Universidad de las Ciencias Informáticas

Facultad3

Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

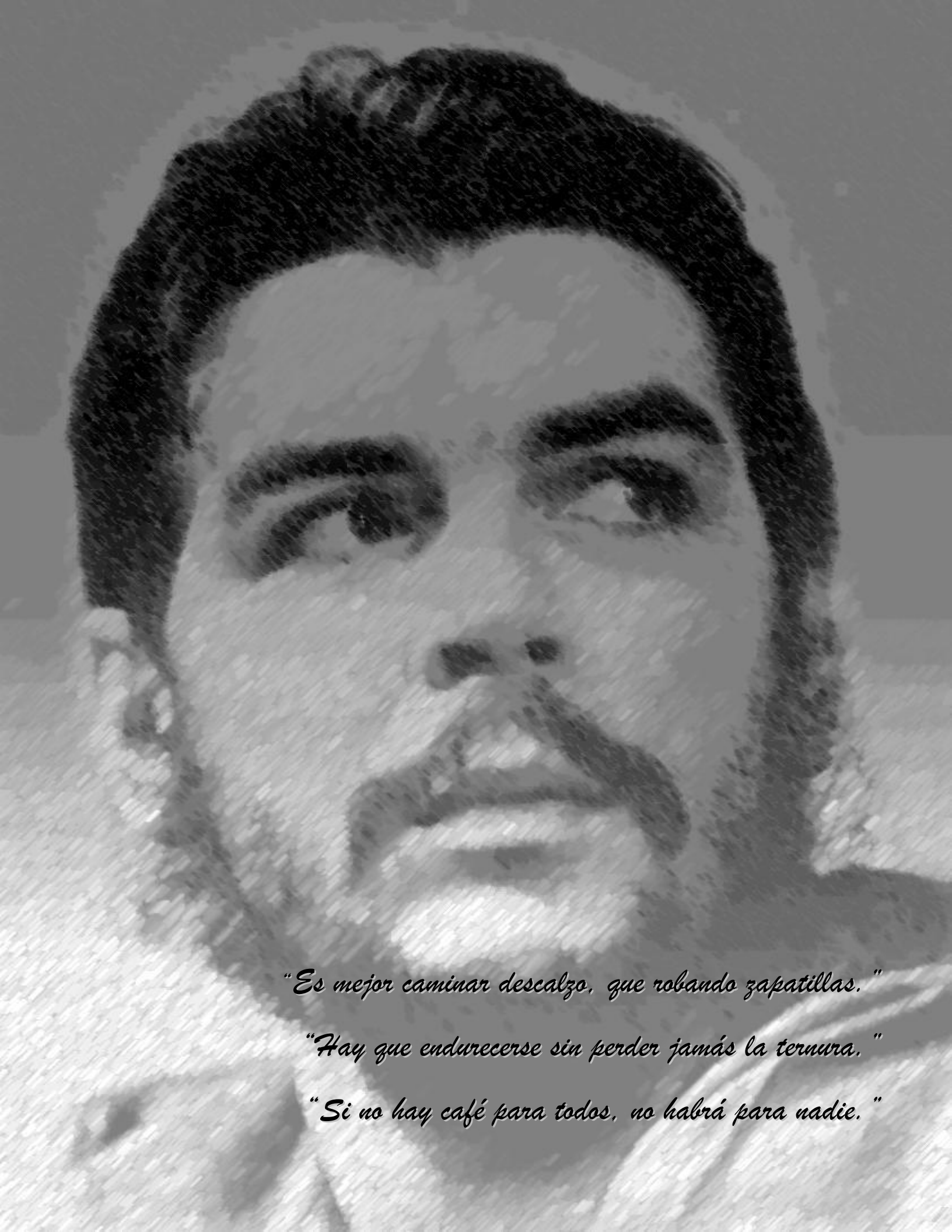
Autores: Carlos Rafael Madrigal García

Frank Carlos García Lorenzo

Tutores: M Sc. Yoan Martínez Márquez

Ing. Luis Manuel Borges Rivero

Junio 2014
La Habana



"Es mejor caminar descalzo, que robando zapatillas."

"Hay que endurecerse sin perder jamás la ternura."

"Si no hay café para todos, no habrá para nadie."

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____días del mes de _____del año _____.

Carlos Rafael Madrigal García

_____
Firma del Autor

Frank Carlos García Lorenzo

_____
Firma del Autor

M Sc. Yoan Martínez Márquez

_____
Firma del tutor

Ing. Luis Manuel Borges Rivero

_____
Firma del tutor

Agradecimientos

Carlos Rafael Madrigal García

Para llevar a cabo mi sueño fueron muchas las personas que influyeron a lo largo de la carrera.

Quiero agradecer en primer lugar a mi querida mamá, que es lo más grande y lindo que me ha dado la vida, gracias por ayudarme a llegar hasta aquí y convertirme en el hombre que soy hoy.

Gracias a mi papá por ser la imagen y el ejemplo a seguir para llevar a cabo todas mis metas.

Gracias a mi segunda mamá Eida, porque a pesar de ser mi tía, has sabido comportarte como una madre para mí. Eres amor y sabiduría.

A mi hermanita Liz porque aunque te encuentres lejos siempre has estado presente para ayudarme.

Mil gracias a mis tutores por su paciencia y dedicación y por hacer de mí una persona mejor.

Al tribunal por señalamientos oportunos que contribuyeron a alcanzar el resultado que aquí se muestra.

Le agradezco a mi compañero de tesis por saber aguantarme, con el cual pasé días y noches trabajando juntos para poder alcanzar este resultado.

Gracias a todos mis amigos, en especial a Alberto y al Fide por todas sus bromas.

Gracias a Fidel y a la Revolución por brindarme esta oportunidad.

A todas aquellas personas que de alguna forma u otra hicieron posible este momento, mil gracias.

Frank Carlos García Lorenzo

Quiero agradecer ante todo a mi familia, en especial a mis padres, sin ellos el sueño de ser ingeniero jamás podría haber sido posible.

A mi tía Odalis y mi prima Lisandra por su apoyo incondicional.

A mi tía Blanquita y a mi tío Gari por los buenos consejos.

A Blanqui y a Diani por toda la ayuda este último año.

A mi compañero de tesis por aguantarme todo este tiempo, Charles sin ti no hubiera sido posible este resultado.

Agradecer a mis tutores, Yoan tu ejemplo y tu dedicación ante el deber es admirable.

A mis amigos del café y a todas las personas que hicieron posible la culminación de esta tesis…

Dedicatoria

Carlos Rafael Madrigal García

Dedico el presente trabajo de culminación de la carrera:

A mi mamá, por ser la persona más especial que hay en mi vida.

A mi papá, por ser el ejemplo a seguir y el motor impulsor para lograr terminar la carrera.

A mi segunda mamá Eida, porque aunque seas mi tía en estos 5 años te has portado como una madre para mí.

A mi abuelita Fela, por ser la voz de la sabiduría que me ha guiado en la realización la misma.

A mis hermanos Liz y José por estar siempre ahí cuando los necesité.

A mi madrastra Eida, por ayudarme y quererme como el hijo que no tuvo.

A mis sobrinitas Ivette y Helen, por ser las luces que iluminan mi vida.

A mi familia, en especial mis tías Nélida, Julia, Eivet, Erel y Eva.

A mis amigos….

A todas esas personas que hicieron posible este momento.

Frank Carlos García Lorenzo

A mi mamá

A mi papá

A mi familia

A Emily

A mí

Resumen

La certificación de roles es un proceso complejo basado en el procesamiento de evidencias, consistentes en artefactos generados por los estudiantes a partir su actividad en las asignaturas de la disciplina de Producción, Investigación y Desarrollo (PID). Estos artefactos constituyen entregables de las diferentes tareas que se generan en el desarrollo de software y que se contemplan en el plan de formación según establecen los objetivos de la disciplina. La cantidad de estudiantes que optan por certificar un rol disminuyen considerablemente cada curso escolar, que a pesar de ser voluntario es de gran importancia pues permite constatar las competencias desarrolladas por el estudiante durante su desempeño en la actividad productiva. Existen un grupo de factores que atentan contra este proceso: la carencia de evidencias que avalen las tareas realizadas, así como lo engorroso del proceso de generación, clasificación, almacenamiento y revisión de las mismas. En el presente trabajo se lleva a cabo el desarrollo de un portafolio para la gestión de evidencias del proceso de certificación de roles en la Universidad de las Ciencias Informáticas. El mismo persigue como objetivo contribuir a la generación, almacenamiento, clasificación y revisión de las evidencias elaboradas por los estudiantes en su labor docente-productiva y a la toma de decisiones sobre la correspondencia del rol a certificar con las competencias y habilidades, que a pesar de ser voluntario es de gran importancia pues permite constatar las competencias desarrolladas por el estudiante durante su desempeño en la actividad productiva. Existen los artefactos aportados.

Palabras claves: evidencias, competencias profesionales, portafolio electrónico.

Índice de Contenidos

Introducción.....1

CAPÍTULO I: Fundamentos de un Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles. 6

Epígrafe 1.1. Los referentes teóricos de los sistemas de gestión académica basados en evidencias.....9

Epígrafe 1.2. Sistemas similares para la gestión académica basados en evidencias. 11

Epígrafe 1.3.El proceso de certificación de roles en la Universidad de las Ciencias Informáticas. 13

Epígrafe 1.4. La selección de la metodología, las herramientas y las tecnologías a utilizar para el desarrollo del sistema para gestionar la certificación de roles. 15

CAPÍTULO II: Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles. 31

Epígrafe 2.1. Características del sistema 31

Epígrafe 2.2 Planificación 35

Epígrafe 2.3 Diseño de la solución. 42

Epígrafe 2.4. Codificación de la solución 49

CAPÍTULO III: Valoración de la efectividad del Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles. 56

Epígrafe 3.1. Validación de los requisitos 56

Epígrafe 3.2. Validación del diseño 57

Epígrafe 3.3. Las pruebas realizadas al sistema 59

Conclusiones Generales 66

Recomendaciones 67

Bibliografía ..... **¡Error! Marcador no definido.**

Índice de Ilustraciones

Ilustración 1 Flujo básico del sistema 32

Ilustración 3 Arquitectura del sistema 43

Ilustración 4 Patrón MVC 44

Ilustración 5 Estructura de Clases 45

Ilustración 6 Estructura de Carpetas 54

Ilustración 7 Ejemplo de clase gestora 54

Ilustración 8 Ejemplo de llamar a una función 54

Ilustración 9 Resultado de la aplicación de la métrica TOC 57

Ilustración 10Resultados de la aplicación de la métrica RC 58

Índice de Tablas

Tabla 1 Cronograma del Proceso de certificación en la UCI 14
Tabla 2 Metodologías ágiles vs tradicionales 16
Tabla 3 Comparación entre Metodologías Ágiles 19
Tabla 4 Comparación entre Marcos de Trabajo 22
Tabla 5 Comparación entre Entornos de Desarrollo 23
Tabla 6 Comparación entre Gestores de Bases de Datos 28
Tabla 7 Comparación entre Patrones Arquitectónicos 29
Tabla 8 Usuarios relacionados con el sistema y su descripción 34
Tabla 9 Desgloso de HU 38
Tabla 11 Estimación de esfuerzos por HU 40
Tabla 12 Distribución del Plan de Iteraciones 42
Tabla 13 Tarjeta CRC de TareaController 49
Tabla 14 HU Alojar evidencias 50
Tabla 15 HU Eliminar evidencia 51
Tabla 16 HU Visualizar evidencia 51
Tabla 17 HU Listar evidencias 52
Tabla 18 HU Aprobar evidencias 52
Tabla 19 HU Rechazar evidencia 53
Tabla 20 PA Autenticar usuario 61
Tabla 21 PA Importar plan de formación 61
Tabla 22PA Registrar tarea 62
Tabla 23 PA Alojar artefactos 63
Tabla 24 Resultado de la aplicación de las pruebas unitarias 65

Introducción

En la actualidad, fenómenos objetivos tales como: la globalización, el intercambio científico, el desarrollo vertiginoso de la informática con nuevos espacios de socialización y nueva división del trabajo profesional, asociados a una demandante necesidad de desarrollo social; influyen en el propósito por el que se aprovecha la informática, un tema apasionante en todos los sentidos, que hace soñar sobre el futuro y de las nuevas posibilidades de desarrollo individual y de aprendizaje con la inserción de la computadora; referirse a la informática es hacer mención a la educación, la productividad y el desarrollo.

La informática ha extendido su aplicación mediante sistemas que gestionan la información en diferentes procesos. Ha dado solución a problemas de las más disímiles áreas del conocimiento, la producción y los servicios como el control de los stocks y de la producción, y del tráfico en aeropuertos y estaciones de trenes. Por otra parte, ha ido cobrando gran importancia en muchos ámbitos de la vida, incluso en la vida cotidiana de las personas, especialmente, en su acceso a diferentes servicios e información a través del desarrollo de Internet.

Cuba se encuentra inmersa en un proceso de transformaciones tecnológicas, para lo cual se han creado diversas instituciones a nivel nacional que favorecen el crecimiento del país en esta esfera. Una de estas instituciones es la Universidad de las Ciencias Informáticas (UCI), universidad nueva en su tipo, fundada por el Comandante en Jefe Fidel Castro en el año 2002, con el objetivo de formar jóvenes comprometidos con la Revolución y capaces de llevar a cabo el proceso de informatización de la sociedad cubana. Posee una infraestructura productiva formada por diferentes centros de desarrollo de software que brindan soluciones informáticas a diferentes problemas existentes en disímiles esferas de la sociedad. En estos centros los estudiantes juegan un papel fundamental a través del desempeño de diferentes roles[1] productivos desde las asignaturas de la disciplina de PID. A lo largo de la carrera los estudiantes deben desarrollar un sistema de habilidades y competencias establecidas dentro del plan de formación elaborado por el profesor de la asignatura. El plan de formación está compuesto por macro-tareas que se subdividen en tareas, a las cuales deben dar solución mediante la generación de artefactos que servirán de evidencia en caso de que opten por certificar algún rol.

[1] función que alguien cumple en determinado contexto, una responsabilidad profesional, conducta esperada según el nivel social y cultural (wordreference, 2014).

Introducción

La certificación de roles se basa en el procesamiento de artefactos generados por los estudiantes en su labor en las asignaturas de la disciplina PID y que constituyen evidencias de su desempeño en el desarrollo de software. Las evidencias que un estudiante recopila a través del proceso constituyen indicadores que forman parte de los Escenarios de Competencias y son almacenados en un expediente elaborado en una hoja de cálculo. Este expediente recoge además los datos del estudiante, el listado de las tareas de las cuales se tienen evidencias y un grupo de evaluaciones por cada rol a certificar. Esta concepción del expediente constituye un inconveniente para el almacenamiento de las evidencias.

Actualmente este proceso se realiza con el apoyo de una hoja de cálculo en que se involucran: el profesor de PID, el tutor de PID, el estudiante, el subdirector de formación y el tribunal de certificación. En la misma se establecen las competencias, habilidades, actividades y demás elementos a tener en cuenta para cada rol a certificar. La cantidad de actores involucrados en este proceso y el cúmulo de información a gestionar con solo el apoyo de una hoja de cálculo, constituyen limitaciones para su desarrollo en tiempo y con la calidad requerida. Además, el proceso resulta engorroso por lo complejo de la generación, la clasificación, el almacenamiento y la revisión de las evidencias aportadas en la realización de cada una de las tareas del plan de formación.

Al finalizar un semestre, los estudiantes pueden solicitar al subdirector de formación del centro al que se encuentran vinculados, la posibilidad de certificar los roles para los cuales ha logrado acumular las evidencias necesarias, presentando el expediente con el cúmulo de evidencias que soporten las tareas realizadas y las competencias a las que tributan con sus evaluaciones y un comprimido con los artefactos generados en las mismas. Una vez que el subdirector de formación del centro tenga el listado de solicitudes, debe analizar y decidir cuáles de estos estudiantes pueden presentarse ante el tribunal de certificación, teniendo en cuenta los requisitos necesarios para solicitar la certificación.

Los estudiantes deben presentar las evidencias del trabajo en el rol ante el tribunal de certificación. Se debe realizar una presentación, donde se expongan las actividades y tareas desarrolladas asociadas al desempeño del rol que desea certificar. Además será objeto de evaluación la preparación del estudiante para responder preguntas relacionadas con los elementos teóricos, las herramientas, los métodos, los procedimientos empleados, los resultados y la novedad de las soluciones obtenidas. Una vez culminado el ejercicio, el tribunal decidirá si otorga o no la certificación al estudiante y nivel alcanzado.

