los otros módulos, donde los campos de estos deberán tener una correcta relación y acoplamiento.

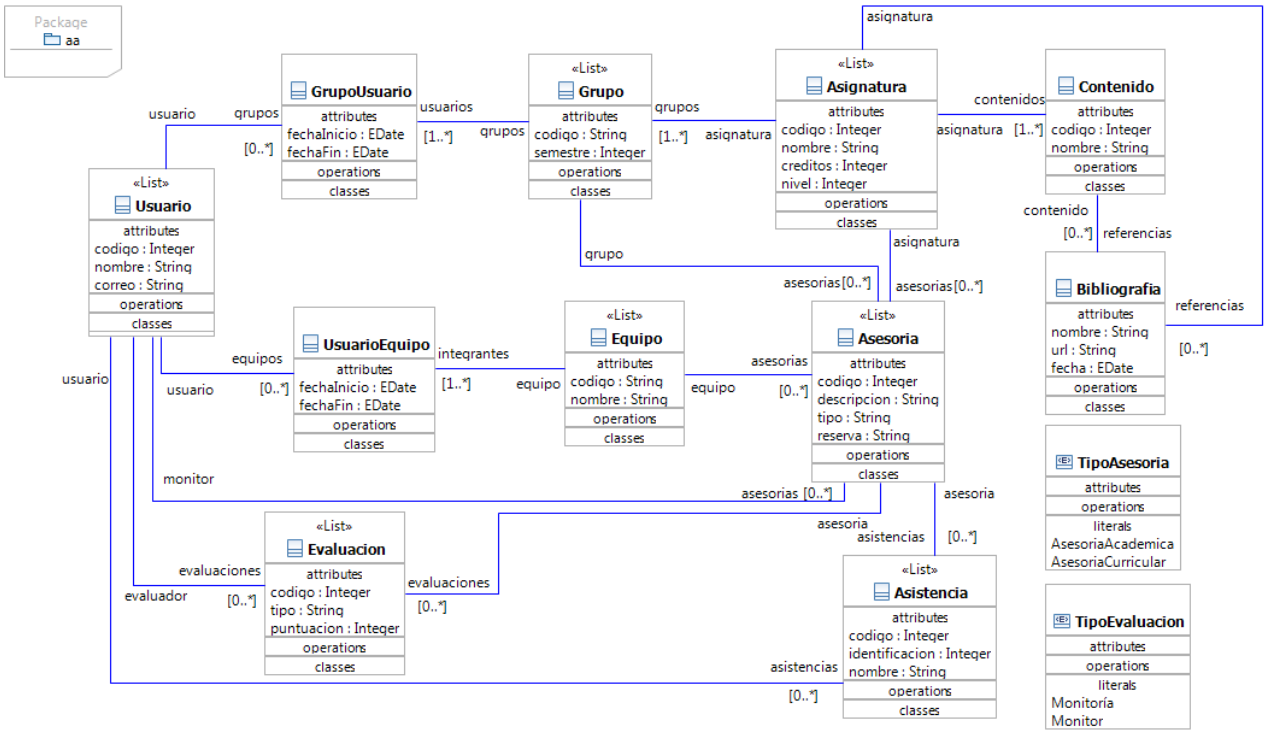

Figura 6. Modelo general del módulo de orientación académica (Elaboración propia)

Al ya tener todo el módulo modelado, los pasos siguientes fueron la generación automática del código fuente por medio de transformaciones M2T a través de la herramienta MarTE y la realización de ajustes manuales a este código, para obtener de esta forma la versión final del módulo funcionando. Finalmente, al módulo creado se le realizó una evaluación por medio de juicio de expertos, para conseguir una visión externa de su funcionamiento y experiencia de usuario, logrando de esta manera determinar su nivel de madurez para salir a un ambiente de producción y una retroalimentación que sirve para mejorar este y todo el software en general.

