Universidad
de las
Ciencias Informáticas
Facultad 4

Componente para la integración de la Plataforma Educativa ZERA 2.0 con el Sistema de Gestión Universitaria

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Javier Alejandro Arango López

Tutores: Ing. Aliander Capdezuñer González

Ing. Evelyn Pérez Rosa

Cotutor: Ing. Lester Collado Rolo

"Año 59 de la Revolución"

Ciudad de la Habana, junio 2017


Pensamiento
“El futuro de nuestra Patria tiene que ser necesariamente un futuro de hombres de ciencia”

Declaración de autoría

Declaro ser autor del presente trabajo de diploma y autorizo a la Facultad 4 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Javier Alejandro Arango López

_____

Firma del autor

Aliander Capdezuñer González

_____

Firma del tutor

Evelyn Pérez Rosa

_____

Firma del tutor

Dedicatoria

En esta tesis queda reflejado todo el sacrificio realizado durante estos cinco años por eso quiero dedicárselo a las personas más importantes en mi vida:

A la persona más importante, cariñosa y hermosa del mundo, mi mamá Marlene, juntos pasamos la universidad y estando lejos nunca se separó de mí, es sencillamente indispensable e irremplazable.

Al hombre más recto que conozco, pero a la vez capaz de producir un amor y un cariño tan grande que no cabe en su pecho, mi papá Javier.

A mis 4 abuelos, porque su amor es un regalo de la vida, es un sentimiento incomparable e indescriptible, por no perderse ni un detalle de mi formación como profesional, ustedes son la esencia de mi vida.

A mi confidente, amigo en malas y buenas, a la persona capaz de llevarme a donde él quisiera, mi hermano Carlos Javier.

A toda mi familia que, a pesar de la distancia, siempre estuvieron dispuestos a sacrificarse y brindarme lo que estaba al alcance de sus posibilidades.

A la memoria de mi abuela Rafaela, porque me brindó todo el amor que un niño pueda recibir y aunque no pudo acompañarme en esta etapa importante de mi vida, sé que donde quiera que se encuentre se siente orgullosa de mí.

A todas las otras que personas que de alguna manera forman parte de mi vida, en especial a Alejandro, Ariel , Patricia y Osniel que son como mis hermanos.

Agradecimientos

En primer lugar, quiero agradecerle a mi madre, por siempre estar cuando más te necesité, por todo el amor y cariño infinito que profesas cuando se trata de tus hijos, por tus jalones de oreja cuando la situación lo ameritaba, por tu esfuerzo, sacrificio, dedicación, porque junto a mí volviste a cursar la universidad, por saber guiarme por el camino correcto y no rendirte jamás. A ti te debo la vida y te agradezco profundamente haberme educado y formado de la manera que lo hiciste. Si hoy soy ingeniero en ciencias informáticas es gracias a ti. Te amo mamá.

A mi motor impulsor, mi referente, mi ídolo, eres el padre que cualquier hijo quisiera tener, te doy las gracias por enseñarme hacer el hombre que soy hoy en día, te doy las gracias porque sé que detrás de ese carácter fuerte, hay un gran padre amable, fiel, amigo y que por tus hijos haces cualquier sacrificio. Tus consejos nunca fallaron, siempre me dijiste que esto era como una inversión y que las inversiones no se hacían para perder, hoy ganamos todos te amo papá.

A mi hermano, que no importa las veces que discutamos, tu amistad y tu apoyo nunca me faltó, no me imagino una vida sin ti.

A mis abuelas Eulalia y Manuela, porque casi siempre cumplieron todos mi caprichos y antojos, por su amor incondicional, sus mimos y su protección cuando más lo necesitaba.

A mis abuelos Arango y Pedro, por sus consejos siempre constructivos, por siempre confiar en mi ciegamente, por su gran apoyo cuando decidí escoger esta carrera y en mi primer año.

A mis tías María Luisa (Mapucha), Esperanza, Maité, Tomasita, mis tíos Pedro Luis, Ariel, Renier, César, mis primos Manuel Enrique y Kirenia y a mi padrino Luis Orlando porque nunca me faltó un consejo de ustedes, porque lo que estuviera en sus manos para ofrecerme lo daban con todo el amor del mundo.

Al resto de mi familia que es bastante grande, porque de una forma u otra ayudaron para que este sueño se hiciera realidad y siempre creyeron que yo podía lograrlo.

A mis amigos de la adolescencia, en especial a Alejandro, Patricia, Ariel y Osniel que son mis otros hermanos, por nunca perder la comunicación conmigo aun estando lejos y preocuparse siempre como iban mis estudios.

A mi otra familia aquí en la escuela, estos cinco años han sido perfectos a su lado, gracias por compartir su amistad y permitirme vivir junto a ustedes momentos malos y buenos. Gracias en especial a mis compañeros de cuarto Oscar Luis, Jeanne, Adrián, Manuel, Ernesto y Daniel que saben que este triunfo es de ustedes también con mucho sacrificio.

A mis tutores Evelyn, Lester y Aliander por su dedicación, entrega y paciencia y a otras personas que me brindaron su ayuda cuando la necesitaba.

A la Revolución Cubana por darme la posibilidad de formarme y convertirme en el profesional que soy hoy.

Resumen

La Plataforma Educativa ZERA 2.0 es un Sistema de Gestión de Aprendizaje que integra un gran número de funcionalidades encaminadas a facilitar la interacción entre los docentes y estudiantes. En la plataforma no se puede consultar información vinculada a las actividades de pregrado, postgrado e investigación. Estas actividades se gestionan en el Sistema de Gestión Universitaria, lo cual es un impedimento para la búsqueda rápida de esta información por el usuario. El siguiente trabajo se enmarca en desarrollar un componente que permita la integración entre la Plataforma Educativa ZERA 2.0 y el Sistema de Gestión Universitaria en relación a las actividades de pregrado, postgrado e investigación. Para guiar el desarrollo de la solución propuesta se utilizó la metodología AUPUCI, así como el servidor de bases de datos PostgreSQL, el entorno integrado de desarrollo Netbeans, el servidor de aplicación Nginx, el *framework* Symfony, entre otras herramientas y tecnologías. Se realizaron pruebas unitarias, de sistemas y de integración, evidenciando que el componente se encontraba libre de errores y listo para ser utilizado. Como resultado se obtiene un componente funcional documentado que permite la integración de la Plataforma Educativa ZERA 2.0 con el Sistema de Gestión Universitaria.

Palabras Claves: componente, funcionalidades, integración, investigación, plataforma educativa, postgrado, pregrado.

Índice

Introducción

Las Tecnologías de la Información y la Comunicaciones (TIC) son un conjunto de servicios, redes y *software* que tienen como objetivo mejorar la calidad de vida de las personas dentro de un entorno y que se integran a un sistema de información interconectado y complementario. En un mundo cada vez más tecnológico, las TIC están en constante evolución y mantenerse informado es clave para conocer y aprovechar los nuevos servicios y ser competitivos. Estas han transformado la manera de trabajar de las personas, haciéndolos capaces de producir mucho más, con mejor calidad y en menos tiempo. (1)

Hoy día, Cuba no está excepto de los avances tecnológicos que existen en el mundo. El sector de la educación es uno de los más favorecidos con estos adelantos, pues ha permitido una mejor forma de enseñar y aprender. Una de las herramientas utilizadas en el proceso de enseñanza y aprendizaje son los Sistemas de Gestión de Aprendizaje (LMS por sus siglas en inglés) los cuales integran un gran número de funcionalidades encaminadas a facilitar la interacción entre los docentes y estudiantes.

Los LMS integran materiales didácticos, herramientas de comunicación, colaboración y gestión educativa, permiten administrar, distribuir, monitorear, evaluar y apoyar las diferentes actividades de un proceso de aprendizaje. Estos sistemas pueden utilizarse como núcleo del aprendizaje a distancia o como un complemento del aprendizaje presencial. Los LMS facilitan el seguimiento del proceso de aprendizaje de cada alumno, realizan evaluaciones, generan informes y ofrecen muchas herramientas de comunicación como pueden ser foros, chats o incluso videoconferencias. (2)

Un LMS tiene entre sus principales funciones: facilitar procesos de gestión (matrículas, selección de asignaturas optativas, etc.) facilitando las funciones de secretaría, creación y difusión de contenidos, planificación y organización de la formación. Existen diferentes tipos de LMS que brindan gran número de facilidades tales como Moodle, Sakai, Canvas, Blackboard, WebCT. (2)

La Universidad de Ciencias Informáticas (UCI) es una de las instituciones encargadas de producir aplicaciones y servicios informáticos en Cuba, a partir de la vinculación estudio-trabajo como modelo de formación. El Centro de Tecnologías para la Formación FORTES que pertenece a la Facultad 4 de la UCI, se encarga de brindar soluciones informáticas para

la educación. Dentro de él se desarrolla la Plataforma Educativa ZERA 2.0 que integra los principales conceptos de los Hiperentornos de Aprendizaje.

La posibilidad de diseñar cursos y el seguimiento específico de las actividades del estudiante, es una de las principales acciones que se pueden realizar en la Plataforma Educativa ZERA 2.0. Además, brinda la oportunidad de colocar al alcance del estudiante un gran cúmulo de ejercicios interactivos con una variedad de recursos asociados que posibilitan desarrollar en él, las más disímiles habilidades.

La UCI cuenta con un Sistema de Gestión Universitaria (Akademos) que gestiona información relacionada con las actividades del usuario en la entidad. A pesar de las ventajas que proporciona la Plataforma Educativa ZERA 2.0, esta posee algunas limitaciones; un usuario que se encuentre activo en la plataforma no tiene la oportunidad de consultar información vinculada a las actividades de pregrado, postgrado e investigación. Estas actividades se gestionan en el Sistema de Gestión Universitaria, lo cual es un impedimento para la búsqueda rápida de esta información por el usuario, pues tiene que salir de la Plataforma Educativa ZERA 2.0 e interactuar con el Sistema de Gestión Universitaria. La Plataforma Educativa ZERA 2.0 al ser un Sistema de Gestión de Aprendizaje de categoría en la Universidad de Ciencias Informáticas desea brindarles un mejor servicio a sus usuarios.

Tomando como punto de partida de esta investigación la situación descrita previamente, se identifica el siguiente **problema científico:** ¿Cómo brindar acceso a la información relacionada con las actividades de pregrado, postgrado e investigación del Sistema de Gestión Universitaria desde la Plataforma Educativa ZERA 2.0?

Por lo que se deduce como **objeto de estudio** la integración entre sistemas informáticos y como **campo de acción**, la integración de la Plataforma Educativa ZERA 2.0 con el Sistema de Gestión Universitaria en relación a las actividades de pregrado, postgrado e investigación.

Para dar solución al problema planteado se definió como **objetivo general** de esta investigación: Implementar un componente que posibilite la integración de la Plataforma Educativa ZERA 2.0 con el Sistema de Gestión Universitaria en relación a las actividades de pregrado, postgrado e investigación.

Del **objetivo general** trazado, se derivan los siguientes **objetivos específicos:**

- ✓ Establecer los referentes teóricos y metodológicos relacionados a la integración entre sistemas informáticos y en específico las plataformas virtuales y sistemas de gestión de información.
- ✓ Realizar el análisis y diseño del componente propuesto para la integración de la Plataforma Educativa ZERA 2.0 con el Sistema de Gestión Universitaria.
- ✓ Implementar un componente que permita la integración de la Plataforma Educativa ZERA 2.0 con el Sistema de Gestión Universitaria.
- ✓ Realizar pruebas de *software* al componente desarrollado**.**

Para dar cumplimiento a los **objetivos específicos** planteados, se proponen las siguientes **tareas de investigación:**

- ✓ Análisis de soluciones similares existentes.
- ✓ Investigación y selección de las técnicas, herramientas y metodologías a emplear en el desarrollo del componente.
- ✓ Implementación de un componente para la integración de la Plataforma Educativa ZERA 2.0 con el Sistema de Gestión Universitaria.
- ✓ Definición de los instrumentos y métodos para la validación de la propuesta de solución.
- ✓ Validación de la propuesta de solución.

Como **posible resultado** se desarrollará componente que garantice la integración de la Plataforma Educativa ZERA 2.0 con el Sistema de Gestión Académico. Un trabajo de tesis bien documentado el cual contiene información sobre la realización del componente.

Para la realización de la investigación fueron empleados **métodos empíricos y teóricos**, con el objetivo de dar solución a la problemática planteada.

Métodos teóricos

- ✓ ***Análisis histórico – Lógico:*** se utiliza con el fin de analizar bibliografías y sistemas homólogos para sintetizar los elementos importantes que se relacionan con esta investigación. Este método se utiliza también para tomar decisiones en cuanto a los resultados arrojados por las encuestas y las entrevistas realizadas y para asentar las bases teóricas para el desarrollo de la tesis.
- ✓ ***Analítico – Sintético***: utilizado para el análisis y estudio de documentos que se basen en el marco teórico de la investigación con el objetivo de conocer

teóricamente cómo han evolucionado los sistemas de gestión de información referente a los procesos de investigación, desarrollo e innovación tecnológica.

- ✓ ***Modelación***: se utiliza con el objetivo de crear modelos y diagramas que son abstracciones del producto final, permitiendo tener un dominio inicial de la información que se va a modelar, representar de forma estática los requisitos y obtener una versión del sistema original para validar los requisitos con el cliente.

Métodos empíricos

- ✓ ***Observación*:** usado para la recopilación de datos y necesidades existentes en el desarrollo del proyecto, permite comparar observaciones obtenidas a través de vías diferentes y escoger la mejor.
- ✓ ***Entrevista:*** se utiliza para conocer las necesidades del cliente y recopilar información para el desarrollo de la propuesta de solución. Se realiza de forma semi-abierta mediante conversaciones de carácter profesional con el cliente.

Estructura capitular

Capítulo I. Fundamentación teórica: en este capítulo se hace referencia a los elementos teóricos en los cuales está basado la investigación, se incluye un estudio del estado del arte de este tema. Se exponen los lenguajes de programación utilizados, así como las metodologías, herramientas y tecnologías en el desarrollo de la solución. Se encuentran los principales conceptos relacionados con el contenido.