En este contexto se identifica como **situación problemática** los siguientes elementos que limitan la realización de la certificación de roles:

- La cantidad de actores involucrados y el cúmulo de información a gestionar con solo el apoyo de una hoja de cálculo dificultan el desarrollo de este proceso en tiempo y con la calidad requerida.
- La tipología diversa de los artefactos y su ponderación manual eleva la complejidad de la certificación de roles.
- Lo engorroso del proceso de generación, clasificación, almacenamiento y revisión de las evidencias, ya que están condicionadas por diversos factores de acuerdo a su tipología y ponderación en el documento que regula la certificación de roles.
- La carencia de las evidencias, consistentes en artefactos, que sustenten las tareas realizadas.

Teniendo en cuenta la problemática planteada anteriormente se define como **problema a resolver**: ¿Cómo gestionar el proceso de certificación de roles de manera que contribuya a la generación, clasificación, almacenamiento y revisión de las evidencias de las tareas de los estudiantes en las asignaturas de la disciplina de Producción Investigación y Desarrollo?

El **objeto de estudio** consiste en el desarrollo de software en ámbitos académicos; enmarcado en el desarrollo de sistemas de gestión académica basados en evidencias como **campo de acción**.

El compromiso de los investigadores se manifiesta en plantear como **objetivo general**: desarrollar un sistema para gestionar la certificación de roles que contribuya a la generación, clasificación, almacenamiento y revisión de las evidencias de las tareas de los estudiantes en las asignaturas de la disciplina de Producción Investigación y Desarrollo.

Para su consecución se definen los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación mediante el estudio de los referentes teóricos de la gestión académica basada en evidencias, la metodología de desarrollo, las técnicas y herramientas de software.
2. Realizar el análisis y diseño del Sistema para la Gestión de la Certificación de Roles, mediante la metodología de desarrollo de software definida.
3. Implementar el Sistema para la Gestión de la Certificación de Roles, mediante las herramientas y tecnologías seleccionadas.
4. Valorar la efectividad del Sistema para la Gestión de la Certificación de Roles de los

estudiantes en las asignaturas de la disciplina de Producción Investigación y Desarrollo; mediante la realización de pruebas de software, la obtención de criterios de los clientes y la liberación por el grupo de calidad.

Para el cumplimiento del objetivo se tienen en cuenta las siguientes **tareas**:

1. Establecimiento de los referentes teóricos de los sistemas de gestión académica basados en evidencias.
2. Determinación de la metodología, herramientas y tecnologías a utilizar para el desarrollo del sistema para gestionar la certificación de roles.
3. Realización del análisis y el diseño del sistema.
4. Obtención del modelo de implementación del Sistema para la Gestión de la Certificación de Roles.
5. Ejecución de pruebas unitarias al código fuente obtenido en la implementación del Sistema para la Gestión de la Certificación de Roles.
6. Valoración del Sistema para la Gestión de la Certificación de Roles mediante la ejecución de pruebas de aceptación para comprobar su correcto funcionamiento.

En la presente investigación se define como **idea a defender** que con el desarrollo de un sistema para gestionar la certificación de roles se contribuirá a la generación, clasificación, almacenamiento y revisión de las evidencias de las tareas de los estudiantes en las asignaturas de la disciplina de Producción Investigación y Desarrollo.

Como **métodos científicos de la investigación** se emplearon los que a continuación se describen, propuestos en (Hernández,R. y Coello,S. 2011):

Métodos teóricos:

- ✓ Método analítico-sintético: con el uso de este método se realizó un análisis de los documentos, las publicaciones, bibliografías y en general toda la información relacionada con la certificación de roles y el empleo de sistemas de gestión académica basados en evidencias. Se realizó un estudio sobre las tendencias actuales, posibilitando hacer una síntesis de estas principales fuentes y llegar a conclusiones parciales sobre el tema en estudio.

- ✓ Modelación: Este método se encarga de realizar una reproducción simplificada de la realidad, abstracciones, con el objetivo de explicarla y comprender mejor los procesos

del negocio de la presente investigación. Es uno de los más importantes en el campo de la construcción de software y se utiliza en el diseño de modelos de datos, diagrama de clases, de componentes y otros.

Métodos empíricos:

- ✓ Entrevista: este método se utilizó en encuentros sostenidos con la Jefa de Departamento Docente de Ingeniería de Software de la Facultad 3 y con la asesora de las asignaturas de la disciplina de Producción, Investigación y Desarrollo del Departamento Metodológico Central. Se persiguió como objetivo recopilar información referente al proceso de certificación de roles en la Universidad de las Ciencias Informáticas que contribuyera al entendimiento de este negocio, así como a la validación de las funcionalidades identificadas para el desarrollo de la solución.

Estructura del documento:

El presente trabajo consta de tres capítulos, que se describen a continuación:

CAPÍTULO 1. Fundamentos de un Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles: en este capítulo se abordan los aspectos teóricos relacionados con la investigación tales como: sistemas similares en la gestión académica basados en evidencias, metodología de desarrollo de software, patrón arquitectónico, marco de trabajo, tecnologías, lenguajes y herramientas a emplear en la construcción de la solución.

CAPÍTULO 2. Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles: en este capítulo, se presenta el diseño del software a través de los artefactos propuestos en las fases de planificación, diseño y codificación de la metodología de desarrollo de software seleccionada. Se identifican y organizan las clases relevantes para las funcionalidades del sistema, así como los patrones arquitectónicos y de diseño utilizados para la realización del Portafolio.

CAPÍTULO 3. Valoración de la efectividad del Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles: en este capítulo se abordan las pruebas realizadas a cada uno de los módulos de la solución. Se muestran los resultados de la aplicación de las técnicas y métricas para validar los requisitos, así como los resultados arrojados después de haber realizado las pruebas de caja negra y caja blanca.

CAPÍTULO I: Fundamentos de un Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles

En el presente capítulo se lleva a cabo la descripción de los conceptos fundamentales que sustentan la investigación y que pueden resultar difíciles de comprender. Se realiza un estudio de diferentes portafolios electrónicos existentes en el mundo y en Cuba para determinar las características, procesos y funcionales asociados a los mismos. También, se hace un análisis de las tecnologías actuales y las principales herramientas que pudieran adecuarse en la construcción del sistema a desarrollar.

La siguiente investigación se encuentra enmarcada en un contexto estudiantil, lo cual trae consigo el uso de procedimientos de acuerdo a las políticas y reglamentos de la universidad. El rendimiento de los estudiantes se mide a través de evaluaciones de desempeño en distintas actividades de la vida estudiantil. Un estudiante universitario en los cinco años de su carrera promedia un cumulo bastante significativo de evaluaciones, las cuales puede llegar a perder en caso de no tener un sistema el cual se encargue de gestionar las evidencias de su desempeño como estudiante en su labor en PID.

La presencia de portafolios en centros educativos, empresas y administraciones ha tenido un gran impacto en los últimos años, creando diferentes prácticas que responden también a objetivos diversos.

Basado en su uso genérico (Barberá, 2005) define un portafolio como "*un instrumento que tiene como objetivo común la selección de muestras de trabajo o evidencias de consecución de objetivos personales o profesionales que, ordenados y presentados de un determinado modo, cumplen la función de potenciar la reflexión sobre cada una de las prácticas (educativas, profesionales o civiles)"*.

De acuerdo con (López, 2002) este puede llegar a contener casi cualquier cosa. Su formato puede ser físico (por lo general a través de una carpeta con argollas), electrónico, digital (a través de archivos en una computadora personal (PC)), o en línea (utilizando los recursos que en la actualidad ofrecen las páginas web).

Los autores del trabajo, después de haber analizado las fuentes anteriores, son partidarios de su coincidencia con la esencia de la presente investigación para la definición de un portafolio de manera genérica. En este sentido el principal objetivo de un portafolio está asociado a cuatro puntos, independientemente del tipo de portafolio del que se trate o del formato de presentación:

1. Almacenamiento de trabajos
2. Evaluación de contenidos
3. Proceso de interacción entre el autor y otras personas
4. Objeto de aprendizaje

El portafolio se puede entender como un sistema de evaluación integrado en el proceso de enseñanza y aprendizaje. Consiste en una selección de evidencias/muestras que conforman una carpeta 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 (certificados acreditativos, fragmentos de películas, entrevistas, actividades académicas, apuntes, trabajos de asignaturas, entre otras) permiten al alumno demostrar que está aprendiendo y a la vez posibilitan al profesor un seguimiento del progreso de este aprendizaje. Esta cualidad de reflexión constante sobre el propio aprendizaje convierte el portafolio en un sistema de evaluación coherente en el marco de la evaluación continuada y formativa. (Cesar Coll, 2004)
Los tipos de portafolio según la Universidad de Puebla (UDLAP, 2010) y la posición que defienden los autores son:

- ✓ **Showcase (vitrina):** Contiene evidencia limitada. Es útil cuando se tienen evidencias físicas que no pueden traducirse en un documento, por ejemplo si se trata de desarrollar prototipos físicos. No está exento de que incluya una evidencia documental como complemento de la evidencia física. Por ejemplo si se trata de un prototipo físico, puede incluir un manual de usuario, describir un procedimiento, etc.

- ✓ **Checklist (lista de chequeo):** es el más utilizado en la educación y se conforma por un número predeterminado de evidencias (el alumno debe saber desde el principio del curso, cuáles serán las evidencias que se deberán entregar.

- ✓ **Portafolio electrónico:** es una colección de materiales digitalizados que incluyen demostraciones, recursos y logros que representan a un individuo, un grupo o una institución. Puede comprender texto, gráficos o elementos multimedia que se pueden

consultar en Internet o en otro medio electrónico como un CD-ROM o un DVD. Es más que una simple colección, también sirve como herramienta para administrar, organizar y controlar los trabajos creados con diferentes aplicaciones. Incluyen reflexión personal y por lo regular promueven el intercambio de ideas y retroalimentación.

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) aportan muchas potencialidades a la hora de diseñar o elaborar un portafolio. Ayudan en la elaboración de las evidencias que conforman el portafolio y también actúan como plataforma base del proceso de enseñanza y aprendizaje.

En las últimas décadas la incorporación de las TIC a la enseñanza mediante entornos de enseñanza-aprendizaje, han favorecido la proliferación de experiencias y estudios centrados en los portafolios electrónicos en la educación superior. Sin embargo, estos trabajos se enmarcan básicamente en el uso de este sistema de evaluación, en la estructura que tiene que tener, y a menudo se deja de lado la reflexión en qué y cómo va aprendiendo, al mismo tiempo que le permiten regular su proceso de aprendizaje y de estudio. (Cesar Coll, 2004)

(Galloway, 2001) hace referencia al uso de un portafolio enunciando lo que considera son sus desventajas y ventajas, posición compartida por los autores de la presente investigación.

Las principales desventajas son:

- ✓ Debe de formar parte de la cultura institucional para ser considerado como una herramienta valiosa.
- ✓ El nivel de alfabetización tecnológica no siempre es el idóneo, sobre todo entre profesores, más que entre alumnos.
- ✓ Los docentes con poca experiencia en el uso y desarrollo del portafolio pueden considerarlo como algo demasiado complejo para ser llevado a cabo.

Las principales ventajas son:

- Es más fácil el almacenamiento de información ya que su espacio ha dejado de ser físico y ha pasado al mundo de la virtualidad.
- Su distribución es fácil, barata y de reproducción inmediata, tanto para profesores como para estudiantes.
- Su elaboración es permanente, con posibilidad de reeditarse o ampliarse cuando se necesite.

- La forma de compartirlo es muy amplia, ya que se puede hacer llegar a cualquier persona en cualquier lugar.

La utilización de los portafolios electrónicos en el proceso estudiantil ha demostrado satisfacer las necesidades para las que han sido creados. Los sistemas basados en evidencias constituyen una muestra de las aplicaciones de un portafolio electrónico como sistema de evaluación integrada; en el siguiente epígrafe se especifican los conceptos y se profundiza en el estudio de los mismos.

Epígrafe 1.1. Los referentes teóricos de los sistemas de gestión académica basados en evidencias.

Se considera necesario comenzar por abordar los elementos fundamentales que contribuyen a un acercamiento a los elementos fundamentales de la definición de un portafolio. En este sentido se considera como" *una recopilación de evidencias (documentos, artículos, notas, diarios, trabajos, ensayos, etc.) consideradas de interés para ser conservadas, debido a los significados que con ellas se han construido"* (Universidad Autónoma de Guadalajara, 2010).

Una vez establecido lo que se entiende por portafolio se procede a establecer lo que se entiende por portafolio de evidencias a los efectos de la presente investigación. El mismo se considera como "*una colección de documentos con ciertas características que tienen como propósito evaluar el nivel de aprendizaje que se ha adquirido, es decir, sus logros, esfuerzos y transformaciones a lo largo de un curso. Implica toda una metodología de trabajo y de estrategias didácticas en la interacción entre el profesor y el alumno, y por otro lado es un método de evaluación que permite unir y coordinar un conjunto de evidencias para emitir una valoración más apegada a la realidad"* (Roa, 2010).

Este se conforma para recopilar información en donde se manifiesten los avances del aprendizaje, actitudinales y procedimentales de los alumnos. Además de permitir al alumno participar en la evaluación de su propio desempeño y que el profesor pueda llevar un registro objetivo y documentado de los avances de sus alumnos (Roa, 2010).

Los portafolios de evidencias son usados comúnmente para acreditar en cierta medida la realización de un evento educativo, la evaluación de ciertas habilidades creadas dentro del proceso de aprendizaje y formación del estudiante, y/o en cualquier situación en donde se necesite evaluar el desempeño de algún individuo o la necesidad de determinar conclusiones a partir de un grupo de evidencias. Un portafolio de evidencias con fines educativos debe ser

capaz de llevar una memoria técnica del proceso de formación, el mismo plan de formación del estudiante, un directorio de participantes, definición de población o conjuntos de individuos involucrados en el proceso y el rol que desempeñan en este. Según (Roa, 2010) pudiera incluir:

- ✓ el perfil de los involucrados,
- ✓ el formato de evaluación y la acreditación del mismo por los involucrados,
- ✓ el conocimiento del nivel de aprendizaje resultado del esfuerzo,
- ✓ las evaluaciones frecuentes y las evidencias del desempeño del estudiante a lo largo de un determinado periodo de tiempo,
- ✓ los instrumentos de evaluación diagnóstica, intermedia y final, aplicados a los participantes y acordes a la estrategia y criterios de evaluación plasmados en el plan de formación,
- ✓ la posibilidad de guardar, eliminar y modificar cada una de las evidencias por sobre todas las cosas,
- ✓ la consulta de cada una de ellas y de realizar reportes los cuales podrán tomarse como evidencias para otras actividades relacionadas o no con el proceso educativo en el que están involucrados.

Las evidencias tienen que acompañarse de la justificación y la reflexión del estudiante, en que pongan de manifiesto la relación entre la evidencia y el aprendizaje. Estas contribuciones le ayudarán a tomar consciencia de qué y cómo va aprendiendo, al mismo tiempo que le permitirá regular su proceso de aprendizaje y de estudio. Esta cualidad de reflexión constante sobre el propio aprendizaje convierte el portafolio en un sistema de evaluación coherente en el marco de la evaluación continuada y formativa (Cesar Coll, 2004).

Según (Cesar Coll, 2004) las evidencias pueden clasificarse en:

- ✓ **Manuscritas:** son elaboradas a mano, pueden realizarse en el aula (resumen, descripción, mapas mentales, etc.) o como parte de alguna tarea.
- ✓ **Evidencias digitales:** videos, audios, simulaciones, software, fragmentos de código etc.
- ✓ **Evidencias impresas:** investigaciones documentales, definiciones, fotocopias o cualquier documento solicitado por computadora.
- ✓ **Evidencias físicas:** prototipos, objetos físicos, etc.

Se puede resumir que la presente investigación se centra en los portafolios de evidencias digitales para darle solución a la problemática planteada. Por lo tanto se procede a realizar el

estudio de los sistemas similares con posibilidades de aportar al desarrollo de la propuesta de solución

Epígrafe 1.2. Sistemas similares para la gestión académica basados en evidencias.

Para el desarrollo de la investigación fue necesario realizar un estudio minucioso de los sistemas informáticos existentes tanto en Cuba como en el resto del mundo. A continuación se presenta algunos de los sistemas informáticos estudiados que dan muestra de la utilización de los portafolios digitales para el proceso de enseñanza y aprendizaje.

