UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Portafolio de Desarrollo Personal para la centralización de las evidencias de trabajo v2.0

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

Autores: *Yenisleidy Herrera Hernández*

Denny Villar Luciano

Tutores: *Ing. Mairelis Gari Maribona*

Ing. Eduardo Alejandro López Urquiaga

Junio - 2015

La Habana, Cuba

Declaración de autoría

Declaramos ser los únicos autores de este trabajo y concedemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año 2015.

_____ _____

Yenisleidy Herrera Hernández Denny Villar Luciano

_____ _____

Ing. Mairelis Gari Maribona Ing. Eduardo A. López Urquiaga

Dedicatoria

De Yenisleidy Herrera Hernández:

Le dedico este logro a toda mi familia y a los que creyeron en mí, en especial a mi mamá, mi papá y mis abuelos porque este es también parte de su sueño.

De Denny Villar Luciano:

A mis dos madres, Lesbia y María, y a mis dos padres Daniel y Roly.

A mi hermana Dianys.

A mis tíos Nancy, Reidel, Amado y Carlos.

A mi prima Mairelis.

A mis abuelas Clara y Lidia.

A mi novia y compañera Yeny y a su familia.

A Yudith y Juancy en España.

A mi piquete de gente querida de amigos en esta universidad.

A mis tutores.

A mi equipo de Beisbol de la Facultad.

A mis profesores desde la primaria hasta el día de hoy, principalmente a Caridad, Margarita, María Victoria, Nuria y Omaida.

Agradecimientos

De Yenisleidy Herrera Hernández:

A mi mamá y mi papá por todo su apoyo, dedicación y preocupación a lo largo de mi vida y especialmente en estos últimos cinco años.

A mima y a pipo por su preocupación cada fin de semana y su apoyo para inspirarme a continuar.

A mi hermano por los momentos divertidos y de pelea que hemos tenido, y por tenerme como ejemplo a seguir.

A mis primos Betty, Alejandro, Dayanna, Abraham, Alizarín y la nueva adquisición de la familia Hany con quienes compartí alegrías y tristezas. En especial quiero destacar a José Carlos que aunque no es primo mío así me lo hizo sentir y brindó su apoyo en primer año.

A mis tutores Mairelis y Eduardo por habernos guiado y ayudado en esta difícil pero no imposible tarea.

A mi novio y compañero de tesis porque a pesar de las preocupaciones y las dificultades se mantuvo a mi lado brindándome su amor incondicional.

A todos los profesores que durante cinco cursos me han enseñado con mucha dedicación lo que necesitaba saber no sólo para mi carrera, sino también para la vida.

Al resto de mi familia de la Habana a tía Cacha, mima y Danay, de Pinar del Río en especial a tía Odalis, Maniloto y Manolo, Miriam y el galleguito y de Villa Clara a Lesbia, Dianys, Roly, Daniel y María que me aceptaron con mucho cariño.

A todos mis compañeros de grupo, de apartamento y a los amigos y amigas de la UCI con los que he compartido momentos felices y estresantes, mementos de estudio, de deporte y sobre todo de mucha diversión.

De Denny Villar Luciano:

A mi mamá Lesbia y a mi padrastro Roly por su apoyo siempre y porque son los principales causales de que hoy se esté formando un hombre de bien, preparado para la vida y la sociedad.

A mi papá Daniel y a mi madrasta María por su apoyo y por enseñarme a ser una mejor persona brindándoles ayuda incondicional a los que me rodean.

A mis tíos Nancy, Reidel, Amado y Carlos y mi prima Mairelis por estar siempre presente cuando se les necesita.

A mi hermana Dianys porque con sus locuras ha hecho de mi vida y mi hogar algo muy confortante y alegre.

A mi novia y compañera de tesis Yeny por soportarme y ser el acoplamiento necesario en estos 5 años para poder salir adelante en los momentos más tensos. Por enseñarme a tomar las cosas con calma y a reírme de las situaciones, siempre que estas lo permitan.

A mis tutores porque sin su ayuda hubiera sido imposible convertirme en Ingeniero.

A mis compañeros y amigos en estos 5 años, porque con ustedes he compartido una etapa de mi vida que nunca voy a olvidar y siempre ocuparan un lugar muy importante en mi corazón y en las historias a contar a mi hijos.

Resumen

Los portafolios digitales son herramientas que facilitan el almacenamiento de archivos utilizados como evidencias de trabajo a lo largo de la vida. En los últimos tiempos han adquirido mayor relevancia y numerables ventajas. Se caracterizan por ser fáciles y económicos de distribuir. Pueden ser compartidos de amplias maneras llegando incluso a cualquier persona en lugares muy alejados. Admiten todo tipo de archivos y su elaboración es permanente aunque se puede editar en cualquier momento. La presente investigación tiene como objetivo desarrollar nuevas funcionalidades que permitan al portafolio creado en el Centro de Tecnologías para la Formación de la Universidad de Ciencias Informáticas mayor interoperabilidad, retroalimentación entre los usuarios y personalización del espacio de trabajo. Antes de comenzar con el desarrollo se realizó un estudio de sistemas de igual propósito determinar las características y rasgos fundamentales, así como estándares y tecnologías más utilizadas. La investigación fue guiada por la metodología AUP que permitió generar artefactos que sirvieron de entrada en el proceso de desarrollo y una vez concluido se aplicaron pruebas de software a la propuesta de solución para garantizar el correcto funcionamiento del mismo y validar que cumpla con los requisitos planteados por el cliente. Finalmente, se obtuvo un software más interoperable capaz de importar portafolio y exportar archivos, evidencias y secciones. Mediante el envío de mensajes se obtuvo mayor retroalimentación entre los usuarios y la personalización del entorno de trabajo quedó garantizada brindando la posibilidad al usuario de seleccionar entre varios colores el de su preferencia.

Índice

Índice de tablas

Índice de ilustraciones

Introducción

Con la era de Internet el mundo educativo ha evolucionado y los profesionales de la educación tienen múltiples razones para aprovechar al máximo las nuevas Tecnologías de la Información y las Comunicaciones (TIC). Para impulsar la incorporación de las TIC al proceso de enseñanza - aprendizaje se han creado herramientas (…) para la gestión de contenidos académicos, permitiendo el seguimiento y la evaluación de los mismos. (Maribona et al., 2013)

Los portafolios electrónicos han surgido como una valiosa herramienta en línea que los estudiantes y las instituciones pueden utilizar para recolectar, almacenar, actualizar y compartir información. Permiten que los estudiantes reflejen su aprendizaje, se comuniquen con sus instructores, tengan información escolar y muestren ejemplos de su trabajo a empleadores potenciales. (Maribona et al., 2013)

Un portafolio digital es una selección deliberada del alumno o del docente que busca dar a conocer los esfuerzos, progresos y estrategias que sigue para lograr determinados objetivos. La selección de trabajos que constituyen el portafolio se realiza de manera sistemática al constituir una secuencia cronológica que permite observar la evolución de conocimientos, habilidades y actitudes del alumno o del docente en una o más asignaturas. Los trabajos contenidos en el portafolio están acompañados de una narrativa reflexiva, la misma que posibilita la comprensión del proceso de aprendizaje de éste en la construcción de conceptos, habilidades y actitudes. Dicha comprensión se propicia en el alumno, pero también en el propio docente. (Hernández, 2014)

La información de los portafolios digitales también puede aparecer en una variedad de medios, como textos, fotografías, ilustraciones, diagramas, material de la web, archivos de audio, hojas de cálculo y presentaciones de PowerPoint. (Sulbarán, 2010)

Los portafolios digitales son una herramienta utilizada por el docente para valorar las competencias que va consiguiendo el estudiante en el proceso de enseñanza-aprendizaje. En este sentido deben conseguir fomentar en el alumno, según Gavari (Sánchez and Gámiz, 2011), la capacidad de incorporar las nuevas

tecnologías de información y comunicación al proceso de aprendizaje; facilitar el aprendizaje; despertar interés, la motivación y el gusto por aprender de forma autónoma y a través de la interacción con otros; y fomentar las aptitudes para la comunicación interpersonal y el trabajo en equipo.

En el Centro de Tecnologías para la Formación (FORTES) de la Universidad de las Ciencias Informáticas (UCI) se desarrolló un portafolio totalmente independiente y que puede integrarse con Plataformas educativas, permitiendo la interoperabilidad de los datos almacenados y contribuyendo además al desarrollo de habilidades para la reflexión y autoevaluación, como parte de una tesis de grado titulada Portafolio de Desarrollo Personal para la Centralización de Evidencias de Trabajo desarrollada en el curso 2013-2014. Este portafolio ayuda a la recogida y selección de la información, facilita también las prácticas de colaboración e intercambio de experiencias y el cultivo de las creencias y conocimientos sobre la profesión docente y la práctica, propiciando el intercambio de ideas en una comunidad de aprendizaje.

Las personas individuales pueden recopilar, gestionar y mostrar la información para apoyar su aprendizaje, el desarrollo, la planificación y la acción en cualquier momento y en cualquier área de la vida. La información recopilada puede ser útil en otros contextos, y varias herramientas o servicios podrían ser utilizados para ayudar a estos procesos. Para ello este portafolio permite exportar archivos y lo hace mediante el formato Leap2A. (Leap2A, 2015)

Este estándar ofrece un modelo para representar la información y los recursos creados, controlados o recogidos por las personas, descripciones de lo que han logrado, creado, hecho o experimentado; información sobre sí mismos, sus habilidades y cualidades; cualquier tipo de información de apoyo o pruebas de cualquier fuente, sus pensamientos o reflexiones sobre lo pasado, presente o futuro, sus planes y las presentaciones de esta información a otras personas. (Maribona et al., 2013)

Hasta el momento el software existente permite sólo exportar el portafolio completo, presentando como deficiencias la posibilidad de importar paquetes en formato Leap2A o exportar recursos seleccionados por el usuario, impidiendo mayor usabilidad y versatilidad del software. Entre las opciones de personalización de los espacios de trabajo no permite hacer varios temas y cambiarlos, ya sea el color de la barra de navegación, los paneles, botones, enlaces, entre otros, impidiendo que el usuario se sienta identificado con el mismo. El portafolio aún no incorpora retroalimentación entre los usuarios porque hasta el momento

la comunicación entre ellos está limitada a las evidencias compartidas, o sea, los usuarios no pueden intercambiar información, criterios y consejos fuera de las evidencias. No se permite compartir portafolios por secciones. Los usuarios no cuentan con la posibilidad de escoger el tipo de notificación acerca del tema que desean por lo que reciben las notificaciones de todos los eventos existentes.

Lo anteriormente planteado permite definir como **problema investigativo** ¿Cómo lograr en el portafolio digital mayor interoperabilidad, retroalimentación entre los usuarios y personalización del espacio de trabajo?

El **objeto de estudio** se enfoca en los portafolios digitales, por lo que el **campo de acción** se sitúa en los procesos de interoperabilidad, retroalimentación y personalización del espacio de trabajo del portafolio digital.

Se tiene como **objetivo general:** Desarrollar funcionalidades que permitan mayor interoperabilidad, retroalimentación entre los usuarios y personalización del espacio de trabajo.

El objetivo general se puede desglosar en los siguientes **objetivos específicos:**

1. Investigar las especificaciones asociadas al desarrollo de portafolios digitales para conformar el marco teórico conceptual de la investigación.
2. Implementar las nuevas funcionalidades que permitan mayor eficiencia y comodidad en el trabajo con el portafolio digital.
3. Comprobar el funcionamiento del sistema, en su nueva versión, mediante pruebas de software.

Para guiar la investigación se plantea la siguiente **hipótesis**: Con una mayor interoperabilidad, retroalimentación entre los usuarios y personalización del espacio de trabajo, se contribuirá al perfeccionamiento del portafolio digital existente.

Los **métodos de investigación científica** a utilizar son:

Métodos Teóricos:

- **Modelación:** Se realizará una modelación simplificada de la realidad permitiendo establecer esquemas, modelar ideas de forma gráfica, además de posibilitar la utilización de diagramas para expresar información de manera organizada.
- **Análisis histórico-lógico:** Se realizará la investigación acerca del desarrollo y evolución de los portafolios digitales en aplicaciones anteriormente realizadas por otros autores, se analizarán los conceptos y estándares a utilizar, lo que ofrecerá la posibilidad de seleccionar las herramientas y metodologías más apropiadas para el desarrollo de la aplicación.
- **Analítico-sintético:** Se analizarán todos los procesos separados en pequeños grupos o partes y después se sintetizarán los resultados obtenidos para llegar a conclusiones.

Resultados Esperados:

Con la realización del presente trabajo se pretende seguir fortaleciendo el desarrollo de la herramienta portafolio de desarrollo personal en el Centro de Tecnologías para la Formación (FORTES). Se espera que al concluir esta versión, la aplicación sea más competente siendo capaz de importar y exportar recursos, obtener un mayor nivel de personalización del entorno de trabajo, así como mayor retroalimentación entre los usuarios. De manera general se pretende que el portafolio se ajuste más a las necesidades de los usuarios.

Para una mejor organización y entendimiento del trabajo, la presente investigación se estructuró de la siguiente forma:

Capítulo 1: “Fundamentación Teórica”

Se realiza un estudio el estado del arte donde se detallan los elementos fundamentales sobre la selección de la herramienta, metodología y tecnología utilizada para el desarrollo del software.

Capítulo 2: “Propuesta de Solución”

Se determinan los requisitos funcionales y no funcionales del software. Se identifican y describen los casos de usos. Se realiza el diseño del sistema, definiendo la arquitectura y los componentes principales. De forma general, se describe el funcionamiento y estructura interna del sistema.

Capítulo 3: “Implementación y Prueba”

Se documenta todo el proceso de implementación y los resultados de pruebas a emplear para comprobar el correcto funcionamiento del software.

Capítulo 1: Fundamentación teórica

Para comenzar con el estudio la investigación, es necesario conocer primeramente todo lo referente a los portafolios digitales, su utilización, aceptación y evolución, tanto en Cuba como en el resto del mundo. Para ello, se estudiarán productos de igual propósito con el fin de determinar las características y rasgos fundamentales a tener en cuenta para complementar el software ya existente. Como parte de la investigación también es necesario conocer las herramientas, metodologías tecnologías y estándares más utilizados en este tipo de producto.