Capítulo II. Análisis y diseño del sistema: en este capítulo se describe la solución propuesta para integrar los sistemas antes mencionados. Se detallan las principales características, así como los requisitos funcionales y no funcionales. Se presenta una descripción de las funcionalidades a implementar. Además, se realizan una serie de diagramas relacionados a la investigación.

Capítulo III. Implementación y prueba: en este apartado se describen las fases de implementación y pruebas. Se realiza la implementación de todas las funcionalidades identificadas, logrando un componente que satisface las principales necesidades del cliente. Se detallan además las pruebas realizadas al sistema, una vez que concluye la implementación, para asegurar que este cumple con las especificaciones requeridas, para de esa manera asegurar la calidad y eficiencia de la solución.

Capítulo 1: Fundamentación teórica

En este capítulo se exponen los principales conceptos relacionados con la fundamentación teórica. También se analizan las soluciones similares a nivel mundial hasta la actualidad. Además, se incluyen las descripciones de las herramientas, tecnologías, metodologías y lenguajes de programación que se utilizarán en el desarrollo de la solución.

1.1 Conceptos asociados al dominio del problema

Para una mejor comprensión del problema a resolver y su solución, se relacionan a continuación diferentes términos propios vinculados al dominio del problema.

Sistemas informáticos

Es un sistema que permite almacenar y procesar información: es el conjunto de partes interrelacionadas: hardware, *software* y personal informático. (3)

Plataformas educativas

Con el nombre de “Plataforma” es como genéricamente se conoce a la herramienta tecnológica usada para distribuir el conocimiento. En contextos de formación, se refiere al conjunto de equipos y *software* básico sobre el cual va a funcionar un sistema que se desea diseñar, desarrollar, o instalar para apoyar actividades de aprendizaje electrónico. (4)

E-Learning es un sistema de educación electrónico o a distancia en el que se integra el uso de las tecnologías de la información y elementos pedagógicos (didácticos) para la formación, capacitación y enseñanza de los usuarios o estudiantes en línea, es decir, se basa en adquirir conocimientos por medios electrónicos, se puede entender como una modalidad de aprendizaje dentro de la educación a distancia. Utiliza herramientas y medios diversos como Internet, Intranets, CD-ROM, producciones multimedia (textos, imágenes, audio, video, etc.), entre otros. Literalmente, *e-learning* es aprendizaje con medios electrónicos: enseñanza dirigida por la tecnología. (5)

Después de conocer la definición sobre plataformas y sistemas *e-learning* podemos decir que una plataforma *e-learning* o también conocido por el término "enseñanza virtual", es un sistema de formación interactivo para desarrollar programas de enseñanza, que hace uso masivo de los medios electrónicos para llegar a un alumnado generalmente remoto. O sea,

es una capacitación no presencial que, a través de plataformas tecnológicas, posibilita y flexibiliza el acceso y el tiempo en el proceso de enseñanza-aprendizaje, adecuándolos a las habilidades, necesidades y disponibilidades de cada docente.

Plataforma Educativa ZERA 2.0

Plataforma de gestión de aprendizaje, actualmente en desarrollo, donde un centro educativo, institución o empresa, gestiona recursos educativos proporcionados por unos docentes y organiza el acceso a esos recursos por los estudiantes y además permite la comunicación entre todos los implicados (alumnado y profesorado), contribuyendo de esta manera a la evolución de los procesos de enseñanza y aprendizaje. Es decir, la Plataforma Educativa ZERA 2.0 se encarga, entre otros aspectos, de presentar los cursos a los usuarios y del seguimiento de la actividad de los estudiantes.

Se puede afirmar que las plataformas educativas son un entorno virtual que acogen variadas herramientas para fines docentes. Su función es permitir la creación y gestión de cursos que permitan elevar el nivel de conocimiento de sus usuarios.

Hiperentornos de Aprendizaje

Sistema informático basado en tecnología hipermedia, en la que se mezclan disímiles dispositivos que representan las diversas tipologías de *software* educativo con el designio de atender las necesidades didácticas. (6)

Interoperabilidad

La norma estadounidense ANSI/NISO Z39.19:2005 y la norma británica BS8723-4:2007, coinciden en definir la interoperabilidad como la capacidad que tienen dos o más sistemas o componentes de intercambiar información y usar esa información que se ha intercambiado. (7)

Después de analizados los conceptos anteriores, se puede afirmar que las plataformas educativas son un entorno virtual que acogen variadas herramientas para fines docentes. Su función es permitir la creación y gestión de cursos que permitan elevar el nivel de conocimiento de sus usuarios. El conocimiento adquirido a través de ellos, brindará un mejor apoyo a la hora del estudio de soluciones similares.

1.2 Análisis de soluciones similares en el mundo

Con el objetivo de obtener conocimiento y hallar una solución al problema existente fue necesario realizar un estudio sobre las tendencias actuales sobre integración de sistemas que existen. Algunas de las soluciones relevantes con respecto a la temática en cuestión son analizadas a continuación.

Integración entre Moodle y Mahara (Mahoodle)

¿Qué es Moodle?

Moodle es un paquete de *software* para la creación de cursos y sitios web basados en internet, es decir, una aplicación para crear y gestionar plataformas educativas o espacios donde un centro educativo, institución o empresa, gestiona recursos educativos proporcionados por un(os) docente(s) y organiza el acceso a esos recursos por los estudiantes y además permite la comunicación entre todos los implicados (alumno y profesor). (8)

¿Qué es Mahara?

Mahara es un sistema combinado entre un portafolio y una red social, (un portafolio donde los estudiantes dejan evidencias del avance y desarrollo de su aprendizaje permanente, estos pueden ser ensayos, obras de arte, o cualquier otra labor realizada que pueda ser almacenada en un formato digital). (9)

La integración entre Moodle y Mahara ofrece soluciones que mejoran significativamente la productividad de los clientes. Con esto en mente decidieron integrar la plataforma de gestión de aprendizaje en línea (LMS) Moodle con la plataforma ePortafolio Mahara (integración moodle epotafolio), la cual brindará a los clientes una experiencia sólida, una interfaz gráfica agradable con los beneficios de las dos plataformas integradas en su totalidad. (9)

Integración entre la Plataforma Virtual Educativa y el HRIS Sistema de Recursos Humanos

La integración permite que los datos de los empleados importantes como departamento, cargo, ubicación, salario, etc., se mantengan al día a través de su plataforma virtual y el HRIS sistema de recursos humanos. Cuando se añade un nuevo empleado al sistema de recursos humanos HRIS, un perfil se crea automáticamente para ellos en la plataforma

virtual de una manera amigable y sencilla importando los datos más importantes a la plataforma educativa virtual LMS. En cuestión la integración ayuda a la eficacia y agilidad del trabajador de recursos humanos. (10)

Integración entre el sistema de gestión académica SIU-guaraní de La Universidad Nacional de La Plata y la plataforma Moodle

La Universidad Nacional de La Plata (UNLP) es la institución de educación superior pública, de Argentina, 2° en el país en cantidad de alumnos. Fundada en 1905, incluye 18 unidades académicas, 4 escuelas de pregrado, 220 laboratorios de investigación, 8 secretarías y más de 30 direcciones que dependen del rectorado y permiten la gestión de más de 107.000 alumnos de grado, 9864 de postgrado. La facultad de informática, al último año informado 2010, cuenta con 3.700 alumnos y un promedio de 800 ingresantes anuales. En el caso particular de la facultad, desde hace más de 10 años se utilizan plataformas virtuales para el complemento de las clases presenciales. (11)

En la facultad de informática de la UNLP se utilizan plataformas virtuales, sistemas de gestión académica y también las redes sociales, que ya no se constituyen como sistemas aislados sino como sistemas destinados a interactuar y comunicarse. En la UNLP se usa el sistema de gestión académica SIU-guaraní, que permite la gestión de alumnos dentro de cada facultad, desde que ingresan hasta que se diploman y provee una serie de controles y pistas de auditoría para los organismos de gestión de títulos. Fue desarrollado por el Ministerio de Educación de la Nación para todas las universidades nacionales. En el año 2011, se realizó la integración de este sistema con la plataforma Moodle y de esta manera es posible hacer una sincronización entre el sistema administrativo y la herramienta educativa que permite compartir cursos y usuarios. (11)

1.3 Análisis de soluciones similares en Cuba

En nuestro país también se han desarrollado hasta la fecha algunos proyectos que tributan a la integración de sistemas. Algunos de los más destacados por sus logros y resultados obtenidos en tal sentido se muestran a continuación.

Sistemas de interacción por correo a los jueces en línea de programación: COJMail v1.0

Un sistema creado por la Universidad de las Ciencias Informáticas (UCI) en el 2011. Su función principal era establecer comunicación y consumir servicios de los Jueces en Línea Caribeño (COJ) a través de correo electrónico. Las instituciones mediante el correo podían interactuar con el COJ y su resultado era como si estuvieran accediendo directamente a la web. El principal problema de esta primera versión del COJMail es que no es compatible con la arquitectura actual que presenta el COJ, razón por la cual no está en condiciones de realizar prestaciones para este juez en línea. (12)

Interacción entre la plataforma Croda y Rhoda

Rhoda

Es un Repositorio de Objetos de Aprendizaje (ROA), desarrollado por especialistas de la UCI. Tiene como objetivo fundamental almacenar y gestionar los objetos de aprendizaje que se crean por parte de la comunidad universitaria. Es una aplicación web modular y multiplataforma, que provee un lugar común accesible a través de un navegador, donde los usuarios pueden almacenar, recuperar y consultar objetos de aprendizaje, destinados a fortalecer el proceso de enseñanza-aprendizaje.

Croda

Es una herramienta de autor web creada en la UCI, que facilita la creación de Objetos de Aprendizaje interoperables, reutilizables, accesibles y duraderos, de manera flexible, aplicando para ello el estándar para la descripción de recursos Learning Object Metadata (LOM) y el estándar de contenido Sharable Content Object Reference Model (SCORM). Esta herramienta de autor web permite la creación de cursos para los distintos entornos educativos.

La herramienta Croda consume servicios web del repositorio Rhoda de tipo SOAP, aunque estas dos plataformas se encuentran ya actualmente en desuso, tienen una gran importancia en cuanto a ejemplo de consumir servicios web.

Luego de un estudio, se llega a la conclusión de que ninguno de estos sistemas aporta a la solución del problema, debido a que no se adaptan a las necesidades del proyecto, utilizan diferentes protocolos para la integración y trabajan con diferentes herramientas privativas, aunque permiten tener una idea de la propuesta de solución y complejidad del problema.

1.4 Metodología de desarrollo de software

Es la forma en que se realiza algo o el método con el cual se llevará a cabo el proceso de desarrollo de *software*, en este caso. Básicamente consiste en seguir al pie de la letra los pasos para llevar a cabo el proceso, mediante el cual se creará un programa o sistema o cualquier tipo de *software*. Estas se clasifican en dos grandes grupos.

Las metodologías ágiles

Se caracterizan por hacer énfasis en la comunicación cara a cara, es decir, se basan en una fuerte y constante interacción, donde clientes y desarrolladores trabajan constantemente juntos. Estas metodologías están orientadas al resultado del producto y no a la documentación; exige que el proceso sea adaptable, permitiendo realizar cambios de último momento. Se puede hacer mención dentro de las metodologías ágiles a: XP (por sus siglas en inglés Extreme Programming), Scrum, Crystal Methodologies, Proceso Unificado Ágil o Agile Unified Process (AUP). (13)

Las metodologías robustas o tradicionales

Están guiadas por una fuerte planificación. Centran su atención en llevar una documentación exhaustiva de todo el proceso de desarrollo y en cumplir con un plan de proyecto, definido en la fase inicial del mismo. Entre las metodologías robustas se encuentran: MSF (por sus siglas en inglés Microsoft Solution Framework), MÉTRICA 3 y RUP (siglas de Rational Unified Process). (14)

A continuación, se muestran claramente las diferencias entre metodologías ágiles y tradicionales, que no se refieren solo al proceso en sí, sino también al contexto de equipo y organización que es más favorable a cada uno de estas filosofías de procesos de desarrollo de *software*. (14)

Tabla 1. Diferencias entre metodologías ágiles y tradicionales.

Metodologías ágiles	Metodologías tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo de desarrollo).	Impuestas extremamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (menos de diez integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Muchos artefactos.
Pocos roles.	Muchos roles.
Menos énfasis en la arquitectura de software.	La arquitectura de software es esencial y se expresa mediante modelos.

Metodología a utilizar y su fundamentación

La UCI decidió hacer una variación de la metodología Proceso Unificado Ágil o Agile Unified Process (AUP) llamada AUP-UCI. Con esta adaptación de AUP se logra estandarizar el proceso de desarrollo de *software*, dando cumplimiento además a las buenas prácticas que define el estándar internacional de calidad CMMI-DEV v1.3. También permite que se adapte el ciclo de vida definido para la actividad productiva de la institución. (15)

Esta metodología facilita el trabajo en proyectos de pequeña envergadura y proporciona un ambiente de desarrollo de *software* iterativo e incremental. En AUP-UCI sólo se utilizan los artefactos que son imprescindibles y realmente necesarios para la realización del producto. Aplica técnicas ágiles incluyendo: desarrollo dirigido por pruebas, modelado ágil, gestión de

cambios ágil y refactorización de base de datos para mejorar la productividad y también que permite realizar cambios durante la duración del proyecto.

1.5 Herramientas y tecnologías asociadas al desarrollo del componente

A continuación, se da una relación de las principales herramientas y tecnologías que se emplearán durante el desarrollo del componente, así como la fundamentación de la elección realizada.