Se comienza el análisis de los sistemas similares en el marco internacional entre los cuales destacan los siguientes sistemas informáticos de referencia:

1- **Portafolio electrónico: desarrollo de competencias profesionales en la red Universidad de Barcelona (UAB) España:** expone el sentido y las características de un portafolio electrónico aplicado en un contexto universitario de enseñanza en línea. El portafolio que se presenta pone su énfasis en el seguimiento de los trabajos de los estudiantes basándose en el progreso continuo de las competencias profesionales que se han desarrollado a lo largo de los estudios de psicopedagogía. El avance de las competencias se expone mediante la publicación de evidencias que documentan el nivel de progreso de las citadas competencias. El portafolio electrónico que se plantea es de carácter individual y está sostenido por un mecanismo de apoyo por parte del profesor donde se aporta retroalimentación continuada a los estudiantes, lo que facilita el ajuste de sus actuaciones a las competencias profesionales planteadas. Es capaz de vincular a un profesor y a un estudiante facilitando la retroalimentación, pero disminuyendo la capacidad de involucrar a más individuos. Está diseñado para estudiantes de la carrera de psicopedagogía que tiene un objeto de la profesión diferente al de la carrera en ciencias informáticas, descartando la posibilidad de usarlo para la resolución de la problemática planteada por la tipología y el contexto de las tareas que desempeñan (Barberá,E y otros. 2006).

2- **El desarrollo de la práctica reflexiva sobre el quehacer docente, apoyada en el uso de un portafolio digital, en el marco de un programa de formación para académicos de la Universidad Centroamericana de Nicaragua:** analiza el nivel de la calidad reflexiva de los profesores universitarios en el marco de la experiencia de la reflexión docente y la mejora de sus prácticas educativas. Se trata de una investigación

cualitativa sobre formación docente utilizando la metodología del estudio de casos y donde se pretende contribuir a la mejora e innovación de la docencia universitaria. Se introduce el uso del portafolio digital, herramienta para el registro de evidencias de un proceso de desarrollo de aprendizaje y que puede facilitar la reflexión sistemática acerca de la práctica docente. Los resultados obtenidos muestran que el proceso reflexivo, con unas directrices bien definidas, contribuyó a una percepción positiva sobre la mejora de la docencia, la planificación de clases y de la interacción con los alumnos. El estudio parte del interés de valorar la práctica reflexiva de los profesores universitarios, mediada y gestionada a través de un portafolio digital, en el ámbito de la formación docente, imposibilitando el uso de esta práctica para involucrar a más de un individuo en el proceso. Este último aspecto limita su uso en el contexto de la presente investigación (Rodrigues,R. 2013).

3- **Portafolio digital. Una innovación docente del sistema evaluativo de los aprendizajes universitarios de la Universidad Autónoma de Barcelona:** resume parte de una investigación doctoral enmarcada en el uso pedagógico de las Tecnologías de la Información y la Comunicación ("TIC") y de las nuevas metodologías de enseñanza y aprendizaje universitario. Centra su interés en la innovación educativa denominada internacionalmente como "e-portfolio" o precisando su soporte tecnológico "portafolio digital". Se describe una propuesta aplicada con un diseño instruccional desarrollado en 2011 centrada en el uso de blogs y espacios de almacenamiento en red, pero están en conflicto con las políticas institucionales de la UCI y la capacidades logísticas y adaptativas de la red en Cuba. (López,O. 2006)

Se relacionan a continuación los sistemas estudiados en la UCI:

1- **Integración de los instrumentos de evaluación para la gestión de las evidencias en la plataforma educativa Zera:** expone la integración de los instrumentos de evaluación que han sido identificados en la plataforma Zera, brindándole al estudiante una potente herramienta de aval donde se almacenen las evidencias de aprendizaje, logros y habilidades alcanzadas por el estudiante durante su vida estudiantil. No contempla la posibilidad de que pueda certificar un rol informático el estudiante que logre almacenar el cúmulo de evidencias necesarias para solicitar el mismo. (Pérez , L. 2012)
2- **Análisis y diseño del portafolio digital del profesor en la Universidad de las Ciencias Informáticas:** se realiza un estudio de los sistemas similares y un análisis de

las principales tecnologías y herramientas definidas. Se recogen los conceptos esenciales de los procesos del portafolio digital del profesor, siendo este el principal protagonista invalidando el uso de este para un estudiante (Albelo,D. 2011).

3- **Análisis y diseño del Portafolio digital del estudiante de la Universidad de las Ciencias Informáticas:** solución informática personalizada para la gestión electrónica de evidencias documentales generadas por el proceso de enseñanza aprendizaje del estudiante. Solo está definido el análisis y diseño de la investigación y existen pocas referencias acerca de la solución presentada.

4- **Implementación del Portafolio Digital de la Universidad de las Ciencias Informáticas:** Se crea una solución informática, que permite almacenar de manera segura la información de las personas de la universidad en un único expediente, retroalimentándose de todas las evidencias documentales que se generan asociadas a una persona en un centro de estudios, la cual maneja toda la documentación relacionada con las personas de una institución. Sin embargo, su aprovechamiento en la práctica se encuentra limitado por su enfoque de repositorio que no contempla la clasificación, generación y revisión de los artefactos para su constitución en evidencias (Martín,L. 2011).

Se puede resumir que con el estudio de los sistemas similares de gestión académica basados en evidencias se identificaron como principales funcionalidades a incluir en el sistema a desarrollar: generar portafolio, asignar credenciales, generar expediente. Las mismas contribuyeron a perfeccionar las nociones sobre la forma en que se desarrolla el proceso de certificación de roles en la UCI.

Epígrafe 1.3.El proceso de certificación de roles en la Universidad de las Ciencias Informáticas.

El proceso de certificación de roles en la UCI se realiza mediante el apoyo de un expediente conformado en una hoja de cálculo donde se establecen las competencias y escenarios por cada rol a certificar. Además se ponen de manifiesto cada una de las tareas contempladas en el plan de formación y que fueron realizadas por el estudiante, así como las evidencias que soportan su realización.

La certificación de rol, brinda un cuerpo de conocimientos que permite desempeñar un rol específico ya que no sólo requiere de un conocimiento de marcas tecnológicas sino de métodos y una disciplina de trabajo que beneficie al empleado y a su empleador. Sirve de aval

para el estudiante que desee desempeñarse profesionalmente en determinado rol en el mundo de la informática, y se considera que no tendrá mayor crecimiento y demanda hasta que la acreditación internacional de universidades se consolide en el mundo.

Los posibles roles a certificar por los estudiantes que cuenten con las habilidades y competencias necesarias para el mismo son:

1. Analista
2. Desarrollador
3. Diseñador de pruebas de software
4. Administrador de base de datos
5. Diseñador de base de datos
6. Administrador de la gestión de la configuración y cambios
7. Soporte de hardware y software
8. Analista de proceso de negocio

A continuación, en la Tabla 1, se muestra un ejemplo del cronograma del proceso de certificación de rol en la UCI.

Actividades	Involucrados	Fecha
-Entrega del Expediente de Certificación de Roles al Subdirector de Formación	Estudiante interesado en certificar el rol	Del 1ro al 4 de Octubre
-Selección de estudiantes a iniciar el Proceso de Certificación de Roles. -Entrega de listado de estudiantes a certificar rol al Departamento Metodológico Central	Subdirector del Centro y personas asignadas. Jefe de Departamento de IGSW de la Facultad.	Del 8 al 11 de Octubre
Creación de los tribunales de Certificación de Roles	Departamento Metodológico Central de IGSW y Vicerrectoría de Producción	Del 15 al 18 de Octubre
Sesión de los Tribunales de Certificación de Roles	Tribunales y estudiantes involucrados en el Proceso de Certificación de Roles.	Del 22 al 25 de Octubre En caso necesario sesionaría también : Del 29 al 1ero de Octubre

Tabla 1 Cronograma del Proceso de certificación en la UCI

Después de haber abordado las características, regularidades y el cronograma del proceso de

certificación de roles en la UCI se procede a seleccionar las herramientas y las tecnologías a utilizar en el desarrollo de la solución informática para la problemática planteada.

Epígrafe 1.4. La selección de la metodología, las herramientas y las tecnologías a utilizar para el desarrollo del sistema para gestionar la certificación de roles

El proceso de desarrollo de software, también denominado ciclo de vida del desarrollo de software, es una estructura aplicada al desarrollo de un producto de software. Tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente. Existen varios modelos a seguir para el establecimiento de un proceso para el desarrollo de software, cada uno de los cuales describe un enfoque distinto para diferentes actividades que tienen lugar durante el proceso. Una gran cantidad de organizaciones de desarrollo de software implementan metodologías para el proceso de desarrollo (Gonzalez,M. 2012).

Las metodologías de desarrollo de software contienen un conjunto de procedimientos, técnicas y ayudas con un soporte documental para el desarrollo de productos de software. Indican paso a paso todas las actividades necesarias a realizar para de esta forma lograr el producto deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de desempeñar (Universidad de Murcia, 2011)

Desarrollar un buen software depende de un sinnúmero de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo, en un determinado proyecto es trascendental para el éxito del producto. El papel preponderante de las metodologías es sin duda esencial en un proyecto y en el paso inicial, que debe encajar en el equipo, guiar y organizar actividades que conlleven al logro de las metas trazadas en el grupo.

El éxito del producto depende en gran parte de la metodología escogida por el equipo, ya sea tradicional o ágil, donde los equipos maximicen su potencial, aumenten la calidad del producto con los recursos y tiempos establecidos.

Las metodologías tradicionales (formales) se focalizan en documentación, planificación y procesos. (Plantillas, técnicas de administración, revisiones, etc.), que centran su atención en llevar una documentación exhaustiva de todo el proyecto y centran su atención en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto.

Los métodos ágiles (MAs) son adaptables en lugar de predictivos. Los métodos ingenieriles tienden a intentar planear una parte grande del proceso del software en gran detalle para un plazo grande de tiempo, pero esto funciona bien hasta que las cosas cambian. Así que su naturaleza es resistirse al cambio. Para los métodos ágiles, no obstante, el grado de adaptabilidad es muy alto donde el cambio es bienvenido. Intentan ser procesos que se adaptan y crecen en el cambio, incluso al punto de cambiarse ellos mismos. Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan. Nos lo proponen porque para muchos clientes esta flexibilidad será una ventaja competitiva y porque estar preparados para el cambio significar reducir su coste. Para la selección de la metodología de desarrollo a utilizar, en la Tabla 2, se realiza una comparación entre estos dos grupos donde se exponen las potencialidades del uso de cada uno:

Metodologías Tradicionales	Metodologías Ágiles
Más Roles, más específicos	Pocos Roles, más genéricos y flexibles
Más Artefactos. El modelado es esencial, mantenimiento de modelos	Pocos Artefactos. El modelado es prescindible, modelos desechables.
Existe un contrato prefijado	No existe un contrato tradicional o al menos es bastante flexible
La arquitectura es esencial: Se promueve que la arquitectura se defina tempranamente en el proyecto	Menos énfasis en la arquitectura: La arquitectura se va definiendo y mejorando a lo largo del proyecto
Cierta resistencia a los cambios (estos pueden provocar grandes costos)	Sujeta a cambios en cualquier etapa del proyecto
Dirigidas al proceso: roles, actividades y artefactos	Dirigidas a las personas: el individuo y el trabajo en equipo
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos	Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños y trabajando en el mismo sitio
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo

Tabla 2 Metodologías ágiles vs tradicionales

Luego de realizada una definición de las metodologías ágiles y de compararlas con las tradicionales se selecciona un enfoque ágil ya que el mismo presenta características que se adaptan a los requerimientos del equipo de desarrollo.

Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios, cada metodología tiene características propias y hace énfasis en algunos aspectos más específicos.

CAPÍTULO I

La Programación Extrema (XP): es una metodología ágil centrada 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. Esta se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (Valdarrama,S. 2006).

El proceso de desarrollo ideal definido por Kent Beck para esta metodología se compone de 4 faces en un ciclo de iteraciones hasta la entrega final. La primera fase es la de investigación en la cual el cliente conforma las Historias de usuarios y define la prioridad de las mismas en el negocio, le sigue la planificación en la cual el grupo de desarrollo realiza una estimación del esfuerzo y el riesgo de cada una de las historias de usuario basada en la experiencia de los desarrolladores, a partir de este momento comienza el análisis y diseño del producto el cual lleva a cabo una retroalimentación continua de todo lo realizado conformando pequeñas entregas al cliente pasando por la fase de mantenimiento hasta la entrega final (Gómez,T y otros. 2010).

XP propone un conjunto de buenas prácticas conocidas en programación, las cuales resultan imprescindibles para el desarrollo de productos confiables y de alta calidad:

- **El juego de la planificación:** hay una comunicación frecuente entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.
- **Entregas pequeñas:** producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más tres meses.
- **Metáfora:** el sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).
- **Diseño simple:** se debe diseñar la solución más simple que pueda funcionar y ser

implementada en un momento determinado del proyecto.

- **Pruebas:** la producción de código está dirigida por las pruebas unitarias. Éstas son establecidas por el cliente en la medida que se escriba el código y son ejecutadas constantemente ante cada modificación del sistema.
- **Programación en parejas:** toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores).
- **Propiedad colectiva del código:** cualquier programador puede cambiar cualquier parte del código en cualquier momento.
- **Integración continua:** cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.
- **40 horas por semana:** se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo.
- **Cliente in-situ:** el cliente tiene que estar presente y disponible todo el tiempo para el equipo. Éste es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita.
- **Estándares de programación:** XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

SCRUM: es una de las metodologías ágiles más conocidas para la gestión de proyectos. Permite a las organizaciones eliminar los impedimentos clásicos en el desarrollo de los proyectos, aumentando la satisfacción de los clientes mediante la realización de entregas frecuentes de resultados tangibles e integrándolos activamente en el ciclo de desarrollo, lo cual proporciona además una mayor adaptación y adecuación a sus necesidades. SCRUM potencia la formación de equipos de trabajo autosuficientes y multidisciplinarios, reduciendo la carga de gestión y proporcionando a los miembros del equipo un entorno amigable y productivo para desarrollar sus habilidades al máximo (Gómez,T y otros. 2010).

Dynamic Systems Development Method (DSDM): se caracteriza por su rapidez de desarrollo

atendiendo a las demandas de tecnología de forma eficaz y eficiente previendo que transcurra mucho tiempo y la tecnología cambie. Es una metodología ágil situada dentro de las RAD (desarrollo rápido de aplicaciones), es ideal para proyectos de sistemas de información cuyos presupuestos y agendas son muy apretados. En esta el equipo de desarrollo debe realizar entregas cortas pero frecuentemente, logrando así que todos los cambios pueden ser reversibles (Universidad de Murcia. 2011).

En la Tabla 3 se relacionan los criterios de selección definidos por el equipo de desarrollo de acuerdo a las características del proyecto y a la solución a desarrollar. Se representa con una cruz (x) la presencia del criterio en la metodología y en el total aparece la suma de los indicadores para cada caso.

Criterios	**SCRUM**	**XP**	**DSDM**
Equipo de desarrollo pequeño.	x	x	x
Genera información mínima necesaria		x	
Sujeta a cambios en cualquier etapa del proyecto.		x	
El cliente es parte del equipo de desarrollo		x	
Robustez	x	x	x
Desarrollo iterativo e incremental		x	x
Programación por parejas		x	
Total	2	7	3

Tabla 3 Comparación entre Metodologías Ágiles

Luego de un análisis de las metodologías ágiles se decide utilizar XP, ya que propone una estructura de roles adaptada al proyecto, la realimentación continua entre el cliente y el equipo de desarrollo. Además el software será desarrollado por dos programadores coincidiendo con la característica de la XP programación por pareja, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y la aplicación de las buenas prácticas propuestas por la misma.

Uno de los pasos más importantes en el desarrollo de un software es la selección del marco de trabajo. Un marco de trabajo es una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de aplicaciones. Se puede considerar

como una aplicación genérica incompleta y configurable a la que pueden añadirse las últimas piezas para construir una aplicación concreta (Ga,F. 2008).

Para su selección a continuación se exponen las caractertísticas y oportunidades que se ofrecen con cada uno de los marcos de trabajo estudiados:

Symfony 2.1.7: es un marco de trabajo completo diseñado para optimizar el desarrollo de aplicaciones web. Para empezar, 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. Permite la ejecución automatizada de tareas, brinda buena documentación y además logra la implementación robusta del patrón Modelo Vista Controlador (MVC) con la integración de Doctrine[2], es compatible con la mayoría de gestores de bases de datos y puede ser ejecutado en múltiples plataformas (Franklin,C. 2010).
Symfony fue diseñado para ajustarse a los siguientes requisitos:

- Independiente del sistema gestor de bases de datos. Su capa de abstracción permite cambiar con facilidad de Sistema Gestor de Bases de Datos (SGBD) en cualquier fase del proyecto.
- Utiliza programación orientada a objetos, de ahí que sea imprescindible PHP 5.
- Sencillo de usar en la mayoría de casos, aunque es preferible para el desarrollo de grandes aplicaciones web que para pequeños proyectos.
- Aunque utiliza MVC (Modelo Vista Controlador), tiene su propia forma de trabajo en este punto, con variantes del MVC clásico como la capa de abstracción de base de datos, el controlador frontal y las acciones.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Una potente línea de comandos que facilita la generación de código, lo cual contribuye a ahorrar tiempo de trabajo.