1.1. Conceptos asociados al dominio del problema

1.1.1. Portafolio electrónico:

Un portafolio digital es una colección de trabajos excepcionalmente buenos o relevantes en la trayectoria de una persona (Doval, 2005). Estos les permiten a los artistas, fotógrafos, escritores y otros profesionales creativos compartir fácilmente los ejemplos de su experiencia laboral. (Hamlett, 2008)

Los portafolios digitales, portafolios electrónicos o e-portafolios como también se les conoce, están muy vinculados al proceso de enseñanza y aprendizaje. Así lo demuestra Barberà cuando dice (Sánchez and Gámiz, 2011) que un portafolio digital “Consiste en una selección de evidencias/muestras que tiene que recoger y aportar el estudiante a lo largo de un período de tiempo determinado y que responde a un objetivo concreto (…). Estas evidencias permiten al alumnado demostrar que está aprendiendo, a la vez que posibilitan al profesor un seguimiento del progreso de este aprendizaje”.

Los portafolios digitales, contienen básicamente el mismo material que se puede tener en un portafolio tradicional. Sin embargo, son capturados, organizados, guardados y presentados de manera electrónica. Un portafolio digital puede contener: fotografías digitales, imágenes escaneadas, archivos de texto, audio, video y/o combinaciones de estos formatos. (Sulbarán, 2010)

Al incorporar diversas tecnologías, el portafolio ya no está limitado al secuenciamiento y promueve la utilización de diversos medios de almacenamiento, que son mucho más manejables tales como: CD ROM, pendrive, diskettes, sitios Web, entre otros. (Sulbarán, 2010)

Los beneficios que brindan son innumerables, entre ellos se puede apreciar que es fácil de almacenar, porque su espacio deja de ser físico para pasar al mundo electrónico. Su distribución es fácil, económica y de reproducción inmediata. Su elaboración es permanente, con posibilidad de reeditarse o ampliarse cuando se quiera. La forma de compartirlo es muy amplia, porque se puede hacer llegar a cualquier persona aún en los lugares más alejados. (Sulbarán, 2010)

1.2. Características de los portafolios de desarrollo personal

Los portafolios han adquirido en los últimos tiempos una gran relevancia. Buena parte de este impulso se debe a la red y a la utilización de portafolios electrónicos ya que reducen, en buena medida, el engorroso trabajo de archivar, clasificar, ordenar y reordenar los materiales acumulados. (Doval, 2005)

Los portafolios digitales se pueden encontrar cumpliendo diferentes funciones según ciertos principios pedagógicos y tecnológicos: Suelen usarse en entornos académicos donde se vinculan al proceso de enseñanza aprendizaje o con propósitos personales como currículum, búsqueda de trabajos, intereses, etc). Según su carácter estos pueden ser privados, donde se tiene el control completo del sistema, o pueden ser propiedad del usuario, cuando en su uso como herramienta evaluativa, los portafolios se hacen públicos para que sean analizados por un profesor, dejando de ser propiedad exclusiva de sus autores y siendo compartidos con la institución educativa. (Illera et al., 2009)

Otra característica importante conlleva a que sea el usuario quien decida sobre su utilización o reutilización a lo largo del tiempo. Finalmente, por nombrar un aspecto relativo al diseño tecnológico, destacan muchos por ser sistemas basados únicamente en interacción a través de Internet y poder ser accedidos desde diferentes localizaciones, aunque pueden ser creados con un destino *off line*, basados en medios físicos, o publicados luego en la red. (Illera et al., 2009)

Un aspecto que se hace necesario destacar es la interacción y comunicación entre los usuarios del sistema. En toda situación de enseñanza-aprendizaje es necesario establecer canales de comunicación efectiva y eficaz entre profesor y alumnos. El profesor debe ser capaz de comentar las evidencias, las secciones, o de forma más genérica, el portafolios publicado por los estudiantes. (Illera et al., 2009)

1.3. Estudio de sistemas similares a nivel internacional

1.3.1. Carpeta Digital utilizada en la Universidad de Barcelona

En la Universidad de Barcelona, en el curso 2006 - 2007 se implantó la *Carpeta Digital* en diversas asignaturas de las enseñanzas de Pedagogía, Comunicación Audiovisual Biblioteconomía y Documentación, entre otras. Las principales funcionalidades del sistema fueron agrupadas en 4 subgrupos: gestión documental y construcción de portafolios, visualización y publicación de los resultados, interacción y comunicación entre los usuarios del sistema y gestión de los resultados de evaluación. Se prestó especial atención a la comunicación entre usuarios porque de forma muy sencilla ofrecía la posibilidad de evaluar continuamente el desempeño del estudiante en cada tarea que desarrollaba. (Illera et al., 2009)

El uso de la *Carpeta Digital* en la universidad antes mencionada proporcionó un soporte específico para el profesorado y un soporte técnico general, también para los estudiantes, que les ayudó a resolver problemas y dudas en el proceso de enseñanza-aprendizaje. (Illera et al., 2009)

1.3.2. Portafolio de Desarrollo Personal empleado en la Universidad de Cataluña

En la Universidad de Cataluña se utiliza un Portafolio de Desarrollo Personal que permite documentar evidencias que en conjunto informan del abasto y calidad de la actuación profesional de un individuo para reflexionar y mejorarla, así como promocionarse en su carrera profesional. El portafolio se utiliza en el contexto universitario como instrumento de autoevaluación del profesor con el objetivo de definir sus objetivos docentes y revisar sus estrategias metodológicas y de evaluación, así como la tarea realizada por parte de sus estudiantes con el fin de comprobar si ha conseguido sus objetivos. El uso de este portafolio docente impulsó la eficacia y los resultados de actuación profesional, representando un primer paso hacia el cambio y la mejora. (Feixas and Valero, 2003)

1.4. Estudio de sistemas similares en Cuba

1.4.1. Experiencia de la Universidad Marta Abreu en el uso de un Portafolio Digital

La educación superior en Cuba se encuentra inmersa en un proceso de transformación en correspondencia con las exigencias del contexto universal, las particularidades del proyecto social cubano y las circunstancias concretas del país desde el punto de vista económico, político y educacional. Con la mira puesta en el perfeccionamiento del proceso de enseñanza-aprendizaje se realizan diversos estudios que avalan cada una de las decisiones que se toman. (León and Campdesuñer, 2012)

Ante la imposibilidad de los métodos evaluativos tradicionales para ofrecer una solución coherente a las actuales necesidades de la sociedad y del sistema educativo, se buscan métodos alternativos que permitan avanzar en este sentido. Una técnica que goza de gran aceptación y sobre la cual versan varias investigaciones es la evaluación por portafolio. (León and Campdesuñer, 2012)

En la Universidad Central de Las Villas “Marta Abreu” se utiliza un portafolio digital en las facultades de Psicología, Ciencias de la Información y de la Educación. Los resultados alcanzados durante la aplicación del portafolio digital como medio para la evaluación del aprendizaje fueron alentadores: se materializó un aumento gradual de la calidad de las evidencias aportadas por los estudiantes, los cuales mostraron cómo mejoraban las reflexiones sobre su progreso e incluso la mayoría de los alumnos comenzó a reflexionar sobre las estrategias propias de aprendizaje. (León and Campdesuñer, 2012)

Desde el punto de vista del profesor, también se detectaron diversas decisiones que denotan un proceso de reflexión sobre la enseñanza que se ha practicado. En este sentido, se destaca la utilización de las guías para la reflexión elaboradas por el docente. (León and Campdesuñer, 2012)

De manera general se pudo apreciar que la evaluación del aprendizaje no debe verse como un elemento separado de la evaluación de la enseñanza. En este sentido, resultó muy valiosa la vinculación de la evaluación del estudiante junto con la evaluación del accionar del profesor. Se pudo apreciar cómo el trabajo en grupos y la retroalimentación mutua garantizaron un mejor desenvolvimiento de la mayoría de los estudiantes, los cuales manifestaron, además, un mayor nivel de autonomía de modo general. (León and Campdesuñer, 2012)

1.4.2. Portafolio Digital desarrollado la Universidad de las Ciencias Informáticas

En la Plataforma Educativa ZERA desarrollada en la Universidad de las Ciencias Informáticas en el año 2012 se generó un portafolio de evidencias, el cual brinda la posibilidad de acceder a las evaluaciones obtenidas, compartir evidencias con otros usuarios dentro de la plataforma, así como fomentar evidencias que hayan sido compartidas con anterioridad. De esta forma el estudiante puede consultar sus evaluaciones, y establecer pautas para mejorarlas. (Frías and Sánchez, 2011)

Este sistema desarrollado se encuentra totalmente integrado a ZERA y permite exportar información en formato Leap2A. Por otra parte posibilita hacer un análisis detallado del avance que obtuvo el estudiante en el proceso de enseñanza-aprendizaje. (Frías and Sánchez, 2011)

1.5. Ambiente de desarrollo

Para realizar el trabajo con buena calidad se necesita guiar la investigación mediante el uso de herramientas, metodologías, lenguajes de programación, framework y tecnologías que garanticen la eficiencia y eficacia del producto final, así como el cumplimiento exitoso de todos los requisitos planteados por el cliente. A continuación se describen las características del ambiente de desarrollo del software.

1.5.1. Metodologías de desarrollo de software

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas de metodologías[1] que inciden en distintas etapas del proceso de desarrollo. Por una parte, se tienen aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Por otra parte están las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando

[1] Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de software.

se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Las metodologías ágiles están revolucionando la manera de producir software y, a la vez, generando un amplio debate entre sus seguidores y quienes, por escepticismo o convencimiento, no las ven como una alternativa para las metodologías tradicionales. (Letelier and Penadés, 2006)

Para un mayor entendimiento de las diferencias entre las metodologías ágiles y tradicionales, se establece la siguiente tabla comparativa según (Letelier and Penadés, 2006):

METODOLOGÍAS ÁGILES	METODOLOGÍAS TRADICIONALES
Pocos artefactos. El modelado es prescindible, modelos desechables	Más artefactos. El modelado es esencial, mantenimiento de modelos
Pocos roles, más genéricos y flexibles	Más Roles, más específicos
No existe un contrato tradicional, debe ser bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños (< 10 integrantes) y trabajando en el mismo sitio	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos
La arquitectura se va definiendo y mejorando a lo largo del proyecto	Se promueve que la arquitectura se defina tempranamente en el proyecto
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo	Énfasis en la definición del proceso: roles, actividades y artefactos
Se esperan cambios durante el proyecto	Se espera que no ocurran cambios de gran impacto durante el proyecto

Tabla 1: Comparación entre metodologías ágiles y tradicionales

Por las numerables ventajas de las metodologías ágiles los autores de la presente investigación concuerdan con la selección realizada en la tesis Portafolio de Desarrollo Personal para la Centralización

de Evidencias desarrollado en el curso 2013-2014 y a continuación se profundizará en este grupo de metodologías mediante una breve descripción de las características más distintivas de cada una de ellas:

- **Scrum**: Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante sprints[2]. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. (Schwaber and Beedle, 2002)
- **Crystal Clear Methodologies:** Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. (Cockburn, 2001)
- **XP[3]:** Se centra en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes y simplicidad en las soluciones implementadas. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (Beck, 2000)
- **AUP[4]:** Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP[5]. Abarca siete flujos de trabajos, cuatro ingenieriles (modelado, implementación, prueba, despliegue) y tres de apoyo (gestión de configuración, gestión de proyectos y ambiente) manteniendo como prioridad

[2] Iteraciones con una duración de 30 días.

[3] Del inglés Extreme Programing.

[4] Del inglés Agile Iniciad Process.

[5] Acrónimo de Proceso Unificado de Rational.

fundamental satisfacer al cliente mediante la entrega temprana y continua de software con valor. (Ambler, 2005)

- **Aup-Variación UCI:** De las cuatro fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes tres fases de AUP en una sola llamada Ejecución y se agrega la fase de Cierre.

Después de haber analizado algunas de las metodologías, así como sus propiedades y formas de trabajo se decidió continuar con el estudio de AUP-Variación UCI porque es la adopción de muchas de las técnicas ágiles de XP y otros procesos ágiles que mantiene de las RUP, además de ser la utilizada en el centro FORTES.

La metodología AUP- Variación UCI para lograr el éxito de un producto en la mayor brevedad de tiempo posible define un conjunto fases dentro de su ciclo de desarrollo, estas son según (Rodríguez, 2014):

- **Inicio**: Se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución**: Se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.
- **Cierre**: Se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Esta metodología cuenta con ocho disciplinas:

- **Modelado del negocio:** Comprende los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito.
- **Requisitos:** El esfuerzo principal es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.
- **Análisis y diseño:** En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.
- **Implementación:** A partir de los resultados del Análisis y Diseño se construye el sistema.
- **Pruebas internas:** Se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posibles componentes de pruebas ejecutables para automatizar las pruebas.
- **Prueba de liberación:** Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
- **Prueba de aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.
- **Despliegue:** Constituye la instalación, configuración, adecuación, puesta en marcha de soluciones informáticas y entrenamiento al personal del cliente. (Rodríguez, 2014)

Para su correcto funcionamiento, AUP se basa en seis principios fundamentales:

- **Simplicidad**: Todo se describe concisamente utilizando poca documentación, no miles de ellas.

- **Agilidad**: El ajuste a los valores y principios de La Alianza Ágil cuya esencia es aceptar que los requisitos cambien, incluso en etapas tardías del desarrollo, entregar software funcional frecuentemente y la medida principal de progreso es el software funcionando.
- **Centrarse en actividades de alto valor**: La atención se centra en las actividades que en realidad lo requieren, no en todo el proyecto.
- **Herramienta de la independencia**: Se pueden usar cualquier conjunto de herramientas pero destacan las de código abierto.
- **Se puede adaptar a las necesidades del cliente**: La metodología AUP es un producto de fácil uso con cualquier herramienta y sin necesidad de tomar un curso.
- **El equipo de trabajo sabe lo que hace:** El cliente no va a leer la documentación detallada del proceso, pero van a querer un poco de orientación de alto nivel de vez en cuando. El producto AUP ofrece enlaces a muchos de los detalles para cuando sea de interés consultarlos. (Ambler, 2005)

1.5.2. Lenguaje de modelado

La creciente complejidad de los sistemas informáticos representa un reto importante para los ingenieros y arquitectos del software. De la preocupación inicial sobre la definición de la estructura y calidad del código final, se ha pasado a dedicar cada vez más tiempo, atención y esfuerzo al diseño y modelado del sistema. Los modelos proporcionan un mayor nivel de abstracción, permitiendo trabajar con sistemas mayores y más complejos, y facilitando el proceso decodificación e implementación del sistema de forma distribuida y en distintas plataformas. (Fuentes and Vallecillo, 2004)

Entre los lenguajes de modelado que define **OMG[6] (Grupo de Gerencia de Objetos)** el más conocido y usado es sin duda **UML[7] (Lenguaje de modelado unificado).** UML es un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan un sistema. UML fue diseñado para ser un lenguaje de modelado de propósito general, por lo que puede utilizarse para especificar la mayoría de los

[6] Del inglés Object Management Group.