Conclusiones y trabajos futuros

Conclusiones

El principal objetivo, el cual era la construcción del módulo de orientación académica para el software OCMS se logró y el módulo se encuentra actualmente funcionando, a la espera de la terminación del resto de módulos del software y de pruebas de usuarios, para terminar de establecer su correcto funcionamiento. Todos los objetivos específicos, los cuales fueron los pasos para llegar al objetivo general, también se lograron a satisfacción, consiguiendo comprender en un buen nivel de detalle los software de administración curricular, explorando

el funcionamiento de la herramienta MarTE, estructurando los requisitos funcionales y no funcionales, creando los *mockups* y las historias de usuario del módulo, estableciendo el modelo completo para el módulo, obteniendo el código fuente a partir del modelo creado por medio de la herramienta MarTE, realizando ajustes manuales al código fuente obtenido, indagando sobre la calidad del producto por medio de validación de juicio de expertos y finalmente realizando una comparación de la experiencia obtenida con ingeniería clásica, versus el desarrollo por medio de la ingeniería dirigida por modelos.

Para finalizar es de destacar que la MDSD tiene un gran potencial de uso, pero aún le falta evolución para llegar a ser una alternativa práctica, debido a que en el punto en el que está, se podría considerar más como un complemento de las metodologías tradicionales, que como una metodología completa en sí misma.

Con respecto a las conclusiones obtenidas a partir del desarrollo del módulo, gracias al juicio de expertos se pudo notar que, aunque el módulo ya funciona y hace todo lo que debe hacer, se tienen varias falencias que se deben corregir principalmente en el tema de ayudas para los usuarios, un poco de usabilidad y algunos otros detalles que se pueden ajustar para mejorar mucho más su *look & feel*.

Líneas de trabajos futuros

La primera línea de trabajo futuro que se podría seguir, es la complementación de la evaluación con pruebas de usuarios, para obtener de este modo un panorama mucho más amplio del correcto funcionamiento del módulo. Una segunda línea es la aplicación de las mejoras sugeridas por los expertos en la evaluación, inicialmente al módulo construido durante el presente trabajo y luego a todo el aplicativo que se está desarrollando.

Otra línea de trabajo futuro, es la continuación de todos los módulos que aún se tienen pendientes por construir; si bien estos módulos ya cuentan con modelos estructurados, aún se tiene pendiente la validación de estos y la generación de todo el código fuente, que posteriormente se deberá añadir a la parte que se tiene funcionando actualmente del software.

Bibliografía

courseleaf.com. (06 de 12 de 2017). Obtenido de courseleaf.com: https://www.courseleaf.com/curriculum/

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research.

Kleppe, A., Warmer, J., & Bast, W. (2003). *MDA Explained: The Model Driven Architecture : Practice and.* Addison Wesley.

kuali.co. (5 de 12 de 2017). Obtenido de kuali.co: https://www.kuali.co/products/student/

OMG. (2003). *MDA Guide Version 1.0.1.* Joaquin Miller and Jishnu Mukerji.

Quintero, J. B. (09 de 12 de 2017). *metafora.abcflex.net.* Obtenido de metafora.abcflex.net: http://metafora.abcflex.net/mde/reports/07.QM/Quorra-MTE%20Report.pdf

Quintero, J., & Anaya, R. (2007). Marco de Referencia para la Evaluación de Herramientas Basadas en MDA. *Memorias de la X Conferencia Iberoamericana de Software Engineering (CIbSE 2007)*, (págs. 225-238). Isla de Margarita, Venezuela.

schoology.com. (4 de 12 de 2017). Obtenido de schoology.com: https://www.schoology.com/about

Stahl, T., & Völter, M. (2006). Model-Driven Software. Technology, Engineering, Management. England: John Wiley & Sons Ltd.

Truyen, F. (2006). *The Fast Guide to Model Driven.* Cephas Consulting Corp.

usc.edu. (06 de 12 de 2017). Obtenido de usc.edu: https://arr.usc.edu/services/curriculum/cms.html