Sistema Gestor de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) o DBMS (DataBase Management System) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Permiten definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. Algunos ejemplos son Oracle, DB2, PostgreSQL, MySQL, MS SQL Server, entre otros. (16)

Permite:

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

Oracle

Es un sistema de gestión de base de datos de tipo objeto-relacional (ORDBMS, por el acrónimo en inglés de Object-Relational Data Base Management System), desarrollado por Oracle Corporation. Se considera como uno de los sistemas de bases de datos más completos, destacando: soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma. Su dominio en el mercado de servidores empresariales había sido casi total hasta que recientemente tiene la competencia del Microsoft SQL Server y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySQL o Firebird. (17)

MySQL

Sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la bases de datos *open source* más popular del mundo junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web. (18)

MS SQL Server

Microsoft SQL Server es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft. Puede ser configurado para utilizar varias instancias en el mismo servidor físico. (19)

SGBD a utilizar y su fundamentación

¿Qué es PostgreSQL?

Es un servidor de base de datos objeto-relacional libre, pues incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional, liberado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group). (20)

Se utilizará el gestor PostgreSQL versión 9.5 debido a su gran estabilidad, potencia, robustez, facilidad de administración, implementación de estándares y además que funciona muy bien con grandes cantidades de datos. Es un *software* con licencia libre y su motor de bases de datos de código abierto es el más potente en la actualidad.

Lenguajes de programación

Lenguaje artificial que puede ser usado para controlar el comportamiento de una máquina, especialmente una computadora. Hay que distinguir dos partes en el lenguaje de programación, lo que se denomina sintaxis del lenguaje y la semántica. Por sintaxis entendemos el conjunto de las construcciones del lenguaje que consideramos correctas en cuanto a su forma, mientras que la semántica es ese mismo conjunto de construcciones que consideramos correctas en cuanto al significado. (21)

Fundamentación de los lenguajes de programación a utilizar

Para el desarrollo de aplicaciones web existe un gran número de lenguajes de programación, divididos en dos grupos, lenguajes del lado del cliente y lenguajes del lado del servidor.

Lenguajes de programación del lado del cliente

Cuando se programa una página web, en la mayoría de los casos se utiliza los que se conocen como “lenguajes del lado del cliente”. El servidor no interviene para nada en el proceso de crear la página web solicitada por el usuario. Los lenguajes utilizados en este caso son el conocido HTML (HiperText Markup Language) o Lenguaje Hipertexto de Marcas, JavaScript, Visual Basic Script (VBScript) o CSS 3 (Cascading Style Sheets) o hojas de estilo en cascada. (22)

HTML v5

Es un lenguaje de etiquetas, donde estas le comunican al navegador cuál es la información a mostrar por pantalla, además del formato de dicha información. Propone estándares para cada aspecto de la web y también un propósito claro para cada una de las tecnologías involucradas. Es un lenguaje muy fácil de comprender y muy utilizado para la presentación de información, permite definir la estructura y el contenido de las páginas, permitiendo combinar textos, imágenes, sonidos, vídeos y enlaces a otras páginas. (23)

JavaScript

Lenguaje de programación universal presentes en numerosas páginas HTML, de manera complementaria a este código, presenta programas, llamados comúnmente scripts. Estos normalmente consisten en unas funciones que son llamadas desde el propio HTML cuando algún evento sucede. De ese modo, podemos añadir efectos como que un botón cambie de forma al pasar el ratón por encima, o abrir una ventana nueva al pulsar en un enlace, entre otros. JavaScript contribuye mucho a la facilidad de uso de un sitio web. (24)

CSS 3

Lenguaje de hojas de estilo en cascada que trabaja junto con HTML para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo, bordes, etc. Mientras que HTML nos permite definir la estructura de una página web, las hojas de estilo en cascada son las que nos ofrecen la posibilidad de definir las reglas y estilos de

representación en diferentes dispositivos, ya sean pantallas de equipos de escritorio, portátiles, móviles, impresoras u otros dispositivos capaces de mostrar contenidos web. (25)

Lenguajes de programación del lado del servidor

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red y otras tareas para crear la página final que verá el cliente. Los lenguajes de lado servidor más ampliamente utilizados para el desarrollo de páginas dinámicas son el ASP, JSP, PERL y PHP. El que se utilizará es el lenguaje PHP debido a que el proyecto al cual va dirigido este trabajo de diploma también trabaja con él. (26)

PHP 7

Es el acrónimo de Hipertext Preprocesor. Lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Fue creado originalmente en 1994 por Rasmus Lerdorf, pero como PHP está desarrollado en política de código abierto, a lo largo de su historia ha tenido muchas contribuciones de otros desarrolladores. (27)

Ventajas

- ✓ Se caracteriza por ser un lenguaje muy rápido.
- ✓ Soporta en cierta medida la orientación a objeto. Clases y herencia.
- ✓ Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- ✓ Capacidad de conexión con la mayoría de los manejadores de base de datos: MysSQL, PostgreSQL, Oracle, MS SQL Server, entre otras.
- ✓ Capacidad de expandir su potencial utilizando módulos.
- ✓ Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones.
- ✓ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- ✓ Incluye gran cantidad de funciones.
- ✓ No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Framework de desarrollo

Desde el punto de vista del desarrollo de *software*, un *framework* es una estructura de soporte definida, en la cual otro proyecto de *software* puede ser organizado y desarrollado. Suelen incluir soporte de programas**,** bibliotecas, lenguaje de *scripting*, *software* para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas. Además, permiten facilitar el desarrollo de *software* y evitar los detalles de bajo nivel, permitiendo concentrar más esfuerzo y tiempo en identificar los requerimientos de *software*. (28)

Framework de desarrollo a utilizar.

Para el desarrollo de la solución serán utilizados los marcos de trabajo JQuery v2.0 y JQuery UI v1.10.0 para el trabajo con el lenguaje JavaScript*,* Bootstrap v3.0, para el trabajo con CSS y como *framework* PHP se utilizará Symfony v2.7. La elección de estos marcos de trabajo viene dada por sus características y porque son los utilizados en el proyecto al cual va dirigido el presente trabajo de diploma.

Symfony v2.7

Framework diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo Vista Controlador. Separa la lógica del negocio, el modelo de datos y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (29)

Symfony es compatible con la mayoría de gestores de bases de datos y se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Bootstrap v3.0

Bootstrap es un *framework* que sirve para crear sitios web responsive (adaptables a móviles) de forma rápida y muy simple. Básicamente Bootstrap es un conjunto de herramientas gratuitas para la creación de sitios y aplicaciones web. No requiere instalación en ningún sistema operativo especial debido a que no es un *software*, sino un paquete de códigos que se descargan e implementan. Contiene combinaciones de código basado en CSS para representar y estilizar tipografías, formas, botones, menús de navegación y

muchos otros componentes de la interfaz de usuario, así como extensiones de JavaScript opcionales para darle interactividad a tus sitios. Bootstrap te libera de escribir decenas de líneas de código CSS y Javascript pues la mayoría de los elementos vienen pre-definidos y listos para usar, sin que ello signifique que no puedas modificarlos o crear tus propios estilos adicionales. Es súper versátil. (30)

JQuery v2.0

Framework para el lenguaje Javascript que implementa una serie de clases (de programación orientada a objetos) que nos permiten programar sin preocuparnos del navegador con el que nos está visitando el usuario, debido a que funcionan de exacta forma en todas las plataformas más habituales. Ofrece una infraestructura con la que tendremos mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Posee licencia para uso en cualquier tipo de plataforma, personal o comercial. (31)

JQuery UI

Es una biblioteca de componentes para el *framework* JQuery que le añaden un conjunto de plugins, widgets y efectos visuales para la creación de aplicaciones web. Cada componente o módulo se desarrolla de acuerdo a la filosofía de JQuery. Es flexible y rápido para el desarrollo web, es open source, tiene plugins y una excelente comunidad de soporte, además los bugs son resueltos rápidamente y posee excelente integración con AJAX. (32)

Entorno de desarrollo integrado.

Un entorno de desarrollo integrado, llamado también IDE (Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). (33)

- ✓ **Editor de código fuente:** Editor de texto que sirve para editar el código fuente de aplicaciones informáticas.
- ✓ **Un compilador:** Es un traductor de código fuente, lo traduce a un lenguaje que sea legible para las máquinas.
- ✓ **Un depurador:** Es una aplicación que tiene como función probar y eliminar posibles errores en un programa en desarrollo.

- ✓ **Constructor de interfaz gráfica:** Herramienta que sirve para crear y diseñar las interfaces con las cuales habrá interacción entre la aplicación y el usuario.

Existen muchos IDE de múltiples lenguajes tales como Eclipse, ActiveState Komodo, IntelliJ IDEA, MyEclipse, Oracle JDeveloper, NetBeans, PhpStorm, Codenvy y Microsoft Visual Studio.

IDE a utilizar y su fundamentación.

NetBeans IDE es un ambiente integrado de desarrollo de código abierto para desarrolladores de *software* que intentan construir aplicaciones usando mayormente Java, o algún otro lenguaje de programación popular como C++, PHP, Python, Groovy, Ruby y otros. La plataforma NetBeans logró ser creada con la contribución de la comunidad de código abierto, siendo un paquete IDE bien diseñado que puede ser usado para la creación de cualquier tipo de aplicaciones de escritorio, web y *mobile*. También hace foco en permitir el desarrollo de aplicaciones usando conjuntos de componentes modulares de *software* que pueden ser creados tanto como por los propietarios de NetBeans, la Corporación Oracle, como por terceros desarrolladores que han logrado expandir sus funcionalidades con numerosos plugins. Tres módulos principales de NetBeans IDE son NetBeans Profiler (que puede monitorear aplicaciones, reportar problemas y más), el editor de JavaScript NetBeans y la herramienta de diseño GUI. (33)

Servidor web

Programa diseñado para permitir la interacción entre ordenadores. Suele funcionar permaneciendo a la espera de peticiones, cuando las recibe responde a ellas transfiriendo documentos de tipo hipertexto, para ello implementa el protocolo HTTP (HyperText Transfer Protocol). El término también se emplea para referirse al ordenador que ejecuta el programa. (34)

Servidor web a utilizar y su fundamentación.

Nginx 1.10.0-0ubuntu0.16.04.4 es un servidor web y proxy inverso, multiplataforma, ligero y de alto rendimiento, es un *software* libre, liberado bajo licencia BSD. Se puede usar Nginx como caché, permitiendo mejorar la eficiencia de tu aplicación sin tocar la programación de la misma. Este servidor web puede funcionar como balanceador de carga, distribuyendo el

tráfico entre varios servidores, permitiendo mayor escalabilidad. Además, que es el servidor utilizado por los miembros del proyecto al cual va dirigido este trabajo de diploma. (35)

Lenguaje de modelado

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño. Es un lenguaje artificial que sirve para expresar un modelo con símbolos gráficos, en forma de diagramas. Al estar estandarizado permite a los desarrolladores documentar el *software* en cuanto a funcionalidades, procesos de negocios y conceptos asociados. (36)

Lenguaje de modelado a utilizar y su fundamentación.

El Lenguaje de Modelado Unificado (UML: Unified Modeling Language) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s. Fusiona los conceptos de la orientación a objetos aportados por Booch, OMT y OOSE (Booch, G. et al., 1999). Incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. Es el lenguaje utilizado en el proyecto al cual va dirigido este trabajo de tesis. (36)

Protocolos de servicios web

Servicios web

Conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la web. (37)

Estándares y Protocolos

- ✓ XML (Extensible Markup Language), todas las tecnologías de servicios Web se basan en XML. El diseño de XML se deriva de dos fuentes principales: SGML (Standard Generalized Markup Language) y de HTML (HyperText Markup Language).
- ✓ UDDI (Universal Description, Discovery and Integration), es un protocolo para describir los componentes disponibles de servicios Web.

- ✓ SOAP (Simple Object Access Protocol) es un protocolo para iniciar las conversaciones con un servicio UDDI. El SOAP simplifica el acceso a los objetos, permitiendo a las aplicaciones invocar métodos objeto o funciones, que residen en sistemas remotos. Una aplicación SOAP crea una petición bloque en XML, proporcionando los datos necesarios para el método remoto, así como la ubicación misma del objeto remoto.
- ✓ WSDL (Web Service Description Language), es el estándar propuesto para la descripción de los servicios web, el cual consiste en un lenguaje de definición de interfaz (IDL- Interface Definition Language) de servicio basado en XML, que define la interfaz de servicio y sus características de implementación. El WSDL es apuntado en los registros UDDI y describe los mensajes SOAP que definen un servicio Web en particular.

Herramienta para el modelado

Las herramientas CASE (Ingeniería de Software Asistida por Computadoras) son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un *software*. Consiste en conseguir una mejora en la productividad y en la calidad del producto final, reducir mantenimiento de los sistemas informáticos y facilitar el uso de las distintas metodologías propias de la ingeniería del *software*. (38)

Visual Paradigm 8.0

Software de modelado UML que nos permite analizar, diseñar, codificar, probar y desplegar. Dibuja todo tipo de diagramas UML, genera código fuente a partir de dichos diagramas y también posibilita la elaboración de documentos. Soporta UML, SysML (Systems Modeling Language), BPMN (Modelo y Notación de Procesos de Negocio), DFD (Diagrama de Flujo de Datos), ERD (Diagrama Entidad-Relación), diagramas, entre otros. El programa cuenta con innumerables ventajas; una de las más importantes es la aptitud para representar todas las funciones posibles. A pesar de ser privativo la UCI paga la licencia para el estudio en la identidad. (39)

1.6 Conclusiones del capítulo.

- ✓ Los conceptos asociados al dominio del problema enmarcados en este capítulo, enriquecieron el conocimiento sobre los sistemas informáticos, las plataformas y la interoperabilidad.
- ✓ El estudio sobre la integración entre plataformas virtuales, contribuyó a elaborar una propuesta adecuada para dar solución al problema expuesto, aunque estos sistemas no aportaron a la solución del componente, pues no se adaptan a las necesidades del proyecto, estos sistemas no poseen código abierto para consultar información sobre sus funcionalidades, utilizan diferentes protocolos para la integración y trabajan con diferentes herramientas privativas.
- ✓ La metodología AUPUCI, el servidor de bases de datos PostgreSQL, el entorno integrado de desarrollo Netbeans, el servidor de aplicación Nginx, el *framework* Symfony, entre otras herramientas y tecnologías utilizadas se podrá implementar un componente que permita la integración entre la Plataforma Educativa ZERA 2.0 y el Sistema de Gestión Universitaria.