[7] Del inglés Unified Modelling Language.

sistemas basados en objetos o en componentes, y para modelar aplicaciones de muy diversos dominios de aplicación. El hecho de que UML sea un lenguaje de propósito general proporciona una gran flexibilidad y expresividad a la hora de modelar sistemas. (Fuentes and Vallecillo, 2004)

El lenguaje UML en su versión 2.4.1 será empleado para modelar los diagramas necesarios en la presente investigación.

1.5.3. Herramientas para el modelado

En el desarrollo de sistemas de información existe el interés por mejorar la productividad y crear productos de software de alta calidad. Para ello se hace imprescindible tomar una acertada decisión a la hora de escoger una herramienta de modelado con UML. Para tener éxito en esta tarea, es necesario analizar factores importantes como las herramientas que existen en el mercado, tanto comerciales como libres, las características de una buena herramienta de modelado con UML y la manera como las herramientas satisfacen las necesidades de las personas que participan en un proyecto, para apoyar el proceso de Ingeniería de Software. (Quintero et al., 2012)

Visual Paradigm es una herramienta CASE[8] (Ingeniería de Software Asistida por Computación). La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Se caracteriza fundamentalmente por su gran disponibilidad en múltiples plataformas, diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad, uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación, generador de informes, capacidades de ingeniería directa e inversa, licencia gratuita y comercial, soporta aplicaciones Web y varios idiomas. (Pressman, 2002)

Dada las características antes descritas y teniendo en cuenta que es la herramienta que se utiliza en la UCI para el modelado de procesos de desarrollo de software, se empleará la versión 8.0 de Visual

[8] Del inglés Computer Aided Software Engineering.

Paradigm para la construcción de los diagramas de clase, entidad-relación, componente, despliegue, caso de uso y modelo del dominio.

1.5.4. Lenguajes de desarrollo de software

Lenguaje del lado cliente

El lenguaje **HTML (Lenguaje de Marcación de Hiper-Textos)** es uno de los puntales de la Web. Desde hace más de dos décadas ejerce una función primordial en el proceso de distribuir información a través de internet. (Franganillo, 2010)

HTML sirve para crear documentos estructurados, conteniendo en formato texto tanto la estructura como el contenido y utiliza etiquetas o marcas para definir las estructuras, cada navegador las utilizará para mostrar según dichas marcas la información que se ha expresado en su interior. (Pereira-Castillo Nunes, 2012)

Se utilizará en esta investigación HTML en su versión 5.0 para facilitar la inclusión de elementos multimedia ya que proporciona mecanismos para simplificar el trabajo. (Franganillo, 2010)

Mejorar el impacto que las páginas web producen en el cliente es objetivo esencial de los diseñadores de todo el mundo. Las **hojas de estilo en cascada (CSS[9])**, permiten desarrollar la creatividad en el diseño con una intuición sin precedentes. Las CSS constituyen un mecanismo para asociar estilos de composición a documentos estructurados, del tipo HTML o XML. Aplicables a cualquier navegador, admiten un mayor control sobre los distintos elementos de una página, permitiendo definir el estilo de las fuentes, el color, el espaciado del texto, la posición del contenido, e incluso variaciones en el sonido en los elementos auditivos. Estos estilos pueden definirse para luego ser aplicados al código de cualquier documento. (Briggs et al., 2003)

[9] Del inglés Cascading Style Sheets.

Para lograr una mayor personalización del espacio de trabajo en el portafolio se utilizará CSS 3.0 ya que es totalmente compatible con HTML5 y sus nuevas etiquetas, además de que los nuevos selectores dan más flexibilidad a la hora de seleccionar unos u otros elementos. (Pastorini, 2014)

Como lenguaje de programación se utilizará **JavaScript,** puesto que es el lenguaje interpretado más utilizado, principalmente en la construcción de páginas Web, con una sintaxis muy semejante a Java y a C. Se basa en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. (Flanagan, 2007)

Este leguaje se utiliza principalmente para crear páginas web dinámicas que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. (Eguíluz, 2008)

La inigualable popularidad de JavaScript como lenguaje de programación de aplicaciones web se ha extendido a otras aplicaciones y otros entornos no relacionados con la web. (Eguíluz, 2008)

Tecnología del lado cliente

Se utilizará **AJAX**[10] como tecnología del lado cliente aunque en realidad no es una tecnología, sino la unión de varias tecnologías que juntas pueden lograr cosas realmente impresionantes como GoogleMaps, Gmail el Outlook WebAccess o algunas otras aplicaciones muy conocidas. AJAX, en resumen, carga y renderiza una página, luego se mantiene en ella mientras scripts y rutinas van al servidor buscando, en *background*, los datos que son usados para actualizar la página solo renderizando esta y mostrando u ocultando porciones de la misma. (Garrett, 2005)

La tecnología AJAX incorpora presentaciones basadas en estándares usando XHTML[11] y CSS, exhibición e interacción dinámicas usando el Document Object Model, intercambio y manipulación de datos usando

[10] Del inglés Asynchronous JavaScript + XML.

[11] Siglas en inglés de eXtensible HyperText Markup Language.

XML y XSLT[12], recuperación de datos asincrónica usando XMLHttpRequest y JavaScript para unir todas las tecnologías. (Garrett, 2005)

La diferencia de AJAX radica en que elimina la naturaleza "arrancar-frenar-arrancar-frenar" de la interacción en la Web introduciendo un intermediario –un motor AJAX- entre el usuario y el servidor. (Garrett, 2005)

En vez de cargar una página Web, al inicio de la sesión, el navegador carga al motor AJAX (escrito en JavaScript y usualmente "sacado" en un frame oculto). Este motor es el responsable por renderizar la interfaz que el usuario ve y por comunicarse con el servidor en nombre del usuario. El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador y un icono de reloj de arena esperando a que le servidor haga algo. (Garrett, 2005)

Para un mayor entendimiento se muestra la siguiente figura que ilustra el patrón de interacción sincrónica de una aplicación Web tradicional comparada con el patrón asincrónico de una aplicación AJAX.

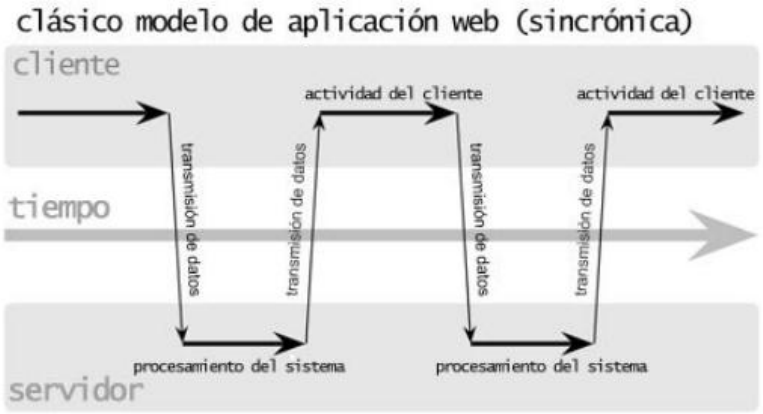


Ilustración 1: Clásico modelo de aplicación web (sincrónica)

[12] Acrónimo de Extensible Stylesheet Language Transformations.

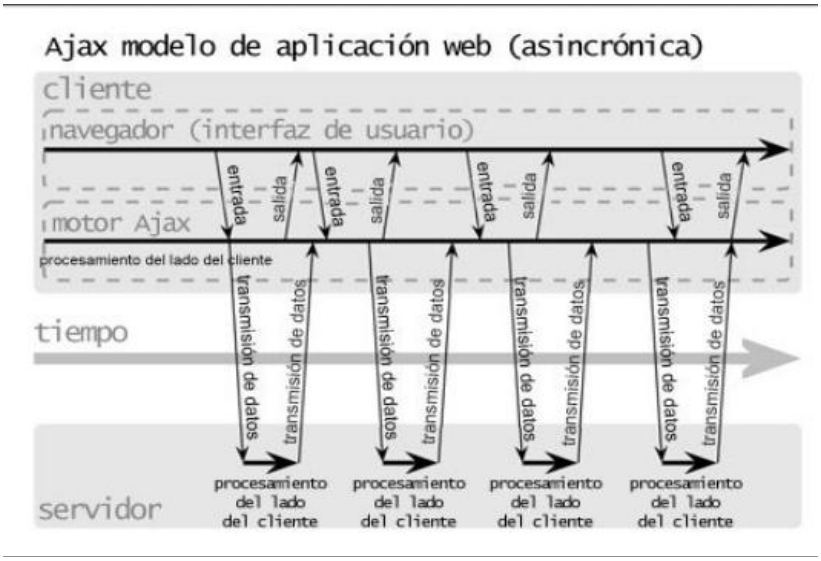


Ilustración 2: Ajax modelo de aplicación web (asincrónica)

La tecnología AJAX será empleada en el presente trabajo para establecer la comunicación asincrónica entre los datos.

Lenguaje del lado servidor

PHP[13] es un lenguaje interpretado del lado del servidor que se caracteriza por su potencia, versatilidad, robustez y modularidad. Los programas escritos en PHP son embebidos directamente en el código HTML y ejecutados por el servidor web a través de un intérprete antes de transferir al cliente que lo ha solicitado un resultado en formato de código HTML puro. Al ser un lenguaje que sigue la corriente *open source*, tanto el intérprete como su código fuente son totalmente accesibles de forma gratuita en la red. (Cobo, 2005)

En comparación con otros tipos de tecnologías similares, PHP resulta más rápido, independiente de la plataforma, está preparado para interactuar con más de 20 tipos de bases de datos y es más sencillo de aprender y utilizar. (Cobo, 2005)

[13] Acrónimo de Procesador de Hipertexto.

Se utilizará PHP en su versión 5.4.16.

1.5.5. Framework

Los ***framework***[14] se utilizan en el ámbito de la programación de aplicaciones desde hace décadas. Genéricamente, un *framework* es un conjunto de herramientas, librerías, convenciones y buenas prácticas que pretenden encapsular las tareas repetitivas en módulos genéricos fácilmente reutilizables. (Pérez, 2007)

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. (Potencier and Zaninotto, 2008)

Capa de presentación

jQuery es un framework para el lenguaje JavaScript, es un producto que simplificará el trabajo de programar en este lenguaje. Implementa una serie de clases (de programación orientada a objetos) que nos permiten programar sin preocuparnos del navegador con el que está visitando el usuario, ya que funcionan de exacta forma en todas las plataformas. (Alvarez, 2010)

Este framework ofrece una infraestructura con mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente como pueden ser las interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de Ajax, etc. Se puede encontrar de manera gratuita, ya que el framework tiene licencia para uso en cualquier tipo de plataforma, personal o comercial. (Alvarez, 2010)

Es importante comentar que jQuery no es el único framework que existe en el mercado. Existen varias soluciones similares pero jQuery es un producto con una aceptación por parte de los programadores muy buena y un grado de penetración en el mercado muy amplio, lo que hace suponer que es una de las

[14] También conocidos como marco de trabajo.

mejores opciones. Además, es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework. Otra cosa muy interesante es la dilatada comunidad de creadores de plugins o componentes, lo que hace fácil encontrar soluciones ya creadas en jQuery para implementar elementos como interfaces de usuario, galerías, votaciones, efectos diversos, etc. (Alvarez, 2010)

Se utilizará el framework jQuery en su versión v1.10.2.

Bootstrap es una herramienta para el desarrollo rápido y correcto de aplicaciones y sitios web. Fomenta las buenas prácticas de diseño y desarrollo web, conforme a estándares W3C[15]. Permite diseñar webs adaptables y fluidas, visualizadas correctamente en múltiples dispositivos. Incluye una robusta base de HTML5**,** CSS3 y JavaScript, también incluye elementos de diseño, tipografías, tablas, formularios, navegación, alertas, etc. (Bootstrap, 2014)

La mayor ventaja es que permite crear interfaces que se adapten a los distintos navegadores, tanto de escritorio como tablets y móviles a distintas escalas y resoluciones, apoyándonos en un framework potente con numerosos componentes webs que nos ahorrarán mucho esfuerzo y tiempo. (Bootstrap, 2014)

Se utilizará Bootstrap en su versión 3.1.1.

Capa lógica del negocio

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor 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. El resultado de todas

[15] El World Wide Web Consortium es un consorcio internacional que produce recomendaciones para la World Wide Web.

estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. (Potencier and Zaninotto, 2008)

Symfony 2.3 está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. Es sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos. Sigue la mayoría de mejores prácticas y patrones de diseño para la web y es fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. (Potencier and Zaninotto, 2008)

La versión de Symfony a utilizar será la v2.3.7 para la implementación de las nuevas funcionalidades del portafolio.

Capa de acceso a datos

El **mapeo objeto-relacional (ORM)** es una técnica de programación para convertir datos del sistema de tipos utilizado en un lenguaje de programación orientado a objetos al utilizado en una base de datos relacional. En la práctica esto crea una base de datos virtual orientada a objetos sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (esencialmente la herencia y el polimorfismo). (del Busto and Enriquez, 2012)

Entre las ventajas que ofrecen los ORM se encuentran: rapidez en el desarrollo, abstracción de la base de datos, reutilización, seguridad, mantenimiento del código, lenguaje propio para realizar las consultas. (del Busto and Enriquez, 2012)

Doctrine es un ORM para PHP 5.3.0 que proporciona persistencia transparente de los objetos desde el lenguaje PHP. Utiliza el patrón Data Mapper1 en el núcleo del proyecto, logrando separar la lógica de dominio / negocio de la persistencia en un sistema de bases de datos relacional. Una de las principales características de Doctrine es su lenguaje SQL llamado Lenguaje de Consulta de Doctrine (DQL)[16]

[16] Del inglés Doctrine Query Language.

inspirado en el Lenguaje de Consulta de Hibernate (HQL)[17] y el bajo nivel de configuración que se necesita para comenzar un proyecto. Soporta las operaciones de CRUD[18], desde la creación de nuevos registros a la actualización de los antiguos. Crea manualmente y automáticamente el modelo de base de datos a implementar. Soportan varios motores de bases de datos (como MySQL, PostgreSQL, Microsoft SQL). (Rosales Morales et al., 2013)

Se utilizará la versión 2.3 de Doctrine porque marca el comienzo de un nuevo enfoque ORM. Entre las ventajas que traen consigo cambios en el desacoplamiento de su lógica de negocio de la capa de persistencia y capacidad de prueba fácil de su modelo de dominio. Además de una reescritura de más de 90% de la base del código existente, sus objetos persistentes (llamadas entidades en Doctrine 2) no están obligados a extender más de una clase base abstracta. Doctrine 2 permite el uso de simples objetos de PHP, presenta una mejor arquitectura y potentes algoritmos que logran realizar un funcionamiento más rápido en comparación con la versión anterior, soporta una API[19] que permite transformar una sentencia SQL arbitraria en un objeto-estructura y la herencia no es un problema en la actualidad, entre otras características. (Rosales Morales et al., 2013)

1.5.6. Sistema gestor de base de datos

PostgreSQL es un gestor de bases de datos orientadas a objetos muy conocido y usado en entornos de software libre porque cumple los estándares SQL92 y SQL99, y también por el conjunto de funcionalidades avanzadas que soporta, lo que lo sitúa al mismo a un mejor nivel que muchos SGBD[20] comerciales. Funciona en múltiples plataformas. Cuenta con un rico conjunto de tipos de datos, permitiendo además su extensión mediante tipos y operadores definidos y programados por el usuario. Su administración se basa en usuarios y privilegios. Puede extenderse con librerías externas para soportar

[17] Del inglés Hibernate Query Language.

