Universidad de las Ciencias Informáticas

Facultad 6

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

Título: Asistente planificador para la asignación de actividades.

Autor

Yosbel Emilio Rodríguez Puentes

Tutor

Ing. Alíen Góngora Rodríguez

La Habana, 2015.

"Año 57 de la Revolución"

Declaración de autoría

Declaro ser el único autor de la presente tesis y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio. Para que así conste, firmo la presente a los ___ días del mes de _____ del año 2015.

Yosbel Emilio Rodríguez Puentes

_____

Firma del Autor

Alíen Góngora Rodríguez

_____

Firma del Tutor

Datos de contacto

Síntesis del tutor:

Ing. Alíen Góngora Rodríguez.

Ingeniero en Ciencias Informáticas, UCI, 2008.

Instructor de las asignaturas de Programación 2, Programación 3 y Programación 4 en la UCI.

Desarrollador del Sistema de Gestión Penitenciario de Venezuela (SIGEP).

Desarrollador del Sistema de Identificación.

Desarrollador del Sistema de Control del Acceso.

Desarrollador del Marco Arquitectónico del CENIA.

Desarrollador Del SIGEDOC.

Agradecimientos

Agradezco a todas aquellas personas que colaboraron con el desarrollo de este trabajo, en especial a Alíen Góngora Rodríguez, Adolfo Corona, Ángel López por dedicar parte de su tiempo en la revisión, corrección de errores y brindar ideas para la solución.

Resumen

En la actualidad el proceso de planificación y asignación de actividades a los profesores de la Universidad de las Ciencias Informáticas se realiza de forma manual. Dicho proceso resulta engorroso y en algunos casos la información no es recibida a tiempo, propiciando de esta forma incumplimiento en la realización de las actividades planificadas.

A raíz de esta situación, se identificó la necesidad de realizar un Sistema Informático que actúe como asistente planificador con el fin de lograr de manera automatizada la planificación, el control y asignación de actividades a cada usuario en dependencia de sus afectaciones de manera tal que se pueda optimizar el tiempo y recurso empleado en este proceso.

Este asistente se desarrollará sobre la utilización de diversas tecnologías libres tales como el lenguaje de programación PHP y el Sistema de Gestor de Base de Datos PostgresSQL, de manera general con el objetivo de obtener un sistema sencillo y con una interfaz amigable capaz de llevar a cabo los procesos de gestión de información antes mencionados.

Palabras clave: Planificación, control, asignación, actividades, gestión de información

Abstrac

At present the process of planning and allocation of activities to teachers at the University of Information Sciences is done manually. This process is cumbersome and in some cases the information is not received in time, thus promoting noncompliance in performing the planned activities. Following this, It requires a computer system serving as assistant planner in order to achieve automated way the planning, control and allocation of activities each user depending on his affectations so identified so as to optimize the time and resources used in this process. This wizard will take place on the use of various open source technologies such as PHP programming language and the System Manager PostgresSQL database, generally with the aim of obtaining a simple system with a friendly interface and able to carry out management processes aforementioned information.

Índice general

Índice de Tablas

Índice de Figuras

Introducción

En la actualidad el sistema de trabajo de las empresas ha sufrido diversos cambios, se puede decir que la empresa del ayer no guarda relación alguna con la de nuestros días, en cuanto a la planificación de las actividades y la forma de delegar las mismas. Hoy día las instituciones son objeto de transformaciones que de una forma u otra afectan el accionar cotidiano de cada una de las empresas que existen; junto a esto, cada elemento relacionado con cada actividad debe acoplarse para obtener el ajuste óptimo de la empresa a estos cambios.

Cada factor productivo debe desempeñarse de forma eficaz en aras de lograr los objetivos de la empresa de la manera más eficiente posible y es aquí donde se llega a realizar el tratamiento del recurso humano como capital humano, es a este factor a quien debe considerarse de real importancia para aumentar sus capacidades y poner a su disposición los medios y recursos necesarios para elevar sus aptitudes al punto tal en que se encuentre como un factor capaz de valerse por sí mismo y entregarle lo mejor de sí a su trabajo, sintiéndose conforme con lo que realiza y cómo es reconocido. La clave del éxito está en una gestión acertada y planificación adecuada de las actividades y las personas que participan en ella.

La empresa tras el logro de una correcta planificación y asignación de tareas vislumbra elementos de utilidad en los diferentes tipos de sistemas basados en la información que influyen en su accionar, el de la información que se encuentra almacenada en el sistema como de la propia generada por el usuario ya una vez implicada con dichas afectaciones, para llevar a cabo análisis de tendencias y proporcionar las recomendaciones más ajustadas en cada momento, teniendo en cuenta las afectaciones planificadas por el mismo con antelación y así logrando la máxima cohesión entre la actividad y el usuario que la desempeña, de tal forma que perfile a la empresa a lograr una máxima cantidad de objetivos en un tiempo específico.

La creación del “Proyecto Futuro”, nombrado posteriormente Universidad de las Ciencias Informáticas (UCI),el cual cuenta con un plan de estudio diferente al resto de las universidades del país, incluye a la producción como parte del proceso enseñanza-aprendizaje, de esto se derivan las misiones más importantes de este centro de altos estudios: formar profesionales comprometidos con su Patria, altamente calificados en la rama de la informática, producir software y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación, en aras de lograr solidificar la vinculación del estudio, el trabajo y contribuir a un correcto funcionamiento de las dos aristas.

Sin embargo, se percibe la existencia de una **situación problemática**, en los diferentes departamentos docentes de la Universidad de las Ciencias Informáticas, la información crece de manera sensible y el trabajo con la misma, generalmente se gestiona, en la actualidad, en formatos duros, lo cual conlleva a una agobiante, y en ocasiones, incorrecta gestión y manipulación de la información, la pérdida de tiempo en la búsqueda de datos, existe el riesgo de cometer errores, la coincidencia o solapamiento de actividades incurriendo en el incumplimientos de las mismas, lo que evidencia una serie de problemas los cuales atentan contra el adecuado desempeño y funcionamiento del departamento docente , afectando la toma de decisiones y la asignación de actividades a los profesores que laborar en el departamento docente según sus afectaciones.

En aras de delimitar la amplia gama de respuestas que entraña la situación problemática se plantea el siguiente **problema a resolver:**

¿Cómo gestionar la información en los departamentos de la Universidad de las Ciencias Informáticas para lograr un adecuado cumplimiento de las actividades?

A raíz de este cuestionamiento, se precisa como **objeto de estudio** al proceso de gestión de la información mediante el empleo de un sistema informático para la realización de actividades.

Una vez definido el objeto de estudio se propone como **campo de acción**: el proceso de planificación, control y asignación de actividades al personal que labora en el departamento docente de técnicas de programación de la facultad 1 de la Universidad de las Ciencias Informáticas.

Se precisa como **objetivo general** de la investigación: Desarrollar un sistema informático planificador, para la realización de actividades, que mejore la planificación, control y asignación de tareas en los diferentes departamentos de la UCI, especialmente el departamento docente de técnicas de programación en la facultad 1.

En correspondencia con esto, se plantea como **objetivos específicos**:

- ✓ Realizar la modelación del negocio, para identificar los principales procesos que intervienen en el proceso de asignación de actividades.
- ✓ Realizar el levantamiento de requisitos que permita captar las condiciones, capacidades y cualidades que debe tener el sistema.
- ✓ Realizar el análisis y diseño del sistema.

- ✓ Implementar una solución informática para la correcta planificación y asignación de actividades.
- ✓ Validar la solución propuesta mediante la realización de pruebas de software.

Para alcanzar los objetivos previamente planteados se definen las siguientes tareas de investigación:

- ✓ Estudio del estado del arte de los sistemas o herramientas gestores de información.
- ✓ Entrevistas con el personal de diferentes departamentos de la UCI.
- ✓ Identificación de los entornos de desarrollo y herramientas de modelado para el desarrollo del componente.
- ✓ Selección de la metodología a utilizar para el desarrollo del componente.
- ✓ Selección de los lenguajes de programación y marcos de trabajo a utilizar.
- ✓ Desarrollo de una base de datos con todos los datos necesarios para el manejo de la información así como para el almacenamiento de la misma.
- ✓ Implementación de las funcionalidades definidas para la solución propuesta.
- ✓ Realización de las pruebas de software, para validar la solución.
- ✓ Documentar el sistema (métodos y modos de uso).

Métodos científicos de la investigación:

Para la realización del proceso investigativo se recaban los métodos teóricos y empíricos, que a continuación se reseñan:

Métodos Teóricos:

- ✓ Histórico – Lógico: Este método facilita la realización de un análisis acerca de la evolución y tendencias actuales de diversas herramientas, entre ellas, PHP y *PostgresSQL*, así como la evolución de las diferentes herramientas empleadas para la gestión de la información.
- ✓ Analítico – Sintético: Se emplea para poder guiarse por el análisis de las teorías, documentos, libros, artículos, permitiendo el procesamiento de la información para arribar a diversas conclusiones prácticas y teóricas acerca del trabajo. Además el uso de este método, ha permitido identificar y analizar los elementos y términos más significativos dentro del campo que comprende el proceso de

gestión de la información, permitiendo extraer lo esencial de la bibliografía consultada en lo referente a la gestión del conocimiento y la información.

- ✓ Modelación: Este método permitió mediante la elaboración del modelo de dato, mostrar gráficamente la relación de los conceptos más significativos relacionados entre sí y para la representación de abstracciones con vistas a explicar la realidad del problema (1).

Métodos Empíricos:

- ✓ Entrevista: Este método se usó como medio para obtener mayor información sobre cómo se lleva a cabo el proceso de gestión de la información en la UCI. Se han consultado diferentes proyectos, en los cuales se usan herramientas las cuales realizan la asignación de actividades a diferentes usuarios, haciendo uso de la gestión de la información. Se le aplicó la entrevista a la analista de la Suite de Gestión de Proyectos (GESPRO) para obtener información y referencias acerca de la realización de gestión de procesos. También para el desarrollo de este método fueron entrevistados de manera informal, programadores con experiencia en la utilización del lenguaje PHP y PostgresSQL, el cliente, y diversas personas con experiencia en la redacción de documentos de trabajos de diplomas.

Estructuración del contenido.

El presente trabajo está compuesto por tres capítulos, los cuales se describen brevemente a continuación.

Capítulo 1. Fundamentación Teórica: En este capítulo se lleva a cabo el proceso de fundamentación teórica de la investigación para un mejor entendimiento de la situación problemática planteada, abarcando los principales conceptos relacionados con la Gestión de la Información y su uso en la UCI. También se realiza un análisis del estado del arte a partir de las diferentes soluciones informáticas existentes, así como la selección de la metodología, tecnologías, lenguajes de implementación y herramientas a utilizar.

Capítulo 2. Características del Sistema: En este capítulo se lleva a cabo la descripción de los procesos del negocio que intervienen en la asignación de actividades, así como la captura de los requerimientos con que debe cumplir el sistema.

Capítulo 3. Diseño e implementación: En este capítulo se realiza el diseño del sistema, ofreciendo una visión del mismo, el modelado del negocio y se obtienen los requerimientos definidos para la aplicación en el capítulo anterior y el flujo de trabajo, donde se describen las partes de la aplicación desde el punto de vista de su arquitectura, así como los ficheros que se utilizaran. Además se realizan las pruebas de funcionalidad, de aceptación y de carga o estrés a la aplicación.

Capítulo 1: Fundamentación teórica

En el presente capítulo se definen una serie de conceptos relacionados con los procesos de gestión de la información, que marcan la base de la investigación científica. Además se realiza un análisis del estado del arte sobre los principales sistemas que existen en la actualidad cuyo objetivo principal es llevar a cabo el proceso de gestión de la información, resaltando sus principales ventajas y desventajas. También se realiza un análisis para seleccionar las herramientas, tecnologías y lenguajes de programación a utilizar, se seleccionará la metodología que guiará el proceso de desarrollo del sistema.

1.1 Conceptos fundamentales

1.1.1 Gestión de la información

En la actualidad, las grandes tendencias que definen la gestión de las organizaciones llevan implícito un aspecto fundamental, que en muchas ocasiones no se considera o se deja de lado, como lo es la información, sus sistemas y proceso de gestión. De manera general la gestión de la información es el proceso que se encarga de suministrar los recursos necesarios para la toma de decisiones, la obtención de nuevos conocimientos, así como para mejorar los procesos, productos y servicios de las organizaciones (2).

La gestión de la información es a su vez un proceso mediatizado por un conjunto de actividades que permiten la obtención de información, lo más pertinente, relevante y económica posible, para ser empleada en el desarrollo y el éxito de una determinada organización. Este proceso es además un subproceso y por tanto requiere acción, decisión y evaluación.

Entre los principales objetivos de la gestión de la información se encuentran:

- Determinar responsabilidades para el uso efectivo, eficiente y económico de información.
- Asegurar un suministro continuo de la información.
- Determinar las necesidades internas de información (relativas a las funciones, actividades y procesos administrativos de la organización) y satisfacerlas competitivamente.

Garantizar una mejor planificación de los recursos y las actividades desempeñadas por una organización.

1.1.2 Características de los Sistemas de Gestión de Información

De manera general, entre las principales características que debe tener un Sistema de Gestión de Información (SGI) se encuentran las que se describen a continuación:

- ✓ Administración de un sistema de base de datos: esto se refiere a la capacidad de un SGI para almacenar datos y facilitar la recuperación de información por los usuarios del sistema.
- ✓ Informes: esto se refiere a que un SGI es bueno en dependencia de la forma en la cual genera informes, pues la capacidad de producir información que ayuda en el proceso de toma de decisiones y la planificación es clave en estos sistemas. La mayoría de los SGI proporciona plantillas de informes mientras que otros ofrecen la posibilidad de crear informes específicos y guardarlos como una plantilla para que otros los utilicen.
- ✓ Acceso abierto: esto hace referencia a que un SGI que permite el acceso a su arquitectura brinda a la organización la posibilidad de cumplir con mayor facilidad con las regulaciones externas y requisitos internos. De forma específica esto permite que una organización pueda integrar más fácilmente un SGI con los sistemas ya existentes en la misma.
- ✓ Integración: esto se refiere a que un buen SGI permite su integración con los sistemas heredados, lo cual propicia que se conserven de forma exitosa los recursos existentes y empleados en la organización.
- ✓ Escalabilidad: esto se refiere a que debido a que no todas las organizaciones requieren la oferta completa de algunos SGI, la escalabilidad se convierte en un factor de suma importancia pues en ocasiones solo es necesario una pequeña parte de los mismos. (3)

La utilización adecuada de los SGI permite acceder a una certificación externa de reconocido prestigio internacional, simplificando la documentación necesaria, ya que al ser la única, esta brinda mayor transparencia y facilita el manejo de la misma (4).