Capítulo 2: Análisis y diseño del sistema

El desarrollo de *software* que este trabajo propone sigue un proceso de análisis y diseño que proporciona los cimientos bajo los cuales se va a desarrollar la aplicación. Es por esto que en este capítulo se detallan los procesos de ingeniería de *software*, análisis y diseño que se involucran para el desarrollo de una aplicación de *software*. Se exponen como artefactos que permiten describir la propuesta de solución: modelo de dominio, la especificación de requisitos funcionales y no funcionales, las historias de usuario que incluyen a su vez los prototipos de interfaz de usuario, el modelo de análisis (se identifican las clases del análisis y se realizan los diagramas de clases y de colaboración) y el modelo de diseño (encargado de realizar los diagramas de clases y de secuencia con estereotipos web). Otro aspecto que se define son los patrones de diseño a utilizar, así como el modelo de entidad relación de la base de datos y el diagrama de despliegue.

2.1 Modelo de dominio

Brinda los conceptos significativos para el domino del problema. El lenguaje UML ofrece la notación en diagramas de estructuras estáticas para graficar los modelos conceptuales. (40)

Conceptos del dominio

Plataforma Educativa ZERA 2.0: plataforma educativa destinada a apoyar el proceso de enseñanza-aprendizaje en la UCI, permite a los distintos usuarios del sistema intervenir en un ambiente dinámico e interactivo.

Aplicación externa: herramienta externa a la plataforma creada con el fin de brindar información (académica, extensionista, residencia, entre otras) sobre el usuario, en este caso el Sistema de Gestión de Universitaria (Akademos) que proveerá los servicios para mostrar en la plataforma.

Usuario: persona que interactúa con la plataforma. Existen dos tipos de usuarios: estudiantes y profesores. Ambos consumen recursos y servicios de la plataforma.

Información: contiene datos del usuario (estudiantes o profesor) en la rama académica como resumen de evaluaciones, registro de asistencia, actividades realizadas.

Diagrama del modelo de dominio

El diagrama de la Figura 1 muestra la relación que existe entre los conceptos del dominio identificados y descritos previamente, para un mayor entendimiento del problema.

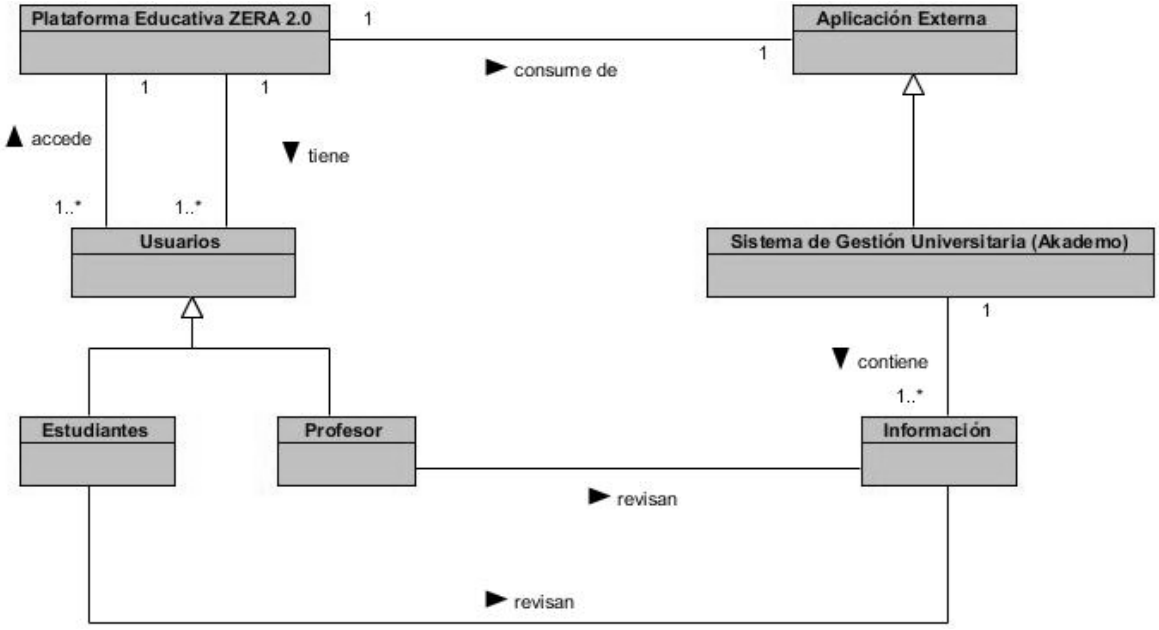


Figura 1. Modelo de dominio

2.2 Descripción del sistema propuesto

El sistema propuesto tiene como objetivo la interacción de información entre la Plataforma Educativa ZERA 2.0 y una aplicación externa como el Sistema de Gestión Académico. Las funcionalidades propuestas se localizan en el perfil del usuario, encargado de agrupar la información relacionada con las actividades de pregrado, postgrado e investigación enviadas desde el Sistema de Gestión Académico. Los servicios son recibidos a través de direcciones Uniform Resources Locator (URL), una de las funcionalidades es configurar esas direcciones con el nombre de la función y sus parámetros de forma visual en los formularios, sin la necesidad de ir al código de la aplicación. Solo un usuario con rol administrador puede configurar las direcciones URL. Este componente brinda a los usuarios la posibilidad de contar con una plataforma donde se muestra información real de sus actividades en la Universidad, brindando un buen servicio de calidad y aprovechamiento de su tiempo.

2.3 Requerimientos del sistema

Comprende todas las tareas relacionadas con la determinación de las necesidades o de las condiciones a satisfacer para un *software* nuevo o modificado, tomando en cuenta los diversos requerimientos de los clientes. Estos se dividen en dos grupos: requisitos funcionales (RF) y requisitos no funcionales (RNF).

Requisitos funcionales

Los requisitos funcionales permiten identificar las condiciones que debe cumplir un sistema para satisfacer un contrato, estándar, especificación u otra documentación formalmente impuesta. (41)

Para el correcto funcionamiento de la solución propuesta se espera que el sistema permita:

RF1. Ver resumen de evaluaciones de un usuario: Permite al usuario revisar sus evaluaciones por cada asignatura, su promedio general, al igual que los puntos por bonificaciones y examen de premios.

RF2. Ver registro de asistencias de un usuario: Permite que el usuario pueda verificar su registro de asistencias en cada asignatura cursada, desglosada en tres aspectos (presente, ausente e injustificadas) cada una en forma de por ciento.

RF3. Ver los cursos realizados por un usuario con rol profesor: Muestra todos los cursos en los que ha participado el profesor autenticado.

RF4. Ver los eventos propuestos para un usuario: Muestra todos los eventos que están disponibles para elección por el usuario autenticado.

RF5. Ver las publicaciones de un usuario: Muestra las publicaciones realizadas por el usuario autenticado.

RF6. Ver los premios de un usuario: Muestra el listado de premios obtenidos por el usuario autenticado.

RF7. Permitir configurar servicios web: Posibilita al usuario con rol administrador, añadir un servicio, al configurar la dirección, el nombre, tipo de servicio y los parámetros de este.

RF8. Ver datos de un usuario: Muestra los datos personales y docentes del usuario autenticado.

Requisitos no funcionales

Requerimientos que permiten definir las cualidades o propiedades que el *software* debe tener, con el objetivo de crear un producto final que sea fácil de utilizar, rápido y confiable. A continuación, se listan los requisitos no funcionales identificados:

Software:

- ✓ El sistema debe ser instalado en un entorno con sistema operativo con: Distribución de CentOS.
- ✓ Deberá contar con un navegador web moderno (Navegadores web: Firefox (v10.x en adelante), Chrome (v20.x en adelante), Opera (v10.x en adelante), navegadores de dispositivos móviles actualizados) para acceder a la aplicación y tener instalado el plugin de Java.

Hardware:

- ✓ El sistema será accesible desde estaciones de trabajo de escritorio, laptop, tablets y smartphones.
- ✓ Servidor de aplicaciones Nginx: 2.7.x con memoria RAM: 16 GB, disco Duro: 500 GB y microprocesador: 6 x 800 GHz.
- ✓ Servidor de bases de datos relacional PostgreSQL 9.4.x con memoria RAM: 16 GB, disco Duro: 100 GB y microprocesador: 6 x 800 GHz.

Apariencia o interfaz externa:

- ✓ Cumplir con las pautas de diseño establecidas en la Estrategia Marcaria de la Universidad.

Seguridad:

- ✓ Garantizar el acceso a las funcionalidades definidas para los usuarios de acuerdo a los roles que posean.

Usabilidad:

- ✓ El sistema podrá ser usado por personas que posean conocimientos básicos de informática y del manejo de ordenadores.
- ✓ Se debe mantener informado al usuario del resultado de las acciones realizadas.

Accesibilidad:

- ✓ El sistema debe estar disponible desde cualquier estación de trabajo conectada a la red y permitir su administración de forma remota.

2.4 Historias de usuarios

Las historias de usuario (HU) son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes. (42)

Las HU tienen un orden que le permite al cliente organizar sus ideas y facilitan al equipo de trabajo identificar cuales tienen más prioridades en el momento del desarrollo de la solución. Son ideas agrupadas de acuerdo con su funcionalidad.

A continuación, se muestra la historia de usuario del requisito Permitir configurar servicios web, las restantes se describen en el Anexo 2 del presente documento:

Tabla 2. Historia de usuario del RF Permitir configurar servicios web.

Número: 7	**Nombre del requisito:** Permitir configurar servicios web
Programador: Javier Alejandro Arango López	**Iteración Asignada:** 1era
Prioridad: alta	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 5 días
Descripción: **1- Objetivo:** Posibilita al usuario con rol administrador añadir un servicio; al configurar la dirección, el nombre, tipo de servicio y los parámetros de este. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para añadir un servicio hay que: - Estar autenticado en el sistema con rol (administrador). **3- Comportamientos válidos y no válidos (flujo central y alternos):** **Url:** campo de carácter que representa la dirección url del servicio a añadir o modificar.	

Admite letras y números y caracteres especiales (/. :).

Nombre: campo de carácter que representa el nombre del servicio a añadir o modificar. Admite solo letras.

Configuración: campo de carácter que representa la configuración del servicio a añadir o modificar. Admite letras y carácter especial (_).

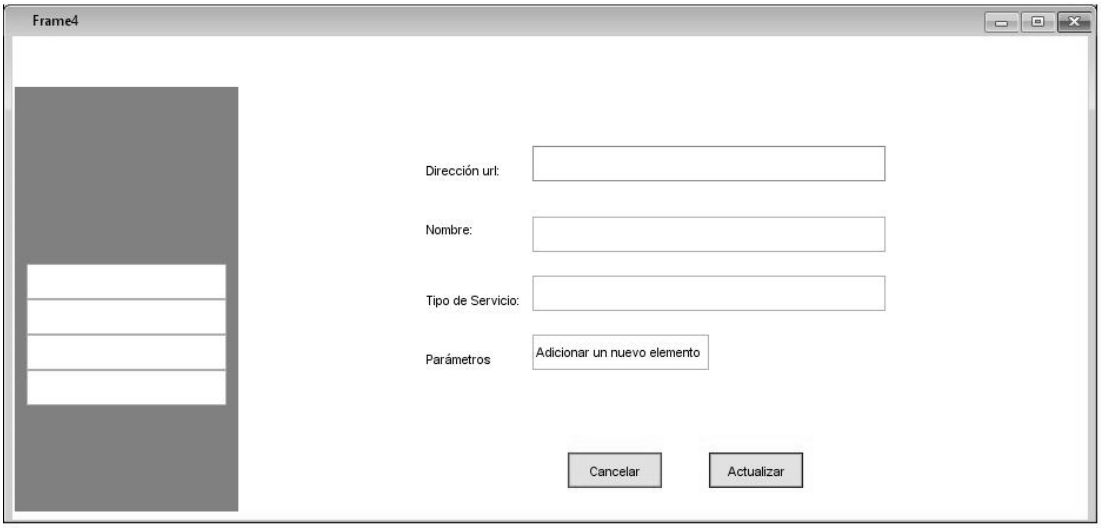
Llave: campo de carácter que representa el nombre del parámetro a modificar. Admite solo letras.

Valor: campo de carácter que representa el valor del parámetro a modificar. Admite letras y números y caracteres especiales.

4- Flujo de la acción a realizar:

El usuario puede seleccionar la opción de “Configuración de servicios de Gestión Académica” que se encuentra en el menú desplegable lateral izquierdo de la página. El usuario puede seleccionar cualquier tipo de configuración de servicio que desee añadir o modificar. En el formulario el usuario puede seleccionar la opción de Actualizar o Cancelar

Prototipo de interfaz:



2.5 Patrones de diseño

Un patrón de diseño (en programación orientada a objetos, POO) es una descripción de diversos objetos y clases preparados para resolver un problema de diseño general aplicado a un contexto específico. Identifica las instancias y clases que participan en dicho patrón además de sus papeles, sus relaciones y sus responsabilidades para llevar a cabo la tarea a resolver. Cada patrón de diseño se centra en resolver un problema particular en la POO. Describe cuando se puede aplicar, si puede ser aplicado desde el punto de vista de las

limitaciones del diseño y las consecuencias tanto positivas como negativas que tiene su utilización. (43)

Patrones de arquitectura.