[18] Acrónimo de Create-Read-Update-Delete.

[19] Mecanismo de extensión proporcionada por Visual Paradigm para extender las funcionalidades del software de cliente.

[20] Sistema de Gestión de Base de Datos.

encriptación y búsquedas por similitud fonética. Controla la concurrencia multi-versión, lo que mejora sensiblemente las operaciones de bloqueo y transacciones en sistemas multi-usuario. (Ginestà and Mora, 2012)

Se utilizará la versión 9.3.4 de PostgreSQL para el trabajo con la base de datos en la presente investigación.

1.5.7. Servidor web

El servidor Web Apache es, según numerosos estudios, el principal servidor de la Web desde hace diez años. Respaldado por una comunidad de desarrollo brillante, se sustenta en un amplio número de personas y organizaciones. Transparencia y diversidad son las principales características de Apache. El código fuente es totalmente abierto. Su arquitectura modular, construida sobre un pequeño núcleo, se adapta a las necesidades específicas de cada usuario. (Kew, 2008)

Entre las características más distintivas de este servidor destaca que es multiplataforma, modular porque puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y la posibilidad de desarrollar módulos específicos. Otro rasgo importante es la extensibilidad ya que gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP. (Apache, 2014)

Se empleará la versión 2.4.6 de Apache en el presente trabajo.

1.5.8. Entorno de desarrollo

NetBeans es el IDE (Integrated Development Environment) oficial de Java. Con sus editores, analizadores de código y convertidores, puede actualizar las aplicaciones de forma rápida y sin problemas. (NetBeans, 2014)

La plataforma de NetBeans ofrece las características de una aplicación de cliente enriquecida en base a su arquitectura modular formada por un conjunto de API's en donde cada módulo se identifica como una

nueva funcionalidad, teniendo la posibilidad de un desarrollo paralelo ya que solo se requiere integrar el módulo a la aplicación en desarrollo. (Ortíz et al.)

Algunas de las características importantes de la plataforma son el marco de trabajo para la interfaz gráfica basada en el paquete Swing, el editor de datos, la personalización de pantalla, los asistentes y sistema de datos, la facilidad de integración, el alto grado de cohesión y coherencia, el editor soporta varios lenguajes de Java, C / C ++, XML y HTML, PHP, Groovy, Javadoc, JavaScript y JSP, y ofrece diferentes vistas de los datos, desde varias ventanas de proyectos a herramientas útiles para la creación de aplicaciones y gestionar de manera eficiente, lo que le permite desglosar sus datos de forma rápida y sencilla. (Ortíz et al.)

Será empleado el IDE NetBeans v8.0.

1.5.9. Sistema de control de versiones

Un Sistema de Control de Versiones (CVS[21]) es un software que administra el acceso a un conjunto de ficheros, y mantiene un historial de cambios realizados. El control de versiones es útil para guardar cualquier documento que cambie con frecuencia, como código fuente, documentación o ficheros de configuración. Un sistema de control de versiones debe proporcionar un mecanismo de almacenamiento de los elementos que gestiona. Debe posibilitar realizar cambios sobre los elementos almacenados y debe contar con un registro histórico de las acciones realizadas con cada elemento. (EcuRed, 2014)

Para la presente investigación se utilizará la herramienta Git, que es un sistema de control de versiones distribuido, gratuito y de código abierto. Fue diseñado para manejar todo, desde pequeños hasta grandes proyectos con rapidez y eficiencia. Lo que distingue a Git de otras herramientas es que es fácil de aprender, presenta características como ramificación barato locales, zonas de descanso convenientes, y varios flujos de trabajo. Además de ser muy rápido porque casi todas las operaciones se realizan a nivel local, lo que supone una enorme ventaja de la velocidad en sistemas centralizados que constantemente tienen que comunicarse con un servidor en alguna parte. (Git, 2014)

[21] Siglas en inglés de Control Versions System.

1.6. Conclusiones

Después de haber realizado un estudio relacionado con los portafolios digitales, tanto a nivel internacional como nacional, se pudo concluir que el empleo de los mismos en el proceso de enseñanza-aprendizaje es beneficioso ya sea para el alumno en la realización de tareas así como para los docentes en la evaluación de las mismas.

En la gran mayoría de los portafolios investigados se encontraron numerables secciones que son comunes para todos, entre ellas destaca una dedicada a la información personal, a los objetivos que el estudiante ha de alcanzar y los criterios de evaluación que se tendrán en cuenta y que han de corresponderse con las competencias contempladas en el currículum institucional, a las evidencias de los trabajos del estudiante, a las reflexiones del alumno sobre las evidencias presentadas, a los comentarios de otros estudiantes, a la valoración y evaluación realizada por el tutor.

Fueron seleccionados como metodología de desarrollo AUP, UML como lenguaje de modelado. Entre los lenguajes de programación escogidos del lado cliente se encuentran CCS3, HTML5, XML y JavaScript; y del lado servidor PHP. Como framework se ratificó Symfony para el desarrollo web PHP y Bootstrap para hacer interfaces de aplicaciones web más amigables, ya que este define una serie de etiquetas que posibilitan un maquetado de las interfaces de usuario a una mayor precisión. Se seleccionó PostgreSQL y Apache como Sistema Gestor de Base de Datos y servidor web Apache respectivamente. Por último se ratificó NetBeans como entorno de desarrollo integrado.

Capítulo 2: Propuesta de solución

Luego de haber realizado un profundo análisis sobre algunos sistemas de igual propósito desarrollado tanto a nivel nacional como internacional, y seleccionado el entorno de trabajo, se procede a la elaboración de una propuesta de solución donde se definen los requisitos funcionales y no funcionales que tendrá el software junto a la arquitectura y la descripción de los casos de usos que proporcionarán mayor interoperabilidad al portafolio digital.

2.1. Características del sistema

La siguiente propuesta de solución pretende complementar el Portafolio de Desarrollo Personal ya existente por lo que añade funcionalidades como importar y exportar recursos seleccionados por el usuario. El entorno de trabajo se mantendrá, aunque se podrá personalizar logrando que el usuario se identifique con el sitio. Otro de los actores que interactuará con la aplicación será el administrador, pero esta vez gestionando grupos de usuarios, es decir, creando, eliminando y modificando grupos de usuarios. Una de las funcionalidades con que cuenta el sistema es que se pueden compartir archivos, al terminar la nueva versión también se podrá compartir el portafolio completo y por secciones. Por último, el usuario entre sus opciones tendrá la posibilidad de seleccionar cuál o cuáles de las notificaciones que se generan desea que le lleguen, evitando así aquellas alertas que no son de su interés.

2.2. Modelo de dominio

El modelo de dominio puede ser tomado como el punto de partida para el diseño del sistema. Cuando se realiza la programación orientada a objetos, el funcionamiento interno del software va a imitar en alguna medida a la realidad, por lo que el mapa de conceptos del modelo de dominio constituye una primera versión del sistema. (EcuRed, 2015b)

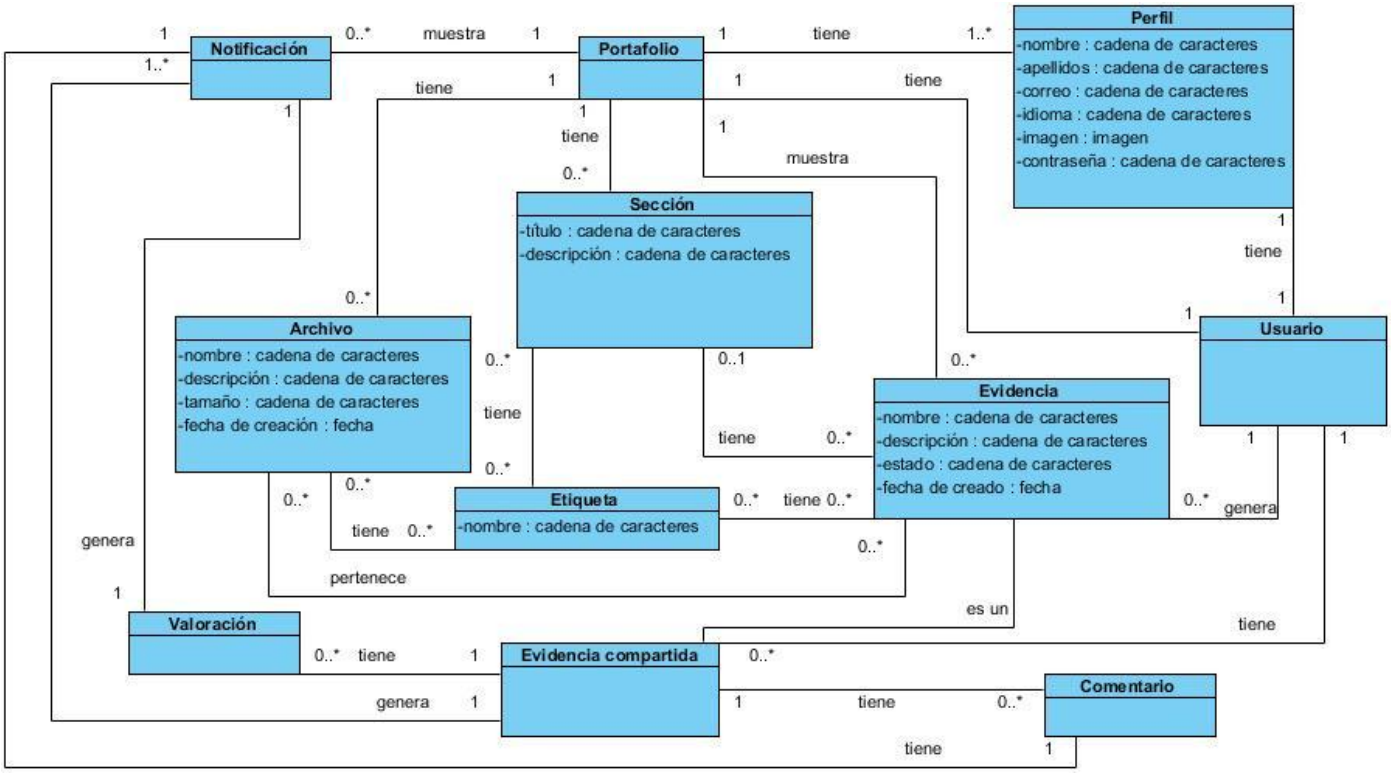


Ilustración 3: Modelo del dominio

2.2.1. Conceptos del dominio

Archivo: Adjunto asignado a una evidencia.

Comentario: Opinión que puede acompañar a una evidencia compartida.

Etiqueta: Nombre que identifica las secciones, archivos y evidencias facilitando su búsqueda en el sistema.

Evidencia: Recurso que crea el usuario para luego reflexionar y comprobar el cumplimiento de sus metas.

Evidencia compartida: Evidencia que comparte el usuario para luego ser comentada y valorada por otros usuarios.

Notificación: Mensaje que indica algún evento existente.

Perfil: Guarda los datos personales del usuario.

Portafolio: Almacena todas las evidencias que va creando el usuario, así como los comentarios, valoraciones, adjuntos, notificaciones, secciones y etiquetas.

Sección: Guarda las evidencias clasificadas por categorías definidas por el usuario.

Usuario: Persona que interactúa con el portafolio digital, creando, comentando y compartiendo las evidencias.

Valoración: Evaluación que se le otorga a una evidencia.

2.3. Especificación de requisitos

La comprensión de los requisitos de un problema está entre las tareas más difíciles que enfrenta un ingeniero de software. La ingeniería de requisitos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye un conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software. (Pressman, 2002)

A continuación se definen los requisitos funcionales y los requisitos no funcionales, pero antes, es preciso entender en qué consisten:

Los **requisitos funcionales** (RF) de un sistema describen lo que el sistema debe hacer. Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. (Pressman, 2002)

Los **requisitos no funcionales** (RnF) son restricciones de los servicios o funciones ofrecidos por el sistema. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces. (Pressman, 2002)

2.3.1. Requisitos funcionales

Los requisitos funcionales identificados fueron:

- **RF 1:** Importar portafolio: El usuario podrá importar hacia el sistema un portafolio desde el menú de *Opciones*.
- **RF 2:** Personalizar el entorno de trabajo: El usuario podrá editar su entorno de trabajo desde el menú de *Opciones* cambiando los colores (rojo, verde, amarillo, rosado, azul, violeta, carmelita y naranja) del banner, botones, secciones, encabezados y fuentes.
- **RF 3:** Compartir el portafolio por secciones: El usuario podrá compartir las secciones del portafolio con todas las evidencias que contienen las mismas.
- **RF 4:** Exportar archivos: El usuario podrá seleccionar una archivo que no se encuentre vinculado a las evidencias para exportarlo mediante el botón Exportar asociado a cada archivo.
- **RF 5:** Exportar evidencia: El usuario podrá seleccionar un recurso para exportarlo mediante el botón *Exportar* asociado a cada evidencia.
- **RF 6:** Exportar secciones: El usuario podrá seleccionar una sección para exportarla mediante el botón *Exportar* asociado a cada sección.
- **RF 7:** Configurar notificaciones: El usuario podrá seleccionar las notificaciones que desea que le lleguen.
- **RF 8:** Enviar mensajes: El usuario podrá enviar mensajes a otros usuarios registrados en el sistema. Se puede acceder desde el botón *Mensaje* ubicado en el panel de navegación.

2.3.2. Requisitos no funcionales

Los requisitos no funcionales coinciden con los de la tesis precedente ya que se trabaja sobre el mismo sistema:

Usabilidad

RnF 1. Los elementos gráficos como los íconos deberán contar con un *tooltype* o mensaje flotante que señalen el tipo de recurso al que se refiere.