1.1.3 Planificación

El proceso de planificación es un factor importante dentro de los SGI. Este se puede definir como un proceso que sirve para organizar determinadas ideas y trabajar sobre una misma línea de trabajo con el fin de fin de resolver un conflicto determinado en una organización u empresa, es un plan que se lleva a la acción para

mejorar un cierto aspecto dentro de la organización. Este proceso incluye entre sus objetivos mejorar la toma de decisiones con la meta de concretar un fin buscado. Por ello, es preciso que una estrategia de planificación tenga en consideración la situación presente y todos aquellos factores ajenos y propios que puedan generar repercusiones para lograr su fin. (5)

1.2 Estado del arte

Desde el punto de vista empresarial, los sistemas de información tienen como propósito perfeccionar las actividades llevadas a cabo en una organización, y así alcanzar ventajas competitivas. Entre algunos de los sistemas analizados se encuentran los siguientes:

Gespro: Suite de Gestión de Proyectos

Ecosistema de software para la Dirección Integrada de Proyectos

Desarrollado por: Laboratorio de Investigaciones en Gestión de Proyectos, Universidad de las Ciencias Informáticas.

Ecosistema GESPRO: Combina el uso de una solución informática para la dirección integrada de proyectos y un sistema de formación especializada en gestión de proyectos. Posibilita la planificación y el control y seguimiento de los proyectos y de los recursos asociados a los mismos en alineación con la proyección estratégica de las organizaciones.

GESPRO se comercializa bajo un modelo de negocios basado en servicios entre los que se destacan: personalización de la solución informática, despliegue en diferentes escenarios, actualizaciones, soporte, consultoría y formación especializada (6).

Gestión del conocimiento y la información en el Polo de Gestión Universitaria:

Diseño metodológico descriptivo para realizar el diagnóstico de la situación actual, cimentado en los resultados de este se elabora la propuesta de procedimiento para la gestión de conocimiento y la información en el Polo de Gestión Universitaria de la Facultad 1. Se definen los responsables de su implementación y las acciones a ejecutar. La solución incluye una planificación que le da un orden lógico a las actividades, culminando con un grupo de instrucciones dirigidas al control.

Ecured: Enciclopedia colaborativa en la red cubana.

Proyecto basado en la gestión de la información con el objetivo de crear y difundir el conocimiento, con todos y para todos, garantizando las peticiones de los usuarios con el fin de obtener la información correcta, en la forma adecuada, para la persona indicada, al costo correcto, en el momento oportuno, en el lugar indicado para tomar la acción precisa.

Creado tras la necesidad de aportar conocimiento universal en un dominio .cu para de esa manera ser accesible a la inmensa mayoría de los cubanos. Su finalidad es generar servicios y productos que respondan a las necesidades y sobrepasen las expectativas de los usuarios, posibilitando que el sistema trabaje eficientemente y económicamente a la vez. La Ecured es un Sistema de Gestión de Información que aprovecha al máximo sus recursos de información en función de la mejora continua y de la toma de decisiones organizacional a todos los niveles jerárquicos desde la cúspide estratégica hasta la base operativa, cumpliendo con el principal pilar de la gestión de la información, logrando la planificación (7).

1.3 Ambiente de desarrollo

A continuación se describe el ambiente de desarrollo para la implementación del Asistente Planificador de actividades.

1.3.1 Metodología de desarrollo

Una metodología hace referencia al conjunto de procedimientos racionales, utilizados para alcanzar una gama de objetivos que rigen en una investigación científica o tareas que requieren habilidades, conocimientos o cuidados específicos. Alternativamente puede definirse como el estudio o elección de un método pertinente para un objetivo determinado. En otras palabras la metodología es la ciencia que nos enseña a dirigir determinado proceso de manera eficiente y eficaz para alcanzar los resultados deseados y tiene como objetivo darnos la estrategia a seguir en el proceso (8).

Esto llevado al desarrollo de software, se refiere a una serie de tácticas, prácticas, herramientas y documentos principales que auxilian a los desarrolladores de software en sus esfuerzos por realizar nuevos sistemas de información. Está integrada por períodos que guiarán a los desarrolladores a elegir las técnicas adecuadas en cada instante del proyecto y además a planificarlo, gestionarlo, controlarlo y evaluarlo.

1.3.2 Metodologías tradicionales

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar. Un ejemplo de metodología tradicional es el Proceso Unificado de Desarrollo (RUP) (9).

1.3.3 Metodologías ágiles

- ✓ Las metodologías ágiles se basan en la planificación adaptativa donde los requisitos y soluciones evolucionan mediante la colaboración de grupos autos organizados y multidisciplinarios. Existen principios recogidos en el llamado Manifiesto Ágil, que justifican su aparición:
- ✓ Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.
- ✓ Es más importante crear un producto software que funcione que escribir documentación exhaustiva.
- ✓ La colaboración con el cliente debe prevalecer sobre la negociación de contratos (10).

1.3.4 Metodología XP

La metodología XP (*Extreme Programming*) es ágil y está 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. XP 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 (11)

El ciclo de vida ideal de XP consiste de cuatro fases: planeación, diseño, codificación y pruebas, tal y como se muestra en la figura a continuación (12):

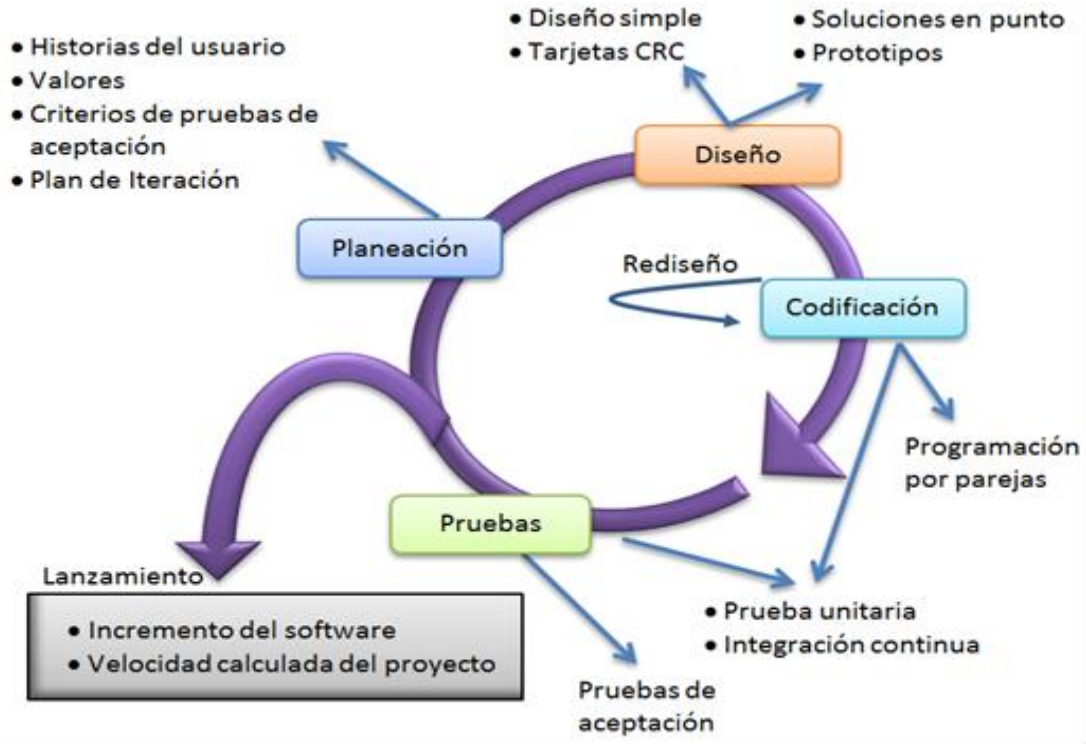


FIG. 1: CICLO DE VIDA DE LA METODOLOGÍA XP

Fase 1: Planeación

La actividad de planeación comienza creando una serie de historias de usuarios que describen las características y la funcionalidad requeridas para el software que se construirá.

Fase 2: Diseño

Esta fase sigue de manera rigurosa el principio de mantenerlo simple. Además el diseño ofrece una guía de implementación para una historia de usuario como está escrita y emplea el uso de tarjetas CRC para identificar y organizar las clases que son relevantes para el incremento del software. Aquí tiene lugar además, la creación de prototipos operacionales cuyo propósito es reducir riesgos cuando comience la verdadera implementación.

Fase 3: Codificación

Un concepto clave dentro de esta fase es la programación en pareja, es decir, se recomienda a dos personas que trabajen juntas en una estación de trabajo de computadora para crear el código de una historia de usuario. Esto proporciona un mecanismo para la resolución de problemas en tiempo real y el aseguramiento de la calidad en las mismas condiciones. También alienta a que los desarrolladores se mantengan centrados en el problema que se tiene a mano.

Fase 4: Pruebas

Aquí se hace referencia a dos aspectos fundamentales: las pruebas de unidad y las pruebas de aceptación; las pruebas de unidad que se crean deben implementarse con un marco de trabajo que permita automatizarlas. Esto apoya una estrategia de regresión de prueba cuando el código se modifica.

Por otra parte las pruebas de aceptación, también llamadas pruebas del cliente, las especifica el cliente y se enfocan en las características generales y la funcionalidad del sistema, elementos visibles y revisables por el cliente. Las pruebas de aceptación se derivan de las historias de usuario que se han implementado como parte de un lanzamiento de software.

Es importante decir que el cliente tiene que estar presente y disponible todo el tiempo para el equipo. Gran parte del éxito del proyecto XP se debe a que el cliente es quien conduce constantemente el trabajo debido a que el mismo forma parte del equipo de desarrollo, hacia lo que aportará mayor valor de negocio, y los programadores que pueden resolver de manera inmediata cualquier duda asociada, dándole respuestas apropiadas a los cambios y realizando un software sencillo, libre de complejidad, siguiendo los pasos que define la metodología para la obtención de un resultado satisfactorio (13).

1.4 Herramientas y tecnologías

1.4.1 Herramienta CASE

La industria de computadoras ha desarrollado un soporte automatizado para el desarrollo y mantenimiento de software. Este es llamado herramientas CASE (Ingeniería de Software Asistida por Computadora o *Competer Aided Software Engineering*). Se pueden definir como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida del desarrollo de un software.

Entre los beneficios más significativos de las herramientas CASE se encuentran los siguientes:

- ✓ Facilidad para la revisión de aplicaciones.
- ✓ Soporte para el desarrollo de prototipos de sistemas.
- ✓ Generación de código.
- ✓ Mejora en la habilidad para satisfacer los requerimientos del usuario.
- ✓ Soporte interactivo para el proceso de desarrollo.

Actualmente existen una gran variedad de herramientas CASE para el proceso de desarrollo de software pero la seleccionada para el análisis y diseño del componente fue la herramienta *Visual Paradigm for UML* 8.0 (14).

1.4.2 Herramientas y tecnologías para el desarrollo de la aplicación

Es una herramienta CASE para el desarrollo de aplicaciones utilizando modelado UML, la cual es ideal para el trabajo de ingenieros de software, analistas y arquitectos de sistemas que están interesados en la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos.

El propósito de elegir *Visual Paradimg*, es que esta herramienta ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software y además, por sus características, entre las cuales se encuentran las siguientes:

- ✓ Disponibilidad en múltiples plataformas (Windows, Linux).
- ✓ Uso de un lenguaje estándar, común a todo el equipo de desarrollo, lo cual facilita la comunicación.
- ✓ Capacidades de ingeniería directa e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de múltiples versiones, con diferentes especificaciones.
- ✓ Licencia gratuita y comercial.
- ✓ Soporta aplicaciones web.
- ✓ Fácil de instalar y actualizar.
- ✓ Editor de figuras.

- ✓ Generador de informes.
- ✓ Compatibilidad entre ediciones.

Esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa, ayudando así a la búsqueda de soluciones para los requisitos. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la ingeniería de software (15).

1.4.1 Lenguaje de programación

El término lenguaje de programación en informática, hace referencia a cualquier lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por un ordenador o computadora. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

1.4.2 Lenguaje de modelado: UML

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, ojear, configurar, mantener, y controlar la información sobre tales sistemas. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

Entre los principales objetivos que están detrás del desarrollo de UML, el primero y más importante, es que UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores.

No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática. Incorpora buenas prácticas de diseño, tales como la encapsulación, separación de los temas, y la captura de la intención del modelo construido. Pretende abordar los problemas actuales del desarrollo de software, tales como gran tamaño, distribución, concurrencia, patrones, y desarrollo en equipo (16).

Un objetivo final de UML es ser tan simple como fuera posible, pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesitan construir.

1.4.3 HTML 5.0

El Lenguaje para Marcación de Hipertexto (HTML)

HTML5 es la quinta revisión importante del lenguaje básico de la *World Wide Web*; HTML, diseñado para ser utilizable por todos los desarrolladores de web abierta.

De manera general HTML5 es la evolución por un lado de lo que ha sido hasta ahora el lenguaje de marcado HTML4 y por otro lado, de la API *Document Object Model* 2 (DOM2). A través de HTML5 vamos a disponer de nuevas API´s que van a intentar ayudar a los desarrolladores a generar aplicaciones web más dinámicas y ricas. Dentro de sus ventajas:

- ✓ Es nativo, y por tanto independiente de plugins de terceros. Es decir, no pertenece a nadie, es de código abierto.
- ✓ Es más semántico, con etiquetas que permiten clasificar y ordenar en distintos niveles y estructurar el contenido. Además, incorpora metadatos que favorecen la accesibilidad.
- ✓ El código es más simple, lo que permite hacer páginas más ligeras que se cargan más.
- ✓ Ofrece una compatibilidad mayor con los navegadores de dispositivos móviles.
- ✓ Posibilita la inserción de videos y audio de forma directa.
- ✓ Tiene la capacidad de ejecutar páginas sin estar conectado.
- ✓ Dispone de nuevas capacidades CSS3 como posibilidad de usar cualquier fuente o tipografía en HTML, columnas de texto, opacidad, trasparencia, contraste, saturación, brillo, animaciones de transición y trasformación, bordes redondeados, gradientes y sombras.

Permite realizar diseños adaptables a distintos dispositivos (web, tablets, móviles).

1.4.4 CSS 3.0

Las hojas de estilo en cascada hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas. Su aplicación más común es dar estilo a páginas web escritas en lenguaje HTML y XHTML, pero también puede ser aplicado a cualquier tipo de documentos XML.

CSS tiene una sintaxis muy sencilla, que usa unas cuantas palabras claves tomadas del inglés para especificar los nombres de varias propiedades de estilo. Una hoja de estilo se compone de una lista de reglas. Entre las principales ventajas que ofrece CSS3 es que se obtiene un mayor control de la presentación del sitio al poder tener todo el código CSS reunido en uno, lo que facilita su modificación. Al poder elegir el archivo CSS que deseamos mostrar, puede aumentar la accesibilidad. Se consigue hacer más legible el código HTML al tener código CSS aparte (siempre que no se use estilos en línea). Las novedades de CSS3 nos permiten ahorrarnos tiempo y trabajo al poder seguir varias técnicas (bordes redondeados, sombra en el texto y sombra en las cajas por mencionar algunas) sin necesidad de usar un editor gráfico.