Los patrones arquitectónicos se utilizan para expresar una estructura de organización base o esquema para un *software*. Proporcionando un conjunto de subsistemas predefinidos, especificando sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos. Heredan mucha de la terminología y conceptos de patrones de diseño, pero se centran en proporcionar modelos y métodos re-utilizables específicamente para la arquitectura general de los sistemas de información. Dentro de los patrones arquitectónicos podemos encontrar Inyección de dependencias, Arquitectura dirigida por eventos (Event-driven architecture o EDA), Arquitectura orientada a servicios o Modelo Vista Controlador. (44)

El patrón arquitectónico seleccionado es el Modelo Vista Controlador pues es el utilizado por el *framework* symfony. Además, este patrón separa los datos de una aplicación la interfaz de usuario y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista pueden ser hechas con un mínimo impacto en el componente del modelo de datos. (45)

Modelo: representa la información con la que trabaja la aplicación, o sea, su lógica de negocio.

Vista: convierte el modelo en una página web que facilita al usuario interactuar con ella.

Controlador: es el encargado de procesar las interacciones del usuario y ejecuta los cambios adecuados en el modelo o en la vista. El controlador es el encargado de aislar al modelo y a la vista de los detalles del protocolo usado para las peticiones (HTTP, consola de comandos, email, etc.).

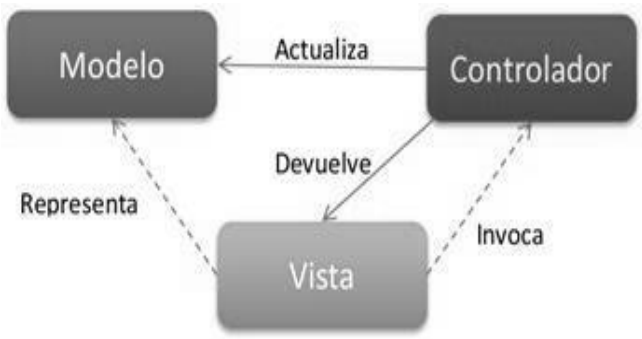


Figura 2. Modelo Vista Controlador.

Patrones GRASP

Se decide emplear para estructurar el diseño del sistema Patrones Generales de *Software* para Asignar Responsabilidades, más conocidos como patrones GRASP (General Responsibility Asignment Software Patterns, por sus siglas en inglés), los cuales serán muy útiles para lograr un *software* con alto grado de usabilidad.

Describe los principios fundamentales de la asignación de responsabilidades a objetos expresados en forma de patrones (46). Dentro de los patrones GRASP utilizados para el diseño del sistema y que incluye por defecto en su arquitectura el *framework* de desarrollo seleccionado se destacan los siguientes:

Experto: consiste en asignar una responsabilidad al experto en información, se asigna la responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Se evidencia en las clases que extienden de la clase Entidad, las cuales son expertas en su propia información, tales como Assistance, Awards, Courses, Data, Evaluation, Events, Point y Publications. (46)

Creador: permite crear objetos de una clase determinada. Es utilizado en la mayoría de las clases controladoras para crear instancias de formularios y entidades, para la vista del usuario, se evidencia en la clase DefaultController. (46)

Controlador: se basa en asignar la responsabilidad de todos los eventos realizados a una clase específica que constituye el único punto de entrada para cada evento. En la solución propuesta este patrón está evidenciado en las clases controladoras DefaultController. (46)

Alta Cohesión: en la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que

no realicen un trabajo enorme. En la solución propuesta este patrón está evidenciado en las clases que extienden de la clase Repository y Manager. (46)

Bajo Acoplamiento: el acoplamiento mide el grado en que una clase está conectada, tiene conocimiento o de alguna manera depende de otra. Este patrón consiste en asignar la responsabilidad de manera que el acoplamiento permanezca bajo. El bajo acoplamiento permite crear clases más independientes, más reutilizables, lo que implica mayor productividad. En la solución del componente se evidencia en las clases que extienden de Repository y Manager. (46)

Patrones GOF

Los patrones de diseño GOF se clasifican en tres grupos: creacionales, estructurales y de comportamiento y comprenden un total de 23 patrones (Gamma y otros, 1994). A continuación, se muestran algunos de los que se evidencian en la solución. (46)

Lazi Initialization: asegura, que de una clase solo existe una instancia y que esta ofrece un punto de acceso a ella. Donde mejor se refleja es en la creación de los servicios, ejemplo en el manager que crea un servicio al cual a través de este acceden al mismo. (46)

Patrón Decorator: el motor de plantillas Twig, está provisto de un mecanismo de herencia gracias al cual la decoración de plantillas se realiza de una manera flexible y versátil (Sensio Labs, 2012). Este patrón se observa en los ficheros “layout.html.twig” ubicados en el directorio app/Resources/views, que son los que contienen el código HTML que es común para todas las páginas, por lo que cada página que se crea heredará de estos. (46)

Patrón Inyección de Dependencias: se puede apreciar en el objeto especial entitymanager (em) a través del cual Doctrine2 realiza la manipulación de la información (buscar, insertar, modificar y eliminar registros en las tablas) sin que el programador tenga que escribir sentencias SQL. (46)

Composite: sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, debido que al poseer todos ellos una interfaz común, se tratan todos de la misma manera. Este se evidencia en las clases formulario. (46)

2.6 Modelo de diseño

Diagrama de clases del diseño

Los diagramas de clases del diseño son una representación más concreta y detallada que los diagramas de clases del análisis, aunque también representan la parte estática del sistema conteniendo las clases y sus relaciones. Son empleados para representar las relaciones que se establecen entre las clases. (47)

Se presenta a continuación el diagrama de clases del diseño de la historia de usuario Permitir configurar servicios web. Para el estudio de los demás diagramas de clase del diseño remitirse al Anexo 2.

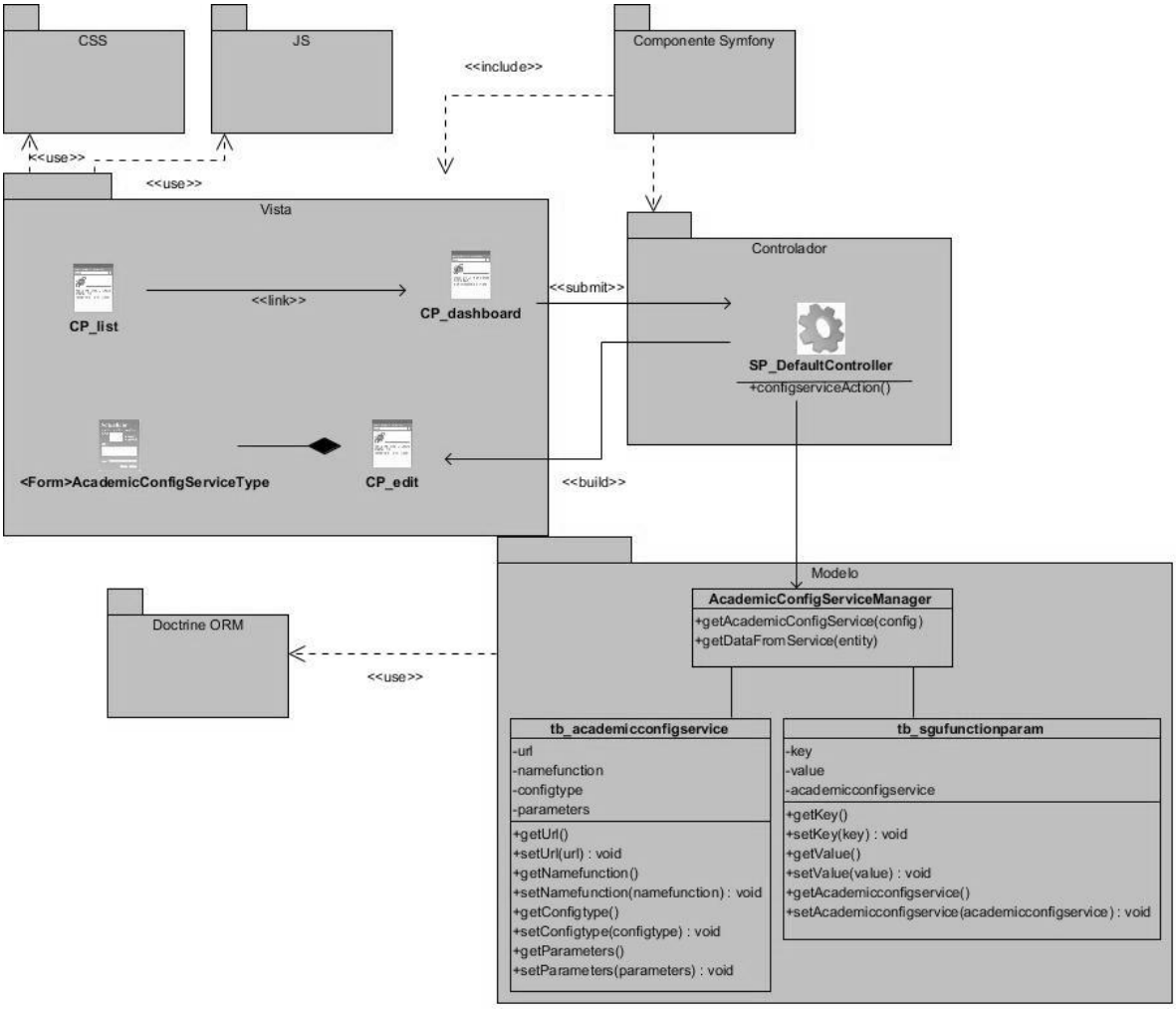


Figura 3. Diagrama de clases del diseño_HU_ Permitir configurar servicios web.

Diagrama de secuencia del diseño

Los modelos de secuencia son modelos dinámicos que documentan para cada nodo de interacción la secuencia de interacciones que tienen lugar entre los objetos. El tiempo se representa verticalmente, por lo que esta avanza hacia abajo sobre las líneas punteadas. Las interacciones entre los objetos se representan por flechas etiquetadas que vinculan a las líneas verticales. Los rectángulos delgados sobre la línea de vida del objeto representan el tiempo en el cual el objeto es el que tiene el control del sistema. (47)

Se presenta a continuación el diagrama de secuencia del diseño de la historia de usuario Permitir configurar servicios web. Para el estudio de los demás diagramas de secuencia del diseño remitirse al Anexo 3.

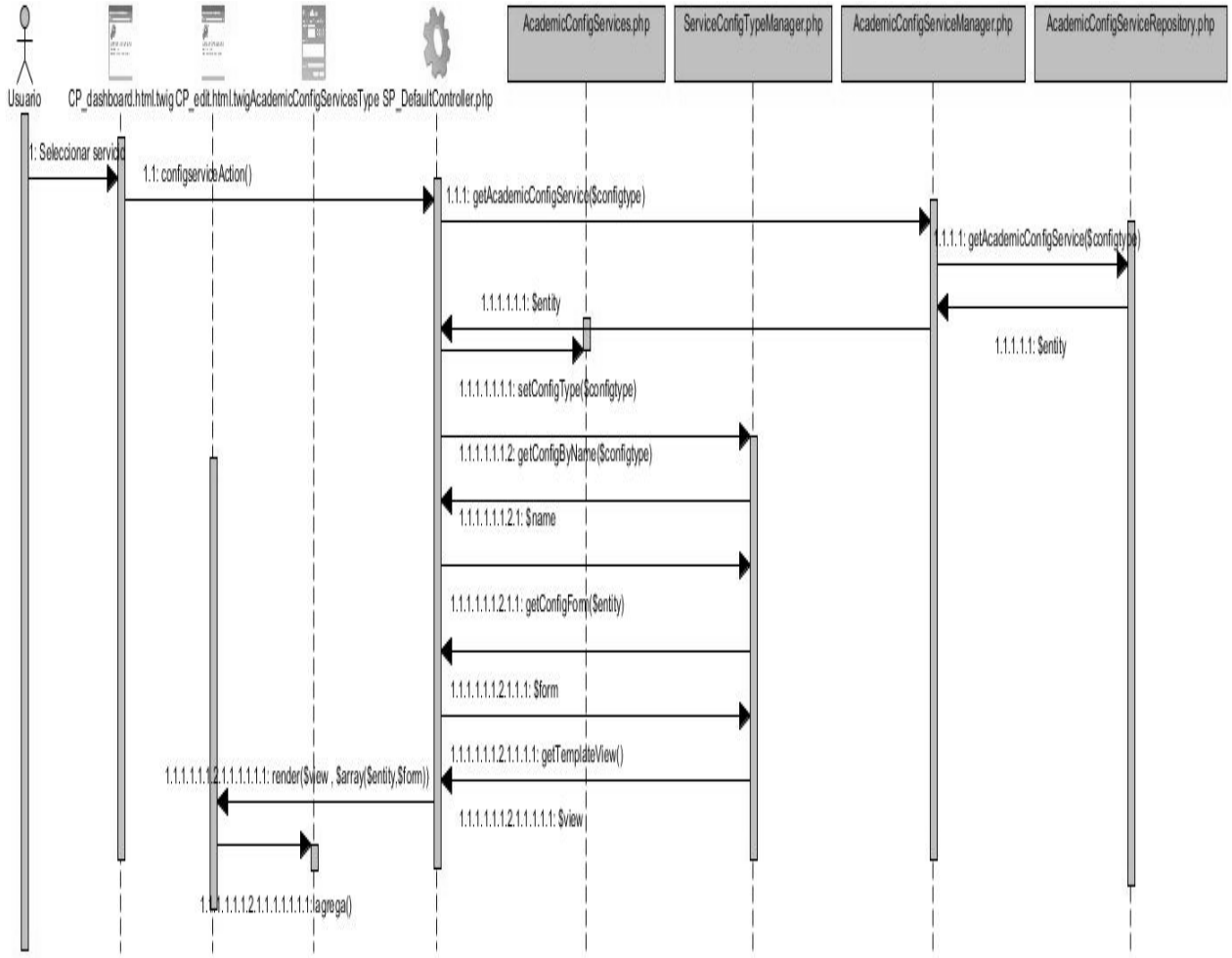


Figura 4. Diagrama de secuencia del diseño_HU_ Permitir configurar servicios web.

2.7 Diagrama de despliegue

El Diagrama de Despliegue es un tipo de diagrama del lenguaje unificado de modelado que se utiliza para modelar la disposición física de los artefactos *software* en nodos.