RnF 2. La aplicación deberá contar con una interfaz y navegación funcional, asequible a todo tipo de

usuario. Se debe tener acceso al menú general desde cualquiera de las páginas.

RnF 3. El sistema debe mantener al usuario informado de las operaciones que este realiza a través de la interacción con el mismo.

Soporte

RnF 4. La aplicación debe ser implementada bajo tecnología web, puesto que la misma será accedida a través de Internet cuando se encuentre prestando servicios como una aplicación independiente.

RnF 5. Ejecutarse sobre cualquier navegador, siendo como mínimo compatible con:

- Explorer 8.0 y superior.
- Mozilla Firefox 7.0 y superior.
- Opera 10.0.0 y superior.
- Chrome 7.0 y superior.

Hardware

RnF 6. Los usuarios finales deberán contar como mínimo con:

- Procesador Pentium II o superior.
- 512 MB de RAM.
- 15 GB de HDD.
- Si no cuentan con un servidor local: conexión de banda ancha de 256Kbps como mínimo.

Restricciones de diseño e implementación

RnF 7. Se aplicará la programación orientada a objetos.

RnF 8. El marco de trabajo de desarrollo que se utilizará es: Symfony v2.3.7.

RnF 9. Como IDE se empleará NetBeans v7.4.

RnF 10. Se empleará como SGBD PostgreSQL v9.3

RnF 11. Como servidor web se empleará Apache v2.4.7.

RnF 12. Se utilizará un estándar de codificación definido por el Centro FORTES.

Eficiencia

RnF 13. Cuando una sesión permanece inactiva durante un tiempo especificado, el sistema debe cerrar dicha sesión de forma automática.

Apariencia o Interfaz externa

RnF 14. Mantendrá en todo momento un diseño sencillo, con pocas imágenes y gráficos, con el objetivo de elevar el nivel de respuesta del sistema ante las peticiones del usuario.

RnF 15. Se deben usar letras no muy pequeñas y emplear buen contraste entre el fondo y el texto.

2.4. Modelo de caso de uso del sistema

El **modelo de caso de uso** ayuda al cliente, a los usuarios y a los desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema. La mayoría de los sistemas tienen muchos tipos de usuarios. Cada tipo de usuario se representa mediante un actor. Los actores utilizan el sistema al interactuar con los casos de uso. Un caso de uso (CU) especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto. (Jacobson et al., 2000)

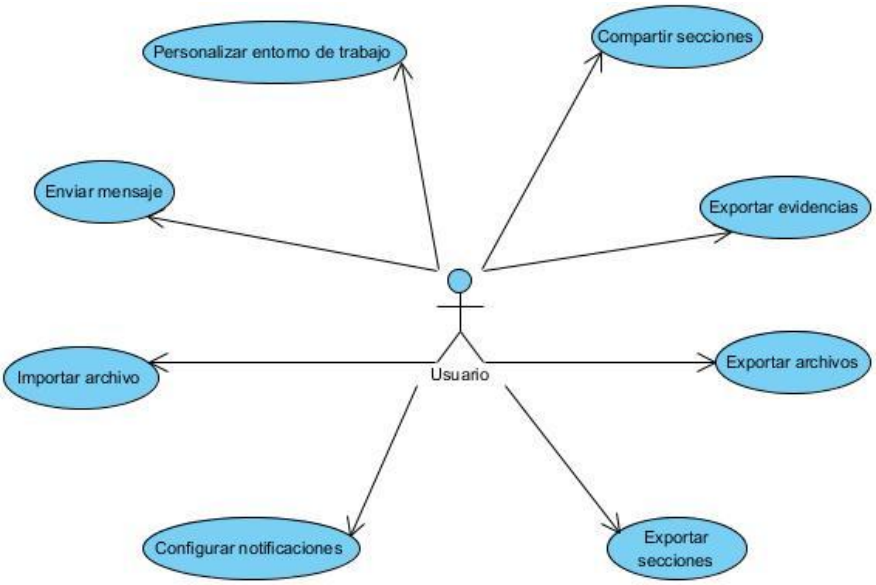


Ilustración 4: Diagrama de caso de uso

2.4.1. Casos de usos del sistema

- **CU 1:** Importar archivo.
- **CU 2:** Personalizar entorno de trabajo.
- **CU 3:** Compartir secciones.
- **CU 4:** Exportar archivos.
- **CU 5:** Exportar evidencias.
- **CU 6:** Exportar secciones.
- **CU 7:** Configurar notificaciones.
- **CU 8:** Enviar mensajes.

2.4.2. Actor del sistema

Cada tipo de usuario se representa mediante un actor. No todos los actores representan a las personas. Pueden ser actores otros sistemas o hardware externo que interactúa con el sistema. Un usuario físico

puede actuar como uno o varios actores, desempeñando los papeles de esos actores. (Jacobson et al., 2000)

ACTOR	DESCRIPCIÓN
Usuario	Actor previamente registrado que interactúa con las secciones del portafolio digital, permitiéndole crear evidencias y secciones, compartirlas, comentarlas, importar y exportar archivos, personalizar el entorno de trabajo y enviar mensajes a otros usuarios.

Tabla 2: Actor del sistema

2.5. Especificación de caso de uso

Para comprender mejor el funcionamiento del sistema a continuación se muestra la descripción del CU Importar archivo. En el Anexo 1 se encuentra un resumen de las especificaciones de los CU más relevantes.

Objetivo	Importar portafolio
Actores	Usuario (inicia)
Resumen	El usuario podrá importar portafolios hacia su sistema.
Complejidad	Alta
Precondiciones	El usuario ha sido autenticado en el sistema correctamente.
Postcondiciones	Se importó un portafolio.
Flujo de eventos	
Flujo básico Importar archivo	

Actor	Sistema
1. El caso de uso inicia cuando el actor selecciona la opción *Opciones*. 2. Selecciona el botón *Importar*. 3. Selecciona el botón *Seleccione un archivo.*	

	4. Muestra una ventana para buscar el portafolio que se desea incluir.
4. Busca la ubicación del portafolio que desea importar y lo selecciona. 5. Selecciona el botón *Subir*.	
	6. Valida el formato del portafolio a importar. 7. Muestra el mensaje "Información importada correctamente". 8. Termina el caso de uso.
Flujos alternos	
4. a El actor selecciona la opción Cancelar	
Actor	**Sistema**
	4. a.1 Regresa a la vista anterior. 4. a.2 Termina el caso de uso.
6. a El formato del portafolio no es compatible con el portafolio	
Actor	**Sistema**
	6. a.1 Muestra el mensaje "¡Fallo en la importación de la información!". 6. a.2 Termina el caso de uso.

Tabla 3: Descripción del CU Importar portafolio

2.6. Arquitectura

Una arquitectura de software de un programa o un sistema computacional es la estructura del sistema, la cual comprende elementos de software, las propiedades externamente visibles de esos elementos, y las relaciones entre ellos. (Fernández, 2006)

2.6.1. Estilos arquitectónicos

Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante composición de los estilos fundamentales. (Reynoso, 2004)

Utilización de los estilos arquitectónicos:

- Sirven para sintetizar estructuras de soluciones.
- Pocos estilos abstractos encapsulan una enorme variedad de configuraciones concretas.
- Definen los patrones posibles de las aplicaciones.
- Permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos no funcionales. (EcuRed, 2015a)

Existen diferentes estilos arquitectónicos entre ellos:

- **Estilo de llamada y retorno:** Enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala.
- **Estilos de código móvil:** Este grupo de estilos enfatiza la portabilidad.
- **Sistemas de flujo de datos:** Esta familia de estilos enfatiza la reutilización y la modificabilidad. Es apropiada para sistemas que implementan transformaciones de datos en pasos sucesivos.
- **Estilos centrados en datos:** Enfatiza la integrabilidad de los datos. Se estima apropiada para sistemas que se fundan en acceso y actualización de datos en estructuras de almacenamiento.
- **Estilos Peer – to – Peer:** Enfatiza la modificabilidad por medio de la separación de las diversas partes que intervienen en la computación. Consiste por lo general en procesos independientes o entidades que se comunican a través de mensajes. (Nodarse, 2013)

Fue seleccionado el estilo de llamada y retorno porque es utilizado en grandes sistemas de software, persiguen escalabilidad y modificabilidad, y la descomposición en módulos disminuye la complejidad. (EcuRed, 2015a)

El estilo de llamada y retorno se caracteriza por una descomposición jerárquica en subrutinas (componentes) que solucionan una tarea o función definida. Los datos son pasados como parámetros y el manejador principal proporciona un ciclo de control sobre las subrutinas. Reflejan la estructura del lenguaje de programación. Permite al diseñador del software construir una estructura de programa relativamente fácil de modificar y ajustar a escala. Se basan en la bien conocida abstracción de procedimientos/funciones/métodos. (EcuRed, 2015a)

2.6.2. Patrones arquitectónicos

Los patrones arquitectónicos ofrecen soluciones a problemas de arquitectura de software en Ingeniería de Software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Expresan un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. (Nodarse, 2013)

Dentro del estilo de llamada y retorno existen diferentes patrones, estos son: (Nodarse, 2013)

- **Modelo – Vista – Controlador:** Separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes (el Modelo, la Vista y el Controlador).
- **Arquitecturas en Capas:** Organización jerárquica, tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior.
- **Arquitecturas Orientadas a Objetos:** En las arquitecturas Orientadas a Objeto (OO), los componentes del estilo se basan en principios OO, tales como: el encapsulamiento, la herencia y polimorfismo.
- **Arquitecturas Basadas en Componentes:** Los sistemas basados en este estilo se basan en principios definidos por una ingeniería de software específica. Un componente de software, es una unidad de composición con interfaces especificadas y dependencias del contexto explícitas.

Se continuará con el uso del patrón Modelo-Vista-Controlador (MVC), pues presenta como ventajas: (Bascón Pantoja, 2011)

- La aplicación está implementada modularmente.
- Sus vistas muestran información actualizada siempre.
- Las modificaciones a las vistas no afectan en absoluto a los otros módulos de la aplicación.
- MVC es bastante utilizado en la actualidad en marcos de aplicación orientados a objeto desarrollados para construir aplicaciones de gran tamaño.
- Las aplicaciones que lo implementan presentan una extensibilidad y una mantenibilidad únicas comparadas con otras aplicaciones basadas en otros patrones.

MVC considera dividir una aplicación en tres módulos claramente identificables y con funcionalidad bien definida: El Modelo, las Vistas y el Controlador. (Bascón Pantoja, 2011)

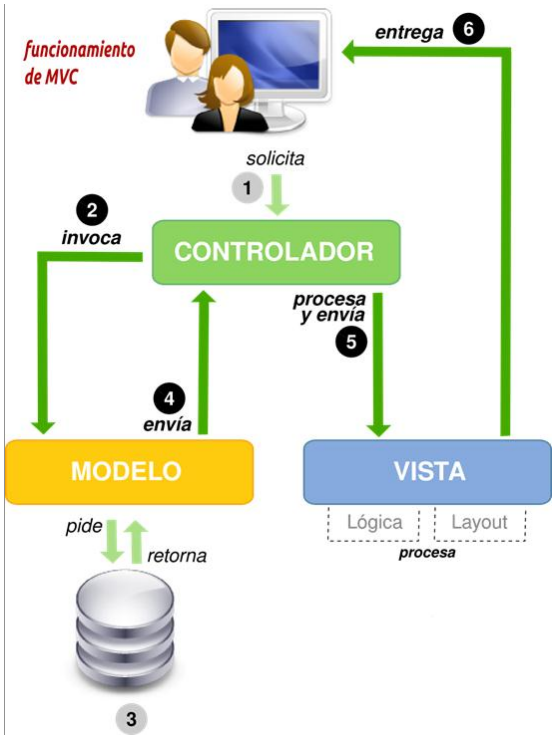


Ilustración 5: Funcionamiento del patrón Modelo-Vista-Controlador

- El **modelo** es un conjunto de clases que representan la información del mundo real que el sistema debe procesar.
- Las **vistas** son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo.
- El **controlador** es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador.

2.6.3. Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Debemos tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios). (Paredes, 2012)

Patrones GRASP [22]

Los patrones GRASP, más que patrones propiamente dichos, son una serie de “buenas prácticas” de aplicación recomendable en el diseño de software.

- **Controlador:** Es el encargado de recibir la petición del usuario y en función de ella, envía la solicitud a las distintas clases para que sea procesada (Fernandez et al., 2010). Como ejemplo, en la aplicación fue creada la clase *SeccionCompartidaControlador* para facilitar la centralización de actividades relacionadas con la sección compartida.
- **Experto:** Posibilita una adecuada asignación de responsabilidades facilitando la comprensión del sistema, su mantenimiento y adaptación a los cambios con reutilización de componentes (Tabares,

[22] Patrones de software para la asignación general de responsabilidades.

2011). Se evidencia en la capa modelo donde las clases existentes tienen la responsabilidad de realizar las consultas a la base de datos ya que cuentan con los atributos necesarios para ello.

- **Bajo Acoplamiento**: Es una medida de la fuerza con que una clase se relaciona con otras, porque las conoce y recurre a ellas; una clase con bajo acoplamiento no depende de muchas otras, mientras que otra con alto acoplamiento presenta varios inconvenientes: es difícil entender cuando está aislada, es ardua de reutilizar porque requiere la presencia de otras clases con las que esté conectada y es cambiante a nivel local cuando se modifican las clases afines (Tabares, 2011). En la aplicación fueron separadas las clases correspondientes al modelo, la vista y el controlador.
- **Creador**: Guía la asignación de responsabilidades relacionadas con la creación de objetos. (Larman, 1999). Este patrón queda evidenciado cuando se crea una instancia de la entidad *Comentario* en la clase *EvidenciaController.php*.

```php
private function createComment(Evidencia $evidencia)
{
    $comentario = new Comentario();
    $comentario->setEvidencia($evidencia);

    return $comentario;
}
```

***Patrones GoF*[23]**

Los patrones GoF indican resoluciones técnicas basadas en Programación Orientada a Objetos. Ayudan a construir software basado en la reutilización y a construir clases reutilizables. Se clasifican según su propósito: (EcuRed, 2015c)

Estructurales: Plantean las relaciones entre clases, las combinan y forman estructuras mayores. Tratan de conseguir que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. Estudian cómo se relacionan los objetos en tiempo de ejecución y sirven para diseñar las interconexiones entre ellos. (EcuRed, 2015c)

[23] Acrónimo en inglés de Gang of Four.