1.4.5 JavaScript 1.8

El lenguaje *JavaScript* surgió a partir de la necesidad de ampliar las posibilidades de HTML, permitiendo a los desarrolladores crear acciones en sus páginas web. Además, es un lenguaje que no requiere de compilación ya que funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos. JavaScript tiene la ventaja de ser incorporado en cualquier página web, puede ser ejecutado sin la necesidad de instalar otro programa para ser visualizado. Entre algunas de características se puede mencionar que es un lenguaje basado en acciones que posee menos restricciones. Gran de la programación de este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, así como cargas de páginas (17).

1.4.6 PHP 5.3.6

PHP (acrónimo de *Hypertext Preprocessor*) es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Con PHP no se encuentra limitado a resultados en HTML. Entre las habilidades de PHP se incluyen: creación de imágenes, archivos PDF y películas Flash (usando *libswf y Ming*) sobre la marcha. También puede presentar otros resultados, como XHTM y archivos XML. PHP puede autogenerar estos archivos y almacenarlos en el sistema de archivos en vez de presentarlos en la pantalla. Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía Web para una base de datos es una tarea simple con PHP.

1.4.7 CodeIgniter 2.1.3

CodeIgniter es un marco de trabajo para el desarrollo de aplicaciones Web usando PHP. Este permite el desarrollo de proyectos mucho más rápidos que si se escribiera código desde cero. Provee una rica colección de librerías para las tareas necesarias más comunes. Permite concentrarse en el desarrollo del proyecto en cuestión, minimizando la cantidad de código necesaria para realizar las tareas. Usa el patrón de diseño arquitectónico Modelo-Vista-Controlador como paradigma de arquitectura de desarrollo, el cual separa en tres capas distintas: la representación de datos, el interfaz de usuario y el controlador de eventos respectivamente (18).

1.4.8 PostgresSQL 9.2

Como gestor de base de datos se utilizó PostgreSQL, el cual se distribuye bajo la licencia de PostgreSQL, un liberal de licencia de código abierto, similar a las licencias BSD (*Berkeley Software Distribution*) o MIT (*Massachusetts Institute of Technology*). Ofrece ventajas entre las que se tienen: (19)

- ✓ Instalación ilimitada: es frecuente que las bases de datos comerciales sean instaladas en más servidores de lo que permite la licencia. Algunos proveedores comerciales consideran a esto la principal fuente de incumplimiento de licencia. Se puede usar PostgreSQL, pues no estaría violando acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.
- ✓ Modelos de negocios más rentables con instalaciones a gran escala.
- ✓ No existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento.
- ✓ Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.

PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente solo se podían ver en productos comerciales de alto calibre.

1.4.9 Jquery 1.11

JQuery es un nuevo tipo de biblioteca o marco de trabajo de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX al sistema. JQuery, al igual que otras librerías, ofrece una serie de

funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. Es decir, con las funciones propias de esta librería se logran grandes resultados en menos tiempo y espacio. La gran ventaja es que permite cambiar el contenido de la página Web sin necesidad de recargarla, utilizando DOM y AJAX de manera extremadamente sencilla gracias a su sintaxis.

1.5 Conclusiones parciales

En este capítulo se detallaron las condiciones y problemas actuales que rodean al objeto de estudio enmarcado en este trabajo, para obtener una mejor compresión y visión del problema permitiendo formar la base para una solución eficiente. Se abordaron conceptos generales propios del tema central de la investigación, importantes para el desarrollo del proyecto y significativos para la comprensión de las características del sistema. Además se seleccionó la metodología XP para dirigir el proceso de desarrollo del Sistema, así como *Java Script*, PHP y HTML como lenguaje de programación usando el marco de trabajo CodeIgniter.

Capítulo 2: Características del sistema

En el presente capítulo se ofrece una descripción de la solución propuesta con el objetivo de solucionar el problema inicialmente planteado. Se define el modelo conceptual en el que se reflejan los principales conceptos que se deben abarcar para desarrollar el Asistente Planificador de Actividades. Posteriormente se plantean los requisitos funcionales, no funcionales y las historias de usuarios correspondientes.

2.1 Flujo actual de los procesos de negocio

Inicialmente en los departamentos de la UCI la gestión de la información se realiza de forma manual, existiendo perdida de la misma y falta de comunicación, ya que esta es almacenada en formatos duros, este proceso se realiza también mediante la vía de correo electrónico, a través del cual se les informa a los profesores de sus actividades pertinentes, pero trae consigo mucha inestabilidad e inseguridad, ya que existe poca planificación y vinculación de los mismos con sus actividades.

Se decide entonces el desarrollo de una aplicación para la planificación de dichas actividades en correspondencia con la afectación de cada uno de los profesores que radican en los departamentos, así informatizar dicho proceso con el fin de lograr de manera automatizada la planificación, el control y asignación de actividades a cada usuario en dependencia de sus afectaciones de manera tal que se pueda optimizar el tiempo y recurso empleado en este proceso y dar cumplimiento a todas las actividades asignadas.

2.1.1 Modelo de datos

Un modelo de datos captura los tipos más importantes de objetos en el contexto del sistema. Las entidades representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema. Muchos de los objetos pueden obtenerse de una especificación de requisitos mediante la entrevista con el cliente. En el siguiente modelo de dato se describen las entidades más importantes dentro del contexto del sistema, mostrando los principales conceptos con los que se trabaja y se definen las relaciones fundamentales: (20)

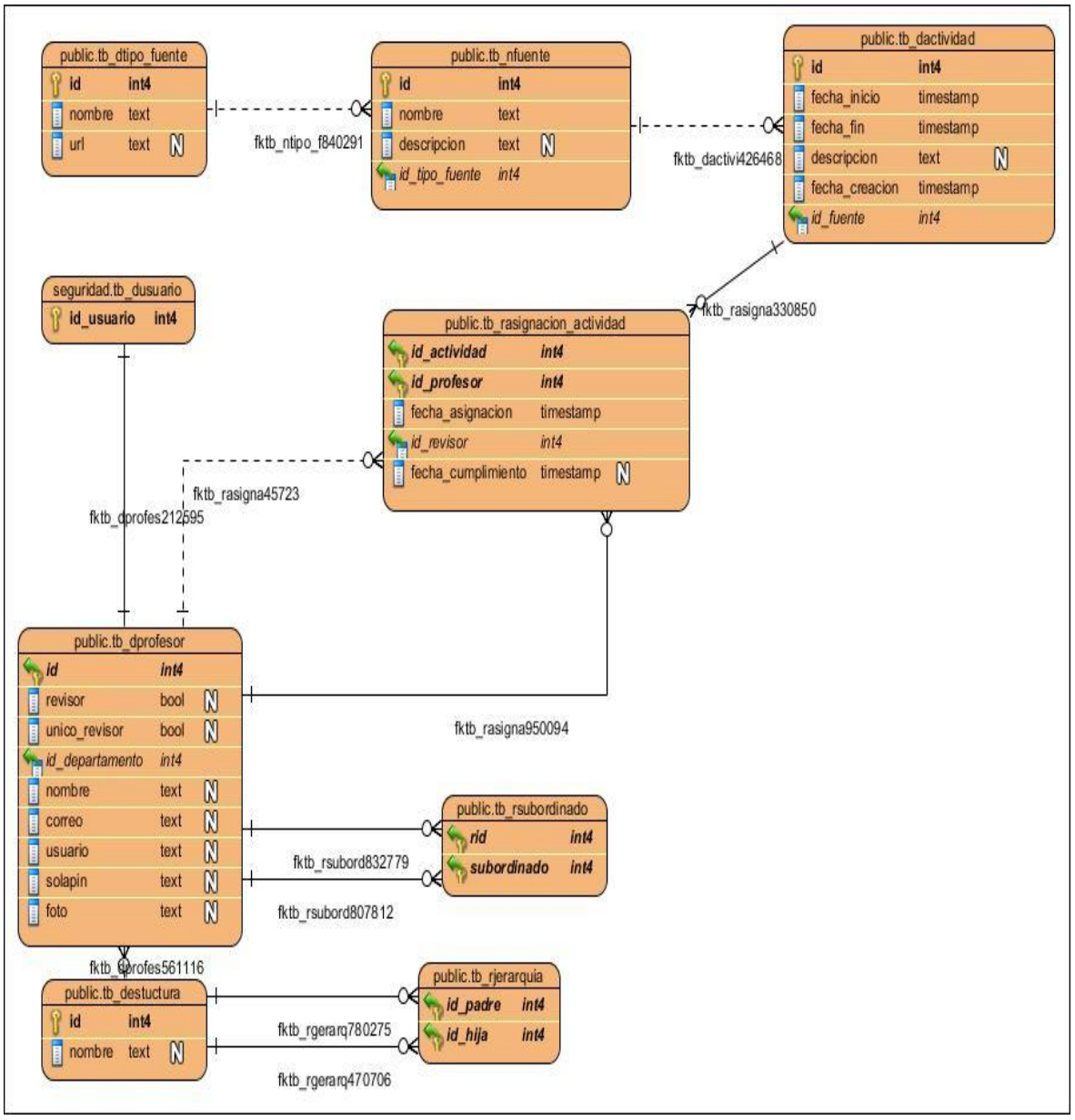


FIG. 2: MODELO DE DATOS

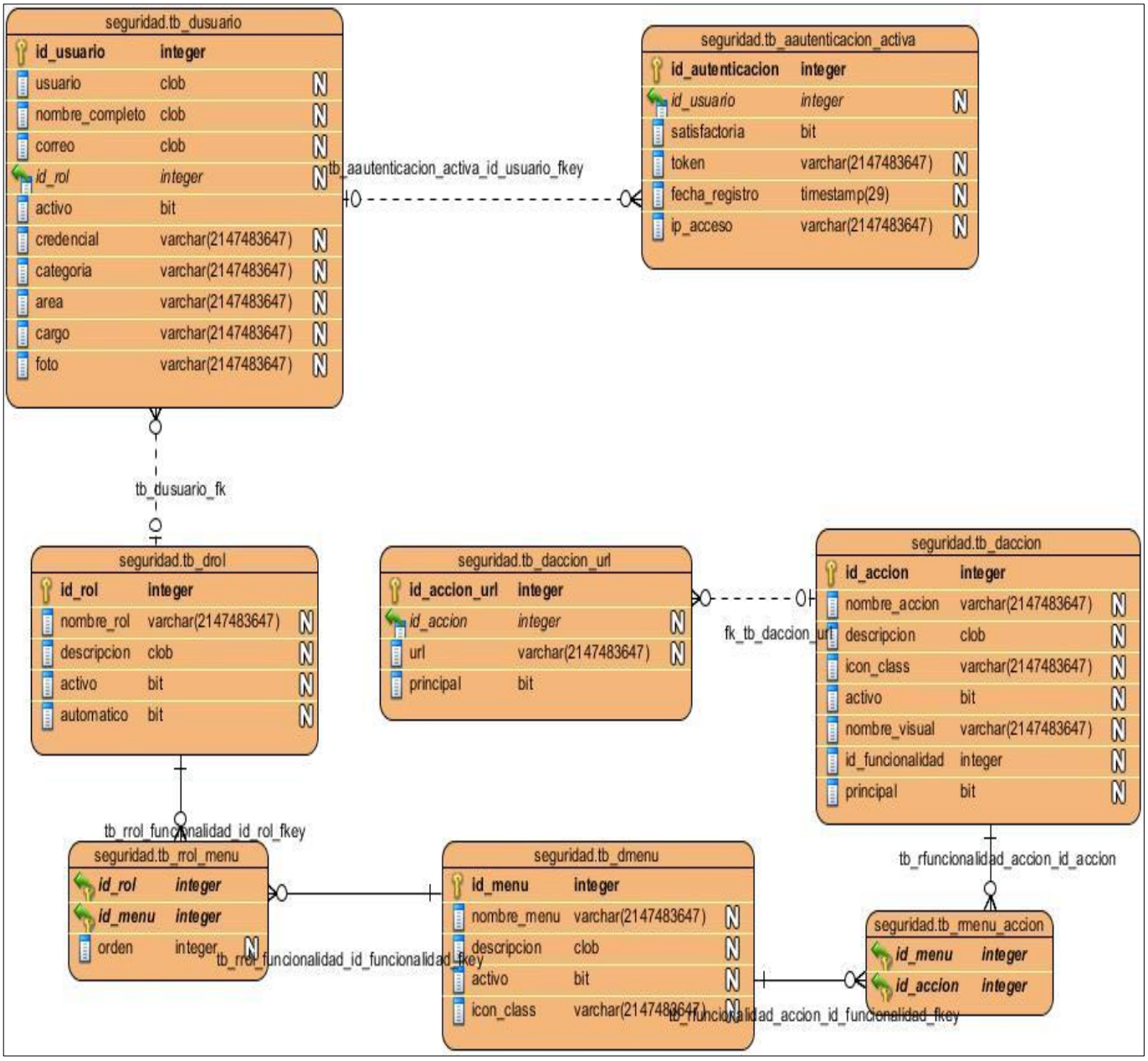


FIG. 3: MODELO DE DATOS

2.2 Funcionalidades del software

Las funcionalidades del software son la descripción de los servicios y restricciones de un sistema, es decir, lo que el software debe hacer y bajo qué circunstancias debe hacerlo.

2.2.1 Funcionalidades

Una funcionalidad define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica (21).

Realizadas varias consultas al cliente se definieron las que se listan a continuación:

RF1: Crear actividad.

RF2: Eliminar actividad.

RF3: Asignar actividad.

RF4: Modificar actividad.

RF5: Dar cumplimiento a la actividad.

RF6: Listar actividades.

RF7: Ver detalles de una actividad.

RF8: Listar notificaciones.

RF9: Leer notificación.

RF10: Listar profesores.

RF11: Adicionar profesores.

RF12: Modificar profesores.

RF13: Eliminar profesores.

RF14: Adicionar departamentos por facultad.

RF15: Modificar departamentos por facultad.

RF16: Eliminar departamentos por facultad.

RF17: Autenticarse en el sistema.

RF18: Salir del sistema.

RF19: Listar funcionalidad existente.

RF20: Registrar la funcionalidad.

RF21: Modificar la funcionalidad.

RF22: Eliminar la funcionalidad.

RF23: Listar menú existente en el sistema.

RF24: Registrar menú existente en el sistema.

RF25: Modificar menú existente en el sistema.

RF26: Eliminar menú existente en el sistema.

RF27: Listar rol existente en el sistema.

RF28: Registrar rol existente en el sistema.

RF29: Modificar rol existente en el sistema.

RF30: Eliminar rol existente en el sistema.

RF31: Asociar menú a rol.

RF32: Listar usuario existente en el sistema.

RF33: Eliminar usuario existente en el sistema.

RF34: Modificar usuario existente en el sistema.

RF35: Mostar los datos del usuario.

RF36: Listar actividades de revisor.