Los elementos usados por este tipo de diagrama son nodos (representados como un prisma), componentes (representados como una caja rectangular con dos protuberancias del lado izquierdo) y asociaciones. (48)

El diagrama de despliegue que se muestra a continuación representa la distribución física del sistema a través de nodos. Es decir, modela de manera detallada los nodos computacionales que intervienen en el funcionamiento del sistema, las conexiones que existen entre estos y los protocolos de comunicación que serán utilizados.

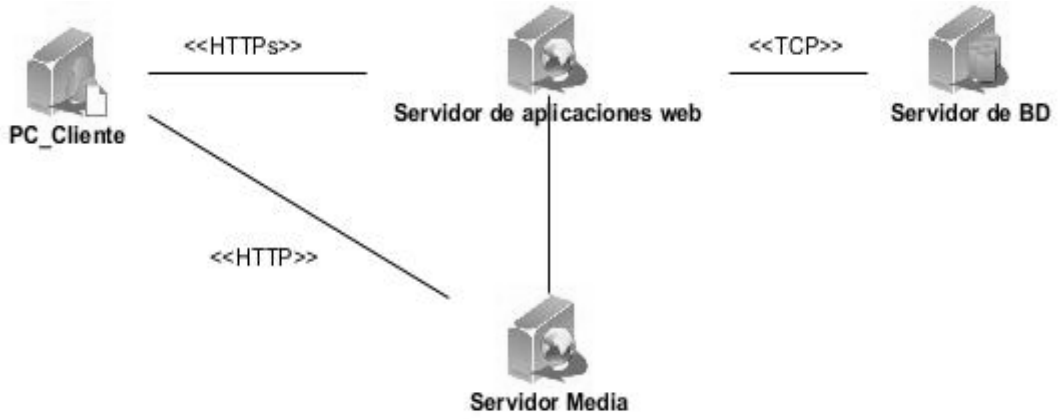


Figura 5. Diagrama de despliegue del sistema

2.8 Modelo de datos

Un modelo de datos es un conjunto de herramientas conceptuales para describir datos, sus relaciones, su significado y sus restricciones de consistencia. Es el proceso de analizar los aspectos de interés para una organización y la relación que tienen unos con otros. Tiene como meta registrar los requerimientos de datos de un proceso de negocio. Está compuesto por las entidades que conformarán las tablas de la base de datos que serán utilizadas por las funcionalidades a implementar. (49)

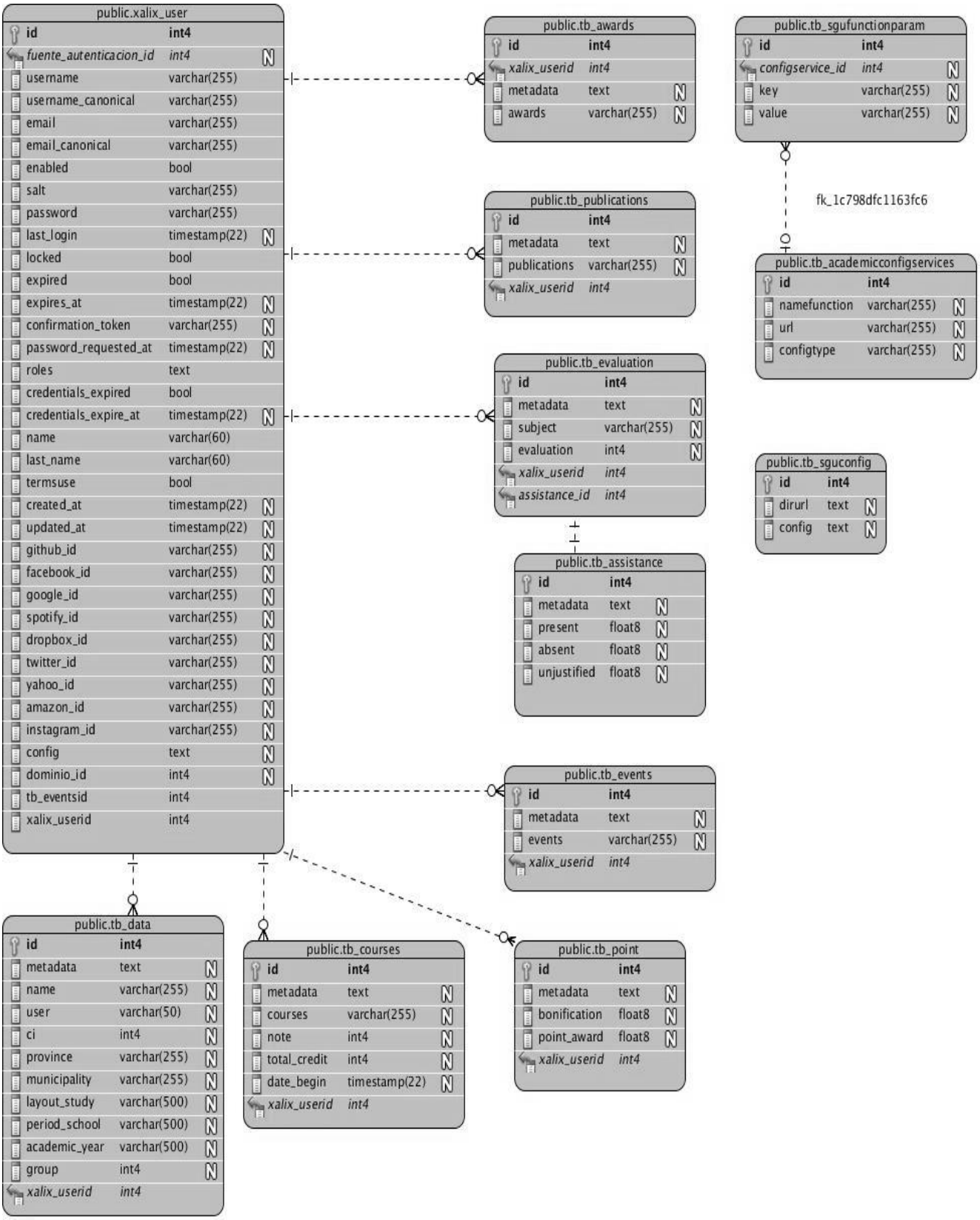


Figura 7. Modelo de datos del sistema

Descripción de las tablas de las bases de datos

A continuación, se muestra la descripción de la tabla public. tb_academicconfigservices

Tabla 3. Descripción de la tabla public. tb_academicconfigservices.

public.tb_academicconfigservices		
Descripción: en esta tabla se agrupa la información correspondiente a los servicios web .		
Atributo	**Tipo**	**Descripción**
Id	Integer	Etiqueta única que identifica el objeto en la tabla.
Namefunction	varchar(255)	Almacena el nombre del servicio web.
url	varchar (255)	Almacena la dirección del servicio web.
Configtype	varchar (255)	Almacena el tipo de configuración del servicio.

2.9 Conclusiones de capítulo

- ✓ La propuesta de solución sustentada por 8 requisitos funcionales y 10 no funcionales brindó un enfoque del componente a realizar.
- ✓ El desarrollo del modelo del diseño, permitió definir las bases necesarias para la implementación del componente y posibilitó que se obtuvieran de una forma concreta y detallada las relaciones que se establecen entre las clases.
- ✓ El esclarecimiento de la arquitectura y los patrones de diseños a utilizar, contribuyó a lograr una implementación centrada en el diseño realizado.

Capítulo 3: Implementación y prueba

En este capítulo se describen los elementos establecidos por el proceso de desarrollo empleado, que responden a la implementación de la solución. Los cuales se modelan en el diagrama de componentes. Además, se incluyen los resultados de las pruebas y las validaciones realizadas al componente con el propósito de validar si el sistema cumple con los requisitos establecidos.

3.1 Modelo de implementación

En el modelo de implementación son descritos cómo los elementos del modelo del diseño se implementan en términos de componentes, partiendo de los resultados obtenidos en el diseño. Se describe el diagrama de componentes que se presenta a continuación. (50)

Diagrama de componente

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas y mostrar las relaciones entre los elementos. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación especificado. Es otra forma de representar una vista estática del sistema, que representa la organización y dependencia entre los componentes físicos que se necesitan para ejecutar la aplicación, sean estos componentes de código fuente, librerías, binarios o ejecutables. A continuación, se muestran los componentes creados en la solución propuesta, a través del diagrama de componentes. (51)

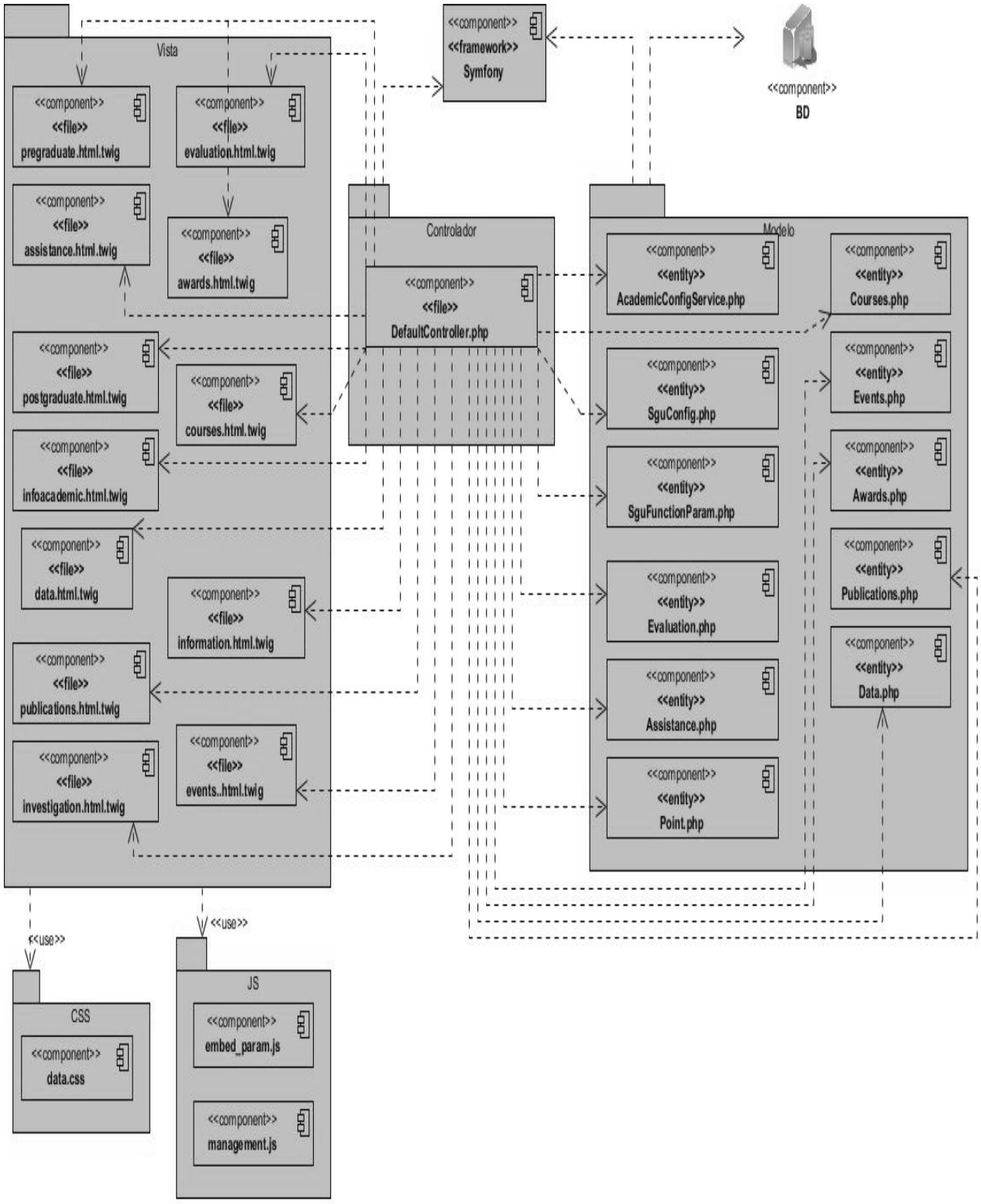


Figura 8. Diagrama de Componente

3.2 Pruebas de software

El desarrollo del *software* implica en sí, errores que pueden empezar a ocurrir desde el primer momento del proceso en el que los requerimientos pueden estar especificados de forma errónea, así como en los posteriores pasos del diseño e implementación. Es por esto, que se hace necesario contar con una actividad que garantice la calidad. Las pruebas es la actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados y una evaluación es hecha de algún aspecto del sistema o componente. (52)

Niveles de prueba

Los niveles de prueba son diferentes formas de verificar y validar un producto de *software*. A continuación, se distinguen los siguientes niveles de prueba empleados para la identificación de los errores. (53)

Pruebas unitarias: son las pruebas que se realizan a los elementos más pequeños del componente. Son aplicables a los componentes representados en el modelo de implementación para verificar que los flujos de control y de datos estén cubiertos y que funcionen como se espera. Constituyen la primera fase de las pruebas que se aplican a un sistema y su objetivo es verificar que estos componentes estén correctamente codificados. (53)

Pruebas de integración: son las pruebas que se realizan para asegurarse de que los componentes en el modelo de implementación funcionen correctamente cuando se combinan para ejecutar una funcionalidad. Estas pruebas descubren errores en las especificaciones de las interfaces de los paquetes. Verifican que las interfaces entre las entidades externas y las aplicaciones funcionan correctamente y que las especificaciones de diseño sean alcanzadas. Es el proceso de combinar y probar múltiples componentes juntos. (53)

Pruebas de sistema: son las pruebas que se realizan cuando el *software* está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de *software* y hardware han sido integrados. En un ciclo iterativo estas pruebas ocurren más temprano, tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados. El objetivo de esta prueba es asegurar

la apropiada navegación dentro del componente, ingreso de datos, procesamiento y recuperación. (53)

Métodos de prueba

Pruebas de caja blanca o estructural: la prueba de caja blanca, en ocasiones llamada prueba de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Estas comprueban los caminos lógicos del *software*, proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Requieren del conocimiento de la estructura interna del sistema y son derivadas a partir de las especificaciones internas de diseño o el código. (53)

Pruebas de caja negra o funcional: las pruebas de caja negra, también denominadas pruebas de comportamiento, se concentran en los requisitos funcionales del *software*. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Permiten encontrar funciones incorrectas o ausentes, errores de interfaz, rendimiento, inicialización y terminación, así como errores en estructuras de datos o en accesos a la base de datos externas. La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del *software* y son realizadas sin tener conocimiento interno del sistema. (53)

Dentro del método de caja negra se hace uso de la técnica Partición Equivalente por ser una de las más efectivas para examinar los valores válidos e inválidos de las entradas existentes en el sistema. La Partición Equivalente divide el dominio de entrada de un programa en clases de datos a partir de las cuales se derivan casos de prueba. (53)

Diseño de casos de prueba

Un caso de prueba específica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse. Comprueba que el sistema cumpla con las necesidades del cliente. (54)

A continuación, se presenta el diseño del caso de prueba perteneciente a la historia de usuario Permitir configurar servicios web, el resto de los artefactos de este tipo se encuentran en el Anexo 4.