- Decorator (Envoltorio): Añade funcionalidad a una clase dinámicamente. En este trabajo las plantillas *base.html.twig* y *frontend.html.twig* utilizan este patrón estableciendo un maquetado para todas las páginas de la aplicación.

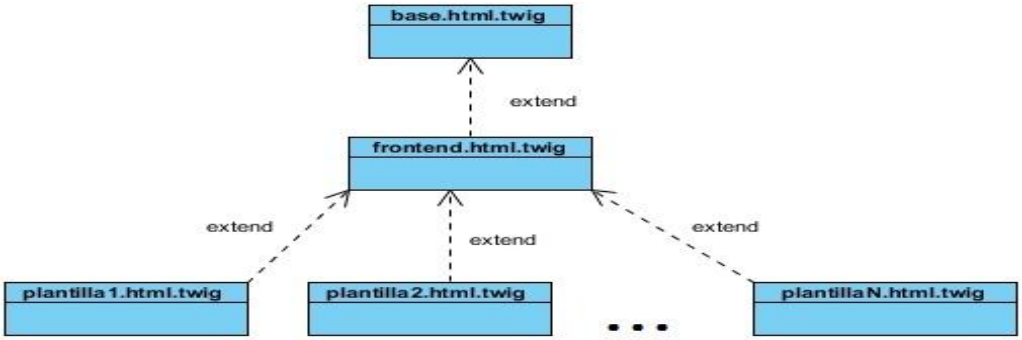


Ilustración 6: Patrón Decorator en Symfony

De comportamiento: Plantea la interacción y cooperación entre las clases. (EcuRed, 2015c)

- Strategy (Estrategia): Permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución. En la aplicación se evidencia cuando se necesita visualizar una entidad *Sección.*

2.7. Conclusiones

En el transcurso del capítulo se documentaron los elementos fundamentales en el diseño del software. Quedaron descritos los requisitos funcionales y no funcionales del sistema. Fue conformada la arquitectura mediante la selección de estilos y patrones arquitectónicos, así como los patrones de diseño garantizando las buenas prácticas y reutilización de funcionalidades. Además se representaron los diagramas del dominio y de caso de uso que servirán como artefactos de entrada en el proceso de implementación.

Capítulo 3: Implementación y prueba

Una vez realizada la propuesta solución, el presente capítulo abordará todo lo relacionado con el diseño de la base de datos, los diagramas de componentes, de clase y los necesarios para el desarrollo de la misma. Como parte del proceso de implementación se desarrollarán las nuevas funcionalidades identificadas a fin de dar cumplimiento a los objetivos planteados. Por último, se determinará la estrategia de prueba a utilizar y los resultados de la ejecución de las mismas.

3.1. Estilos y estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe quedar como si un único programador hubiera escrito todo el código de una sola vez. Al inicio de un desarrollo de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Si fuera necesario incorporar código fuente previo, o bien realizar el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con el código existente. (Díaz, 2013)

A continuación se muestran algunos de los estándares definidos para el desarrollo de la aplicación:

- Los comentarios se escriben entre los caracteres ***/**** y *****/*** o también se encuentran comenzando cada línea de código comentada por los caracteres ***//***.
- Los nombres de los métodos se escriben siguiendo el formato **CamelCase** mientras que los nombres de las clases siguen el formato **UpperCamelCase**.
- Los nombres de las clases se escriben con letra inicial de cada palabra en mayúscula.
- Los nombres de las tablas de la base de datos comienzan con el prefijo **xl**_ .
- El acceso a los atributos de las clases se hace mediante los getAtributo[24] y setAtributo[25].

[24] Método que permite obtener el valor de un atributo especificado.

3.2. Diagrama de clases

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. (Larman, 1999)

Se mostrarán los diagramas de las clases que han sido agregadas o modificadas, el resto se mantiene como aparece en la versión anterior. Seguidamente aparece el diagrama de clase del *bundle* portafolio, el resto se puede consultar en el Anexo 3.

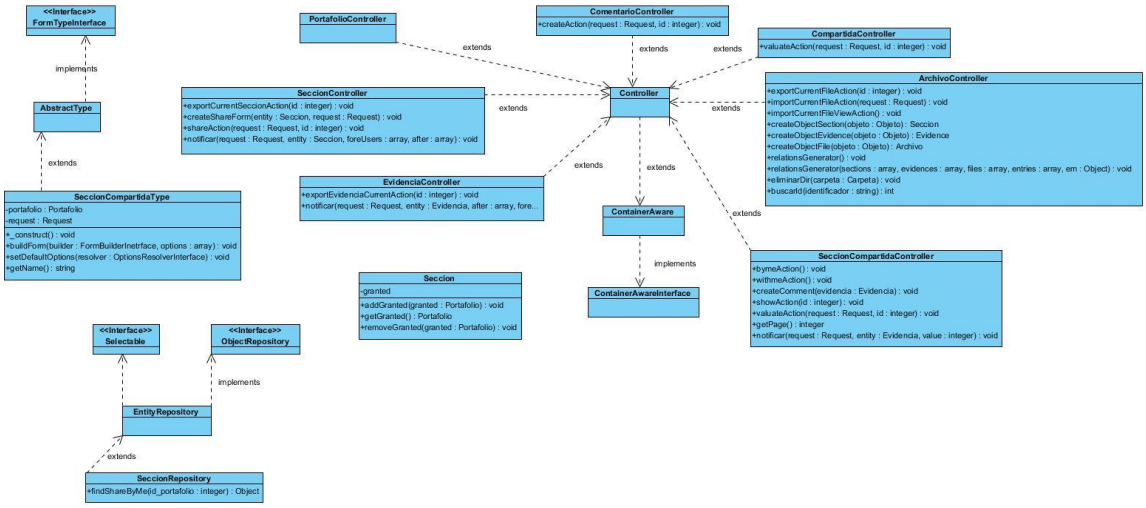


Ilustración 7: Diagrama de Clase PortafolioBundle

3.3. Diseño de la base de datos

Un sistema de base de datos es básicamente un sistema computarizado para guardar registros; es decir, es un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base en peticiones. (Date, 2001)

Cuando se diseña correctamente una base de datos se obtendrá la información precisa y actualizada. A continuación se muestra el diagrama entidad-relación correspondiente al sistema en desarrollo donde se

[25] Método que permite cambiarle el valor a un atributo especificado.

crearon y modificaron algunas tablas de la versión anterior. Entre ellas se encuentra *xl_mensaje_jabber* que fue añadido para guardar los datos correspondientes a la funcionalidad de *Enviar Mensaje*. La tabla *xl_usuario* fue editada con atributos nuevos necesarios para almacenar la información relacionada con la configuración de notificaciones.

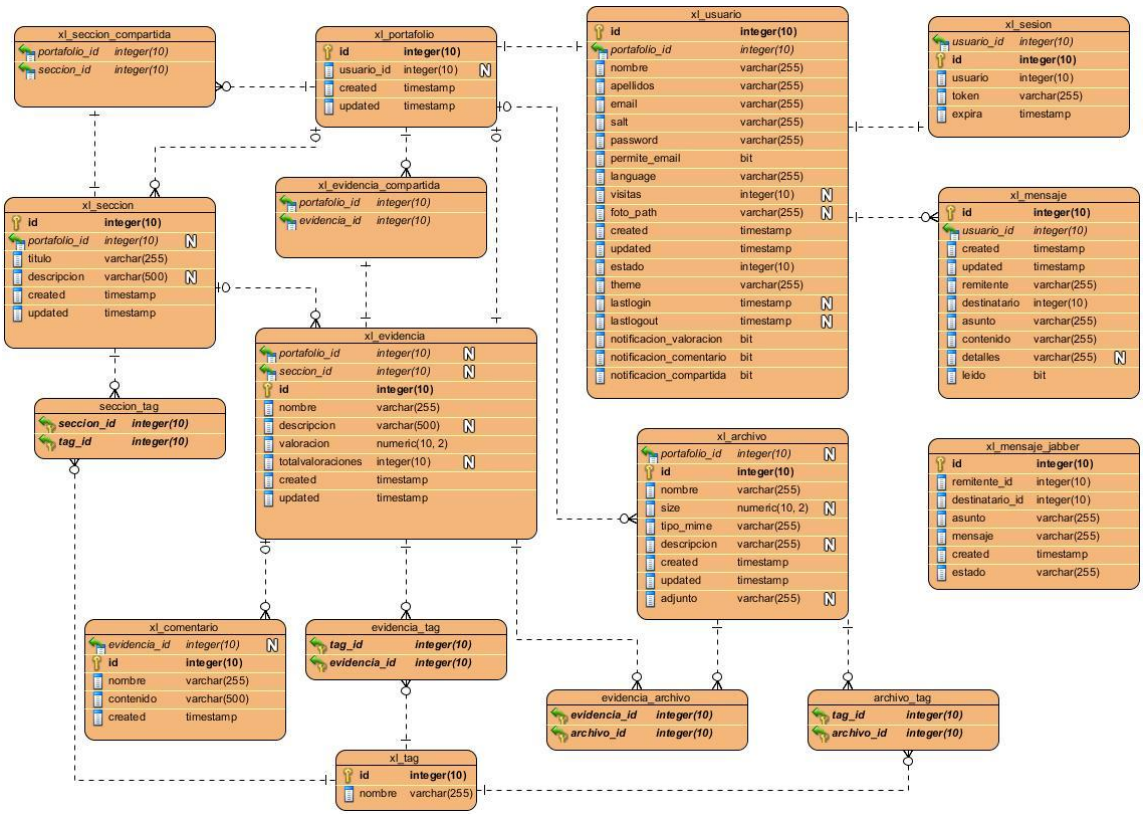


Ilustración 8: Diseño de la base de datos

3.4. Diagrama de componentes

Un diagrama de Componentes permite modelar la estructura del software y la dependencia entre componentes. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente, etc. (Booch et al., 1999)

Se mostrarán sólo los bundles en los que se han modificado o añadido nuevas clases, el resto se mantiene como está descrito en la versión anterior. A continuación se encuentra el diagrama de componente del bundle portafolio, el resto se puede consultar en el Anexo 2.

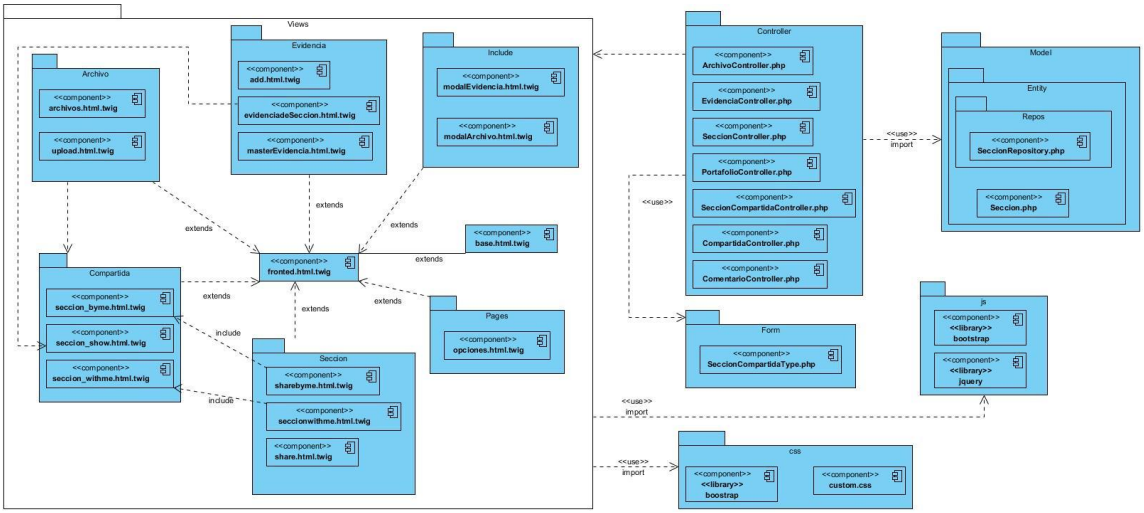
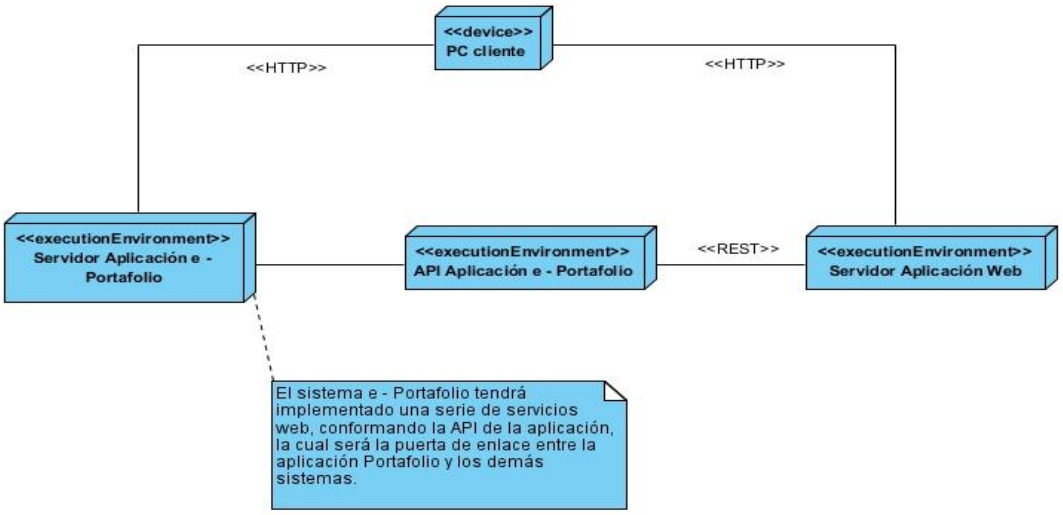


Ilustración 9: Diagrama de componentes. Paquete PortafolioBundle.

3.5. Diagrama de despliegue

Los Diagramas de Despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. (Booch et al., 1999)

El diagrama de despliegue no sufrió cambios con respecto al diseñado en la versión anterior.



3.6. Pruebas de software

La prueba es un conjunto de actividades que se planean con anticipación y se realizan de manera sistemática. Por tanto, se debe definir una plantilla para las pruebas del software (un conjunto de pasos en que se puedan incluir técnicas y métodos específicos del diseño de casos de prueba). (Pressman, 2002)

3.6.1. Pruebas realizadas por iteraciones

Las **pruebas unitarias** prueban que un pequeño trozo de código de la aplicación funciona tal y como debería hacerlo. Idealmente, los trozos de código son la parte más pequeña posible que se pueda probar. En la práctica suelen probarse clases enteras, a menos que sean muy complejas y haya que probar sus métodos por separado. (Eguiluz, 2012)

Las **pruebas de aceptación** tienen como propósito demostrar al cliente el cumplimiento de un requisito del software. Describe un escenario (secuencia de pasos) de ejecución o uso del sistema desde la perspectiva del cliente. Puede estar asociada a requisitos funcionales o no funcionales. (Letelier, 2007)

3.6.2. Resultados obtenidos en las pruebas unitarias

En la realización de las pruebas unitarias se utilizó la librería PHPUnit. Esta devolvió un informe con un total de 21 pruebas ejecutadas donde se realizaron 106 comprobaciones. A continuación se muestran los resultados obtenidos. En el Anexo 4 se muestran algunas de las pruebas unitarias implementadas.