RF37: Realizar evaluación.

RF38: Listar reportes.

2.2.2 Requisitos no funcionales del sistema

Los requisitos no funcionales son las condiciones que necesita un software para que su rendimiento sea óptimo. También hacen que sea aceptado por los clientes ya que son los encargados de hacer que el software sea de interés, pues definen la rapidez con que responde el mismo al usuario, la interfaz gráfica que se usa, entre otros aspectos (22).

A continuación se muestran los requisitos no funcionales definidos para la solución de la investigación:

Usabilidad:

RNF1: El sistema debe presentar un menú lateral y una barra de iconos flotantes que permitan el acceso rápido a la información por parte de los usuarios.

RNF2: Las vistas del sistema deben indicar en cada momento la acción que se está realizando, así como los íconos deben estar representados por una imagen acorde a la acción que se realiza.

RNF3: Sólo se mostrarán a los usuarios aquellas acciones o informaciones del menú lateral a las que por su responsabilidad o rol dentro del negocio necesiten acceder.

Software:

RNF4: Para el despliegue del sistema se debe contar en el servidor de bases de datos con PostgreSQL 9.2

RNF5: Para el despliegue del sistema se debe contar en el servidor de aplicaciones web con: PHP v 5.6 con las librerías php5-ldap, php5-gd, php5-mcrypt, php5-pgsql, php5-xsl, php5-openssl; Apache 2.2 con el módulo *rewrite* activado.

RNF6: Para el uso del sistema se requiere como mínimo una PC cliente con el navegador web Mozilla Firefox 26.0.

Confiabilidad:

RNF7: El tratamiento de las excepciones permitirá un seguimiento hasta guardar información, acerca del lugar dónde se produjo el error y de los parámetros utilizados en el sistema que lo provocaron. Cuando ocurre una excepción el sistema mostrará un mensaje explicativo del error ocurrido.

RNF8: El sistema puede permanecer inactivo durante 10 minutos. Al cumplirse este término se cerrará la sesión teniendo que autenticarse el usuario nuevamente.

RNF10: El sistema soportará la conexión simultánea de varios usuarios, con un promedio de 1000 y un máximo de 3000.

Interfaz:

RNF11: La interfaz es sencilla con colores suaves a la vista y sin cúmulo de imágenes u objetos que distraigan al cliente del objetivo.

Interfaz de comunicación:

RNF12: La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo de conexión TCP/IP.

RNF13: La comunicación entre el cliente y el servidor de aplicaciones se realiza a través del protocolo HTTPS.

Hardware

RNF14: Para la ejecución del sistema se requiere que la PC cliente tenga las siguientes características de hardware: Pentium 4 o superior, 512 MB RAM y 200 MB disco duro disponible como mínimo.

Requisitos legales de derecho de autor y otros:

RNF15: El sistema debe ser sometido a un análisis legal por parte de los abogados y personal autorizado con vistas a declarar su autenticidad y evitar restricciones legales para su uso y comercialización; así mismo se debe proceder a una evaluación y certificación por parte del cliente del producto.

2.2.3 Especificación de funcionalidades

A continuación se muestra la especificación de funcionalidades para cada funcionalidad del sistema:

TABLA 1: ESPECIFICACIÓN DE FUNCIONALIDADES

Código	Nombre	Prioridad	Complejidad
1	Crear actividad.	Alta	Alta
2	Eliminar actividad.	Alta	Baja
3	Asignar actividad.	Alta	Alta
4	Modificar actividad.	Alta	Baja
5	Dar cumplimiento a la actividad.	Alta	Alta
6	Listar actividades.	Alta	Alta
7	Ver detalles de una actividad.	Alta	Baja
8	Listar notificaciones.	Alta	Alta
9	Leer notificación.	Media	Baja
10	Listar profesores.	Media	Baja
11	Adicionar profesores.	Alta	Baja
12	Modificar profesores.	Alta	Baja
13	Eliminar profesores.	Alta	Baja
14	Adicionar departamentos por facultad.	Alta	Alta
15	Modificar departamentos por facultad.	Alta	Baja
16	Eliminar departamentos por facultad.	Media	Alta
17	Autenticarse en el sistema.	Alta	Alta
18	Salir del sistema.	Alta	Baja
19	Listar funcionalidades existentes.	Alta	Alta
20	Registrar las funcionalidades.	Alta	Alta
21	Modificar las funcionalidades.	Baja	Alta
22	Eliminar las funcionalidades.	Baja	Baja
23	Listar los menuces existentes en el sistema.	Baja	Alta
24	Registrar los menuces existentes en el sistema.	Alta	Alta

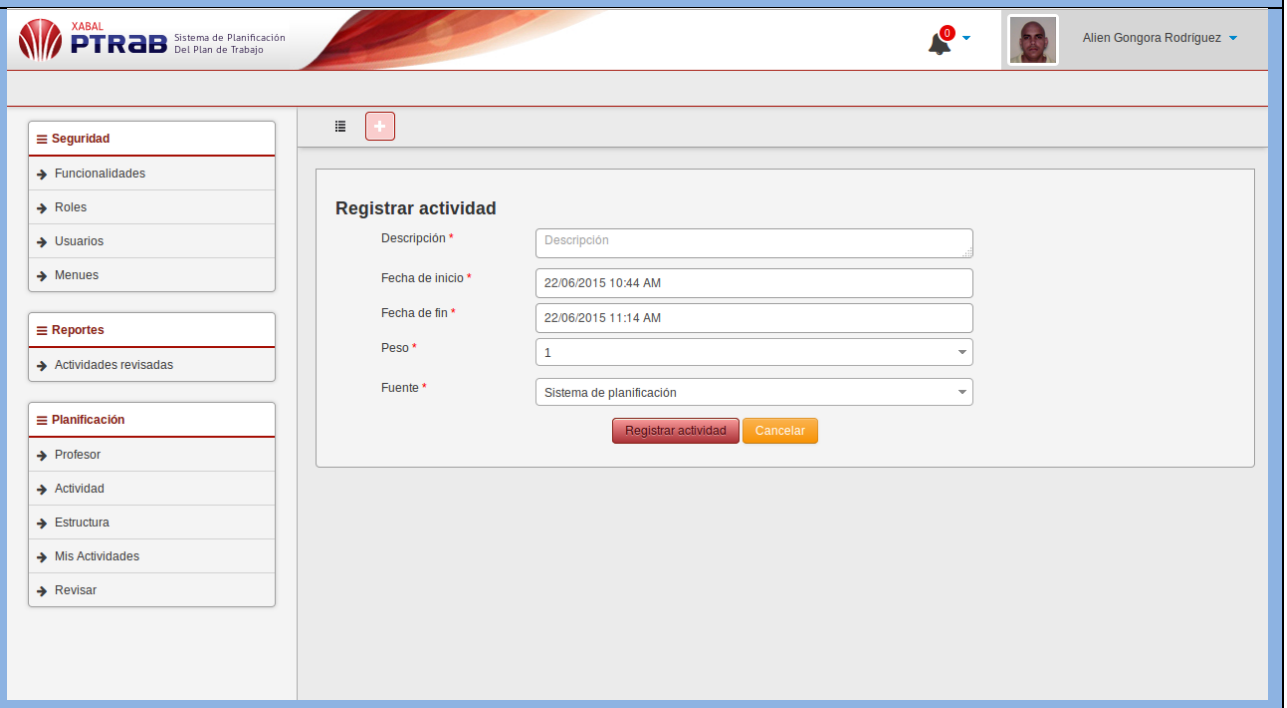
25	Modificar los menuces existentes en el sistema.	Alta	Baja
26	Eliminar los menuces existentes en el sistema.	Baja	Baja
27	Listar los roles existentes en el sistema.	Media	Alta
28	Registrar los roles existentes en el sistema.	Media	Alta
29	Modificar los roles existentes en el sistema.	Alta	Baja
30	Eliminar los roles existentes en el sistema.	Alta	Baja
31	Asociar menú a rol.	Alta	Alta
32	Listar los usuarios existentes en el sistema.	Media	Media
33	Eliminar los usuarios existentes en el sistema.	Media	Media
34	Modificar los usuarios existentes en el sistema.	Alta	Media
35	Mostar los datos del usuario	Media	Alta
36	Listar actividades de revisor.	Alta	Alta
37	Realizar evaluación.	Alta	Alta
38	Listar reportes.	Alta	Alta
39	Listar actividades de revisor.	Alta	Alta

2.2.4 Descripción de funcionalidades

La descripción de las funcionalidades es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto, ya que estos son un reflejo detallado de las necesidades de los clientes o usuarios del sistema. A continuación, se muestran dos funcionalidades específicas. Para consultar las especificaciones de las restantes funcionalidades, diríjase a los Anexos.

TABLA 2: DESCRIPCIÓN DE FUNCIONALIDAD: REGISTRAR ACTIVIDAD

No.	Nombre	Descripción	Complejidad	Prioridad para el cliente
1	Crear actividad	Los usuarios en dependencia de su roll como profesores del departamento, tendrán opciones de crear una determinada actividad, insertando la	Alta	Alta

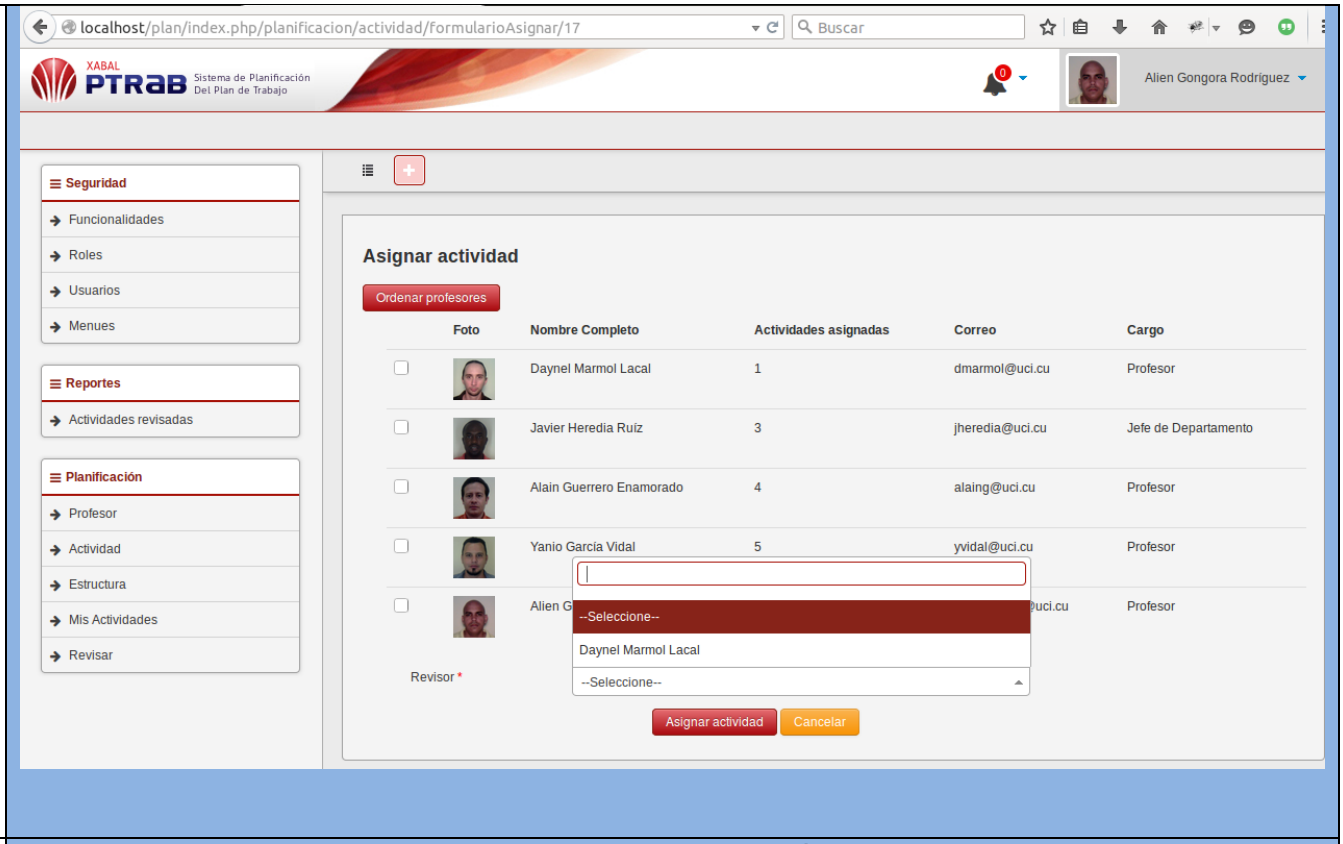
		fecha de inicio, fecha de fin, descripción de la actividad, la fecha de creación de la misma, el peso de importancia de la misma y la fuente de donde proviene la misma.		
	Prototipo			
				
	Prototipo Crear actividad			

	Campos	**Tipos de datos**	**Reglas o restricciones**
	Descripción	texto	Campo obligatorio/ Admite cualquier cadena de carácter
	Fecha Inicio	Int	Campo obligatorio/No procede
	Fecha fin	Int	Campo obligatorio/No procede
	Peso	No procede	Campo obligatorio
	Fuente	No procede	Campo obligatorio/No procede
	Observaciones	En caso de cancelar, se le muestra al usuario una notificación de confirmación, si realmente desea salir.	

TABLA 3: DESCRIPCIÓN DE FUNCIONALIDAD: ASIGNAR/SUGERIR ASIGNACIÓN DE ACTIVIDAD

No	Nombre	Descripción	Complejidad	Prioridad para el cliente
3	Asignar actividad	Los usuarios en dependencia de su roll como profesores del departamento, tendrán opciones de asignar una determinada actividad a un determinado profesor, vinculando la actividad con el profesor seleccionado, especificando la fecha de asignación de dicha actividad y el revisor de la actividad, y fijando la fecha de cumplimiento. Existen dos variantes de asignar la actividad, actividades por designación: son aquellas actividades como las del horario que ya están asignadas a una determinada persona, y existe la asignación de las mismas teniendo en cuenta la optimización de la asignación según la importancia de la actividad (peso para lograr una mayor eficiencia de la planificación.	Alta	Alta
	Prototipo Sugerir asignación de actividad			

Prototipo Asignar actividad



Prototipo Asignar/Sugerir asignación de actividad

Campos	Tipos de datos	Reglas o restricciones
Revisor	No procede	Campo obligatorio
Observaciones:	Se realizan varias formas de asignar una determinada actividad, vinculando la actividad con el profesor que te sugiere la búsqueda de la aplicación que debe ser el profesor indicado para la realización de la misma, la asignación si debe ser manual, interfiere el factor humano, definiendo el revisor	

2.3 Fase de planificación

En esta fase las personas del negocio necesitan determinar las posibles funcionalidades de la aplicación, generalmente representadas por historias de usuarios, y establecer prioridades y estimaciones de su realización. Se determinan además los elementos que van a componer cada una de las versiones así como la fecha de entrega de las mismas.