Tabla 4. Diseño de casos de prueba de la DRP Permitir configurar servicios web.

CP Gestionar actividad								
Descripción general Posibilita al usuario con rol administrador añadir un servicio; al configurar la dirección, el nombre, el tipo de servicio y los parámetros de este. **Condiciones de ejecución** Estar autenticado en el sistema con rol (administrador). **SC1 Permitir configurar servicios web**								
Escenario	**Descripción**	**Url**	**Nombre**	**Configuración**	**Llave**	**Valor**	**Respuesta del sistema**	**Flujo central**
EC1.1 Seleccionar la opción “Configuración de servicios de Gestión Académica”	El usuario puede seleccionar la opción de “Configuración de servicios de Gestión Académica” que se encuentra en el menú desplegable lateral izquierdo de la página.						El sistema muestra en el panel del centro los distintos tipos de configuraciones de servicios.	Inicio/Administración/Configuraciones/Configuración de servicios de Gestión Académica
EC1.2 Seleccionar cualquier tipo de configuración	El usuario puede seleccionar cualquier tipo de configuración de servicio que desee añadir o modificar.						El sistema muestra un formulario con los campos a llenar por el usuario (*) url: (*) nombre: (*) configuración: parámetros:	Inicio/Administración/Configuraciones/Configuración de servicios de Gestión Académica/tipo de configuración elegida

							Este último contiene los campos de llave y valor. También se muestran dos opciones de Actualizar y Cancelar.	
EC1.3 Seleccionar opción Actualizar	Selecciona la opción de Actualizar.	V	V	V	N/A	N/A	El sistema valida los datos, los inserta en la Base Dato y regresa al EC 1.2 mostrando un mensaje de confirmación.	Inicio/Administración/Configuraciones/Configuración de servicios de Gestión Académica/tipo de configuración elegida/Actualizar
EC1.4 Seleccionar opción Cancelar	El usuario puede seleccionar la opción de Cancelar.	N/A	N/A	N/A	N/A	N/A	El sistema te envía hacia el EC 1.2 mostrando un mensaje de cancelación.	Inicio/Administración/Configuraciones/Configuración de servicios de Gestión Académica/tipo de configuración elegida/Cancelar
EC1.5 Datos incompletos	Existen datos incompletos.	I	V	V	N/A	N/A	Muestra un mensaje de información. Muestra un indicador sobre los campos vacíos.	Inicio/Administración/Configuraciones/Configuración de servicios de Gestión Académica/tipo de configuración elegida/Actualizar
		V	I	V	N/A	N/A		
		V	V	I	N/A	N/A		
EC1.6 Datos incorrectos.	Existen datos incorrectos.	I	V	V	N/A	N/A	Muestra un mensaje de información. Muestra un indicador sobre los campos incorrectos.	Inicio/Administración/Configuraciones/Configuración de servicios de Gestión Académica/tipo de configuración elegida/Actualiza
		V	I	V	N/A	N/A		
		V	V	I	N/A	N/A		
Descripción de las variables								

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Url	Campo de texto	No	Campo de carácter que representa la dirección url del servicio a añadir o modificar. Admite letras y números y caracteres especiales (/. :) .
2	Nombre	Campo de texto	No	Campo de carácter que representa el nombre del servicio a añadir o modificar. Admite solo letras.
3	Configuración	Campo de texto	No	Campo de carácter que representa la configuración del servicio a añadir o modificar. Admite letras y carácter especial (_) .
4	Llave	Campo de texto	Si	Campo de carácter que representa el nombre del parámetro a modificar. Admite solo letras.
5	Valor	Campo de texto	Si	Campo de carácter que representa el valor del parámetro a modificar. Admite letras y números y caracteres especiales.

Resultados obtenidos

Resultados de las pruebas unitarias

Durante el desarrollo del componente se llevaron a cabo pruebas unitarias para asegurarnos de que el código implementado de una funcionalidad específica cumple con el resultado esperado. Estas pruebas fueron realizadas por el desarrollador empleando el método de caja blanca, aprovechando las ventajas que brinda el framework de pruebas PHPUnit.

Resultados de las pruebas de integración

Durante la implementación de las funcionalidades se realizaron las pruebas de integración. Estas pruebas validaron la integración de la Plataforma Educativa ZERA 2.0 con el Sistema de Gestión Universitaria. Las realizaciones de las pruebas unitarias permitieron afirmar que las funcionalidades establecidas a integrar y demás elementos del sistema operaron satisfactoriamente, ajustándose a los requisitos especificados.

Resultados de las pruebas de sistema

Se empleó el método de caja negra sobre las interfaces gráficas del sistema. La aplicación de este método tiene como objetivo que el producto final cuente con el menor número de errores posibles. Es decir, se centra en el cumplimiento de las funcionalidades del componente que se obtiene.

Tabla 5. Tabla de no conformidades por casos de prueba.

Iteraciones	Cantidad de casos de prueba	No conformidades detectadas			
		Alta	Media	Baja	Total
1	8	10	5	10	25
2	8	1	2	7	10
3	8	0	0	0	0

Las no conformidades (NC) medias y bajas, se centraron en errores ortográficos como: omisiones de tildes y cambio de mayúscula por minúscula y mensajes sin traducir al inglés. Las altas consistieron en errores de validación, aceptar letras donde se esperaban valores numéricos. Al realizarse la primera iteración se identificaron 25 NC que fueron corregidas al término de esta, se procedió hacer una segunda iteración y fueron encontradas 10 NC que también fueron corregidas al final de la misma y para comprobar que el *software* quedó libre de errores se hizo una tercera iteración donde no se encontró ninguna NC.

A continuación, se muestran dos gráficos en el primero se puntualiza por iteraciones el total de no conformidades identificadas, las cuales fueron evaluadas en un rango (Alta, Media, Baja) y el segundo muestra por iteración la cantidad de pruebas unitarias, de sistemas y las NC identificadas entre las dos.

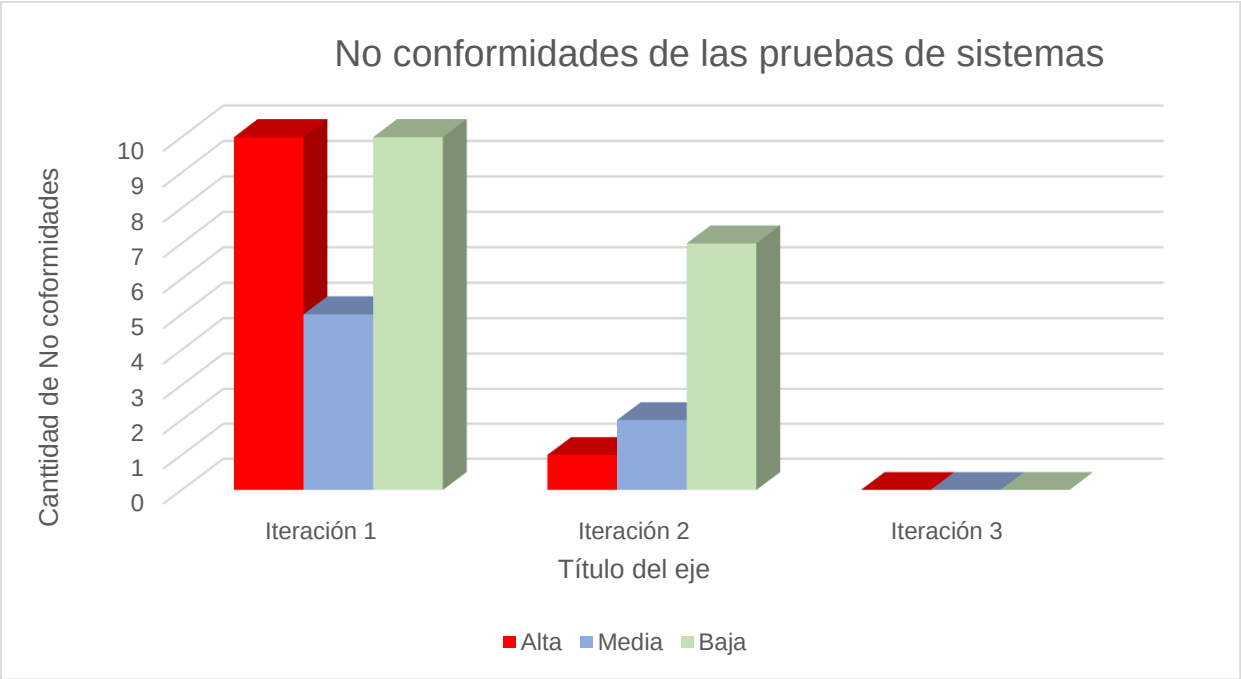


Gráfico 1. Resultados de las pruebas de sistema.

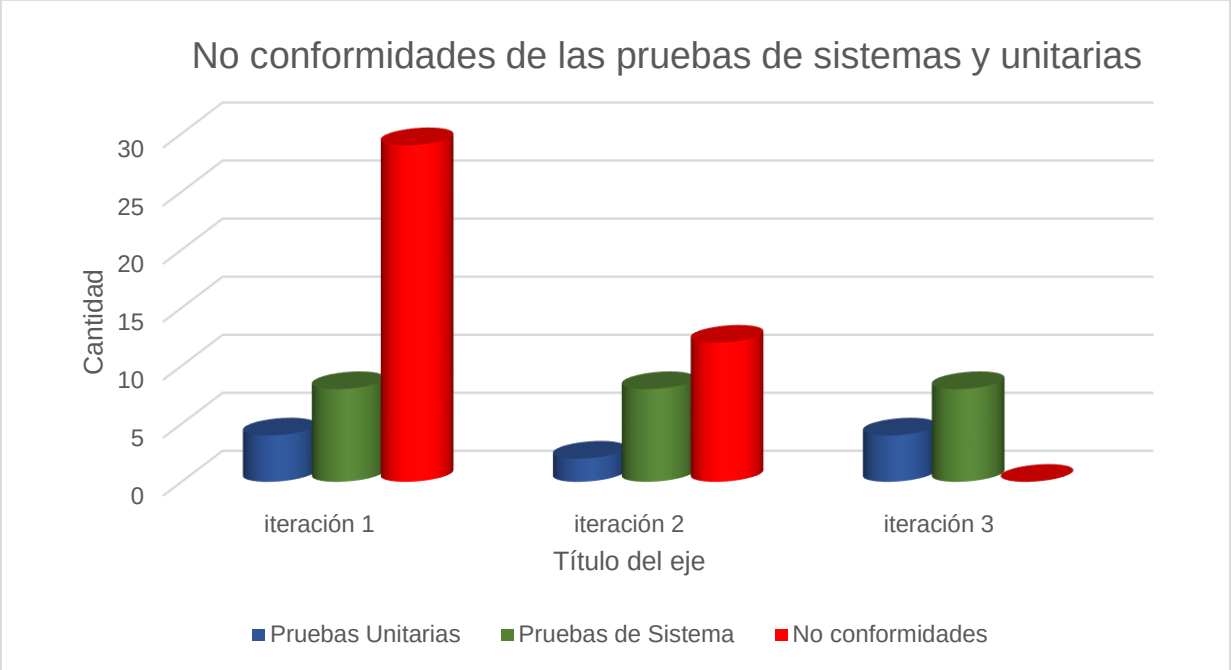


Gráfico 2. Resultados de las pruebas de sistema y pruebas unitarias.

3.3 Estándares de Codificación

Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible.

CSS

- ✓ Una declaración CSS siempre termina en punto y coma y los conjuntos de declaraciones se colocan entre llaves.
- ✓ Para hacer un código CSS legible, ponga una declaración en cada línea.
- ✓ Coloque la llave que cierra en una línea nueva.
- ✓ Para nombres compuestos de una propiedad utilice el guion (-).

En la figura 9 se muestra un ejemplo de este tipo reflejado en el código de la aplicación.

```
.portlet-title > .nav-tabs > li.active > a, .portlet-title > .nav-tabs > li:hover > a {
    margin: 0;
    background: none;
    color: #333;
}
```

Figura 9. Código CSS.

HTML

- ✓ No coloque espacios entre la relación atributos-valor.
- ✓ Utilice *lowercase* para el nombre de los atributos de las etiquetas.

En la figura 10 se muestra un ejemplo de este tipo reflejado en el código de la aplicación.

```
{% trans_default_domain "AcademicManagementBundle" %}
<link href="{{asset("bundles/academicmanagement/css/data.css")}}" rel="stylesheet">
<div class="profile-content" >
    <div class="row">
        <div class="col-md-12">
```

Figura 10. Código HTML.

JS

- ✓ Utilice la notación *camelCase* para los nombres de variables y funciones.
- ✓ Todos los nombres comienzan con una letra.
- ✓ Siempre utilice la palabra reservada "var" para declarar variables.