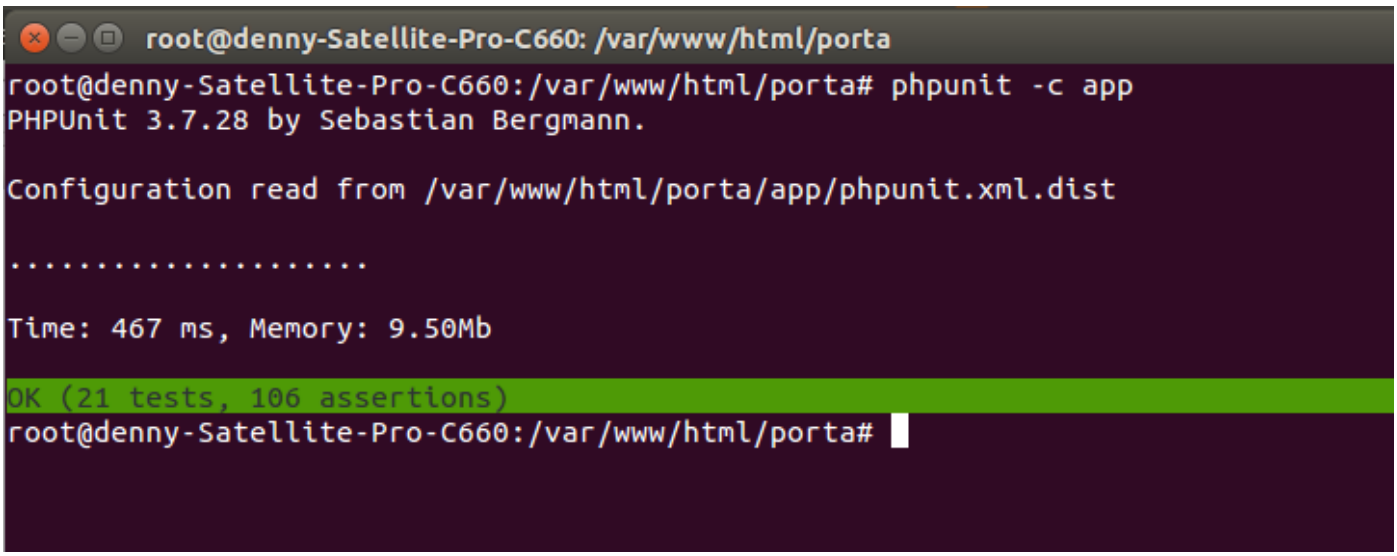


Ilustración 10: Resultados de las pruebas unitarias

3.6.3. Resultados obtenidos en las pruebas de aceptación

Se realizaron las pruebas de aceptación en tres iteraciones donde se identificaron recomendaciones y no conformidades significativas y no significativas. La primera iteración estuvo relacionada con los requisitos de personalización del entorno de trabajo, exportar archivos, exportar evidencias y exportar secciones. La segunda iteración se enfocó en los requisitos importar portafolio, compartir secciones, configurar notificaciones y enviar mensajes. Por último, se desarrolló una tercera iteración donde se comprobó el funcionamiento general del software. Todas las no conformidades fueron resueltas de forma satisfactoria.

A continuación se muestran los resultados obtenidos en cada iteración de pruebas:

	Iteración 1	Iteración 2	Iteración 3

No conformidades	12	7	2
Significativas	8	5	0
No significativas	4	2	2
Recomendaciones	3	3	1

Tabla 4: Resultados obtenidos en la prueba de aceptación.

3.7. Conclusiones

Al concluir el presente capítulo se obtuvo un portafolio digital interoperable, pues se complementó con algunas funcionalidades que permiten realizar acciones como importar y exportar evidencias seleccionadas.

Una vez finalizado el desarrollo de las nuevas funcionalidades fueron aplicadas algunas pruebas de software con el objetivo de detectar y corregir inconsistencias propias del desarrollo que hubiesen surgido en la etapa de implementación.

Conclusiones generales

Una vez finalizada la presente investigación se concluye lo siguiente:

- Se realizó un estudio acerca de las tendencias actuales de los portafolios digitales, mediante el cual se identificaron las herramientas y estándares más empleados en la creación de estos. Quedando conformado el marco teórico conceptual guiado por la metodología AUP y utilizando los lenguajes de programación del lado cliente CCS3, HTML5, XML y JavaScript; y del lado servidor PHP en los *framework* de desarrollo Symfony, jQuery y Bootstrap.
- Se implementaron funcionalidades que proporcionaron que en la nueva versión el portafolio digital sea mucho más interoperable permitiendo importar y exportar recursos, compartir evidencias, configurar notificaciones, personalizar el entorno de trabajo y enviar mensajes.
- La realización de las pruebas unitarias y de aceptación, permitieron detectar errores y corregirlos, garantizando de esta manera el correcto funcionamiento del sistema.

Recomendaciones

Tomando como base la investigación realizada se recomienda para futuros trabajos:

- Brindar la posibilidad de que el usuario pueda generar archivos de texto, video y audio desde el portafolio digital.
- Crear los roles de estudiante y profesor donde se le permita al profesor gestionar grupos de estudiantes e interactuar con ellos compartiendo información y enviando mensajes de grupos.
- Ofrecer la posibilidad al usuario de importar tanto secciones y como evidencias.

Referencias bibliográficas

ALVAREZ, M. A. 2010. Manual de jQuery. *Recuperado el,* 17.

AMBLER, S. W. 2005. *The Agile Unified Process* [Online]. Available: http://www.ambysoft.com/unifiedprocess/agileUP.html.

APACHE, C. 2014. *Apache* [Online]. Available: https://es.opensuse.org/Apache.

BASCÓN PANTOJA, E. 2011. El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing. *Revista Acta Nova,* 2.

BECK, K. 2000. *Extreme programming explained: embrace change*, Addison-Wesley Professional.

BOOCH, G., RUMBAUGH, J., JACOBSON, I., MARTÍNEZ, J. S. & MOLINA, J. J. G. 1999. *El lenguaje unificado de modelado*, Addison-Wesley.

BOOTSTRAP, C. 2014. *Framework, Twitter Bootstrap!* [Online]. Tutorial Bootstrap. Available: http://internoma.github.io/tutorial-bootstrap/ [Accessed Diciembre 2014].

BRIGGS, O., CHAMPEON, S., COSTELLO, E. & PATTERSON, M. 2003. *Cascading Style Sheets*.

COBO, A. 2005. *PHP y MySQL: Tecnología para el desarrollo de aplicaciones web*, Ediciones Díaz de Santos.

COCKBURN, A. 2001. Agile software development. *Computer,* 34**,** 131-133.

DATE, C. J. 2001. *Introducción a los sistemas de base de datos*.

DEL BUSTO, H. G. & ENRIQUEZ, O. Y. 2012. Mapeo Objeto/Relacional (ORM). *Revista Telem@ tica,* 10**,** 1-7.

DÍAZ, E. V. P. 2013. Pautas de codificación. Available: http://gespro.fortes.prod.uci.cu/attachments/20023/Estandar_de_codificación_Xalix.pdf.

DOVAL, F. M. G. 2005. El papel de los portafolios electrónicos en la enseñanza-aprendizaje de las lenguas. *Glosas didácticas: revista electrónica internacional de didáctica de las lengua y sus culturas***,** 10.

ECURED. 2014. *Sistema de Control de Versiones* [Online]. Available: http://www.ecured.cu/index.php/Sistemas_de_control_de_versiones [Accessed diciembre 2014].

ECURED. 2015a. *Estilos arquitectónicos* [Online]. Available: http://www.ecured.cu/index.php/Estilos_arquitect%C3%B3nicos.

ECURED. 2015b. *Modelo del dominio* [Online]. Available: http://www.ecured.cu/index.php/Modelo_de_dominio [Accessed febrero 2015].

ECURED. 2015c. *Patrones GoF* [Online]. Available: http://www.ecured.cu/index.php/Patrones_Gof.

EGUILUZ, J. 2012. Desarrollo Web Ágil con Symfony 2. *easybook.*

EGUÍLUZ, J. 2008. Introducción a JavaScript. *Versión electrónica consultada en www. librosweb. es el,* 20.

FEIXAS, M. & VALERO, M. 2003. El Portafolios y el SEEQ como Herramientas para el Desarrollo Profesional. *Estrategias formativas para el cambio organizacional. CISSPRAXIS. Barcelona. Formato web en [Consulta el 30 de abril de 2008]: http://www. udg. es/ice/FUniversit/Portafolios. pdf*.

FERNÁNDEZ, L. F. 2006. Arquitectura de Software. *Software Guru,* 2**,** 40-45.

FERNANDEZ, V. M., HIDALGO, D. F. & ORTIZ, Y. V. 2010. Propuesta de diseño para proyectos informáticos que utilizan Symfony como framework. *Serie Científica,* 3.

FLANAGAN, D. 2007. *JavaScript. La Guía Definitiva*.

FRANGANILLO, J. 2010. H. 3. Html5: el nuevo estándar básico de la Web.

FRÍAS, E. G. & SÁNCHEZ, E. S. G. 2011. *Desarrollo del Portafolio Electrónico para la Plataforma Educativa ZERA, e-Portafolio.* Universidad de las Ciencias Informáticas.

FUENTES, L. & VALLECILLO, A. 2004. Una introducción a los perfiles UML. *Revista Novatica–Asociación de Técnicos de Informática-España*.

GARRETT, J. J. 2005. Ajax: Un nuevo acercamiento a las aplicaciones web. *Maestros del Web".< http://www. maestrosdelweb. com/editorial/ajax/>(31/03/2009)*.

GINESTÀ, M. G. & MORA, O. P. 2012. Bases de datos en PostgreSQL.

GIT, C. 2014. *Página Oficial de GIT* [Online]. Available: http://git-scm.com/.

HAMLETT, S. 2008. Cómo crear tu portafolio digital. *eHow en español*.

HERNÁNDEZ, X. G. 2014. *Algunas aplicaciones del portafolio en el ámbito educativo* [Online]. Available: http://www.quadernsdigitals.net/datos/hemeroteca/r_47/nr_507/a_7050/7050.pdf.

ILLERA, J. L. R., AGUADO, G., GALVÁN, C. & RUBIO, M. J. 2009. Portafolios electrónicos para propósitos múltiples: aspectos de diseño, de uso y de evaluación. *RED. Revista de Educación a Distancia***,** 1-14.

JACOBSON, I., BOOCH, G. & RUMBAUGH, J. 2000. *El proceso unificado de desarrollo de software*, Addison Wesley Reading.

KEW, N. 2008. *Desarrollo de módulos y aplicaciones con Apache*.

LARMAN, C. 1999. UML y patrones. Introducción al análisis y diseño orientado a objetos.

LEAP2A, C. 2015. *The Leap2A specification for e-portfolio portability and interoperability* [Online]. Available: http://www.leapspecs.org/2A/ [Accessed enero 2015].

LEÓN, Y. F. & CAMPDESUÑER, I. M. 2012. El portafolio digital y su impacto en la calidad del proceso de evaluacion del aprendizaje. *Edutec: Revista electrónica de tecnología educativa***,** 4-16.

LETELIER, P. 2007. Pruebas de Aceptación como conductor del Proceso de Software.

LETELIER, P. & PENADÉS, M. C. 2006. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)* [Online]. Available: http://www.cyta.com.ar/ta0502/b_v5n2a1.htm.

MARIBONA, M. G., RODRÍGUEZ, A. E. L., GIL, D. C. & PÉREZ, L. 2013. *PORTAFOLIO DE DESARROLLO PERSONAL PARA LA PLATAFORMA EDUCATIVA ZERA* [Online]. Available: http://semanatecnologica.fordes.co.cu/sites/default/files/public/p251_0.pdf.

NETBEANS, C. 2014. *NetBeans* [Online]. Available: https://netbeans.org.

NODARSE, I. 2013. Introducción a los estilos y patrones arquitectónicos.

ORTÍZ, C. O. M., LÓPEZ, R. R., MEJÍA, J. D. J. C. & LEÓN, A. R. Una Arquitectura Modular para el desarrollo de un IDE que apoye a la Enseñanza de los Fundamentos de la Programación Orientada a Objetos.

PAREDES, A. 2012. *Patrones de arquitectura vs. Patrones de diseño* [Online]. Available: https://arlethparedes.wordpress.com/2012/08/27/patrones-de-arquitectura-vs-patrones-de-diseno/.

PASTORINI, A. 2014. HTML 5 y CSS 3.

PEREIRA-CASTILLO NUNES, P. H. 2012. Desarrollo de un Juego Web Multiusuario con HTML5.

PÉREZ, J. E. 2007. CSS avanzado. librosweb. es, Ebook en linea http://www. librosweb. es/css_avanzado.

POTENCIER, F. & ZANINOTTO, F. 2008. Symfony 1.2, la guía definitiva. Apress.

PRESSMAN, R. S. 2002. *Ingeniería de Software, un enfoque práctico*.

QUINTERO, J. B., DE PÁEZ, R. A., MARÍN, J. C. & LÓPEZ, A. B. 2012. Un estudio comparativo de herramientas para el modelado con UML. *revista universidad eafit,* 41**,** 60-76.

REYNOSO, C. B. 2004. Introducción a la Arquitectura de Software. *Universidad de Buenos Aires,* 33.

RODRÍGUEZ, T. 2014. Metodología de desarrollo para la Actividad productiva de la UCI.

ROSALES MORALES, Y., MARRERO CLARK, M. E. & TRUJILLO OLIVA, A. 2013. *Extensión de la herramienta Visual Paradigm para la generación de clases de acceso a datos con Doctrine 2.0*.

SÁNCHEZ, E. R. & GÁMIZ, Á. M. E. 2011. *El portafolio digital un nuevo instrumento de evaluación* [Online]. Available: http://www.raco.cat/index.php/dim/article/viewArticle/247586/0 [Accessed 21.

SCHWABER, K. & BEEDLE, M. 2002. Agile Software Development with Scrum.

SULBARÁN, M. 2010. Portafolio digital. *Docencia para la educación superior*.