2.3.1 Descripción de requisitos

El primer paso de cualquier proyecto que siga la metodología XP es definir las historias de usuario con el cliente. La Historia de Usuario (HU) es una técnica utilizada para especificar los requisitos o funcionalidades del software, se utiliza específicamente en XP. Las mismas son tarjetas de papel donde el cliente describe brevemente las características que el sistema debe poseer, sean funcionales o no funcionales. Estas historias de usuario tienen un tratamiento muy dinámico y flexible, en cualquier momento pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. El contenido de estas debe ser concreto y sencillo (23)

La metodología XP no propone un formato específico para las HU. A continuación se muestra la estructura general que tendrán las HU.

TABLA 4: MODELO DE HISTORIA DE USUARIO

HISTORIA DE USUARIO	
Número:HU- #	Nombre de Historia de Usuario
Usuario	
Prioridad en Negocio: (Alta/Media/Baja).	Iteración asignada
Riesgo en Desarrollo: (Alta/Media/Baja).	Puntos estimados (1 punto equivale a 1 día de programación).
Descripción: Breve explicación de lo que hará la historia de usuario	
Observaciones:	

TABLA 5: HU-17: AUTENTICARSE EN EL SISTEMA

HISTORIA DE USUARIO	
Número: HU-17	Nombre: Autenticarse en el sistema.
Usuario: Trabajadores (profesores del departamento).	

Prioridad en Negocio: Alta.	Iteración Asignada: 4
Riesgo en Desarrollo: Alta.	Puntos Estimados: 2
Descripción: Los usuarios podrán interactuar con en el sistema, tras ingresar usuario y contraseña.	
Observaciones: Tras validar los datos ingresados y ser insertado en la base de datos, el administrador le concede el rol al usuario, que solicita el acceso al sistema, en caso de no encontrarse en la base de dato su acceso en negado, mostrando la información.	

TABLA 6: HU-1: CREAR ACTIVIDAD

HISTORIA DE USUARIO	
Número: HU-1	Nombre: Crear actividad.
Usuario: Trabajadores (profesores del departamento).	
Prioridad en Negocio: Alta.	Iteración Asignada: 1
Riesgo en Desarrollo: Alta	Puntos Estimados: 2
Descripción: Los usuarios en dependencia de su roll como profesores del departamento, tendrán opciones de crear una determinada actividad, insertando la fecha de inicio, fecha de fin, descripción de la actividad, la fecha de creación de la misma, peso de la actividad y la fuente de donde proviene la misma.	
Observaciones: Queda adicionada la actividad, tras mostrar la información de confirmación.	

TABLA 7: HU-3: ASIGNAR ACTIVIDAD

HISTORIA DE USUARIO	
Número: HU-3	Nombre: Asignar actividad.
Usuario: Trabajadores (profesores del departamento).	
Prioridad en Negocio: Alta.	Iteración Asignada:2
Riesgo en Desarrollo: Alta	Puntos Estimados: 2
Descripción: Los usuarios en dependencia de su roll como profesores del departamento, tendrán opciones de asignar una determinada actividad a un determinado profesor, vinculando la actividad con el profesor seleccionado, especificando la fecha de asignación de dicha actividad y el revisor de la actividad, y fijando la fecha de cumplimiento. Existen actividades por designación, existe la asignación de las mismas teniendo en cuenta la optimización de la asignación según la importancia de la actividad para lograr una mayor eficiencia.	
Observaciones: Queda asignada la actividad, tras mostrar la información de confirmación.	

TABLA 8: HU-36: LISTAR ACTIVIDADES DE REVISOR

HISTORIA DE USUARIO	
Número: HU-36	Nombre: Listar actividades de revisor.
Usuario: Trabajadores (profesores del departamento).	
Prioridad en Negocio: Alta.	Iteración Asignada: 2
Riesgo en Desarrollo: Alto.	Puntos Estimados: 1
Descripción: Se muestran todas las actividades con sus campos, para la posterior revisión.	
Observaciones:	

TABLA 9: HU-37: REALIZAR EVALUACIÓN

HISTORIA DE USUARIO	
Número: HU-37	Nombre: Realizar evaluación.
Usuario: Trabajadores (profesores del departamento).	
Prioridad en Negocio: Alta.	Iteración Asignada: 2
Riesgo en Desarrollo: Alto.	Puntos Estimados: 1
Descripción: Se selecciona la actividad, la cual se desee revisar, y se le define una evaluación, (bien, regular, mal).	
Observaciones:	

2.4 Estimación de esfuerzo

En este paso se determina el costo asociado a la implementación de cada historia de usuario, utilizando como medida el punto. Se determinó que un punto equivale a 1 día de programación, donde el equipo de desarrollo trabaja bajo condiciones adecuadas y sin interrupciones. La estimación del costo de la implementación de las historias de usuarios es establecida por los programadores, tomando valores que por lo general no exceden los 3 puntos. Esto permite al equipo llevar el registro de la velocidad de desarrollo del proyecto establecida en puntos por iteración (12).

La velocidad del proyecto es una medida que representa la rapidez con la que se desarrolla el proyecto. Estimarla es muy sencillo, basta con contar el número de historias de usuario que se pueden implementar en una iteración; de esta forma, se sabrá el cupo de historias que se pueden desarrollar en las distintas iteraciones. Usando la velocidad del proyecto se controlarán que todas las tareas se puedan desarrollar en el tiempo del que dispone la iteración.

TABLA 10: ESTIMACIÓN DE ESFUERZO POR PUNTOS

No	Historias de Usuarios	Puntos Estimados
1	Crear actividad.	2
2	Eliminar actividad	2
3	Asignar actividad	2

4	Modificar actividad	2
5	Dar cumplimiento a una actividad	1
6	Listar actividades	0.5
7	Ver detalles de una actividad	0.5
8	Listar notificaciones	0.5
9	Leer notificaciones	0.5
10	Listar profesores	0.5
11	Adicionar profesores	2
12	Modificar profesores	0.5
13	Eliminar profesores	0.5
14	Adicionar departamentos por facultad	1
15	Modificar departamentos por facultad	0.5
16	Eliminar departamentos por facultad	0.5
17	Autenticarse en el sistema	2
18	Salir del sistema	1
19	Listar funcionalidades existentes	0.5
20	Registrar funcionalidades existentes	2
21	Modificar funcionalidades existentes	0.5
22	Eliminar funcionalidades existentes	0.5
23	Listar menuces existentes en el sistema	0.5
24	Registrar menuces existentes en el sistema	1
25	Modificar menuces existentes en el sistema	0.5
26	Eliminar menuces existentes en el sistema	0.5
27	Listar los roles existentes en el sistema	0.5

28	Registrar los roles existentes en el sistema	2
29	Modificar los roles existentes en el sistema	1
30	Eliminar los roles existentes en el sistema	0.5
31	Asociar menú a rol.	2
32	Listar los usuarios existentes en el sistema	0.5
33	Eliminarlos usuarios existentes en el sistema	1
34	Modificar los usuarios existentes en el sistema.	2
35	Mostar los datos del usuario	1
36	Listar actividades de revisor	1
37	Realizar evaluación	1
38	Listar reportes	2

2.4.1 Plan de iteraciones

Luego de que las historias de usuarios fueron descritas e identificadas, así como estimado el esfuerzo que cada una de ellas conlleva se procede a implementar el sistema. Este plan muestra el orden en que serán implementadas las historias de usuarios y en qué iteración se desarrollará, así como el tiempo que demorará dicha implementación y la fecha para las liberaciones del producto.

El plan de iteraciones especifica las HU que serán implementadas en cada iteración del sistema, donde el tiempo de 1 iteración debe oscilar de 2 a 3 semanas, trabajándose 40 horas por semana y el proyecto final no debe excederse de los 3 meses. Para la realización del sistema se definieron cuatro iteraciones, las cuales se describen a continuación (23).

TABLA 11: PLAN DE ITERACIONES

Iteración	Descripción de la Iteración	Orden de las HU a implementar	Duración
1	En esta iteración se implementarán las historias de usuarios de mayor prioridad. Al finalizar dicho proceso se contarán con las funcionalidades descritas en las historias de usuario implementadas.	1,2,4,5,11,20,28 ,31	3 semanas
2	Al inicio de la iteración, como propone la metodología XP, se revisan y corrigen las posibles no conformidades o historias de usuario que no superen las pruebas de aceptación de la iteración anterior. Luego se desarrollan las historias de usuario descritas.	3,6,7,8,9,10,12, 13,19,23,32,33, 34,35,36,37	2 semanas 3 días
3	Al igual que en la iteración anterior se revisan y corrigen las posibles no conformidades o historias de usuario que no superen las pruebas de aceptación de la iteración anterior, siempre tratando de suprimir las no conformidades de la iteración anterior Luego se desarrolla la historia de usuario descrita.	14,15,16,21,22, 24,25,26,27,29, 30	1 semana 2 días
4	Al igual que en la iteración anterior se revisan y corrigen las posibles no conformidades o historias de usuario que no superen las pruebas de aceptación de la iteración anterior, siempre tratando de suprimir las no conformidades de la iteración anterior Luego se desarrolla la historia de usuario descrita.	17,18,38	1 semana

2.4.2 Tareas de ingeniería

Las tareas de ingeniería surgen de la división de las historias de usuario en tareas más sencillas para planificar el trabajo desde un punto de vista técnico. Cada una de las historias de usuario se transformará

en tareas que serán desarrolladas por los programadores dentro del equipo de desarrollo, aplicando la práctica de la programación en parejas. Cada tarea de desarrollo corresponderá a un período de uno a tres días de desarrollo. A continuación se relaciona una muestra de las tareas de ingeniería asociadas a algunas de las historias de usuario identificadas: (24)

TABLA 12: TAREAS DE INGENIERÍA

Historia de usuario	Tareas de Ingeniería
Creación de la actividad.	1. Se recoge la información del formulario. 2. Se valida la información recogida. 3. Se hace persistente la información en la base de datos.
Asignación de la actividad	1.Se selecciona la actividad a asignar. 2.Se carga la información de los profesores los cuales no tienen afectaciones. 3.La aplicación tras el uso de un algoritmo calcula el profesor más indicado para la realización de dicha actividad, y los muestra en orden descendente. 4.Se crea la relación actividad-profesor, y se hace persistente en la base de datos.
Registrar profesor	1.Se realiza la búsqueda consumiendo el servicio de capital humano (ASSET). 2.Se consume esa información, a través de los métodos que nos brinda ese servicio y se verifica si ya existe en la base de dato por el identificador de ese profesor. 3.Tras verificar se insertan en la base de dato los campos de profesor.
Revisar actividad	1.Se carga las personas vinculadas a dicha actividad.

	2.Se realiza la asociación profesor, actividad, calificación. 3.Se guarda la información en la base de dato.

2.5 Fases de diseño

El diseño del software es el proceso de aplicar distintas técnicas y principios con el propósito de definir un producto con los suficientes detalles como para permitir su realización. Constituye la primera etapa técnica del proceso de ingeniería del software y el punto de partida para realizar las actividades de implementación traduciendo los requisitos funcionales en una representación lógica del software a desarrollar.

La metodología XP propone que el diseño de la aplicación ha de ser lo más simple posible, siempre y cuando cumpla con las funcionalidades especificadas por el cliente. La complejidad innecesaria y el código extra debe ser removido inmediatamente. Kent Beck, creador de la metodología XP, plantea que en cualquier momento el diseño adecuado para el software es aquel que: supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos

2.5.1 Descripción de la arquitectura y el diseño

En el desarrollo de la solución propuesta se ha utilizado la arquitectura cliente-servidor, haciendo uso del patrón de arquitectura Modelo-Vista-Controlador. El marco de trabajo empleado está basado en dicho patrón, por su flexibilidad, es un enfoque de software que separa la lógica de la aplicación de la presentación. En la práctica, le permite a sus páginas web contener mínimo código ya que la presentación está separada del código PHP.

2.5.2 Arquitectura cliente-servidor

Esta arquitectura se divide en dos partes claramente diferenciadas, la primera es la parte del servidor y la segunda es un conjunto de clientes. Normalmente, el servidor es una máquina bastante potente que actúa de depósito de datos y funciona como un sistema gestor de base de datos. Por otro lado los clientes suelen ser estaciones de trabajo que solicitan varios servicios al servidor. Ambas partes deben estar conectadas entre sí mediante una red.Describe una relación entre el cliente y el servidor en el cual el primero realiza peticiones y el segundo envía respuesta.

Puede usar un amplio rango de protocolos y formatos de datos para compartir la información. Este tipo de estilo arquitectónico es el más utilizado en la actualidad, debido a que es la más avanzada y la que mejor ha evolucionado en estos últimos años. Se puede decir que esta arquitectura necesita tres tipos de software para su correcto funcionamiento:

Software de gestión de datos: este software se encarga de la manipulación y gestión de los datos almacenados y requeridos por las diferentes aplicaciones. Normalmente, este software se aloja en el servidor.

Software de desarrollo: este tipo de software se aloja en los clientes y solo en aquellos que se dedique al desarrollo de aplicaciones.

Software de interacción con los usuarios: también reside en los clientes y es la aplicación gráfica de usuario para la manipulación de datos, siempre a nivel usuario (24).

FIG. 4: ARQUITECTURA CLIENTE-SERVIDOR

2.5.3 Patrón de arquitectura

El patrón MVC es un patrón de arquitectura de software que separa los datos de una aplicación en la interfaz, la lógica y el modelo de datos en tres componentes distintos. Debido a la implementación del patrón por el *framework CodeIgniter*, el mismo ser el encargado de obtener, validar y procesar los datos enviados al servidor, así como las respuestas al cliente, separando la lógica de la aplicación de su representación al usuario y la comunicación al modelo, propiciando una comunicación y procesamiento más rápido y arquitecturas consistentes, reutilizables y más fácilmente mantenibles (25).

Controlador: Es el responsable de procesar las entradas del usuario en forma de peticiones

HTTP (*Hyper Text Transfer Protocol*), invocando las funcionalidades necesarias, expuestas por la lógica de negocio y devolviendo un modelo requerido para ser mostrado.

Modelo: Contiene los datos resultantes de la ejecución de la lógica de negocio, los cuales deberían ser mostrados en la respuesta.

Vistas: Son las encargadas de mostrar los dados del modelo que han sido suministrados por un controlador como respuesta a una petición. La forma de mostrar el modelo podrá ser con diferentes tipos de vistas como HTML (*Hyper Text Markup Languaje*), documentos PDF, documentos de Excel e imágenes,entre otras.