[2] Doctrine es una librería para PHP que permite trabajar con un esquema de base de datos como si fuese un conjunto de objetos, y no de tablas y registros. (Guardado, 2010)

CodeIgniter: es un marco de trabajo para la creación de aplicaciones web dinámicas con PHP. Su principal objetivo es ayudar a que los desarrolladores, puedan realizar proyectos más rápido que creando toda la estructura desde cero, proveyendo un conjunto de bibliotecas para tareas comúnmente necesarias, así como una interfaz simple y estructura lógica para acceder a esas bibliotecas. CodeIgniter le permite enfocarse creativamente en su proyecto al minimizar la cantidad de código necesaria para una tarea dada. La arquitectura de CodeIgniter se basa en el paradigma MVC (modelo, vista, controlador), lo que permite una ágil separación de las distintas capas que configuran la funcionalidad de un software (Martínez,P. 2010).

Zend Framework: es un marco de trabajo de código abierto para desarrollar aplicaciones web y servicios web con PHP5. Ofrece un gran rendimiento, una robusta implementación del patrón Modelo Vista Controlador (MVC),una abstracción de base de datos y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos (Madeja, J. 2008).

CakePHP: es un marco de trabajo de desarrollo rápido para PHP, libre y de código abierto. Facilita al usuario la interacción con la base de datos, además de hacer uso del patrón Modelo Vista Controlador, es compatible con PHP5 y posee un sistema de plantillas rápido y flexible (Madeja, J. 2008).

En la Tabla 4 se procedió de la misma manera que en la Tabla 3 se realizó para la selección de la metodología de desarrollo, pero en este caso para el marco de trabajo.

	Marco de trabajo			
Criterios	**CodeIgniter**	**Zend Framework**	**CakePHP**	**Symfony2.1.7**
Código abierto	X	X	X	X
Robustez	X	X	X	X
Multiplataforma	X	X	X	X
Seguridad	X	X	X	X
Flexibilidad ante cambios	X	X	X	X
Simplicidad	X		X	X
Comunidad y soporte		X		X
Total	6	6	6	7

Tabla 4 Comparación entre Marcos de Trabajo

Se selecciona como marco de trabajo Symfony 2.1.7 ya que el mismo ha sido diseñado para desarrollar y optimizar aplicaciones web así como de obtener el máximo rendimiento de PHP5 y aprovechar todas sus características.

Ya se está en condiciones de pasar a uno de los pasos fundamentales para llevar a cabo el desarrollo de un software es definir el entorno de desarrollo mediante el cual se llevará a cabo la implementación del mismo. Un Entorno de Desarrollo Integrado (IDE, según sus siglas en inglés), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto (García,F 2013).

NetBeans 7.3: es un entorno de desarrollo integrado, modular y de base estándar. Consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general para compilar cualquier tipo de aplicación. Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. NetBeans permite crear aplicaciones web con PHP 5, un potente depurador integrado y además de venir con soporte para Symfony (Díaz,M y otros. 2007).

PhpStorm 7: es un editor de PHP enfocado en la productividad de los desarrolladores. Algunas de las características que posee son: completado de código, fácil configuración de proyectos, editor avanzado de Java Script[3], editor HTML[4]/CSS[5] y herramientas de depuración (Díaz,M y otros. 2007).

[3] Lenguaje ligero e interpretado, orientado a objetos, más conocido como el lenguaje de script para páginas web, pero también usado en muchos entornos sin navegador (MDN, 2012).

[4] HTML, Hyper Text Markup Language (Lenguaje de marcación de Hipertexto) es el lenguaje de marcas de texto utilizado normalmente en la www (World Wide Web) (Ravioli,2001).

[5] Es el acrónimo de Cascade Style Sheet, traducido al español significa Hojas de Estilo en Cascada.

Eclipse: es un entorno de desarrollo integrado de código abierto y multiplataforma. Eclipse dispone de un editor de texto con resaltado de sintaxis. La compilación es en tiempo real y posee la capacidad de expandir su potencial utilizando módulos (Díaz,M y otros. 2007).

En la Tabla 5 se utiliza el mismo procedimiento que en los casos anteriores, pero en esta ocasión para la selcción del Entorno de Desarrollo Integrado.

	Entorno de Desarrollo		
Criterios	**NetBeans**	**phpStorm**	**Eclipse**
Libre y gratuito	10	10	10
Interprete compatible con Symfony	10	10	8
Dominio del Grupo de Desarrollo	10	2	2
Comunidad y soporte	10	5	7
Usabilidad	9	9	10
Multiplataforma	10	10	10
Total	59	46	47

Tabla 5 Comparación entre Entornos de Desarrollo

Se seleccionó como entorno de desarrollo NetBeans 7.3 el cual según el estudio realizado por el grupo de desarrollo es el indicado para realizar la aplicación. Consta de un entorno de desarrollo integrado (IDE) de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear un sistema web, este soporta varias plataformas dentro de ellas PHP, además ofrece una selección variada de plugins[6] para optimizar el trabajo y es completamente compatible con Symfony.

En este contexto para el desarrollo de la solución fueron tomados en cuenta un conjunto de herramientas y tecnologías multiplataforma y de código abierto. Dentro de estas se encuentran las herramientas de Ingeniería de Software Asistida por Ordenador (CASE), aplicaciones

[6] Un plugin es aquella aplicación que, en un programa informático, añade una funcionalidad adicional o una nueva característica al software (definicion.de, 2000).

informáticas que brindan un conjunto de ayudas y/o funcionalidades para el ciclo de vida completo del proceso de desarrollo de software o en algunas de sus fases.

A continuación se realiza un esbozo de la herramienta CASE Visual Paradigm y de sus principales características.

Visual Paradigm 8.0: es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegue. El software de modelado ayuda la construcción rápida de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Facilita la interoperabilidad con otras herramientas CASE, la mayoría de los IDE's principalmente y permite la integración de todos los componentes (Alonso,E. 2007).

Fue seleccionada como herramienta CASE Visual Paradigm 8.0 el cual según el estudio realizado por el grupo de desarrollo es el indicado para el desarrollo de la solución ya que posibilitará su construcción con mayor rapidez, exactitud, a un menor costo y mejor trabajo en equipo. Facilitando la interoperabilidad con otras herramientas CASE, la mayoría de los IDE's y la integración de todos sus componentes.

Una vez definidos el marco de trabajo, el entorno de desarrollo y la herramienta CASE es necesario definir los lenguajes de programación a utilizar así como algunas bibliotecas para el desarrollo de la solución.

Un lenguaje de programación es un lenguaje diseñado que permite crear programas mediante un conjunto de instrucciones, reglas de sintaxis y operadores que un equipo debe ejecutar. Posibilita que los humanos puedan darle instrucciones a un equipo y comunicarse con el hardware y el software.

PHP 5.3 es un lenguaje de programación del lado del servidor[7] gratuito e independiente de plataforma, rápido, con una gran biblioteca de funciones y mucha documentación. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Posee la capacidad de expandir su potencial utilizando módulos

[7] son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él (Torre,A. 2006).

(llamados ext's o extensiones). Entre sus características se encuentran:

- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y de enviar su resultado HTML al navegador Esto hace que la programación PHP sea segura y confiable.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Posee manejo de excepciones (desde PHP5).
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Debido a su flexibilidad ha tenido una gran acogida como lenguaje base para las aplicaciones WEB de manejo de contenido, y es su uso principal.

Twig es un motor de plantillas que posee un amigable ambiente para el diseñador y desarrollador apegado a los principios de PHP permitiendo añadir funcionalidades a los entornos de plantillas. Posee una sintaxis corta y concisa, similar a la de otros lenguajes de programación. Además implementa un mecanismo de herencia de plantillas, para acelerar el rendimiento del sistema que se desarrolla (Pacheco,N. 2011).

Este lenguaje permite compilar las plantillas hasta código PHP regular optimizado, lo que proporciona rapidez durante la implementación. Garantiza la seguridad del código de las plantillas ya que posee un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable. Permite al desarrollador definir sus propias etiquetas y filtros personalizados, garantizando flexibilidad a la aplicación. Posibilita la depuración de las plantillas creadas, mediante mensajes de error con el nombre del archivo y el número de línea donde se produjo el problema (Pacheco,N. 2011).

HTML 5.0: es un lenguaje de programación del lado del cliente, diseñado para estructurar textos para la generación de páginas web. La interpretación de las etiquetas es realizada por el navegador web. El lenguaje HTML es extensible, se le pueden añadir características, etiquetas y funciones adicionales para el diseño de páginas web, generando un producto vistoso, rápido y sencillo (Alvarez,M. 2001).

CSS 3.0: es un lenguaje creado para controlar el aspecto o presentación de los documentos

electrónicos definidos con HTML y XHTML[8]. Es empleado para separar los contenidos y su presentación y es imprescindible para crear páginas web complejas ya que separa la definición de los contenidos y la descripción de su aspecto presentando numerosas ventajas. Mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes (Universidad Politécnica de Madrid, 2005).

Bootstrap 3.0.1: es una de las bibliotecas más utilizadas y conocidas en la actualidad, desarrollado por Twiter, y pensado y diseñado para crear interfaces web. Los diseños creados con Bootstrap son simples, limpios e intuitivos, esto les da agilidad a la hora de cargar y al adaptarse a otros dispositivos. Una de las principales características que posee es que permite crear interfaces web con CSS y JavaScript que adaptan la interfaz dependiendo del tamaño del dispositivo en el que se visualice de forma nativa, es decir, automáticamente se adapta al tamaño de un ordenador o de una Tablet sin que el usuario tenga que hacer nada, esto se denomina diseño adaptativo o Responsive Designy. Es una de las corrientes que marcan esta nueva era digital (Jacob,M. 2014).

JavaScript: es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario (Polo,M. 2008).

JQuery 2.0.3: es una biblioteca JavaScript rápida, pequeña y rica en funciones. Está destinada para documentos HTML de recorrido y la manipulación, manejo de eventos, animación y Ajax. Mucho más simple con un API, fácil de usar que funciona a través de una multitud de navegadores. Con una combinación de versatilidad y capacidad de ampliación, jQuery ha cambiado la forma en que millones de personas escriben JavaScript (Eguiluz,J. 2007).

Para la consecución de la selección de las herramientas y tecnologías a utilizar se procede a realizar una definición y comparación de los sitemas gestores de bases de datos más utilizados.

Sistema gestor de bases de datos.: los Sistemas de Gestión de Base de Datos (SGBD) se usan como interfaz entre las aplicaciones, los usuarios y las bases de datos. Tienen como propósito principal gestionar convenientemente la información a almacenar o recuperar, de manera sencilla y fiable para el usuario (Universidad de Almería, 2003).

PostgreSQL 9.2: es un gestor de bases de datos orientadas a objetos (SGBDOO) muy conocido y usado en entornos de software libre. Está considerado como un SGBD de código abierto potente y, sobre todo, fidedigno con los estándares. Posee uso de transacciones avanzadas, lenguaje procedimental, gran estabilidad y escalabilidad (Pérez,M y otros. 2014).

MySQL: es un SGBD rápido y sencillo de usar que posee y brinda un conjunto muy práctico de características desarrolladas en cooperación muy cercana con los usuarios. Ofrece un rico y muy útil conjunto de funciones para el diseño y desarrollo de Bases de Datos. La conectividad, velocidad y seguridad, hace que MySQL sea altamente conveniente para acceder a bases de datos en Internet (Ortega,J. 2014).

Oracle: es un Sistema de Gestión de Bases de Datos Relacional (SGBDR). La base de datos Oracle es una herramienta confiable y segura, tiene opciones de auditoría, copia de seguridad y aplicaciones para la toma de decisiones que lo diferencian de sus competidores libres y propietarios. Además se basa en la tecnología cliente/servidor. El acceso a los datos se otorga según privilegios concedidos por el administrador y es posible acceder a datos de Oracle usando software de otros fabricantes. Es una de las bases de datos más utilizadas actualmente (Ortega,J. 2014).

	Gestor de Base de Datos		
Criterios	**PostgreSQL**	**MySQL**	**Oracle**
Licencia Libre	10	5	0
Gran Comunidad y soporte	8	8	2
Dinámico	9	7	9
Multiplataforma	10	10	6
Seguro	8	7	9
Soporte PHP	10	10	10

Total	55	47	36

Tabla 6 Comparación entre Gestores de Bases de Datos

Se selecciona el Gestor de Base de Datos PostgreSQL considerando los argumentos analizados anteriormente y teniendo en cuenta la presencia en mayor medida de los criterios presentados en la Tabla 6.

En esta etapa de la investigación se está en condiciones de analizar los patrones arquitectónicos. **Patrón Arquitectónico:** está orientado a representar los diferentes elementos que componen una solución de software y las relaciones entre ellos. Los patrones arquitectónicos forman parte de la llamada Arquitectura de Software (arquitectura lógica de un sistema) que define, de manera abstracta, los componentes que llevan a cabo alguna tarea de proceso de información, sus interfaces y la comunicación entre ellos (Universidad de Almería, 2003).

Modelo-Vista-Controlador (MVC): es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es la capa de abstracción que se sitúa sobre Sistema de Gestión de Base de Datos llamado Doctrine, y la Lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista. La vista representa el modelo en un formato adecuado para interactuar con el usuario. Transforma el modelo en una página web que permite al usuario interactuar con ella. Se encarga de la presentación visual de los datos captados por el modelo para después mostrarlos al usuario. Interactúa con el modelo y muestra los datos al usuario. Los controladores reciben la entrada o sea los eventos producidos por el usuario mediante el uso del teclado o el mouse a través de las vistas. Los eventos son traducidos para servir las demandas del modelo o las vistas. Cuando se realiza algún cambio, entra en acción bien puede ser por cambios en el modelo o en la vista (Universidad de Almería, 2003).

La aplicación de este patrón de arquitectura presenta varias ventajas:

- Reduce el esfuerzo de programación necesario en la implementación de sistemas.
- El modelo, la vista y los controladores se tratan como entidades separadas lo cual hace que un cambio producido en el modelo se refleje automáticamente en las vistas.
- Se pueden construir varias vistas sin necesidad de modificar el modelo.

Cliente-servidor

Se puede definir como un patrón arquitectónico para el desarrollo de sistemas distribuidos. Este tipo de arquitectura distribuye una aplicación entre dos o más componentes especializados cuya ejecución se distribuye entre uno o más equipos. El cliente se define como el proceso que requiere un servicio en particular. El servidor se define como el proceso que provee dicho servicio (Universidad de Almería, 2003).

N capas

Es una de los patrones más comunes que los arquitectos de software utilizan para dividir sistemas de software complicados. Al pensar en un sistema en términos de capas, se imaginan los principales subsistemas de software ubicados de la misma forma que las capas de un pastel, donde cada capa descansa sobre la inferior. En este esquema la capa más alta utiliza varios servicios definidos por la inferior, pero la última es inconsciente de la superior. Además, normalmente cada capa oculta las capas inferiores de las siguientes superiores a esta (Almeira, 2011).

	Patrones		
Criterios	**MVC**	**Cliente-Servidor**	**Por capas**
Fiabilidad	X	X	X
Escalabilidad	X	X	X
Usabilidad	X	X	
Organización de los datos	X	X	X
Disponibilidad	X		X
Total	6	5	5

Tabla 7 Comparación entre Patrones Arquitectónicos

Se seleccionó el patrón MVC, de acuerdo al análisis realizado y la información mostrada en la Tabla 7, ya que el mismo permite el desarrollo de aplicaciones web robustas, además de poder aplicar multilenguaje y distintos diseños de presentación sin que se altere la lógica del negocio. Al separar la aplicación en capas como presentación, lógica de negocio y acceso a datos

resulta imprescindible en el desarrollo de arquitecturas consistentes, reutilizables y de fácil mantenimiento, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos.

Al finalizar el presente capítulo se concluir que:

- ✓ la selección de la metodología de desarrollo XP permitió la organización, planificación y ejecución de las actividades e hitos a cumplir, haciendo énfasis en la generación de artefactos bien documentados.
- ✓ La selección de las herramientas y tecnologías de acuerdo a los criterios establecidos por las características del equipo de desarrollo y de la solución a desarrollar favoreció la pertinencia de las mismas en la presente investigación.

CAPÍTULO II: Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles

En el presente capítulo se elabora una propuesta de solución, teniendo en cuenta cada una de las fases que para este momento dentro del ciclo de vida ideal de la metodología XP definido por Kent Beck, son necesarias abordar. Se presenta el diseño del software a través de los artefactos propuestos en las fases de planificación, diseño y codificación de la metodología de desarrollo seleccionada. Se muestra la lista de reserva de producto que debe cumplir el sistema de cómputo donde se despliega la herramienta. Se identifican y organizan las clases relevantes para las funcionalidades del sistema, así como los patrones arquitectónicos y de diseño utilizados para la realización del Portafolio.