TABARES, R. B. 2011. Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación. *Entre Ciencia e Ingeniería*, 161-173.

Anexos

Anexo 1: Especificación de casos de usos

Objetivo	Personalizar el entorno de trabajo
Actores	Usuario (inicia)
Resumen	El usuario podrá personalizar su entorno de trabajo cambiando el color de los paneles, botones, enlaces y barra de navegación.
Complejidad	Baja
Precondiciones	El usuario ha sido autenticado en el sistema correctamente.
Postcondiciones	Se personalizó el entorno de trabajo.
Flujo de eventos	
Flujo básico Personalizar el entorno de trabajo	

Actor	**Sistema**
1. El caso de uso inicia cuando el actor selecciona la opción *Opciones*.	
	2. Brinda la posibilidad de cambiar los colores de la apariencia entre azul, verde, rojo, amarillo, naranja, violeta, rosado, negro y predeterminado.
3. Selecciona un color.	
	4. Muestra el mensaje “¡Tema cambiado!” 5. Termina el caso de uso.

Tabla 5: Descripción del CU Personalizar entorno de trabajo

Objetivo	Exportar archivo
Actores	Usuario (inicia)
Resumen	El usuario podrá seleccionar el archivo específico que desea exportar.
Complejidad	Baja
Precondiciones	-El usuario ha sido autenticado en el sistema correctamente. -El archivo a exportar debe encontrarse en el portafolio.
Postcondiciones	Se exportó un archivo.
Flujo de eventos	
Flujo básico Exportar recurso	

Actor	**Sistema**
1. El caso de uso inicia cuando el actor selecciona la opción *Archivo*.	
	2. Muestra una interfaz que permite seleccionar el archivo que desea exportar.
3. Selecciona el archivo. 4. Selecciona el botón exportar.	
	5. Muestra el mensaje “Se ha exportado exitosamente”.

Tabla 6: Descripción del CU Exportar archivo

Objetivo	Enviar mensaje
Actores	Usuario (inicia)
Resumen	El usuario podrá enviar un mensaje a otro usuario.
Complejidad	Media

Precondiciones	-El usuario emisor ha sido autenticado en el sistema correctamente. -El usuario destinatario se encuentra registrado en el sistema.
Postcondiciones	Se envió un mensaje.
Flujo de eventos	
Flujo básico Enviar mensaje	

Actor	**Sistema**
1. Selecciona el botón *Mensajes* que se encuentra en el panel de navegación.	
	2. Muestra la interfaz de mensajes.
3. Selecciona Enviar mensaje	
	4. Muestra una interfaz para introducir los siguientes datos: • Asunto • Mensaje • Usuarios destinatarios
5. Introduce los datos requeridos. 6. Selecciona el botón *Enviar.*	
	7. Vuelve a la interfaz de mensajes. 8. Termina el caso de uso
Flujos alternos	
5. a El actor selecciona la opción Cancelar	

Actor	**Sistema**
5. a.1 Selecciona la opción *Cancelar.*	
	5. a. 2 Muestra la interfaz de Secciones

Tabla 7: Descripción del CU Enviar mensaje

Anexo 2: Diagrama de componentes

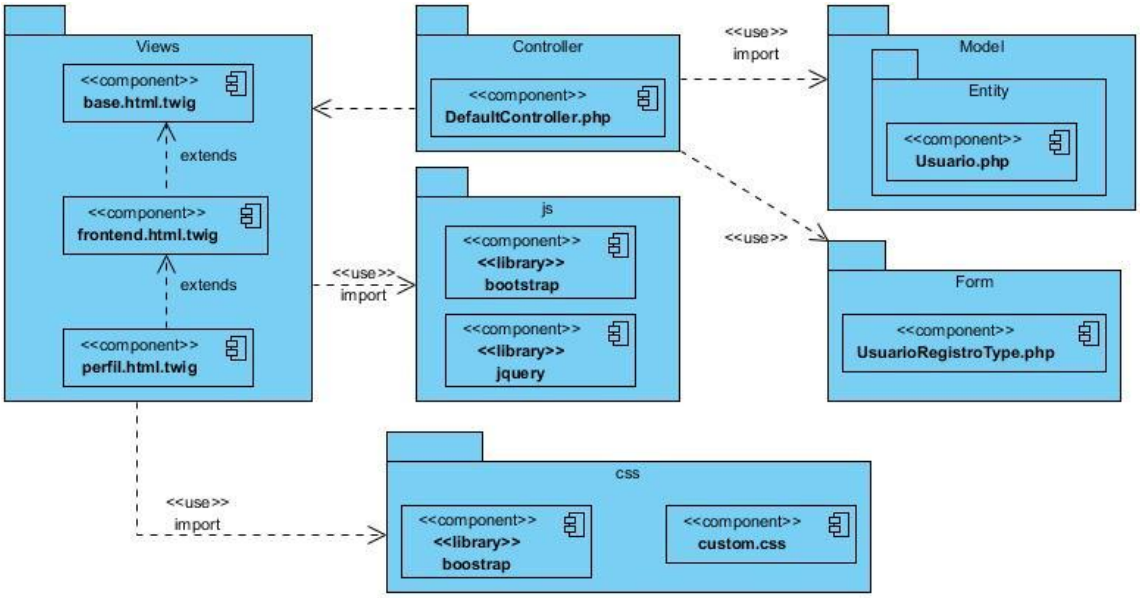


Ilustración 11: Diagrama de componentes. Paquete UsuarioBundle

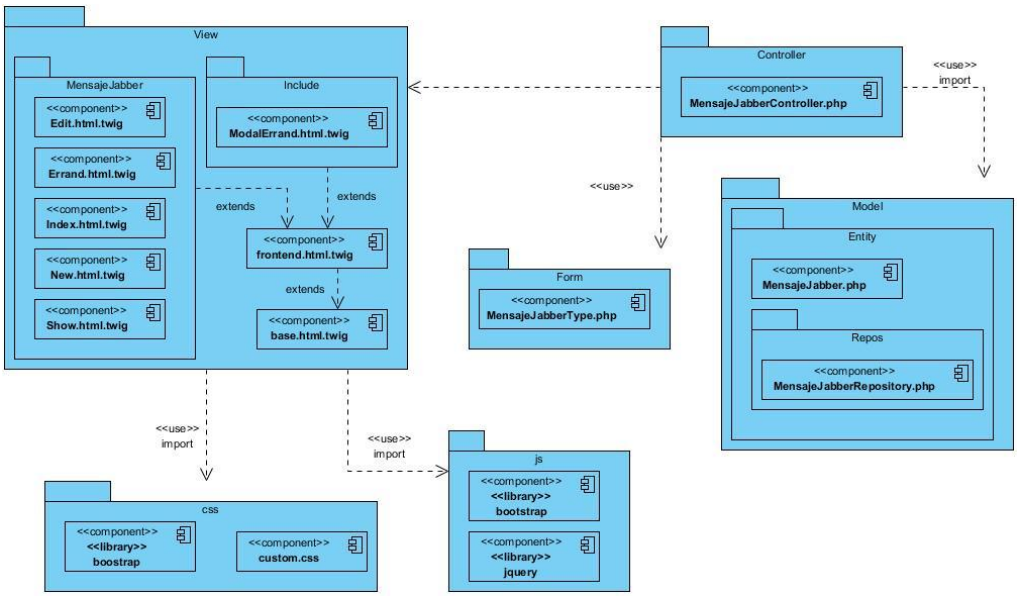


Ilustración 12: Diagrama de componentes. Paquete MensajeBundle

Anexo 3: Diagrama de clases

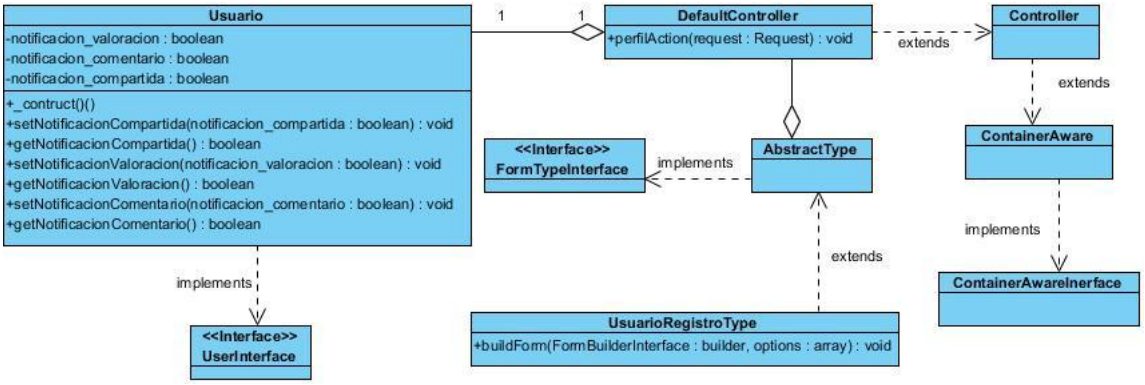


Ilustración 13: Diagrama de Clase UsuarioBundle.

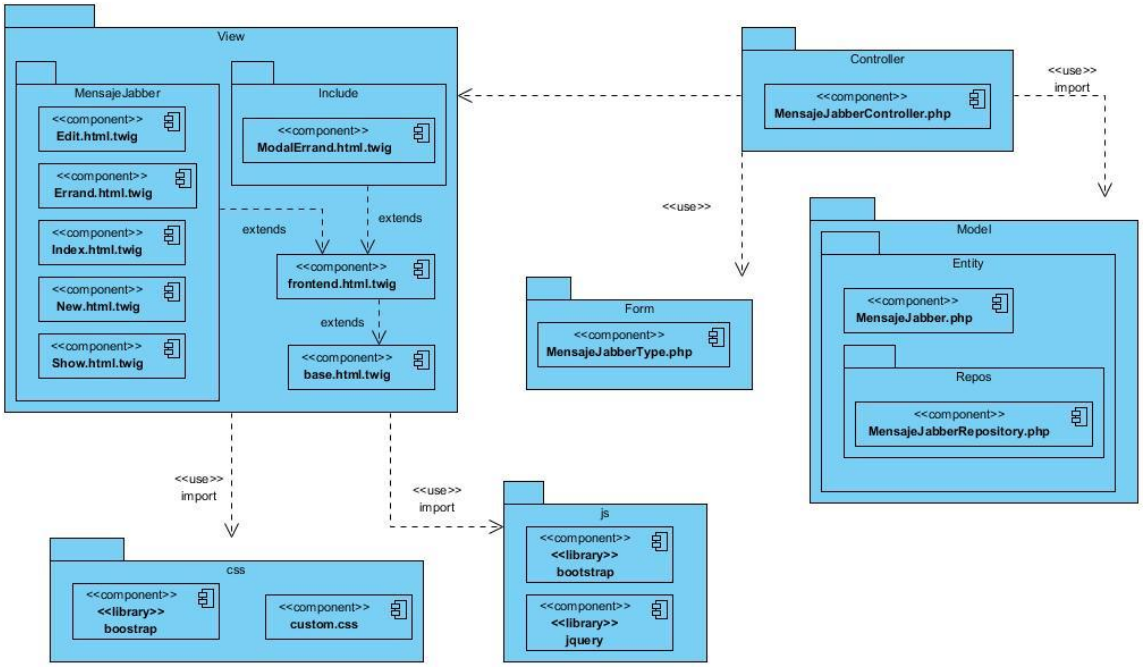


Ilustración 14: Diagrama de Clase MensajeBundle

Anexo 4: Pruebas unitarias

```php
public function testAll()
{
    $offset = 1;
    $limit = 2;

    $evidencias = $this->getEvidencias(2);
    $this->repository->expects($this->once())->method('findBy')
            ->with(array(), null, $limit, $offset)
            ->will($this->returnValue($evidencias));

    $this->pageHandler = $this->createEvidenciaHandler($this->om, static::EVIDENCIA_CLASS, $this->formFactory);

    $all = $this->pageHandler->getAllEvidencias($limit, $offset);

    $this->assertEquals($evidencias, $all);
}
```

Ilustración 15: Prueba unitaria-Obtener todas las evidencias

```php
public function testGet()
{
    $id = 1;
    $evidencia = $this->getEvidencia();
    $this->repository->expects($this->once())->method('find')
            ->with($this->equalTo($id))
            ->will($this->returnValue($evidencia));

    $this->evidenciaHandler = $this->createEvidenciaHandler($this->om, static::EVIDENCIA_CLASS, $this->formFactory);

    $this->evidenciaHandler->getEvidencia($id);
}
```

Ilustración 16: Prueba unitaria- Obtener evidencia según el id

```php
public function testValidarAsunto()
{
    $mensaje = new MensajeJabber();
    $mensaje->setRemitente('John Doe');
    $mensaje->setDestinatario(1);

    //asunto en blanco
    $listaErrores = $this->validator->validate($mensaje);
    $this->assertGreaterThan(0, $listaErrores->count(), 'Formato incorrecto.'
    );

    $error = $listaErrores[0];
    $this->assertRegExp("/This value should not be blank./", $error->getMessage());
    $this->assertEquals('asunto', $error->getPropertyPath());
}
```

Ilustración 17: Prueba unitaria-Validar asunto

```php
public function testValidarRemitente()
{
    $mensaje = new MensajeJabber();

    //remitente en blanco
    $listaErrores = $this->validator->validate($mensaje);
    $this->assertGreaterThan(0, $listaErrores->count(), 'Formato incorrecto.'
    );

    $error = $listaErrores[0];
    $this->assertRegExp("/This value should not be blank./", $error->getMessage());
    $this->assertEquals('remitente', $error->getPropertyPath());
}
```

Ilustración 18: Prueba unitaria-Validar remitente

```php
public function testValidarDestinatario()
{
    $mensaje = new MensajeJabber();
    $mensaje->setRemitente('John Doe');

    //destinatario en blanco
    $listaErrores = $this->validator->validate($mensaje);
    $this->assertGreaterThan(0, $listaErrores->count(), 'Formato incorrecto.'
    );

    $error = $listaErrores[0];
    $this->assertRegExp("/This value should not be blank./", $error->getMessage());
    $this->assertEquals('destinatario', $error->getPropertyPath());

    //destinatario no numero
    $mensaje->setDestinatario('badone');
    $listaErrores = $this->validator->validate($mensaje);
    $this->assertGreaterThan(0, $listaErrores->count(), 'Formato incorrecto.'
    );

    $error = $listaErrores[0];
    $this->assertRegExp("/This value should be of type/", $error->getMessage());
    $this->assertEquals('destinatario', $error->getPropertyPath());
}
```

Ilustración 19: Prueba unitaria- Validar destinatario