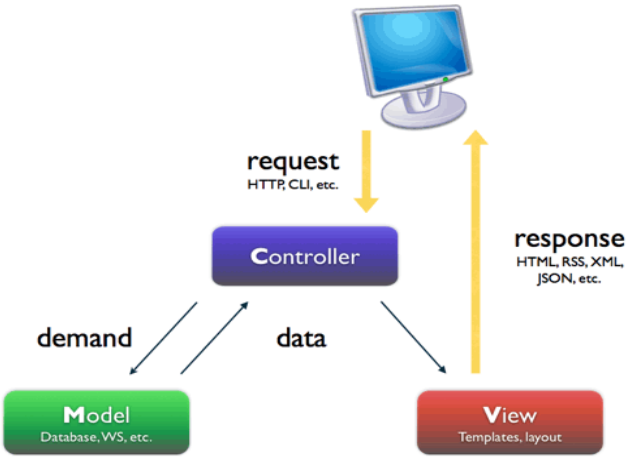


FIG. 5: PATRÓN DE ARQUITECTURA: MODELO-VISTA-CONTROLADOR

2.5.4 Diagrama de despliegue

Los nodos de un diagrama de despliegue representan elementos físicos con capacidad de proceso o facilitadores de algún servicio. Las máquinas físicas y los procesadores se representan como nodos y la construcción interna puede ser representada por nodos o artefactos embebidos. Los estereotipos permiten precisar la naturaleza del equipo: dispositivos, procesadores y memoria. El diagrama de despliegue representado muestra la siguiente distribución:

PC_Cliente: ordenador cliente capaz de conectarse al servidor de aplicaciones mediante el protocolo de comunicaciones HTTP.

Servidor_Web: ordenador en que se encuentra el servidor web Apache, este será el lugar en que se gestione todo el contenido de la aplicación. El mismo establecerá comunicación con los ordenadores clientes mediante protocolo HTTP y con el servidor de base de datos por medio del protocolo TCP/IP,y consumirá servicios a través de la pasarela de autenticación LDAP y de ASSET.

Servidor_BD: ordenador en que se encuentra el gestor de base de datos PostgreSQL capaz de mantener persistente la información generada y a utilizar. (28)

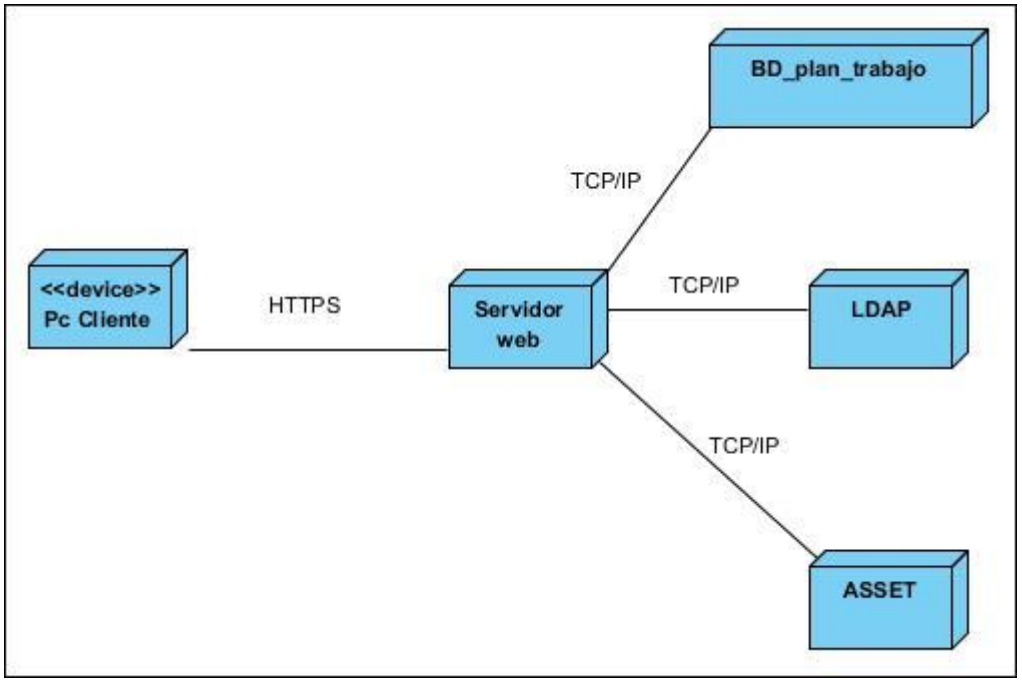


FIG. 6: DIAGRAMA DE DESPLIEGUE

2.5.5 Tarjetas CRC

Para el diseño de aplicaciones informáticas la metodología XP no requiere la presentación del sistema mediante diagramas utilizando UML como lenguaje de modelado. En su lugar se utilizan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad, Colaboración).

La característica más sobresaliente de las tarjetas CRC es su simpleza y adaptabilidad, ya que una tarjeta CRC no es más que una ficha de papel o cartón que representa a una entidad del sistema, a las cuales asigna responsabilidades y colaboraciones. Luego en un estado de diseño avanzado o ya en la implementación del sistema, las tarjetas CRC se convierten en clases con métodos, atributos, relaciones de herencia, composición o dependencia. En la siguiente Tabla Modelo de Tarjetas CRC, se muestra la estructura general de la Tarjetas CRC (24).

TABLA 13: MODELO DE TARJETA CRC

TARJETA CRC	
Clase: (nombre de la clase que tiene el contenido)	
Responsabilidades	Colaboraciones

Durante el proceso de diseño del componente se elaboraron un total de 5 tarjetas CRC. A continuación se muestran las más significativas y el resto están plasmadas en los Anexos. Descripción de las tarjetas CRC.

TABLA 14: MODELO DE TARJETAS CRC: CLASE ACTIVIDAD

TARJETA CRC	
Clase: actividad	
Responsabilidades:	Colaboración:
Permite obtener una actividad Permite asignar o sugerir la asignación de una determinada actividad. Permite obtener las actividades de la persona que se encuentra en el sistema. Permite registrar una actividad. Permite eliminar una actividad. Permite modificar una actividad.	Profesor Usuario

Tabla 15: Modelo de Tarjetas CRC: Clase profesor

TARJETA CRC	
Clase: profesor	
Responsabilidades:	Colaboración:
Permite obtener los profesores dado parámetros Permite obtener el profesor revisor Permite obtener los profesores relacionados con una determinada actividad. Permite registrar un profesor Permite eliminar un profesor Permite modificar datos de un profesor	usuario

TABLA 16: MODELO DE TARJETAS CRC: CLASE USUARIO

TARJETA CRC	
Clase: usuario	
Responsabilidades:	Colaboración:
Listar usuario Buscar usuario Mostrar el rol del usuario Modificar un usuario Eliminar usuario	rol

2.6 Patrones de diseño

Los patrones de diseño expresan esquemas para definir estructuras de diseño o sus relaciones con las que construir sistemas de software. A la hora de desarrollar un software es importante el uso de patrones de

diseño, existen varios patrones y con distintos propósitos entre los que se encuentran los patrones de creación, patrones estructurales, patrones de comportamiento, entre otros. Proporcionan una estructura conocida por todos los programadores, de manera que la forma de trabajar no resulte distinta entre los mismos. La utilización de patrones de diseño, permite ahorrar grandes cantidades de tiempo en la construcción de software. El software construido es más fácil de comprender, mantener y extender (29).

2.6.1 Patrones GRASP

Los patrones GRASP (*General Responsibility Assignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos y son fundamentales en el diseño del software orientado a objeto, debido a que brindan una solución a varios de los problemas que dan la medida de un refinamiento del diseño, siempre con el propósito de lograr un sistema reusable y flexible.

Para el diseño del sistema, se tuvieron en cuenta los siguientes patrones GRASP:

Experto, Creador, Bajo acoplamiento, Alta cohesión y Controlador.

Creador: Asignarle a la clase B la responsabilidad de crear una instancia de clase A, el uso de este patrón se evidencia en la clase loader que es el objeto load de las clases controladoras el cual se encarga de cargar los elementos del marco de trabajo: las librerías y los modelos. También guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Este patrón brinda soporte al bajo acoplamiento y tiene como objetivo principal encontrar un creador, el cual debe conectarse con el objeto producido en cualquier evento, dándole soporte al bajo acoplamiento.

Controlador: Asigna la responsabilidad de manejar un mensaje de un evento del sistema a una clase, la cual representa a la empresa o sistema global. Este patrón debe delegar a otros objetos el trabajo que ha de realizarse mientras se dedica a coordinar la actividad. También ofrece una guía para tomar decisiones apropiadas en la elección de los controladores de eventos. También este patrón se refleja en las clases controladoras, que son las clases que se encargan de obtener datos y enviarlos a las librerías y las vistas. El patrón creador garantiza que los procesos sean manejados por la capa controladora y no por la capa de presentación. La puesta en práctica de este patrón se pone de manifiesto en las clases controladoras, actividad.php, estructura.php, revisor.php, autenticar.php, siendo estas las encargadas de obtener y enviar los datos a las librerías y a las vistas

Alta cohesión y bajo acoplamiento: La alta cohesión asigna una responsabilidad, de modo que la cohesión siga siendo alta, además caracteriza a las clases con responsabilidades relacionadas que no realicen mucho trabajo. Este patrón mejora la calidad y facilidad del diseño, aunque también promueve la reutilización y genera un bajo acoplamiento, también la propia implementación del CodeIgniter contiene estos patrones nivelados, permitiendo el uso de los componentes de forma individual, evidenciando de esta forma el bajo acoplamiento, así como la dependencia entre los componentes o la alta cohesión.

Por su parte el bajo acoplamiento asigna una responsabilidad para mantener bajo acoplamiento. Una clase con bajo acoplamiento no depende de muchas otras, reduciendo el impacto de los cambios, además son más fáciles de reutilizar, aumentando de esta forma la productividad

Experto: Este patrón se evidencia en las clases librerías, actividad_lib, profesor_lib, reporte_lib, funcionalidad_lib, que cuentan con la información necesaria para cumplir con la responsabilidad sobre los elementos del negocio, asignándole una responsabilidad a la clase que tiene la información necesaria para cumplirla.se encarga de asignarle una responsabilidad al experto en información, que es la clase que cuenta con la información necesaria para cumplir con la responsabilidad. Mantiene el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. También brinda beneficios como el soporte a la alta cohesión y al bajo acoplamiento. Su uso permite asignar a una clase la información necesaria para cumplir su responsabilidad, de esta forma, se asignan las mismas de forma adecuada. Estos patrones se evidencian en la implementación del propio marco de trabajo CodeIgniter, el cual permite el uso de forma individual de los componentes, poniendo de manifiesto el bajo acoplamiento, así como la dependencia existente entre ellos o alta cohesión.

2.6.2 Patrones GOF

Los patrones GoF (Gang of Four) describen las formas comunes en que distintos tipos de objetos pueden ser organizados para trabajar unos con otros. Además permiten crear grupos de objetos para ayudar a realizar tareas complejas, tratan la relación entre clases y la formación de estructuras de mayor complejidad. A continuación se explican los patrones GoF utilizados en el desarrollo del sistema

Instancia única (Singleton): garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

El patrón singleton se implementa creando en nuestra clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado).

La utilización de este patrón se puede ver en las clases controladoras pues todas ellas son instancias únicas.

Mediador (Mediator): permite definir un objeto que coordine la comunicación entre los objetos de diferentes clases, pero que funcionan como un conjunto.

Este patrón se pone manifiesto en las librerías, pues estas funcionan como mediadoras entre las clases controladoras y las modelos (acceso a datos). 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. Coordina las relaciones entre sus asociados. Permite la interacción de varios objetos, sin generar acoples fuertes en esas relaciones, con la intención de definir un objeto que encapsule como interactúa un conjunto de objetos.

Observador (Observer): permite definir una dependencia de uno-a-muchos entre objetos, brindando la posibilidad de que cuando un objeto cambie de estado se notifique y actualicen todos los objetos que dependen de él. La puesta en práctica de este patrón se evidencia en la clase loader que es el objeto load de las clases controladoras, la cual se encarga de cargar las librerías y los modelos. Además de actualizar la clase controladora instanciada.

Capítulo 3: Implementación y prueba

Introducción

En el proceso de desarrollo de un software, la etapa de prueba es clave a la hora de detectar errores o fallas. Conceptos como estabilidad, escalabilidad, eficiencia y seguridad se relacionan a la calidad de un producto bien desarrollado. Las pruebas nos muestran que nuestro producto final funciona, cuando no conseguimos pensar en ninguna prueba que pudiese producir un fallo en nuestra aplicación, entonces habremos concluido por completo.

Uno de los pilares de XP es el proceso de pruebas. Esta metodología anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección.

También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. “Las pruebas unitarias y funcionales son el corazón de XP”.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y 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 de aceptación

Las pruebas de aceptación se crean a partir de los relatos de usuario. El cliente es quien define los escenarios de prueba para verificar si el relato de usuario ha sido correctamente implementado. Un relato de usuario puede tener una o varias pruebas de aceptación. Son pruebas de caja negra a nivel del sistema y cada una corresponde a un resultado producido por el sistema. Su corrección debe ser programada para la próxima iteración. Si no hay pruebas de aceptación nuevas no se ha hecho nada nuevo. Un relato de usuario no está completo hasta no haber pasado todas sus pruebas de aceptación.

3.1 Estándares de codificación

CodeIgniter se basa en el estándar de codificación fijado por *Zend Framework* y no en las particularidades del mismo.

Tanto Codeigniter y Zend Framework son *framework* de código abierto para desarrollar aplicaciones web y servicios web con PHP5. La arquitectura utilizada por estos *framework* es modelo –vista-controlador.

Temas incluidos en los estándares de código

Dar formato a archivos PHP

Convenciones de nombrado

Estilo de código

Documentación integrada

Los estándares de código resultan importantes en cualquier proyecto de desarrollo, pero son especialmente importantes cuando muchos desarrolladores trabajan en el mismo proyecto. Los estándares de código ayudan a asegurar que el código tenga una alta calidad, menos errores, y pueda ser mantenido fácilmente.

Pautas generales:

Zend Framework se estandariza una convención de nombres de clases. Todas las clases de Zend Framework están almacenadas jerárquicamente.

Los nombres de clases pueden contener sólo caracteres alfanuméricos. Los números están permitidos en los nombres de clase, pero desaconsejados en la mayoría de los casos. Las barras bajas (_) están permitidas solo como separador de ruta. Si el nombre de una clase está compuesto por más de una palabra, la primera letra de cada palabra debe aparecer en mayúsculas. Poner en mayúsculas las letras siguientes no está permitido, ej: "Zend_PDF" no está permitido, mientras que " Zend_Pdf " es admisible.

Los nombres de funciones pueden contener únicamente caracteres alfanuméricos. Los guiones bajos (_) no están permitidos. Los números están permitidos en los nombres de función pero no se aconseja en la mayoría de los casos.

Los nombres de funciones deben empezar siempre con una letra minúscula. Cuando un nombre de función consiste en más de una palabra, la primera letra de cada nueva palabra debe estar en mayúsculas. Esto es llamado comúnmente como formato "camelCase".

Por norma general, se recomienda la elocuencia. Los nombres de función deben ser lo suficientemente elocuentes como para describir su propósito y comportamiento.