En la figura 11 se muestra un ejemplo de este tipo reflejado en el código de la aplicación.

```
var prototype = $collectionHolder.data('prototype');
var index = $collectionHolder.data('index');
var newForm = prototype.replace(/__name__/g, index);
```

Figura 11. Código JS.

PHP

- ✓ Declare los atributos de las clases antes de los métodos.
- ✓ Utilice la notación *camelCase* para los nombres de variables, funciones, métodos y argumentos.
- ✓ Prefije las clases abstractas con *Abstract*.
- ✓ Sufije las interfaces con Interface.
- ✓ Los comentarios utilizan las anotaciones siguientes:

@param esta etiqueta provee el nombre, el tipo y la descripción de los parámetros de una función

@var esta etiqueta define qué tipo de dato se representa mediante la propiedad

@return esta etiqueta es utilizada para documentar el valor que retorna una función

En la figura 12 y 13 se muestra un ejemplo de este tipo reflejado en el código de la aplicación.

```php
class AcademicConfigServiceType extends AbstractType
{
    private $configClass;

    /**
     * Builder the objects.
     */
    public function __construct()
    {
        $this->configClass = 'FORTES\AcademicManagementBundle\Entity\AcademicConfigServices';
    }
```

Figura 12. Código PHP.

```php
    /**
     * Build the form for this type.
     *
     * @param FormBuilderInterface $builder Builder.
     * @param array                $options Options.
     *
     * @return mixed
     */
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
```

Figura 13. Código PHP.

3.4 Conclusiones del capítulo

- ✓ El modelo de implementación, conformado por el diagrama de componente permitió implementar y organizar los elementos del modelo del diseño.
- ✓ Las pruebas unitarias, de integración y de sistemas validaron que el componente se encontraba libres errores, arrojando 41 no conformidades en total, que fueron corregidas en tres iteraciones.
- ✓ Los estándares definidos por el proyecto contribuyeron a organizar la implementación del componente.

Conclusiones Generales

Con la culminación del presente trabajo de diploma se obtuvieron las siguientes conclusiones:

- ✓ Los referentes teóricos y metodológicos relacionados a la integración entre sistemas informáticos no aportaron a la solución del componente, pues no se adaptan a las necesidades del proyecto y estos sistemas similares no poseen código abierto para consultar información sobre sus funcionalidades, por lo que se ratifica la implementación del componente.
- ✓ La metodología AUPUCI, el servidor de bases de datos PostgreSQL, el entorno integrado de desarrollo Netbeans, el servidor de aplicación Nginx, el *framework* Symfony, entre otras herramientas y tecnologías utilizadas apoyaron la implementación de un componente que permite la integración entre la Plataforma Educativa ZERA 2.0 y el Sistema de Gestión Universitaria.
- ✓ Para un enfoque del resultado final, se realizó una propuesta de solución sustentada por los 8 requisitos funcionales y 10 no funcionales, se definió la arquitectura y los patrones de diseño a utilizar, lo cual contribuyó a lograr una implementación centrada en el diseño realizado.
- ✓ La implementación del componente brindó el acceso a la información relevante de los procesos docentes de pregrado, como la asistencia y las notas; de postgrado, como los cursos realizados y de investigación, como los premios, los eventos y las publicaciones en el Sistema de Gestión Universitaria desde la Plataforma Educativa ZERA 2.0.
- ✓ Se realizaron 10 pruebas unitarias y 24 de sistemas, detectándose 41 no conformidades en tres iteraciones, que fueron corregidas al final de cada una, obteniéndose un producto final libre de errores.

Recomendaciones

Para la continuidad de la presente investigación se recomienda incorporar varias funcionalidades en el componente desarrollado como:

- ✓ Permitir que el usuario en la interfaz visual tenga la posibilidad de sincronizar manualmente la información que se le muestra, sin la necesidad de esperar que el sistema actualice dicha información.
- ✓ Posibilitar que un usuario con rol administrador tenga la opción manualmente de variar el tiempo de días en que la plataforma actualiza la información que envía el Sistema de Gestión Universitaria.

Referencias Bibliográficas

1. Tom Dwyer. «SINTITUL-15 - 6118». Accedido 13 de mayo de 2017. http://revistasinvestigacion.unmsm.edu.pe/index.php/sociales/article/viewFile/6906/6118.
2. ERLA MARIELA MORALES MORGADO. Gestión del conocimiento en sistemas «e-learning», basado en objetos de ... - Erla Mariela Morales Morgado - Google Libros. [online]. [Accessed 13 May 2017].
3. MOYA, José Manuel Huidobro; MARTÍNEZ, David Roldán. *Seguridad en redes y sistemas informáticos*. Thomson Paraninfo, 2005.
4. ALONSO, M. Covadonga López; SERÉ, Arlette. Entornos formativos en el ciberespacio: las plataformas educativas. *Español actual: Revista de español vivo*, 2004, no 82, p. 9-20.
5. Las plataformas e-learning para la enseñanza y el aprendizaje universitario en Internet - E-Prints Complutense. [online]. [Accessed 13 May 2017]. Available from: http://eprints.ucm.es/10682/
6. VIDAL LEDO, María; GÓMEZ MARTÍNEZ, Freddy; RUIZ PIEDRA, Alina María. Hiperentornos educativos. *Educación Médica Superior*, 2011, vol. 25, no 1, p. 123-131.
7. Interoperabilidad de sistemas de organización del conocimiento: el estado del arte. [online]. [Accessed 13 May 2017]. Available from: http://www.scielo.org.ar/scielo.php?script=sci_arttext&pid=S1851-17402011000100002
8. RIVAS, Francisco Cosano. La plataforma de aprendizaje moodle como instrumento para el trabajo social en el contexto del espacio europeo de la educación superior. *Acciones e investigaciones sociales*, 2006, no 1, p. 367.
9. GRANERO-GALLEGOS, Antonio; BAENA-EXTREMERA, Antonio. Diseños de aprendizaje basados en las TIC (Moodle 2.0 y Mahara) para contenidos de Anatomía, Fisiología y Salud en las clases de Educación Física escolar. *International Journal of Morphology*, 2015, vol. 33, no 1, p. 375-381.
10. Sistema de Gestión para Aprendizaje Corporativo | Integracion HRIS. [online]. [Accessed 13 May 2017]. Available from: https://www.paradisosolutions.com/es/soluciones/integracion-hris-lms

11. DÍAZ, J., et al. Integración de plataformas virtuales de aprendizaje, redes sociales y sistemas académicos basados en Software Libre. Una experiencia en la Facultad de Informática de la UNLP. *Universidad Nacional de la Plata*, 2012.
12. Arletys González Madera y Reysel Pérez Avilés. *Componente para la integración del Juez en Línea Caribeño y la Plataforma Educativa ZERA 2.0*. Universidad de las Ciencias Informáticas, [no date].
13. CANÓS, José H.; LETELIER, Patricio; PENADÉS, Mª Carmen. Metodologías ágiles en el desarrollo de software. *Metodologías Ágiles en el Desarrollo de Software*, 2003, vol. 1, no 10, p. 1-8.
14. CARRILLO, A. Metodología RUP de Ingeniería del Software. *Disponible desde: http://es. scribd. com/doc/59141033/Carrillo-Anay-Metodologia-Rup-de-Ingenieria-Del-Software*, 2011.
15. UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS and CARRETERA A SAN ANTONIO KM 2 1⁄2. TORRENS. BOYEROS. LA HABANA. CUBA. MetodologíaUCI2015.pdf.
16. SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Fundamentos de bases de datos. 2002.
17. BOICEA, Alexandru; RADULESCU, Florin; AGAPIN, Laura Ioana. MongoDB vs Oracle-Database Comparison. En *EIDWT*. 2012. p. 330-335.
18. SUEHRING, Steve. *MySQL bible*. John Wiley & Sons, Inc., 2002.
19. AGRAWAL, Sanjay, et al. Database tuning advisor for microsoft SQL server 2005: demo. En *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005. p. 930-932.
20. DOUGLAS, Korry; DOUGLAS, Susan. *PostgreSQL: a comprehensive guide to building, programming, and administering PostgresSQL databases*. SAMS publishing, 2003.
21. TORRES REMÓN, Manuel. FUNDAMENTOS DE PROGRAMACIÓN g LENGUAJES Y TÉCNICAS DE PROGRAMACIÓN. 2012.
22. MORA, Sergio Luján. *Programación de aplicaciones web: historia, principios básicos y clientes web*. Editorial Club Universitario, 2002.
23. RAMOS GRISALES, Evin Alexis. Diseño Web. 2012.

24. MANGER, Jason J.; SOTO, Saúl Flores; CABALLERO, José Francisco Becerril. *Fundamentos de JavaScript*. McGraw-Hill, 1997.
25. GAUCHAT, Juan Diego. *El gran libro de HTML5, CSS3 y Javascript*. Marcombo, 2012.
26. COBO, Ángel. *PHP y MySQL: Tecnología para el desarrollo de aplicaciones web*. Ediciones Díaz de Santos, 2005.
27. GARCÍA, Esteban Trigos. *PHP 4*. Anaya Multimedia-Anaya Interactiva, 2000.
28. TUPE, C.; CISNEROS, J. Evaluación y Selección de Framework de Desarrollo PHP: Symfony. *Kumbia, CakePHP y Zend*, 2008.
29. POTENCIER, Fabien; ZANINOTTO, François. symfony. *Potencier. org [viitattu 05.12. 2015]. Saatavissa: http://fabien. potencier. org/article/65/why-symfony*, 2012.
30. EFRON, Bradley; TIBSHIRANI, Robert J. *An introduction to the bootstrap*. CRC press, 1994.
31. ALVAREZ, Miguel Angel. Manual de jQuery. *Recuperado el*, 2010, vol. 17.
32. SARRION, Eric. *JQuery UI*. " O'Reilly Media, Inc.", 2012.
33. JOHN VERZANI. Getting Started with RStudio: An Integrated Development Environment for R - John Verzani - Google Libros. [online]. [Accessed 16 May 2017]. Available from: https://books.google.com.cu/books?id=KmC-o32qFpQC&printsec=frontcover&dq=Integrated+Development+Environment&hl=es&sa=X&redir_esc=y#v=onepage&q=Integrated%20Development%20Environment&f=false
34. DE RED, PUERTO; FLASH, MEMORIA. Transmission control protocol. 1981.
35. CHI, Xiaoni, et al. Web load balance and cache optimization design based nginx under high-concurrency environment. En *Digital Manufacturing and Automation (ICDMA), 2012 Third International Conference on*. IEEE, 2012. p. 1029-1032.
36. BOOCH, Grady, et al. *El lenguaje unificado de modelado*. Addison-Wesley, 1999.
37. CAULDWELL, Patrick; CHAWLA, Rajesh; CHOPRA, Vivek. *Servicios web XML*. 2002.

38. QUINTERO, Juan Bernardo, et al. Un estudio comparativo de herramientas para el modelado con UML. *revista universidad eafit*, 2012, vol. 41, no 137, p. 60-76.
39. PARADIGM, Visual. Visual paradigm for uml. *Visual Paradigm for UML-UML tool for software application development*, 2013, p. 72.
40. PREFACIO, X. V. Parte I: El Proceso Unificado de Desarrollo de Software. 2000.
41. COYLE, Karen, et al. Guidelines for Dublin Core application profiles. 2009.
42. JOSÉ RUBÉN LAÍNEZ FUENTES. Desarrollo de Software Ágil: Extremme Programming y Scrum. 2ª Edición - José Rubén Laínez Fuentes - Google Libros. [online]. [Accessed 16 May 2017]. Available from: https://books.google.com.cu/books?id=TxRpCwAAQBAJ&pg=PA49&dq=Historias+de+usuario&hl=es&sa=X&redir_esc=y#v=onepage&q=Historias%20de%20usuario&f=false
43. DÍAZ, Mª Paloma; MONTERO, Susana; AEDO, Ignacio. Ingeniería de la web y patrones de diseño. *Pearson. Prentice Hall*, 2005.
44. CHRISTOPHER ALEXANDER, SARA ISHIKAWA, MURRAY SILVERSTEIN - GOOGLE LIBROS. El Lenguaje de Patrones - Christopher Alexander, Sara Ishikawa, Murray Silverstein - Google Libros. [online]. [Accessed 16 May 2017]. Available from: https://books.google.com.cu/books?id=WHArAAAACAAJ&dq=Patrones+de+Arquitectura.&hl=es&sa=X&redir_esc=y
45. GONZÁLEZ, Yanette Díaz; ROMERO, Yenisleidy Fernández. Patrón Modelo-Vista-Controlador. *Revista Telem@ tica*, 2012, vol. 11, no 1, p. 47-57.
46. LARMAN, C. GRASP: Más patrones para asignar responsabilidades. *L. Hernández MONTENEGRO, RODRÍGUEZ Y SALAZAR: Uso de patrones de diseño de software*, 2004, vol. 59.
47. BOOCH, Grady, et al. *El lenguaje unificado de modelado*. Addison-Wesley, 1999.
48. CORNELIO, Omar Mar, et al. Aplicación informática para el control energético de la tecnología utilizando herramienta de monitoreo de red Nmap Application for the power control technology using network monitoring tool Nmap. *Revista Cubana de Ciencias Informáticas (RCCI)*, 2012, vol. 6, no 2.

49. NAVATHE, SHAMKANT B.; ELMASRI, R. Fundamentos de sistemas de bases de datos. *Addison Wesley*, 2002.
50. RIVERA, L. Justificación conceptual de un modelo de implementación de Lean Manufacturing. *Heurística*, 2008, vol. 15, p. 91-106.
51. LÓPEZ, Ramón Ortigosa; RAMIRO, Luis Vázquez. *Ingeniería del software*. Centro de Estudios Financieros, 2011.
52. Pressman, Roger S. Ingeniería del Software: Un Enfoque Práctico.S.I.: McGraw-Hill 2005.
53. SOMMERVILLE, Ian. Ingeniería del software. 7ma. Madrid: Pearson Educación, 2005.