Epígrafe 2.1. Características del sistema

La certificación de roles se realiza con el apoyo de una hoja de cálculo en el que por roles se establecen las competencias, habilidades, actividades, ponderaciones y demás elementos a considerar para cada rol a acreditar. El proceso se basa en el procesamiento de evidencias, consistentes en artefactos generados por los estudiantes en su actividad de la disciplina PID. Estos artefactos se van generando de acuerdo a las tareas que contempla el plan de formación a lo largo del semestre.

La recogida, revisión y aprobación de los expedientes se realiza al finalizar cada semestre. En muchas ocasiones el estudiante se ve afectado por la carencia de las evidencias que constaten las tareas realizadas. La causa fundamental es lo engorroso del proceso de generación, clasificación, almacenamiento y revisión de las evidencias, ya que las mismas están condicionadas por diversos factores de acuerdo a su tipología y ponderación en el documento que regula la certificación de roles.

Teniendo en cuenta las dificultades que presenta el proceso de certificación de roles, se decide desarrollar un sistema que sirva de apoyo en el proceso, un sistema para gestionar la certificación de roles, que contribuya a la generación, clasificación, almacenamiento y revisión de las evidencias, de las tareas de los estudiantes en las disciplina PID de la UCI. El sistema de apoyo al proceso de certificación de roles que se propone, será capaz de integrar las acciones de todos los individuos que intervienen en dicho proceso. Mediante la utilización de la herramienta el profesor de la disciplina PID podrá elaborar un registro con los estudiantes que supervisará en el semestre, definido en el sistema por el subdirector de formación o el administrador y desde el comienzo del período en curso, ir asignándoles tareas a los estudiantes que

están bajo su responsabilidad. El estudiante puede acceder a las tareas que le son asignadas y subir para cada una de ellas los artefactos que la misma genera, y de esta manera ir conformando un expediente asociado a un rol que acreditará, cuando sea capaz de reunir los suficientes artefactos que evidencien su labor productiva, la aplicación debe ser capaz de generar un grupo de reportes que pueden ser tomados también como evidencias en el proceso. Posee un mecanismo para notificaciones, alertas y avisos permitiendo la intercomunicación entre el sistema y los usuarios, y entre los usuarios registrados en la aplicación.

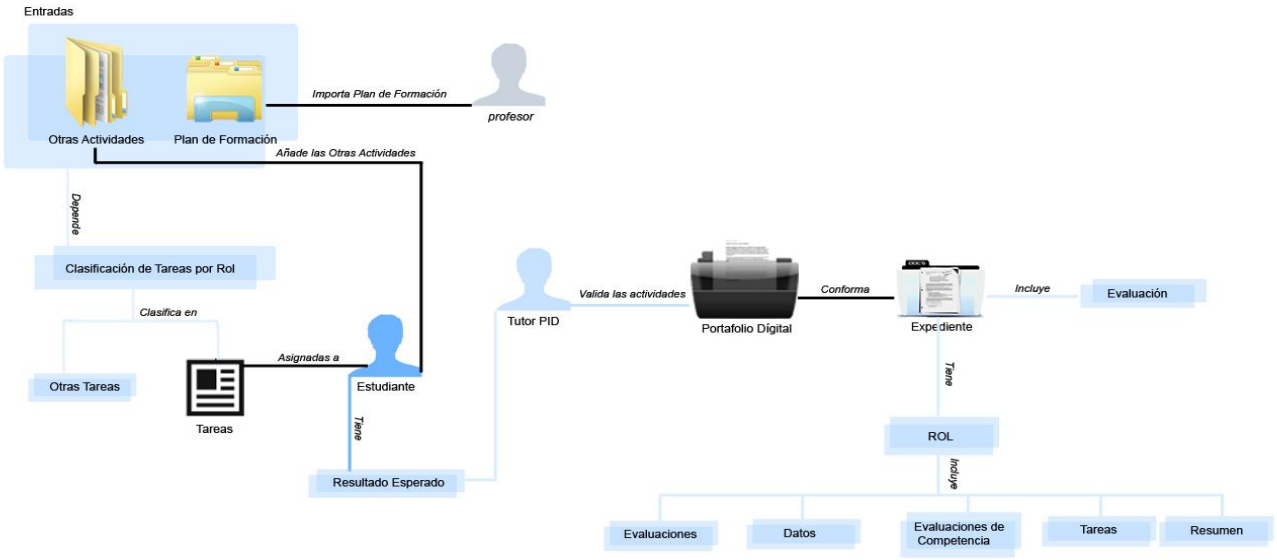


Ilustración 1 Flujo básico del sistema

Un paso fundamental para el desarrollo del sistema es la definición de la metáfora. En el desarrollo con la metodología XP una metáfora puede constituir una analogía a lo que otras metodologías o la mayoría de ellas toman como arquitectura. De esta manera se describirá el funcionamiento del sistema con palabras que sean comunes entre el grupo de desarrollo y los clientes. Así se logra que el cliente conozca y le resulte fácil entender el ciclo de desarrollo del sistema y la evolución del mismo desde un punto de vista bastante cercano a la de los desarrolladores (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).

El estudiante al acumular un grupo de evidencias obtenidas en el desarrollo de las tareas por las tareas asignadas por el plan de formación que fue importado por el profesor PID, se le notifica que puede conformar un expediente del rol al cual acreditan ese grupo de evidencias. Cada tarea tiene una

competencia específica que posee un valor, que se incrementa a medida que logre obtener resultados en cada una de las tareas que vaya realizando.

La tarea de elegir una metáfora para el sistema permitió mantener una coherencia de nombres de todo aquello que se pudo implementar.

Personal relacionado con el sistema

Se define como personal relacionado con el sistema a todas las personas involucradas o no en el proceso de certificación de roles, donde los estudiantes de la UCI son los protagonistas y la principal razón de ser del sistema. Los usuarios relacionados con la aplicación web son las personas que de una forma u otra interactúan con la aplicación, tanto los que obtienen resultados de valor con los procesos que se ejecutan como los que no obtienen ningún resultado. En el presente proyecto se definieron los siguientes usuarios:

Usuario relacionado con el sistema	Descripción
Estudiante	Usuario encargado de incorporar evidencias en el sistema para su posterior procesamiento, encargado de realizar una solicitud para certificar un rol (Pilar del proceso)
Profesor	Usuario encargado de incorporar actividades las cuales validará el profesor PID antes de ser asignadas a un estudiante.
Profesor PID	Usuario encargado de subir el plan de formación correspondiente al Estudiante que le pertenece. Es el encargado de validar cada una de las actividades o tareas asignadas a los estudiantes. Evalúa cada tarea junto con la evidencia que le corresponde subida a la aplicación por el Estudiante.
Administrador del sistema	Usuario encargado de administrar el sistema, con privilegios y credenciales para realizar cualquier cambio en la aplicación.

Subdirector de formación	Usuario encargado de velar por el correcto funcionamiento del proceso a través de la aplicación. Se encarga de definir el comienzo y el fin de un semestre en curso además de incorporar las asignaturas correspondientes a cada uno de los semestres.

Tabla 8 Usuarios relacionados con el sistema y su descripción

Lista de reservas para el producto

Las listas de reserva del producto en una aplicación son de una gran importancia ya que son las cualidades que todo sistema debe poseer para un correcto funcionamiento. A continuación se definen los siguientes requisitos:

- **Usabilidad**

El sistema deberá contar con una interfaz de fácil entendimiento para que usuarios inexpertos puedan interactuar fácilmente con el software.

El sistema podrá ser utilizado por cualquier usuario con las siguientes características:

- Conocimientos básicos relativos del uso de un navegador web
- Para un estudiante solo conocimientos básicos del proceso de certificación de un rol
- Para un profesor conocimientos suficientes en el proceso de certificación de un rol

El sistema se desplegará en lenguaje español.

- **Hardware**

Para el correcto funcionamiento de la aplicación se necesita de un equipo que soporte un navegador web, 512 MB de memoria RAM o superior, una tarjeta de red, conexión al dominio uci.cu, un microprocesador de un 1GHz o superior sin distinción de tecnología, y un disco duro de 80 GB o superior.

- **Software**

La instalación de la aplicación en un equipo que actuará como cliente requiere de un Sistema Operativo

que soporte la versión de Firefox v26.0 o un navegador que soporte CSS 3, PHP 5.0, HTML 5 y Java Script.

Epígrafe 2.2 Planificación

El ciclo de vida de un proyecto realizado con la metodología XP inicia con la fase de planificación. En esta fase, los clientes plantean a grandes rasgos las Historias de Usuarios. Al finalizar el equipo cuenta con suficiente material de trabajo como para producir una primera entrega. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas y tecnologías que serán utilizadas en el proyecto.

Se decidió tratar en el documento solo la funcionalidad alojar evidencias por la relevancia que tiene para el sistema, de la cual a continuación se presenta una descripción:

La funcionalidad alojar evidencias del Portafolio para la gestión de evidencias del proceso de certificación de roles, tiene su basamento en que un estudiante después de haber culminado la realización de una de las tareas que contempla el plan de formación, deberá subir las evidencias consistentes en los artefactos (prototipos, código fuente, trabajos investigativos, documentos, evaluaciones dadas por los tutores en las tareas que responden a alguna de las competencias específicas del rol, certificados o diplomas obtenidos a partir de la presentación de trabajos que constituyan resultados de las actividades realizadas en el desempeño del rol, etc.) de la realización de la misma para posteriormente ser validada por el profesor que se la asignó.

Uno de los artefactos que se generan por la metodología de desarrollo XP para la especificación de requisitos del software y las características del sistema son las Historias de usuarios (HU).

Historias de usuario (HU):

Es la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las Historias de Usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Estas deben proporcionar sólo el detalle suficiente como para poder hacer razonable la estimación de cuánto tiempo requiere la implementación de la historia. También se les asigna un número identificativo, una prioridad en el negocio (alta, media, baja) y la iteración en la que se

implementará. (Beck, 1999)

Historia de Usuario	
Número: 01	**Nombre de la Historia de Usuario:** Alojar artefactos
Cantidad de modificaciones a la Historia de Usuario: ninguna	
Usuario: Carlos Rafael Madrigal García, Frank Carlos García Lorenzo	**Iteración asignada:** 1.0
Prioridad en negocio: Alto	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos reales:** 1
Descripción: Permite a los estudiantes subir las evidencias de las tareas que desarrollaron para que puedan ser validadas posteriormente por el profesor que se las asignó. El sistema mostrará una interfaz con la descripción de la tarea seleccionada dando la posibilidad de alojar las evidencias de la tarea ya culminada.	
Observaciones: Es necesario definir con qué tarea va a estar relacionada la evidencia a alojar y quién va a ser el encargado de validarla. El sistema debe de emitir una notificación al profesor encargado de revisar el resultado esperado de la tarea asignada al estudiante que el mismo subió a la aplicación, y una vez el profesor valide la tarea se debe de emitir una notificación al estudiante que la alojó.	

Figura 1 HU Alojar artefactos

- ✓ **Número:** Posee el número asignado a la HU.
- ✓ **Nombre de HU:** Atributo que contiene el nombre de la HU.
- ✓ **Usuario:** El usuario del sistema que utiliza o protagoniza la HU.
- ✓ **Prioridad en el negocio:** Evidencia el nivel de prioridad de la HU en el negocio.

- ✓ **Riesgo de desarrollo:** Evidencia el nivel de riesgo en caso de no realizarse la HU.
- ✓ **Puntos estimados:** Este atributo no es más que una estimación hecha por el equipo de desarrollo del tiempo de duración de la HU. Cuando el valor es 1 equivale a una semana ideal de trabajo. En la metodología XP está definida una semana ideal como 5 días hábiles trabajando 40 horas, es decir, 8 horas diarias. Por lo que cuando el valor de dicho atributo es 0.5 equivale a 2 días y medio de trabajo, lo que se traduce en 20 horas.
- ✓ **Puntos reales:** Igual que el parámetro anterior, pero en este caso será el tiempo real en el que se realizó la HU.
- ✓ **Descripción:** Posee una breve descripción de lo que realizará la HU.

Estas se pueden clasificar según la prioridad para el negocio en:

- ✓ **Alta**: Se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.
- ✓ **Media**: Se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
- ✓ Baja: Se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.

Según el riesgo para el desarrollo en:

- ✓ **Alta**: Cuando en la implementación de las HU se consideran la posible existencia de errores que conlleven a la inoperatividad del sistema.
- ✓ **Media**: Cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.
- ✓ **Baja**: Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

Fueron diseñadas un total de 44 Historias de Usuario de las cuales 9 resultaron ser de prioridad alta para el negocio y 7 de un alto riesgo para el desarrollo.

	Alto	Medio	Bajo	Total
Prioridad	9	16	19	44
Riesgo	7	25	12	44

Tabla 9 Desgloso de HU

En esta fase el cliente establece la prioridad que tendrá cada HU según sus necesidades más inmediatas, luego los programadores realizan una estimación del esfuerzo que se necesita para cada una de ellas. La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas (release[9]). Se deben incluir varias iteraciones para lograr una entrega. El cronograma fijado en la etapa de planeamiento se realiza en un número de iteraciones, cada una toma de una a cuatro semanas en ejecución.

La planificación se realiza basándose en el tiempo o el alcance. Al planificar por tiempo, se multiplicó el número de iteraciones por la velocidad del proyecto, determinándose cuantos puntos se pueden completar. Al planificar según el alcance, se dividió la suma de puntos de las Historias de Usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

La Tabla 10 muestra la estimación de esfuerzo para cada una de las historias de usuarios definidas en el desarrollo de la solución propuesta.

[9] ciclo desde la entrevista con el usuario hasta la obtención de una solución (Beck, 1999)

Historias de Usuario	Puntos de estimación (semanas)
Visualizar usuario	2
Listar usuarios	2
Autenticar usuario en el sistema	2
Asignar credencial	2
Asociar profesor-estudiante	2
Eliminar asociación	2
Listar estudiantes por profesor	2
Importar Plan de Formación	2
Registrar tarea	2
Modificar tarea	2
Eliminar tarea	2
Visualizar tarea	2
Listar tareas	2
Evaluar tarea	2
Alojar artefacto	2
Eliminar artefacto	2
Visualizar artefacto	2
Listar artefactos	2
Descargar artefactos	2
Adicionar rol	1
Modificar rol	1
Eliminar rol	1
Listar rol	1
Visualizar rol	1
Adicionar competencia	1

Modificar competencia	1
Eliminar competencia	1
Listar competencia	1
Visualizar competencia	1
Adicionar escenario	1
Modificar escenario	1
Listar escenarios	1
Generar expediente	2
Listar expedientes	2
Visualizar expediente	2
Generar comprimido de Evidencias	2
Realizar solicitud de certificación de rol	2
Aceptar solicitud de certificación de rol	2
Rechazar solicitud de certificación de rol	2
Listar solicitudes de certificación de rol	2
Registrar notificación	1
Eliminar notificación	1
Listar notificaciones	1
Generar reportes	2

Tabla 10 Estimación de esfuerzos por HU

Desarrollo de iteraciones

Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta etapa, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo. Las iteraciones son también utilizadas para medir el progreso del proyecto.

CAPÍTULO II

Una vez definidas las Historias de Usuarios (HU) y estimado el esfuerzo propuesto para la realización de cada una de ellas, se distribuyó la realización del sistema en tres iteraciones, las cuales se describen a continuación de manera más detallada:

Iteración I: esta iteración tiene como objetivo realizar las HU1-19 las cuales se refieren a importar el plan de formación en el semestre y curso activo y de asignarle todas las tareas que este contiene al estudiante que se encuentra asociado. El estudiante al que se le asignó el plan de formación importado, a medida de que vaya realizando las tareas que este contiene, irá subiendo las evidencias que constaten la realización de las mismas para posteriormente ser evaluadas.

Iteración II: esta iteración tiene como objetivo realizar las HU 20-33 las cuales son las referentes a gestionar los roles posibles a certificar existentes en el sistema así como todas las competencias y escenarios que estos contienen.

Iteración III: esta iteración tiene como objetivo realizar la HU 34-44 las cuales son las referentes a gestionar el expediente de evidencias y las solicitudes de certificación de rol.

Después de realizados la estimación de esfuerzo y el plan de iteraciones, y continuando los pasos que propone XP se crea el plan de duración de las iteraciones. Este tiene como objetivo fundamental mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una según la prioridad asignada por el cliente.

Iteración	Historias de Usuario	Duración total de las Iteraciones (semanas)
Iteración I	HU(1-19)	8
Iteración II	HU(20-33)	3
Iteración III	HU(34-44)	7
Total	44	18

Tabla 11 Plan de duración de las iteraciones

A continuación se presenta el plan de entregas ideado para la fase de implementación. Atendiendo al mismo se harán entregas del sistema al finalizar cada iteración en la fecha aproximada que se indica en la siguiente tabla.