Formato de archivos PHP:

Todos los nombres de archivos de soporte y código fuente van en minúsculas

No usar CAMELCASE (ejemplo: NombreFuncion), usar guion bajo (ejem: nombre_funcion) versión de php usada: PHP 5.3.5

Contenido de Directorios

assets

Todos los archivos no php van en el directorio assets:

assets \

\js (Los ficheros de java script)

\css (Las hojas de estilos)

\img (Las imágenes)

Los nombres de archivos van en minúsculas y sin espacios, el nombre del archivo debe tener relación con la vista donde será utilizado.

Código PHP de modelos vistas y controladoras:

Todo el código fuente va en las siguientes carpetas

application \

\modules

\nombre_modulo (planificación)

\controllers

\nombre_controlador.php (actividad.php)

\nombre_controlador.php (estructura.php)

...

\libraries

\actividad_lib.php

\estructura_lib.php

...

\models

\actividad_mdl.php

\asignacion_actividad_mdl.php

\profesor_mdl.php

...

...

Controladoras:

// Funciones customizadas del controlador

public function nombre_funcion(){

}

public function nombre_funcion2(){

}

Los nombres de las funciones deben ser verbos o palabras que documenten su finalidad por ejemplo:

public function listarMisActividades()

public function formularioAsociar($id)

public function formularioModificar($id)

Modelos:

Los nombres de los modelos tienen el nombre de la tabla a la que representan agregando el sufijo _model por ejemplo:

asignacion_actividad_mdl.php

actividad_mdl.php

Usar el guión bajo si son más de dos palabras.

// Forma de cargar las librerías:

```
 $this->load->libraries(array('planificacion/actividad_lib','planificacion/profesor_lib'));

$this->load->libraries(array('planificacion/estructura_lib',));
```

Vistas:

En el caso de las vistas se crea una carpeta dentro de la carpeta views con el nombre del

submódulo al que corresponden:

views \

\actividad

Dentro de esta carpeta creamos los archivos de vistas.

3.2 Descripción de las clases implementadas

Tras el uso del marco de trabajo CodeIgniter se han creado varias carpetas contenedoras de los módulos de la aplicación, la carpeta system en la cual se mantiene sin sufrir variaciones, debido que es la carpeta contenedora de las librerías de este marco de trabajo.

La carpeta en la cual se enfoca el mayor peso de la aplicación es en la carpeta aplicattion, la cual contiene varios módulos, Planificación, Reporte y Seguridad encargándose cada una de un peso importante en la aplicación.

El negocio se centra en el módulo Planificación donde se cargan las librerías más importantes las cuales son las estructuras del negocio, entiéndase por las librerías de actividad, profesor, estructura, al igual que reportes y seguridad, las cuales hacen peticiones a los objetos de las clases contenedoras de acceso a dato, mediante el enrutamiento de datos a través de los controladores. En este caso las librerías deben cargar peticiones realizadas al contrato de servicio de ASSET para la obtención de información de un determinado profesor y se usa la pasarela de autenticación LDAP, ambas como fuentes primaria de información.

En la carpeta models se encuentran los archivos que contienen una serie de funciones que permiten devolver, insertar y actualizar la información de la base de datos y por otro lado la carpeta libraries donde

se encuentran los archivos que sirven de intermediarios entre el controlador y el modelo, además de ser usadas para obtener información de los restantes módulos. Esta última es una adaptación realizada para disminuir la cantidad de responsabilidades en el controlador.

En la carpeta views serán almacenadas las páginas o fragmentos de páginas web que mostrarán la información al usuario.

En la carpeta controllers se encuentran los archivos que son los encargados de procesar la información y cargar cualquier recurso necesario para procesar la solicitud HTTP.

3.3 Pruebas funcionales

Se denominan pruebas funcionales a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados cumplan con las funciones específicas para los cuales han sido creados. A este tipo de pruebas se les denomina también pruebas de comportamiento y para realizarlas se emplea el método de caja negra, donde los probadores o analistas de pruebas no enfocan su atención en cómo se generan las respuestas del sistema, sino en el funcionamiento de la interfaz del sistema.

Básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, esto generalmente se define en los casos de prueba preparados antes del inicio de las pruebas

El principal objetivo de este tipo de pruebas es demostrar que las funciones del software son operativas, que la entrada se produce de forma adecuada, que la salida se produce de forma correcta y que se mantiene la integridad de la información. Según la metodología usada se propone realizar tras cada iteración las pruebas funcionales, e ir proporcionando entregas pequeñas al cliente de la aplicación e ir resolviendo incremental e iterativamente las no conformidades de la iteración anterior y de esa forma, lograr la aceptación del cliente y darle una solución final al problema planteado. A continuación se describen algunos de los casos de pruebas asociados a los requisitos de prioridad alta.

TABLA 17: CASOS DE PRUEBAS. RELACIONADA CON ACTIVIDAD

Escenario	Descripción	Variables	Respuesta del sistema	Flujo central
EC 1.1_Registrar Actividad	Mediante este escenario se adiciona una actividad.	Descripción de la actividad Fecha Inicio de la actividad Fecha culminación de la actividad Fuente de la cual proviene la actividad.	El sistema registra la actividad emitiendo una notificación, operación exitosa.	1.El usuario se autentica en el sistema. 2.Después puede navegar en el menú lateral izquierdo, seleccionando la funcionalidad actividad. 3.Se le muestran todos los campos relacionados con dicha funcionalidad. 4.El usuario selecciona la acción de registrar la actividad. 5.El usuario debe

				ingresar todos los campos para la creación de la actividad, debido que existen capos obligatorios y presiona el botón Registrar actividad.
EC 1.2_Asignar Actividad	Mediante este escenario se asigna una actividad.	N/A Se debe tener en cuenta que una vez relacionada la actividad con el profesor adecuado para la realización de la actividad, debe designarse un revisor.	El sistema registra la actividad emitiendo una notificación, operación exitosa.	1. El usuario se autentica en el sistema. 2. Después puede navegar en el menú lateral izquierdo, seleccionando la funcionalidad actividad. 3. Se le muestran todos los campos relacionados con dicha funcionalidad. 4. El usuario selecciona el profesor para crear

				la relación entre la actividad y el profesor al cual le será asignada. 5. El usuario presiona el botón Asignar actividad.
EC 1.2_Asignar Actividad incorrectamente	Mediante este escenario se trata de asignar una actividad, sin seleccionar ningún campo	N/A Se debe tener en cuenta que una vez relacionada la actividad con el profesor adecuado para la realización de la actividad, debe designarse un revisor.	El sistema no registra la actividad emitiendo una notificación, en rojo Seleccione una opción	1. El usuario se autentica en el sistema. 2. Después puede navegar en el menú lateral izquierdo, seleccionando la funcionalidad actividad. 3. Se le muestran todos los campos relacionados con dicha funcionalidad. 4. El usuario selecciona el profesor para crear la relación entre la actividad y el profesor al cual le será asignada, dejando el campo de selección sin marcar.

Tabla 18: Casos de pruebas. Relacionada con actividad del revisor

Escenario	Descripción	Variables	Respuesta del sistema	Flujo central
EC 1.1_Revisar actividad	Mediante este escenario el revisor trata de evaluar una actividad, dejando los campos vacios.	Campos: Bien, Mal, Regular	El sistema no registra la evaluación de la actividad emitiendo una notificación, en rojo Seleccione una opción	1. El usuario se autentica en el sistema. 2. Después puede navegar en el menú lateral izquierdo, seleccionando la funcionalidad actividad. 3. Se le muestran todos los campos relacionados con dicha funcionalidad. 4. El usuario selecciona la acción de revisar la actividad. 5. El usuario debe marcar una evaluación, y presionar el botón revisar actividad.
EC 1.2_ Revisar actividad	Mediante este escenario el revisor le da	N/A Se debe tener en cuenta que una vez relacionada la actividad con el profesor adecuado para la	El sistema registra la actividad emitiendo una	1. El usuario se autentica en el sistema.

incorrectamente	una evaluación a una determinada actividad.	realización de la actividad, debe designarse un revisor.	notificación, operación exitosa.	2. Después puede navegar en el menú lateral izquierdo, seleccionando la funcionalidad actividad. 3. Se le muestran todos los campos relacionados con dicha funcionalidad. 4. El usuario selecciona la acción de revisar la actividad. 5. El usuario no marca ninguna evaluación y presionar el botón revisar actividad.

3.3.1 Prueba de rendimiento

La prueba de rendimiento está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. La prueba de rendimiento se da durante todos los pasos del proceso de la prueba. Incluso al nivel de unidad, se debe asegurar el rendimiento de los módulos individuales. Sin embargo, hasta que no estén completamente integrados todos los elementos del sistema no se puede asegurar realmente el rendimiento del sistema (30).

3.3.2 Pruebas de resistencia(o estrés)

Las pruebas de resistencia están diseñadas para enfrentar a los programas con situaciones anormales. La prueba de resistencia ejecuta un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales. Esencialmente, el responsable de la prueba intenta romper el programa.

3.4 Resultados de las pruebas

Resultados de las pruebas

La pruebas realizadas se llevaron a cabo en cuatro iteraciones, estas pruebas permitieron detectar un grupo de no conformidades las cuales fueron erradicadas. Fueron detectados en las iteraciones errores en la interfaz, ortografía y validación. En la primera iteración fueron detectadas 6 no conformidades para un total de 8 funcionalidades, en la segunda 8 no conformidades de 16 funcionalidades, en la tercera 5 no conformidades de 11 funcionalidades y en la última iteración no se detectaron no conformidades. Permitiendo comprobar el correcto funcionamiento de las funcionales, la calidad de la solución implementada y corregir los errores identificados. A continuación se muestra en la Tabla 7 los resultados de las pruebas aplicadas:

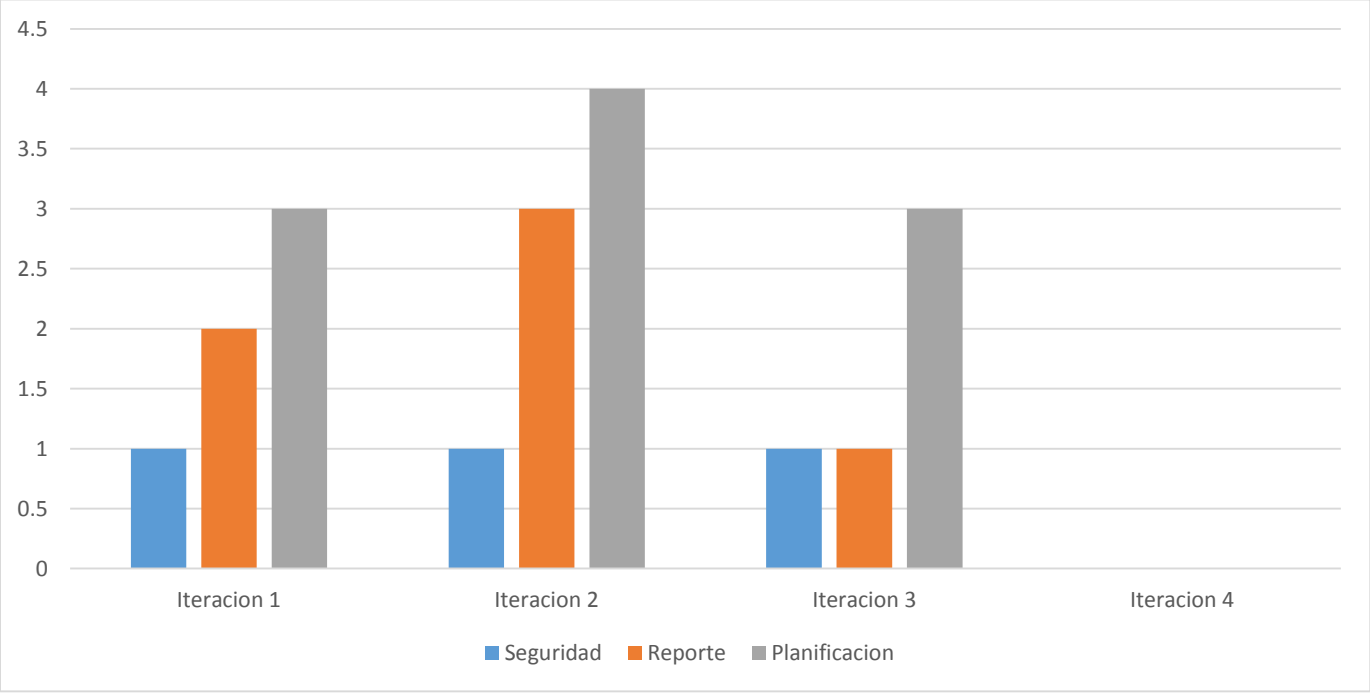
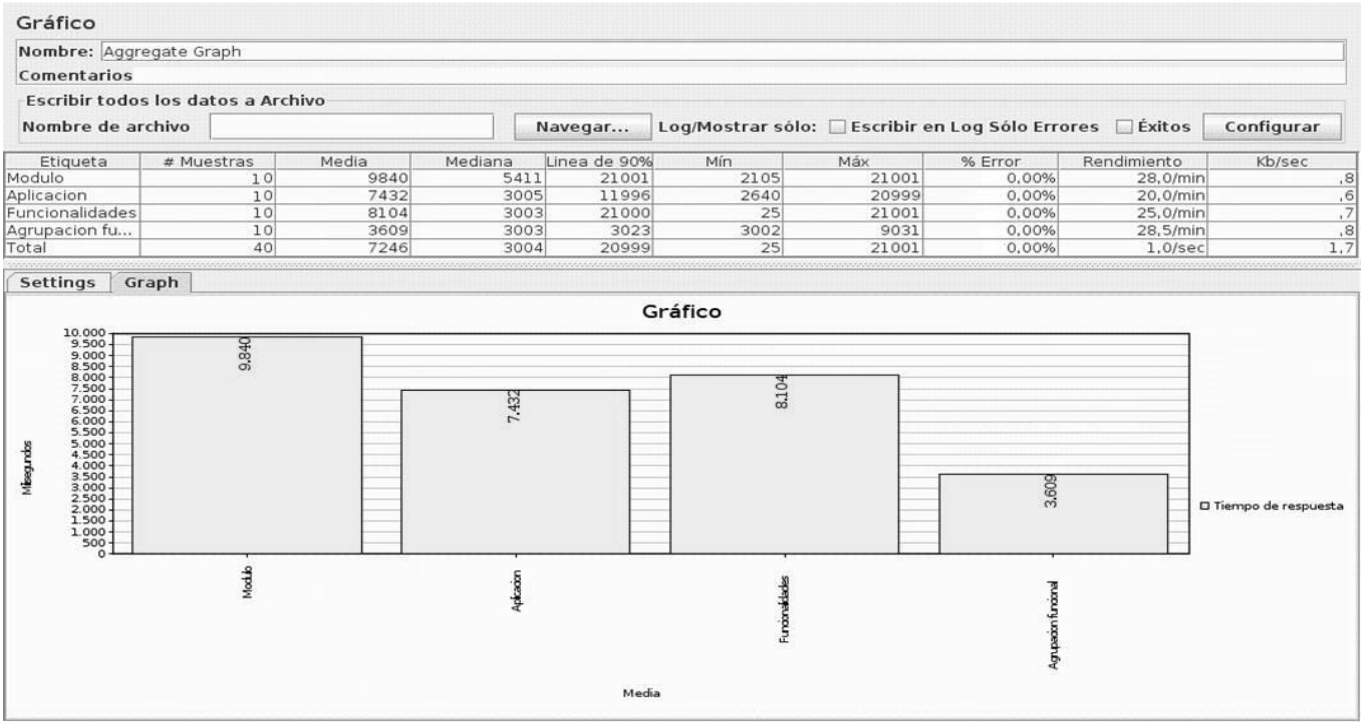


FIG. 7: GRÁFICA DE NO CONFORMIDADES OBTENIDAS EN LAS PRUEBAS FUNCIONALES

Pruebas de rendimiento y estrés



Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Modulo	10	9840	5411	21001	2105	21001	0,00%	28,0/min	,8
Aplicacion	10	7432	3005	11996	2640	20999	0,00%	20,0/min	,6
Funcionalidades	10	8104	3003	21000	25	21001	0,00%	25,0/min	,7
Agrupacion fu...	10	3609	3003	3023	3002	9031	0,00%	28,5/min	,8
Total	40	7246	3004	20999	25	21001	0,00%	1,0/sec	1,7

FIG. 8: GRÁFICA DE LOS RESULTADOS OBTENIDOS DE LAS PRUEBAS DE RENDIMIENTO Y ESTRÉS

Luego de aplicarle al sistema las pruebas, las cuales se apoyaron en el uso de la herramienta Jmeter, que ofrece datos importantes acerca del stres que puede soportar el sistema. Para 40 usuarios conectados en 10 hilos de manera concurrente a 4 de las funcionalidades principales, se obtuvo como resultado el buen funcionamiento del mismo, manteniendo un 0 % de error al ser utilizado por estos usuarios. Por lo que queda comprobado el correcto funcionamiento del sistema.