Iteración	Fecha de Entrega
1	23 de marzo del 2014
2	23 de Junio del 2014
3	10 de junio del 2014

Tabla 12 Distribución del Plan de Iteraciones

Epígrafe 2.3 Diseño de la solución.

La metodología XP propone que la implementación debe ejecutarse de forma iterativa e incremental, alcanzando al final de cada iteración un producto funcional que debe ser examinado y mostrado al cliente. De esta forma se garantiza una constante retroalimentación entre los desarrolladores y clientes posibilitando que los desarrolladores puedan ganar en claridad de lo que debe hacer el producto, apoyándose en la visión de los clientes.

Durante el diseño de la solución, la máxima simplicidad posible es la clave para el éxito de XP. Se debe tener en cuenta que un diseño complejo siempre tarda más en desarrollarse que uno simple, y que siempre es más fácil añadir complejidad a un diseño simple que quitarla de uno complejo.

Arquitectura del sistema

La arquitectura de un sistema es el nivel del diseño donde se definen la estructura y propiedades globales del sistema. Es una representación de alto nivel de la estructura del sistema que describe las partes que lo integran.

La arquitectura del sistema está basada en el patrón arquitectónico MVC el cual separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. (**Ilustración 2**)

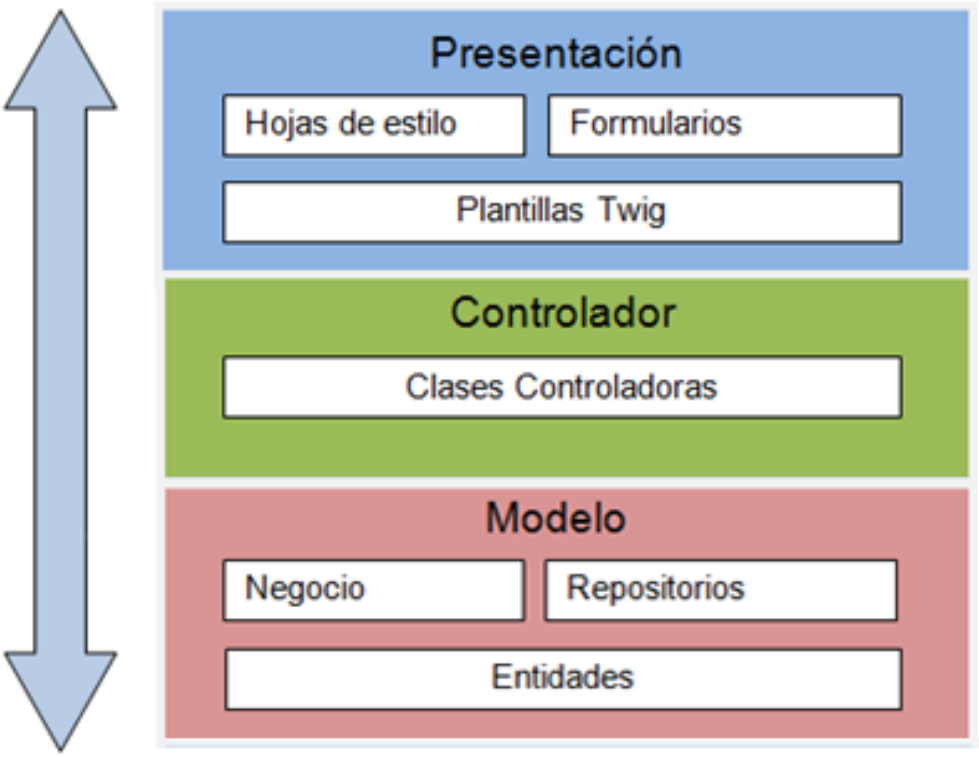


Ilustración 2 Arquitectura del sistema

La arquitectura del sistema en el nivel superior, o sea la capa de presentación, encapsula las interfaces de usuario representadas por las clases twig, hojas de estilo y formularios necesarios para la interacción con

el cliente. La capa controlador contiene las clases controladoras que se encargan de dar respuesta a las peticiones realizadas por el usuario. A continuación se muestra la capa de negocio, la cual contiene las clases de negocio que tienen como principal función separar la lógica de negocio del resto de la aplicación y obtener una mayor reutilización del código. Esta capa se encarga de recibir las peticiones, procesar la información, hacer pedidos a las clases Repositorio y devolver respuestas a las clases Controladoras, las que a su vez las envían a las clases enmarcadas en la capa vista

Inicialmente el usuario interactúa con la interfaz (Vista) de alguna forma. El controlador recibe la petición de la acción solicitada y gestiona el evento. El controlador accede al modelo actualizándolo o buscando la información requerida. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario y mostrar los datos del modelo para generar la interfaz apropiada. (Ver Ilustración 3 Patrón MVC)

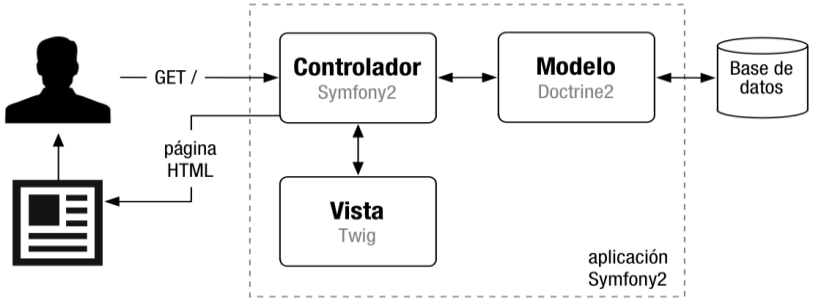


Ilustración 3 Patrón MVC

A continuación se muestra la estructura de carpetas del proyecto siguiendo el patrón arquitectónico MVC obtenido, con la ayuda del marco de trabajo Symfony2 y la herramienta para mapear las clases del modelo Doctrine.

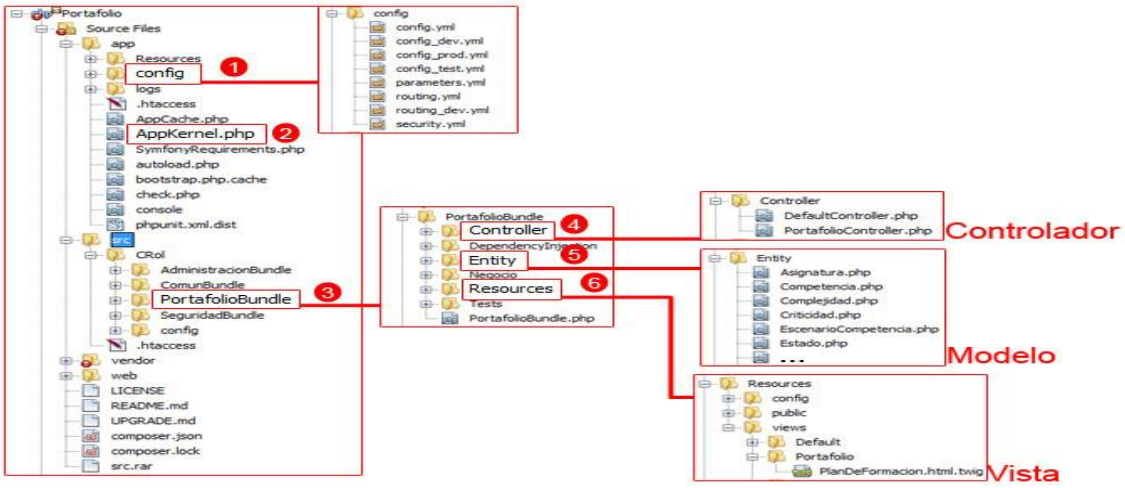


Ilustración 4 Estructura de Clases

Symfony 2 posee una clase controladora genérica la cual se muestra en la Ilustración 4 (2), que es el punto de entrada principal de la aplicación y es la responsable de toda la configuración. Ofrece las rutas de los módulos o Bundles que el usuario necesita para satisfacer su necesidad y que se encuentran en la carpeta config (1). Es el núcleo de Symfony 2 y por tanto uno de los componentes fundamentales para su correcto funcionamiento. En la carpeta src se localiza proyecto llamado Crol, nombre que se deriva de Certificar un Rol y que identifica al sistema creado. El sistema está organizado en cuatro bundles o módulos fundamentales: Administración, Común, Portafolio y Seguridad, los cuales se pueden observar dentro de la estructura de clases mostrada anteriormente. Los módulos poseen una organización común, la cual se muestra en (3) dentro de la misma se localizan las carpetas con los componentes específicos de la arquitectura:

Controller (4): posee los ficheros con los códigos de las clases Controladoras.

Entity (5): en esta carpeta se encuentran las Entidades del Modelo.

Resources (6): dentro de esta se encuentran las carpetas con los CSS, los JS y las TWIG que conforman la Vista.

A continuación se realiza una descripción de cada uno de los Bundles (Módulos) junto con una explicación de su funcionamiento y de sus funcionalidades.

Administración: este módulo contiene las funcionalidades necesarias para gestionar los usuarios que tendrán acceso al sistema. Se podrá a través de esta vista gestionar un grupo de credenciales que permiten la accesibilidad en el sistema, modificar los datos y rol de un usuario existente y eliminar un usuario registrado. El sistema también permitirá listar o mostrar los usuarios registrados en el sistema. Contiene además una funcionalidad para especificar la fecha en curso definiendo el momento inicial de la aplicación del plan de formación en dependencia para cada uno de los años, ya que los semestres no comienzan en el mismo momento. En este módulo también se gestionan todos los datos que resultan básicos en un sistema, así como la gestión de asignaturas encargada de organizar las asignaturas por semestre en curso, la gestión de los roles a certificar, con cada una de las competencias específicas y los estados de competencias de estas.

Seguridad: este módulo de seguridad se encarga de la autenticación de los usuarios en el sistema y de la gestión de las trazas. El equipo de desarrollo decidió gestionar esta funcionalidad independiente para lograr una seguridad de acuerdo a lo que plantean las normas de Seguridad Informáticas en la Universidad de las Ciencias Informáticas (UCI). El módulo autentica los usuarios del dominio uci.cu cuyos permisos valida el Módulo de Administración. La gestión de las trazas es un sistema que va registrando las operaciones que van realizando los usuarios en la aplicación, de esta manera se conoce qué usuario inserta, actualiza, elimina o simplemente entra en la aplicación y en qué momento lo hace.

Común: el módulo común gestiona las vistas y las entidades que son comunes en el sistema permitiendo la reutilización de código. Este módulo además gestiona las notificaciones del sistema posibilitando la intercomunicación entre usuarios – usuarios y sistema –usuarios.
Portafolio: gestiona las entidades y funcionalidades del negocio permitiendo una independencia del código. Las clases de este módulo poseen responsabilidades específicas a cumplir, de acuerdo con la información que manejan.

El marco de trabajo empleado promueve las buenas prácticas del diseño y la implementación haciendo uso de **patrones de diseño**. Los patrones de diseño expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software. El uso de patrones proporciona una estructura conocida por todos los programadores, de manera que la forma de trabajar no resulte distinta entre los mismos. Así la incorporación de un nuevo programador, no requerirá conocimiento de lo

realizado anteriormente por otro. (O'Reilly, 2010)
Los patrones de diseño se dividen en dos grandes grupos los GRASP (del inglés General Responsibility Asignment Software Patterns) (patrones generales de software para asignar responsabilidades) y los GOF (del inglés Gand of Four).

Clasificación según su propósito:

- ✓ **Patrones de Creación:** Los patrones creacionales abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados.
- ✓ **Patrones Estructurales:** Los patrones estructurales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades.
- ✓ **Patrones de Comportamiento:** Los patrones de comportamiento están relacionados con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos.

A continuación se muestran los patrones según su propósito organizados por cada una de sus clasificaciones:

- ✓ **Instancia Única (Singleton)**: Restringe la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Este patrón se muestra en la clase Tarea la cual es la única que puede crear objetos de Tipo Evidencia.

- ✓ **Mediador (Mediator)**: Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente. Este patrón se ilustra en las clases Portafolio y Expediente, esta última interfiere entre Portafolio y Tarea.

Estructurales:

- ✓ **Fachada**: Provee de una única interfaz para acceder a un sistema completo, que actúa como único punto de acceso al mismo, y hace que éste sea más fácil de utilizar. Ejemplo de esto se evidencia en la clase PortafolioGtr que es la encargada del acceso a los datos de la entidad Tarea en la BD.

Creacionales:

- ✓ **Fábrica abstracta:** define una interfaz para crear un objeto, pero delega la responsabilidad de instanciarlo a sus subclases y promueve el encapsulamiento de las partes más variables del sistema. Ejemplo de su uso se aprecia la clase PortafolioController.

Los patrones de asignación de responsabilidad **(GRASP)** empleados fueron:

- ✓ **Alta cohesión:** se encuentra evidenciado en la implementación de las clases que forman parte de las capas controladora, negocio y modelo; las cuales están formadas por diferentes funcionalidades que se encuentran estrechamente relacionadas.
- ✓ **Creador:** se refleja en las clases gestoras donde se encuentran las acciones definidas para las operaciones lógicas del negocio en cuestión y se ejecutan cada una de ellas. En las acciones se crean los objetos de las clases que representan las entidades.
- ✓ **Experto:** se pone en práctica con el uso de clases que poseen responsabilidades específicas a cumplir, de acuerdo con la información que manejan. El uso de este patrón se evidencia en las clases controladoras, modelos y de entidad, que poseen funciones concretas de acuerdo con la información que gestionan.
- ✓ **Bajo acoplamiento:** consiste en tener las clases lo menos relacionadas entre sí posible, para que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión en las demás. Esta característica permitió potenciar la reutilización y disminuyó la dependencia entre las clases.
- ✓ **Controlador:** consolida todas las peticiones de manipulación por analizar a través de un objeto de controlador único. Estas peticiones web son tratadas por un solo controlador frontal que posee el marco de trabajo que se denomina app.php donde es el único punto de entrada a la aplicación.

Tarjetas CRC

Las tarjetas CRC constituyen una metodología creada por Kent Beck que define la relación Clase – Responsabilidad – Colaboración y ayuda a pensar en términos de objetos. En primer lugar deben identificarse las clases que compondrán al sistema. Para comenzar este proceso se utiliza la técnica de tormenta de ideas (Brainstorm) mediante la cual el equipo de desarrollo aporta ideas para localizar todas las posibles clases. Una vez identificadas todas las clases se procede a refinarlas o sea dejar solo aquellas que sean básicas para el sistema que son las que poseen responsabilidades y participan

activamente en el sistema. Las tarjetas CRC se crean a partir de la lista de clases básicas identificadas y constituyen una primera aproximación a los objetos que luego se van a utilizar en el desarrollo del sistema. El resto de las tarjetas CRC puede ser consultado en el **Anexo 2.**

TareaController	
Responsabilidades • Registrar tarea • Modificar tarea • Eliminar tarea • Visualizar tarea • Listar tareas • Obtener tareas por persona • Tareas por expediente	**Colaboradores** • Tarea • Persona • Expediente • TareaGtr

Tabla 12 Tarjeta CRC de TareaController

Una clase es cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son los elementos que conoce y las que realizan, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades. En la práctica conviene tener pequeñas tarjetas de cartón, que se llenan y que son mostradas al cliente, de manera que se pueda llegar a un acuerdo sobre la validez de las abstracciones propuestas. Los pasos a seguir para llenar las tarjetas son los siguientes:

- ✓ Encontrar clases.
- ✓ Encontrar responsabilidades.
- ✓ Definir colaboradores.
- ✓ Disponer las tarjetas.

Epígrafe 2.4. Codificación de la solución

En esta fase XP propone tener en cuenta una serie de aspectos como son la disponibilidad del cliente y el desarrollo en pareja para lograr mayores resultados en la implementación del software.

Disponibilidad del cliente: el cliente formó parte del equipo de desarrollo, describió las HU, guío la toma de decisiones, aprobó las versiones del producto y verificó el cumplimiento de los objetivos trazados.
Desarrollo en pareja: toda la implementación fue realizada por dos personas que trabajaron en forma conjunta.

Para llevar a cabo la correcta implementación de las HU se deben definir por parte del equipo de desarrollo las **Tareas de Ingeniería** (TI) que se realizan en cada una de las iteraciones. Las TI también conocidas como tareas de implementación permiten a los desarrolladores obtener un nivel de detalle más avanzado que el que propicia las HU. A continuación se describen las tareas de ingeniería pertenecientes a la primera iteración:

Tarea de Ingeniería	
Número Tarea:1	**Historia de Usuario:** 1, Alojar evidencias.
Nombre Tarea: Implementar RF_Alojar_ evidencias	
Tipo de Tarea: *Desarrollo.*	**Puntos Estimados:**2
Fecha inicio:05/03/2014	**Fecha Fin:** 19/03/2014
Programador Responsable: *Carlos Rafael Madrigal García, Frank Carlos García Lorenzo.*	
Descripción: Se debe implementar el requisito funcional Alojar evidencias, el cual debe permitir a los estudiantes subir las evidencias de las tareas que desarrollaron para posteriormente ser validadas por el profesor que se las asignó.	

Tabla 13 HU Alojar evidencias

Tarea de Ingeniería	
Número Tarea:2	**Historia de Usuario:** 2, Eliminar evidencia

Nombre Tarea: Implementar RF_Eliminar evidencia.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:**2
Fecha inicio: 05/03/2014	**Fecha Fin:** 19/03/2014
Programador Responsable: Carlos Rafael Madrigal García, Frank Carlos García Lorenzo*.*	
Descripción: Se debe implementar el requisito funcional Eliminar evidencia, el cual debe permitir a los estudiantes eliminar las evidencias que este considere innecesarias.	

Tabla 14 HU Eliminar evidencia

Tarea de Ingeniería	
Número Tarea:3	**Historia de Usuario:** 3, Visualizar evidencia
Nombre Tarea: Implementar RF_Visualizar evidencia.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:**2
Fecha inicio: 05/03/2014	**Fecha Fin:** 19/03/2014
Programador Responsable: Carlos Rafael Madrigal García, Frank Carlos García Lorenzo.	
Descripción: Se debe implementar el requisito funcional Visualizar evidencia, el cual debe permitir visualizar las evidencias seleccionadas.	

Tabla 15 HU Visualizar evidencia

Tarea de Ingeniería	
Número Tarea:4	**Historia de Usuario:** 4, Listar evidencias

Nombre Tarea: Implementar RF_Listar evidencias.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:2**
Fecha inicio: 05/03/2014	**Fecha Fin:** 19/03/2014
Programador Responsable: Carlos Rafael Madrigal García, Frank Carlos García Lorenzo.	
Descripción: Se debe implementar el requisito funcional Listar evidencias, el cual debe permitir listar todas las evidencias pertenecientes a un tarea, competencia o rol seleccionado.	

Tabla 16 HU Listar evidencias

Tarea de Ingeniería	
Número Tarea:5	**Historia de Usuario:** 5,Aprobar evidencia
Nombre Tarea: Implementar RF_Aprobar evidencia.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados:2**
Fecha inicio: 05/03/2014	**Fecha Fin:** 19/03/2014
Programador Responsable: Carlos Rafael Madrigal García, Frank Carlos García Lorenzo.	
Descripción: Se debe implementar el requisito funcional Aprobar evidencia, el cual debe permitir aprobar una evidencia en caso de que esta se corresponda con el cumplimiento de la tarea por la cual fue presentada.	

Tabla 17 HU Aprobar evidencias

Tarea de Ingeniería

Número Tarea:6	**Historia de Usuario:** 6,Rechazar evidencia
Nombre Tarea: Implementar RF_Rechazar evidencia.	
Tipo de Tarea: Desarrollo.	**Puntos Estimados**
Fecha inicio: 05/03/2014	**Fecha Fin:** 19/03/2014
Programador Responsable: Carlos Rafael Madrigal García, Frank Carlos García Lorenzo.	
Descripción: Se debe implementar el requisito funcional Rechazar evidencia, el cual debe permitir rechazar una evidencia en caso de que esta no se corresponda con el complimiento de la tarea por la cual fue presentada.	

Tabla 18 HU Rechazar evidencia

Para una correcta comprensión y ejecución de la codificación resulta imprescindible el uso de estándares de codificación.

Los estándares de codificación: son aquellos que permiten entender de manera rápida y sencilla el código empleado en el desarrollo de un software. Garantizan el mantenimiento óptimo de dicho código por parte del programador. A continuación se muestran algunas pautas del estándar definido por el equipo de desarrollo así como ejemplos de su uso.

- Todas las nomenclaturas a utilizar se definirán en idioma español.
- Los identificadores para las variables y los parámetros serán con letras en minúsculas y en caso de ser un nombre compuesto se divide cada palabra con un guión bajo.
- En caso de nombrarse los métodos con una sola palabra, esta se escribe en minúsculas y en caso de ser un nombre compuesto, las palabras que lo conforman se escriben juntas, de la segunda en adelante se escriben con letra inicial mayúscula. Se emplea la notación Camello variante (LowerCamelCase).

- El bundle del módulo Portafolio se escribe con el nombre Portafolio seguido de la palabra Bundle (PortafolioBundle).

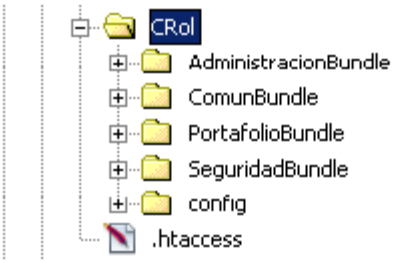


Ilustración 5 Estructura de Carpetas

- Los nombres de las clases se escriben con la primera letra de cada palabra que lo compone en mayúscula, haciendo uso de la notación Camello, con la variante UpperCamelCase.

```
class PersonaController extends BaseController {
```

- Las clases formularios comienzan con el nombre del formulario según su función, seguido de la palabra Type (BuscarEstudianteType).

- Las clases gestoras de negocio comienzan con el nombre del gestor seguido de Gtr(PortafolioGtr).

```
class PortafolioGtr extends BaseGtr {
```

Ilustración 6 Ejemplo de clase gestora

- Las funciones deben ser llamadas sin espacios entre el nombre de la función, el signo de paréntesis y el primer parámetro, espacios entre cada coma por parámetro y sin espacios entre el ultimo paréntesis, el signo de paréntesis cerrado y el signo de punto y coma.

```
epository('PortafolioBundle:Semestre')->findOneBy(array('anno'=>$estudiante->getAnno()));
tRepository('PortafolioBundle:Expediente')->findOneBy(array('portafolio'=>$portafolio, 'semestre'=>$semestre));
```

Ilustración 7 Ejemplo de llamar a una función

- Hacer uso de llaves para ganar en claridad del código.

Al finalizar el presente capítulo se puede arribar a las siguientes conclusiones:

- ✓ Después de realizado el análisis del sistema en términos de solución quedó definida la estructura del sistema, proporcionando una comprensión detallada de las Historias de Usuario, así como de la arquitectura.
- ✓ La propuesta de arquitectura de la aplicación se sustenta en un conjunto de componentes reutilizables que tiene como base el patrón arquitectónico MVC, lo que conforma un sistema robusto y flexible a cambios.
- ✓ El empleo de patrones de diseño garantizó una solución que tiene como premisa la reutilización de código y la resolución de problemas en contextos similares durante la fase de implementación del software.

CAPÍTULO III: Valoración de la efectividad del Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles

En el presente capítulo se describen las pruebas realizadas a cada uno de los módulos de la solución. Estas pruebas son efectuadas con el objetivo de comprobar el cumplimiento de los requisitos. Se muestran los resultados de la aplicación de las técnicas y métricas para validar los requisitos, y también los resultados arrojados después de haber realizado las pruebas de caja negra y caja blanca.

Epígrafe 3.1. Validación de los requisitos

Para verificar que la solución propuesta cumple con todos los requisitos definidos previamente con el cliente, se efectuó un proceso de validación de estos requisitos, donde se emplearon las técnicas siguientes:

- **Construcción de prototipos:** mediante los prototipos se le mostró al cliente un modelo ejecutable de la solución que le permitió tener una visión preliminar de cómo sería el sistema.
- **Generación de casos de prueba de aceptación:** para validar los requisitos funcionales de la solución, se diseñaron casos de pruebas de aceptación para cada una de las Historias de Usuario.

Para medir la calidad de la especificación de requisitos se aplicó una de las métricas definidas por la metodología XP, especificación de requisitos (ER).

Para determinar la ER o ausencia de ambigüedad en los requisitos se realiza la siguiente operación:

ER= Nui/Nr

Dónde:

Nr: es el total de requisitos de especificación.

Nui: es el número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas.

Mientras más cerca de 1 esté el valor de ER, menor será la ambigüedad.

Resultados de la validación de requisitos:

ER=Nui/Nr

ER=44/44

ER=1

Luego de la aplicación de la métrica para medir la calidad de la especificación de los requisitos se obtuvo un resultado satisfactorio con lo que se puede concluir que los requisitos no poseen ambigüedad.

Epígrafe 3.2. Validación del diseño

Para comprobar la calidad del diseño del sistema se emplearon las métricas Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC).

La métrica **TOC** permite medir la responsabilidad, la complejidad de implementación y la reutilización de las clases del diseño. Es importante destacar que para esta métrica, la responsabilidad y la complejidad son inversamente proporcionales a la reutilización, por lo que a mayor responsabilidad y complejidad de implementación de una clase, menor será su nivel de reutilización. Los resultados de la aplicación de estas métricas se muestran en la Ilustración 8.

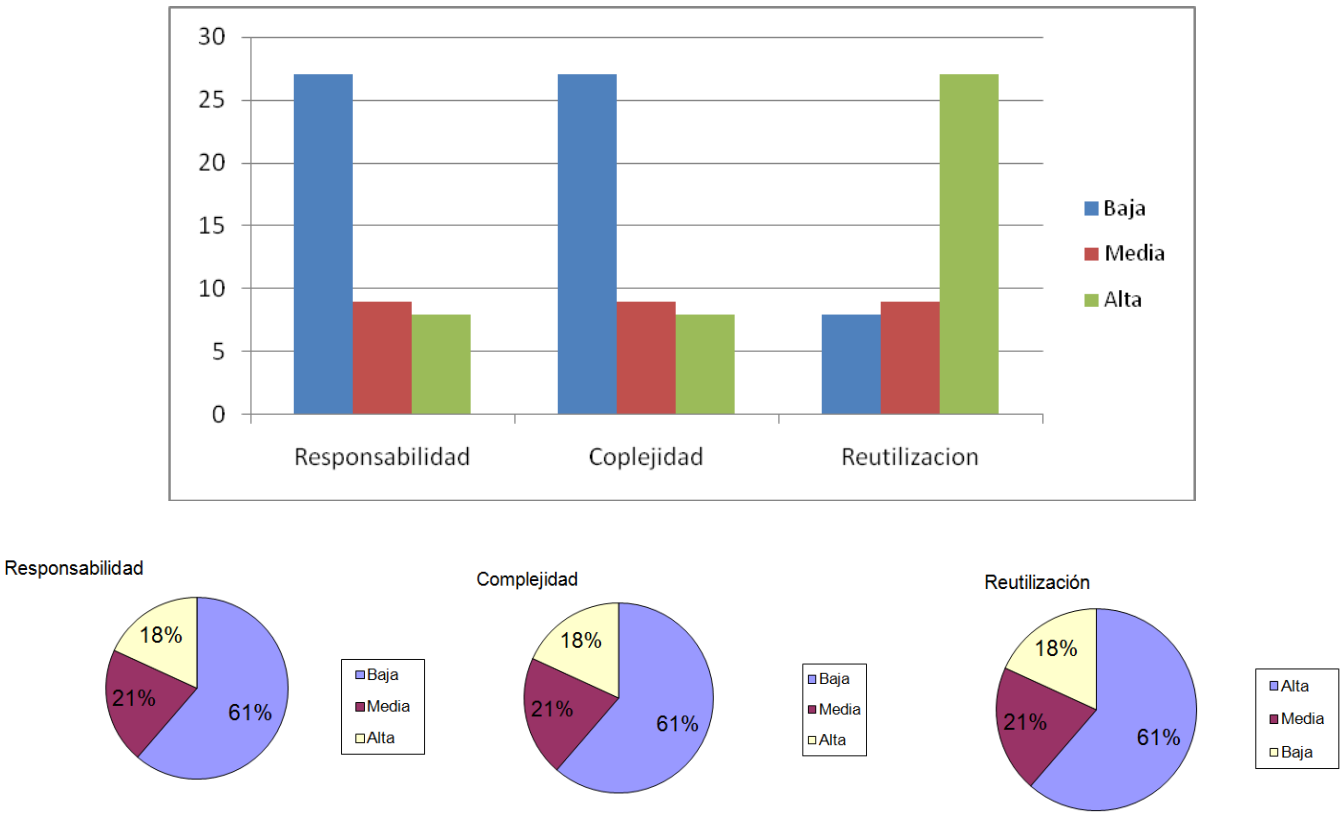


Ilustración 8 Resultado de la aplicación de la métrica TOC

Después de ser aplicada la métrica TOC se puede apreciar en los gráficos anteriores, que las clases del diseño de la solución no presentan una alta sobrecarga en cuanto a los factores expuestos, lo que

beneficia mucho la reutilización de las mismas.

La métrica **RC** permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas de unidad necesarias para probar una clase teniendo en cuenta las relaciones existentes entre ellas.

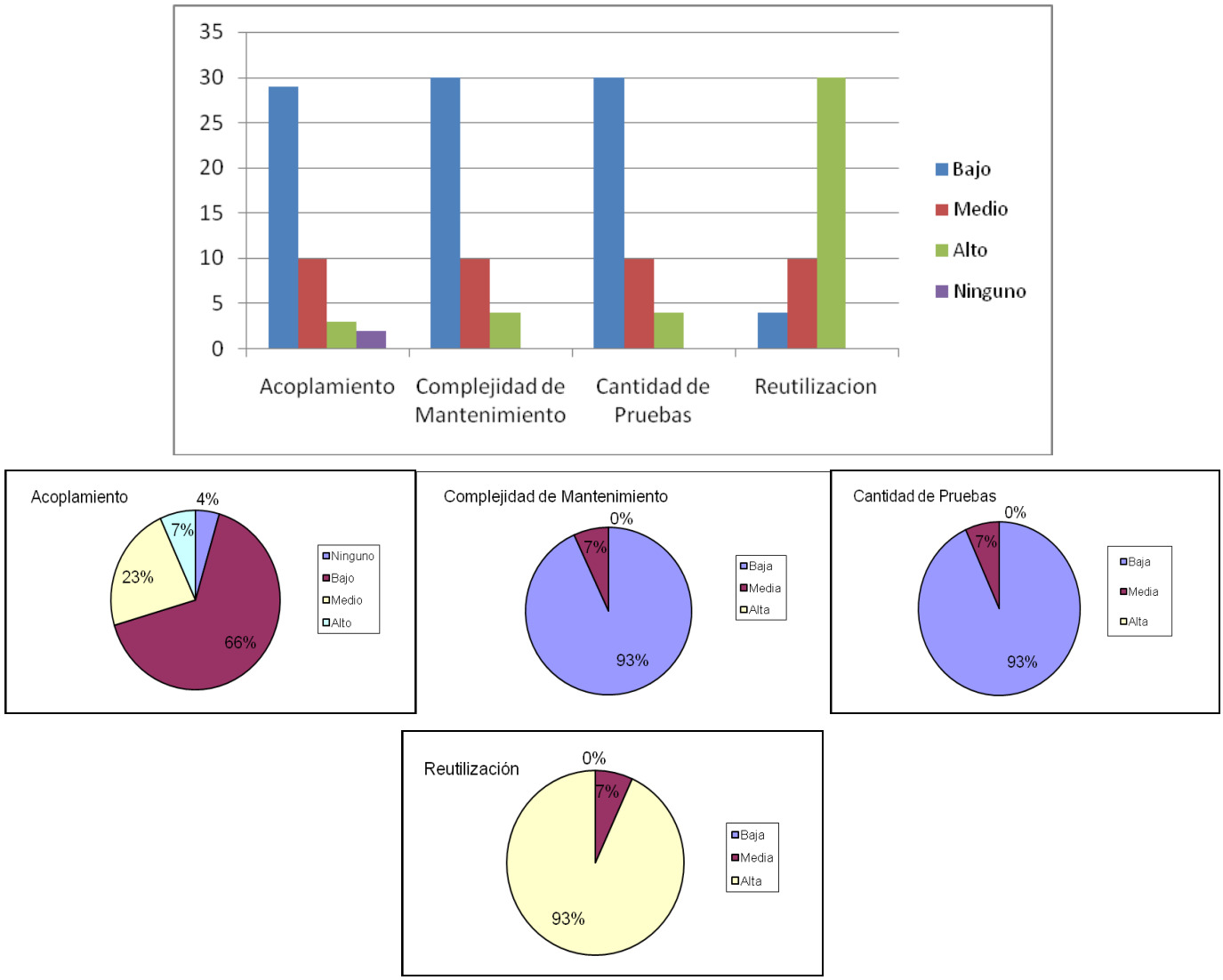


Ilustración 9 Resultados de la aplicación de la métrica RC

En la Ilustración 9 se muestra gráficamente que las clases del diseño de la solución promueven el bajo acoplamiento, que la complejidad de mantenimiento y la cantidad de pruebas no son altas y en

consecuencia el grado de reutilización es alto.

Después de aplicadas las métricas TOC y RC los resultados obtenidos arrojaron que el desarrollo de la solución no es complicado, y que las clases muestran resultados aceptables de acuerdo a los factores medidos anteriormente.

Epígrafe 3.3. Las pruebas realizadas al sistema

Una de las partes fundamentales de XP lo constituye el proceso de pruebas. Mediante este se minimiza la cantidad de errores no detectados y también el tiempo transcurrido entre la introducción de este en la aplicación y su detección. Todo esto contribuye a elevar la calidad del producto y a la seguridad de los desarrolladores a la hora de hacer cambios o modificaciones.

La metodología XP divide las pruebas en dos grupos:

- **Pruebas de aceptación o pruebas funcionales:** destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.
- **Pruebas unitarias:** encargadas de verificar el código y diseñada por los programadores.

Pruebas de Aceptación:

Estas pruebas tienen como objetivo la verificación de los requisitos, por este motivo, los propios requisitos del sistema son la principal fuente de información a la hora de construir las pruebas de aceptación.

Las pruebas de aceptación son creadas a partir de las HU. Durante una iteración la HU seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a probar cuando una HU ha sido correctamente implementada. Una HU puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento. Cada prueba de aceptación representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas. Ninguna HU se considera completa hasta que no supera sus pruebas de aceptación (Riva,J. 2014)**.**

La realización de este tipo de pruebas y la publicación de los resultados deben ser lo más rápido posible, para que los desarrolladores puedan realizar con la mayor rapidez los cambios que sean necesarios.

A cada uno de las Historias de Usuario se le realizaron pruebas de aceptación. Estas pruebas son reflejadas mediante tablas que validan la realización de las mismas. Estas tablas están conformadas por 8

parámetros que dan información acerca de la prueba realizada. Estos parámetros son:

- **Código:** Muestra un identificador para cada prueba realizada (normalmente se pone el nombre del componente seguido de las letras PA: pruebas de aceptación).
- **Nombre de la historia de usuario:** Indica el nombre de la prueba.
- **Descripción:** Se describe cuál es la funcionalidad que se va a medir del componente al que se le esté realizando la prueba.
- **Condiciones de Ejecución:** Este parámetro indica cuáles son las condiciones que se tienen que cumplir para que el componente realice correctamente la funcionalidad que se va a medir.
- **Entrada / Pasos de ejecución:** Indica los pasos a seguir para realizar la prueba.
- **Resultados esperados:** Muestra cuál es el resultado que se obtendría de un correcto funcionamiento de la prueba.
- **Evaluación de la prueba:** Indica el estado de la prueba si es satisfactoria o insatisfactoria.

Caso de prueba de aceptación	
Código:PA3-HU03	**Historia de Usuario:** Autenticar usuario
Nombre: Autenticar usuario.	
Descripción: Para acceder al sistema el usuario debe de autenticarse previamente.	
Condición de ejecución: El usuario de formar parte de los usuarios del dominio .uci.	
Entrada/Pasos de ejecución: - El usuario presiona la opción entrar. - El usuario introduce su usuario. - El usuario introduce su contraseña. - El usuario presiona el botón Entrar.	

Resultado esperado: En caso de que se introduzca algún dato incorrecto se mostrará el siguiente mensaje: usuario o contraseña incorrecta. Si todo esta correcto se entrará a la aplicación sin problemas.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 19 PA Autenticar usuario

Caso de prueba de aceptación	
Código:PA5-HU05	**Historia de Usuario:** Importar plan de formación.
Nombre: Importar plan de formación.	
Descripción: Se importa el plan de formación de un estudiante y son registradas todas las tareas que este contiene.	
Condición de ejecución: Debe de estar seleccionado un estudiante y el plan de formación debe de estar elaborado según la plantilla predefinida para el mismo en una Hoja de Cálculo.	
Entrada/Pasos de ejecución: - El usuario selecciona el estudiante al cual se le van a importar el plan de formación. - El usuario presiona el botón importar plan de formación. - El usuario presiona el botón examinar para importar el plan de formación en formato xlsx. - El usuario presiona el botón aceptar.	
Resultado esperado: Se listan todas las tareas contenidas dentro del plan de formación y son asignadas al estudiante seleccionado.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 20 PA Importar plan de formación

Caso de prueba de aceptación	
Código:PA6-HU06	**Historia de Usuario:** Registrar tarea.
Nombre: Registrar tarea	
Descripción: Se registra una nueva tarea y es asignada al estudiante seleccionado.	
Condición de ejecución: Debe de estar seleccionado un estudiante para poder realizar la operación.	
Entrada/Pasos de ejecución: - El usuario selecciona el estudiante al cual se le va a registrar la tarea. - El usuario presiona el botón registrar tarea. - El usuario selecciona el tipo de actividad. - El usuario selecciona la fecha de inicio. - El usuario selecciona la fecha de culminación. - El usuario introduce la descripción de la tarea. - El usuario introduce las observaciones de la tarea. - El usuario presiona el botón aceptar.	
Resultado esperado: Se registra la tarea y le es asignada al estudiante seleccionado.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 21PA Registrar tarea

Caso de prueba de aceptación

Código:PA6-HU06	**Historia de Usuario:** Alojar artefactos.
Nombre: Alojar Artefactos.	
Descripción: Se registra un artefacto y le es asignado a la tarea seleccionada.	
Condición de ejecución: Debe de estar seleccionada una tarea para poder realizar la operación.	
Entrada/Pasos de ejecución: - El usuario selecciona la tarea a la cual se le va a alojar el artefacto. - El usuario presiona el botón Alojar artefacto. - El usuario selecciona el tipo de artefacto. - El usuario introduce el nombre del artefacto. - El usuario introduce la descripción del artefacto. - El usuario presiona el botón examinar para importar el artefacto. - El usuario presiona el botón aceptar.	
Resultado esperado: Se registra el artefacto y se le es asignado a la tarea seleccionada.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 22 PA Alojar artefactos

Pruebas Unitarias

Con el objetivo de comprobar que las funciones internas del sistema se ejecutan correctamente, se le realizan pruebas al código de las principales funcionalidades de cada uno de los módulos. Para ello se emplearon las Pruebas Unitarias. Estas son una actividad fundamental en XP, que deben ser automatizadas y elaboradas por los desarrolladores antes y durante la implementación de un componente. El objetivo de las pruebas unitarias es el aislamiento de partes del código y la demostración de que estas

partes no contienen errores, y en caso de contener proceder a su pronta eliminación. Luego de terminada la implementación del mismo, asegurando así el buen funcionamiento del componente, es necesario que todas las pruebas sean correctas para poder integrarlo al sistema existente. (Malfará, y otros, 2006)
La implementación de las pruebas unitarias se realizó con la ayuda de la biblioteca PHPUnit, herramienta empleada para realizar las pruebas unitarias al código PHP.

A continuación se muestra una selección de los resultados alojados después de realizar las pruebas en cada una de las iteraciones a las clases más críticas del sistema.

Componente	Iteración	Cantidad de Pruebas	Errores
PortafolioController.php	1	8	3
	2	9	0
	3	5	1
TareaController.php	1	7	4
	2	8	2
	3	7	1
PortafolioGtr.php	1	7	3
	2	9	2
	3	6	0
TareaGtr.php	1	5	2
	2	5	1
	3	5	1
Portafolio.php	1	2	1
	2	1	3
	3	2	0
Tarea.php	1	5	1
	2	2	4
	3	1	2
Expediente.php	1	7	1
	2	5	2

	3	3	0
Soicitud.php	1	5	4
	2	2	2
	3	7	1

Tabla 23 Resultado de la aplicación de las pruebas unitarias

Al finalizar el presente capítulo se puede arribar a las siguientes conclusiones:

- La aplicación de técnicas de validación de requisitos permitió comprobar que los requisitos obtenidos están en correspondencia con las necesidades del cliente.
- Las métricas de validación, las pruebas de aceptación y las pruebas unitarias aplicadas arrojaron resultados favorables asegurando así el buen funcionamiento del sistema.
- Las pruebas permitieron comprobar cada funcionalidad por separado y la integración de estas en el sistema, facilitando la detección de no conformidades para su corrección.

Conclusiones Generales

- Las tecnologías y herramientas seleccionadas, así como la utilización de XP como metodología de desarrollo, en correspondencia con los criterios de selección definidos, contribuyeron a la obtención de una solución informática para la problemática identificada.

- El estudio de los referentes teóricos y de los sistemas existentes contribuyó a profundizar en la fundamentación de la solución; así como a la incorporación de nuevas funcionalidades tales como: generar portafolio y gestionar credenciales.

- Con el empleo de las herramientas y tecnologías seleccionadas, la utilización de patrones de diseño y patrones arquitectónicos, se logró implementar un sistema de acuerdo a los estándares y modelos utilizados en el desarrollo de software que responde a las necesidades del cliente.

- La valoración del sistema mediante la validación de los requisitos, la aplicación de métricas y pruebas de aceptación aportaron valores favorables que contribuyeron a la constatación de su efectividad.

- Como resultado general del trabajo realizado se le dio cumplimiento al objetivo general propuesto, implementándose el Portafolio para la Gestión de Evidencias del Proceso de Certificación de Roles en la Universidad de las Ciencias Informáticas.

Recomendaciones

- Implementar el flujo correspondiente a la segunda etapa del negocio donde se realiza el proceso de confección del tribunal y de la evaluación de la solicitud de certificación de roles.

- Perfeccionar la ponderación establecida por competencia en cada rol a certificar, así como la combinación de dos o más roles en un mismo plan de formación.

Bibliografía

López, Olatz. 2006. Portafolio digital:una innovación docente del sistema evaluativo de los aprendizajes universitarios. 2006.

Albelo Neiro, Denis. 2011. Análisis y diseño del portafolio digital del profesor en la Universidad de las Ciencias Informáticas. 2011.

Almeira. 2011. [En línea] 2011. http://www.roa.unp.edu.ar:8080/graduate/bitstream/123456789/203/1/Tesina%20Arquitectura%20de%20Soft.pdf.

Alonso Menéndez, Evelyn. 2007. Herramientas CASE para el proceso de desarrollo de Software. [En línea] 2007.

Alvarez, Miguel Angel. 2001. Que es html. [En línea] 1 de Enero de 2001. http://www.desarrolloweb.com/articulos/que-es-html.html.

Barberá, Elena. 2005. s.l. : Universidad y Sociedad del Conocimiento, 2005. Vol. 3.

Barberá, Elena, y otros. 2006. Portfolioelectrónico: desarrollo de competencias profesionales en la red. 2006.

Beck, Kent. 1999. Extreme Programming Explained. Embrace Change. s.l. : Pearson Education. , 1999.

Carrasco,J. 2012. Programación en castellano. [En línea] 2012. http://www.programacion.com/articulo/la_nueva_metodologia_219/2.

Cesar Coll, Martín, Elena. 2004. La evaluación del aprendizaje escolar. 2004.

Daniel Díaz, Marisol y López Guzmán, Verónica. 2007. Soluciones de software libre para aplicaciones de bases de datos. 2007.

Daniel Moo Moo, Jose Luis Moo Noh. 2014. Manual de PostgreSQL. 2014.

de la Torre, Aníbal. 2006. [En línea] 2006.

Bibliografía

definicion.de. 2000. definicion.de. definicion.de. [En línea] 2000. http://definicion.de/.

Eguiluz, Javier. 2007. Javascript fácil y rápido con jQuery. [En línea] 24 de Julio de 2007.

Elena Barberà, Guillermo Bautista,Anna Espasa,Teresa Guasch. 2006. Portfolio electrónico: desarrollo de competencias profesionales. 2006.

Franklin. 2010. Symfony para Todos. [En línea] 2010. http://symfony.cubava.cu/introduccion/symfony-2/.

Ga, Francisco José. 2008. Estudio, Comparativa y Aplicación Práctica de Metodologías de Desarrollo de Aplicaciones Web en Java. 2008.

Galloway. 2001. Electronic Portafolios(EP):A how to guide. Technology and teacher education . [En línea] 2001.

García, Fernando. 2013. [En línea] 4 de febrero de 2013.

Gonzalez, Mauris. 2012. Fundamentos de la Ingeniería del Software. 2012.

Guardado, Iván. 2010. web.ontuts. web.ontuts. [En línea] web.ontuts, 6 de julio de 2010. http://web.ontuts.com.

Guardia, Sergio. 2011. Stock. Definición, objetivos y tipos. 2011.

Hernández León, Rolando ALfredo y Coello González, Sayda. 2011. El Proceso de Investigación Científica. Ciudad de La Habana : Editorial Universitaria, 2011.

jacob, Mark Otto. 2014. [En línea] 2014. [Citado el: 24 de 02 de 2014.] http://getbootstrap.com/.

Kioskea. 2014. Kioskea.net. [En línea] Mayo de 2014. http://es.kioskea.net/contents/304-lenguajes-de-programacion.

López, Dr. Miguel Angel. 2002. El portafolio digital como estrategia de autoevaluación. 2002.

Macario Polo, Daniel Villafranca. 2008. Introducción a las aplicaciones Web con Java. [En línea] 2008. [Citado el: 28 de Febrero de 2014.] http://www.inf-cr.uclm.es/www/mpolo/asig/0708/tutorJavaWeb.pdf.

Bibliografía

Madeja, Junta de andalucia. 2008. [En línea] 2008. http://madeja.i-administracion.junta-andalucia.es/servicios/madeja.

Martín Díaz, Luis Benito. 2011. Implementación del Portafolio Digital de la Universidad de las Ciencias Informáticas. 2011.

MDN. 2012. MDN. MDN. [En línea] MDN, 2012. https://developer.mozilla.org.

Navarro, Juan José Moreno. Curso de Software basado en Componentes.

O'Reilly. 2010. Head First Design Patterns por O'Reilly. . 2010.

Ortega, Juan José. 2014. Sistemas gestores de bases de datos comerciales y libres. 2014.

Pablo Martínez, Pablo Ruiz Díaz, Sebastián Waisbrot. 2010. Manual de CodeIgniter. 2010.

Pacheco, Nacho. 2011. Manual de Twig. 2011.

Pérez Cabrera, Leonardo. 2012. Integración de los instrumentos de evaluación para la gestión de las evidencias en la plataforma educativa Zera. 2012.

Pérez Mora, Oscar y Gibert Ginestà, Marc. 2014. Bases de datos en PostgreSQL. s.l. : UOC, 2014.

2014. PHP. [En línea] 2014. http://www.php.net/.

Ravioli, Pablo. Manografias. Manografias. [En línea] http://www.monografias.com.

Riva, José. 2014. Actas de los Talleres de Ingeniería del Software y Base de Datos. [En línea] 2014. http://www.sistedes.es/TJISBD/Vol-3/No-4/PRIS09.pdf..

Roa, María Araceli. 2010. Como elaborar un portafolio de evidencias. 2010.

Rodrigues, Renata. 2013. El desarrollo de la práctica reflexiva sobre el quehacer docente, apoyada en el uso de un portafolio digital, en el marco de un programa de formación para académicos de la Universidad Centroamericana de Nicaragua. 2013.

Sandra, Y Reinaga, Héctor Casas. 2008. Identificación y Modelado de Aspectos Tempranos dirigido por

Bibliografía

Tarjetas de Responsabilidades y Colaboraciones. 2008.

Tinoco Gómez, Oscar, Rosales López, Pedro Pablo y Salas Bacalla, Julio. 2010. Criterios de selección de metodologías de desarrollo de software. 2010.

UDLAP, Puebla, Universidad de las América. 2010. Conoce a los portafolios. [En línea] 2010. http://portafolios.udlap.mx/conoce.aspx..

Universidad Autónoma de Guadalajara. 2010. Desarrollo de productos de utilidad. CARACTERÍSTICAS GENERALES DE UN PORTAFOLIO DE EVIDENCIAS. [En línea] 2010. http://crecea.uag.mx/opciones/pr_porta.htm..

Universidad de Almería. 2003. SISTEMAS DE INFORMACIÓN: ASPECTOS TÉCNICOS Y LEGALES. 2003.

Universidad de Murcia. 2011. Metodologías de desarrollo de software. [En línea] 13 de 10 de 2011. www.um.es/docencia/barzana/IAGP/Iagp2.html..

Universidad Politécnica de Madrid. 2005. Tecnologías y servicios para la sociedad de la información. 2005.

Valdarrama del Pino, Santiago Luis. 2006. Programación extrema en pocos minutos. 2006.

wordreference. 2014. wordreference. wordreference. [En línea] Dicionario de la lengua española, 4 de 06 de 2014. http://www.wordreference.com/definicion/rol.

Yenier Figueroa, Hector Fuentes, Manuel Delgado, Yanisleidy Torres. Octubre 2012. Propuesta de arquitectura de software para proyectos de gestión sobre plataformas web. La Habana Cuba : s.n., Octubre 2012.