3.5 Conclusiones parciales

En el diagrama de despliegue se reflejan los recursos del sistema desarrollado.

Las pruebas realizadas a la solución permitieron detectar las no conformidades existentes y demostrar que dicha solución cumple con todas las especificaciones dadas por el cliente.

Conclusiones Generales

El trabajo investigativo estuvo enfocado al estudio de la gestión de la información para la asignación de actividades y así llevar un control y planificación de la realización de las mismas según su nivel de importancia El proceso de desarrollo de software utilizado guió satisfactoriamente la elaboración de la solución, garantizando la obtención de los artefactos definidos. La validación de los requerimientos y las pruebas realizadas, comprobó que las funcionalidades descritas satisfacen las necesidades del cliente, logrando así el desarrollo de un sistema informático planificador, para la mejora de toma de decisiones y la realización de actividades, que mejore la planificación, control y asignación de tareas en los diferentes departamentos de la UCI, especialmente el departamento docente de técnicas de programación en la facultad 1.

RECOMENDACIONES

Para garantizar el perfeccionamiento de la solución creada se recomienda:

- Perfeccionarlo en aras de satisfacer nuevas requerimientos del cliente y lograr ampliar su campo de acción.
- Desarrollar un algoritmo que permita realizar las asignaciones de forma automatizada, sin la necesidad de un intermediario, con el objetivo de que el proceso sea automático y por consecuente más rápido.

Bibliografía Referenciada.

Bibliografía

1. Zayas, Carlos Alvarez de. *Metodología de la Investigación Científica.* Santiago de Cuba : Academia, 1995. pág. 24.

2. Ponjuan, German. Gestion de Informacion. [En línea] 4 de noviembre de 2014. Ecured.html.

3. Sistema de gestion integrado. *Ventajas y desventajas de implementar un sistema de gestion integrado.* [En línea] 2000. Centanni & Asociados.html.

4. ACIMED. [En línea] ACIMED v.20 n.5 Ciudad de La Habana, nov de 2009. http://scielo.sld.cu/scielo.php?pid=S1024-94352009001100006&script=sci_arttext.

5. Monografias. *Planificación.* [En línea] 2010. http://www.monografias.com/trabajos34/planificacion/planificacion.shtml.

6. Ecured. *Gespro.* [En línea] 2003. [Citado el: 23 de abril de 2015.] http://www.ecured.cu/index.php/Gespro.

7. Ecured. [En línea] 2011. [Citado el: 2 de mayo de 2015.] http://www.ecured.cu/index.php/EcuRed:Enciclopedia_cubana.

8. Metodologia de desarrollo de software. *Metodologia de desarrollo de software.* [En línea] 2011. http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

9. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES . [En línea] 22 de marzo de 2014. [Citado el: 12 de abril de 2015.] https://prezi.com/xpo8pdv2ghqm/metodologias-tradicionales-vs-metodologias-agiles/.

10. Jummp. *Kent Beck y la naturaleza adaptativa del software.* [En línea] mayo de 2011. https://jummp.wordpress.com/2011/05/20/kent-beck-y-la-naturaleza-adaptativa-del-software/.

11. Extreme Programming Explained: Embrace Change. [En línea] 2012. http://www.amazon.com/Extreme-Programming-Explained-Embrace-Change/dp/0201616416.

12. Beck, Kent. *Extreme Programming Explained.* September 1999.

13. Penadés, Patricio Letelier y Mª Carmen. *Métodologías ágiles para el desarrollo de software:.* Valencia : s.n.

14. slideshare. *CASE.* [En línea] http://es.slideshare.net/guestf131a9/herramientas-case.

15. Bustos, Prof. Guillermo. *Guía de Uso de la Herramienta CASE .* Mayo, 2010.

16. Unified Modeling Language. [En línea] 22 de MAYO de 2015. [Citado el: 1 de junio de 2015.] http://www.uml.org/.

17. JavaScript. [En línea] 2012. [Citado el: 2 de ABRIL de 2015.] https://developer.mozilla.org/es/docs/Web/JavaScript.

18. CodeIgniter. *Guía del Usuario de CodeIgniter Versión 2.1.4.* [En línea] 2012. http://www.codeigniter.com/.

19. PostgreSQL. [En línea] 2013. [Citado el: 6 de abril de 2015.] http://www.postgresql.org.es/sobre_postgresql.

20. Fundamento de las bases de datos: Modelo entidad-relación. [En línea] 5 de nov de 2013. [Citado el: 8 de abril de 2015.] http://www.genbetadev.com/bases-de-datos/fundamento-de-las-bases-de-datos-modelo-entidad-relacion.

21. Bustamante, Dayana. *Metodologia actual.* Barinas : s.n., marzo,2014.

22. Ecured. *Requisitos no funcionales.* [En línea] 2015. [Citado el: 6 de abril de 2015.] http://www.ecured.cu/index.php/Requisitos_no_funcionales.

23. Amnada, Kmila,Emilio. *http://www.abacovirtual.edu.pe/chiclayo/filedocente/140318-CEI2012203MB-000029-19072012-173247.pdf.*

24. Priolo, Ing. Sebastian. *Programacion Extrema.*

25. Ingenieria De Software. *Metodologia XP .* [En línea] 12 de julio de 2012. [Citado el: 24 de mayo de 2015.] http://pnfiingenieriadesoftwaregrupocuatro.blogspot.com/2012/07/bienvenidos-al-blog.html.

26. Arquitectura cliente servidor. [En línea] 2014. [Citado el: 4 de mayo de 2015.] http://www.desarrolloWeb.com/articulos/arquitectura-cliente-servidor.htm.

27. Patron MVC. [En línea] 2013. [Citado el: 4 de mayo de 2015.] http://www.proactiva-calidad.com/java/patrones/mvc.html. .

28. UML: Diagrama de Despliegue . [En línea] 12 de abril de 2013. [Citado el: 2 de junio de 2015.] http://umldiagramadespliegue.blogspot.com/.

Bibliografía Consultada

1- SOMERVILLE, Ian. Ingeniería del Software. [En línea]. 7a. ed. Madrid: Pearson Education S.A, 2005. 661p. [En línea], [ref. de 3 de febrero de 2015]. Disponible en web: <http://books.google.com.cu/books?id=gQWd49zSut4C&pg=PA80&dq=Herramientas+Case&hl=es&sa=X&ei=uOG3T-_cE8jpgged2PSiCg&ved=0CEQQ6AEwAw#v=onepage&q=Herramientas%20Case&f=false ISBN: 84-7829-074-5>.

2- HERNÁNDEZ, H. La importancia de un Sistema de Información en las empresas. 5 de enero de 2015 2012, nº Disponible en: http://hectorhernandezadm.blogspot.com/.

3- VIDAL, Y. G. Documento de Arquitectura de Software del proyecto. 5 de febrero de 2015 nº.

4- INC, C. E. L. CODEIGNITER. 30 de enero de 2013 2013, nº Disponible en: http://codeigniter.com/.

5- PROJECT, J. T. J. The jQuery Project. 3 de febrero de 2015 2010, nº Disponible en: http://docs.jquery.com/Release:jQuery_1.3.2.

6- GROUP, P. G. D. A brief history of PostgreSQL. 2 de febrero de 2015 2013, nº Disponible en: http://www.postgresql.org/docs/9.4/interactive/intro-whatis.html.

7- PARADIGM, V. UML tool, business process modeler and database designer for software development team. 3 de febrero de 2015 nº Disponible en: http://www.visual-paradigm.com.

8- MAESTROSDELWEB. ¿Qué es JavaScript? 4 de febrero de 2015 nº Disponible en: http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

9- LIBROSWEB.ES. Introducci'on a CSS. 3 de marzo de 2013 2013, nº Disponible en: http://www.librosweb.es/css/.

10- ECURED. HTML. 5 de marzo de 2015 nº Disponible en: http://www.ecured.cu/index.php/HTML.

11- ÁLVAREZ, R. Qué es y para qué sirve SQL. 4 de marzo de 2015 2001, nº Disponible en: http://www.desarrolloweb.com/articulos/262.php.

12- PRESSMAN, R. S. Ingeniería del software, un enfoque práctico 5ta edición. nº

13- BECK, K. Extreme Programming Explained. Embrace Change. s.l.: Pearson Education, 1999. Traducido al español como: Una explicación de la programación extrema. Aceptar el cambio, Addison Wesley, 2000.

14- Sistema de Gestión Integrado (SGI). [Online] abril 2015. Ventajas y desventajas de implementar un Sistema de Gestión Integrado (SGI). _ Consultor ISO-9001 _ C.A.B.A. _ Centanni & Asociados.htm.

15- Marín Sánchez, J.; Lugo García, J.A.; Piñero Pérez, P.Y.; Santiesteban García, A.M.; Abelardo Santana, F.N.; Menéndez Rizo, J. (2014) Proceso para la planificación y control de proyectos de software utilizando Xedro-GESPRO. Revista Cubana de Ciencias Informáticas, Vol. 8, No. 2. pp.144-161, ISSN 1994-1536.

16- PRESSMAN, Roger S. Ingeniería del Software un enfoque práctico. s.l. : Mc Graw Hill, 2007. ISBN: 97-0105-473-3.

17- Extreme Programming. [En línea] 9 de diciembre de 2002. [Citado el: 8 de abril de 2015.] http://www.infoab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf

18- Larman, Craig. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* [trad.] Luz Maria Henhndez Rodriguez y Humberto Cárdenas Anaya. México : PRENTICE HALL, 1999. pág. 536. ISBN: 970-17-0261-1.

19- datos, Patrones de diseño de base de. EVA. [En línea] 2013. [Citado el: 23 de Febrero de 2015.] http://eva.uci.cu/file.php/180/2._Clases/Tema_1/Materiales_basicos/4.Patrones_de_diseno_de_BD.pdf

20- datos, Patrones de diseño de base de. EVA. [En línea] 2013. [Citado el: 23 de Febrero de 2015.] http://eva.uci.cu/file.php/180/2._Clases/Tema_1/Materiales_basicos/4.Patrones_de_diseno_de_BD.pdf

21- HTML.it s.r.l. – Design. HTMLPOINT - Tutorial ISS - Internet Information Server. HTMLPOINT.com (September 2007).[cited 23 Abril 2015]. Available from World Wide Web: < http:// www. htmlpoint. com/iis/index.html>.

22- Visual Paradigm International. Visual Paradigm for UML. [En línea] http://www.visual-paradigm.com/product/vpuml/.

23- SOFTENG. 2013. SOFTENG Software Engineers. [En línea] 2013. [Citado el: 2015 de Mayo de 21.] http://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum.html.

24- EVA. Patrones de diseño de base de datos. 23 de abril de 2015 2013, nº Disponible en: http://eva.uci.cu/file.php/180/2._Clases/Tema_1/Materiales_basicos/4.Patrones_de_diseno_de_BD.pdf

25- PRESSMAN, Roger S. Ingeniería del Software un enfoque práctico. Pruebas de rendimiento. Quinta Edición. s.l. : Mc Graw Hill, 2007, 309 p, ISBN: 97-0105-473-3

26- 2015. Zend Framework. [Online] 2015. [Cited: abril 2, 24.] https://github.com/zendframework.

27- SCRIBD, C. Diagrama de despliegue. Diagrama de despliegue. [En línea] Marzo de 2009. [Citado el: 1 de Marzo de 2015.] http://www.scribd.com/doc/53551175/11/.

28- GARCÍA VIDAL, Yanio. Documento de Arquitectura de Gestión Universitaria. Patrones GRASP. La Habana : s.n., Marzo de 2012.

29- GARCÍA VIDAL, Yanio. Documento de Arquitectura de Gestión Universitaria. Patrones Gof. La Habana : s.n., Marzo de 2012

30- DESARROLLOWEB.COM. Modelo-vista-controlador. Modelo-vista-controlador. [En línea] Agosto de 2009. [Citado el: 7 de Mayo de 2015.] http://www.desarrolloWeb.com/articulos/modelo-vista-controlador-codeigniter.html.

31- DESARROLLOWEB.COM. Arquitectura cliente-servidor. Arquitectura cliente-servidor. [En línea] Agosto de 2009. [Citado el: 14 de Mayo de 2015] http://www.desarrolloWeb.com/articulos/arquitectura-cliente-servidor.html

Descripción de funcionalidades:

TABLA 19: DESCRIPCIÓN DE FUNCIONALIDAD: AUTENTICARSE EN EL SISTEMA

No	Nombre	Descripción	Complejidad	Prioridad para el cliente
17	Autenticarse en el sistema	Los usuarios podrán interactuar con en el sistema, tras ingresar usuario y contraseña. : Tras validar los datos ingresados y ser insertado en la base de datos, el administrador le concede el roll al usuario, que solicita el acceso al sistema, en caso de no encontrarse en la base de dato su acceso es negado.	Alta	Alta
	Prototipo			

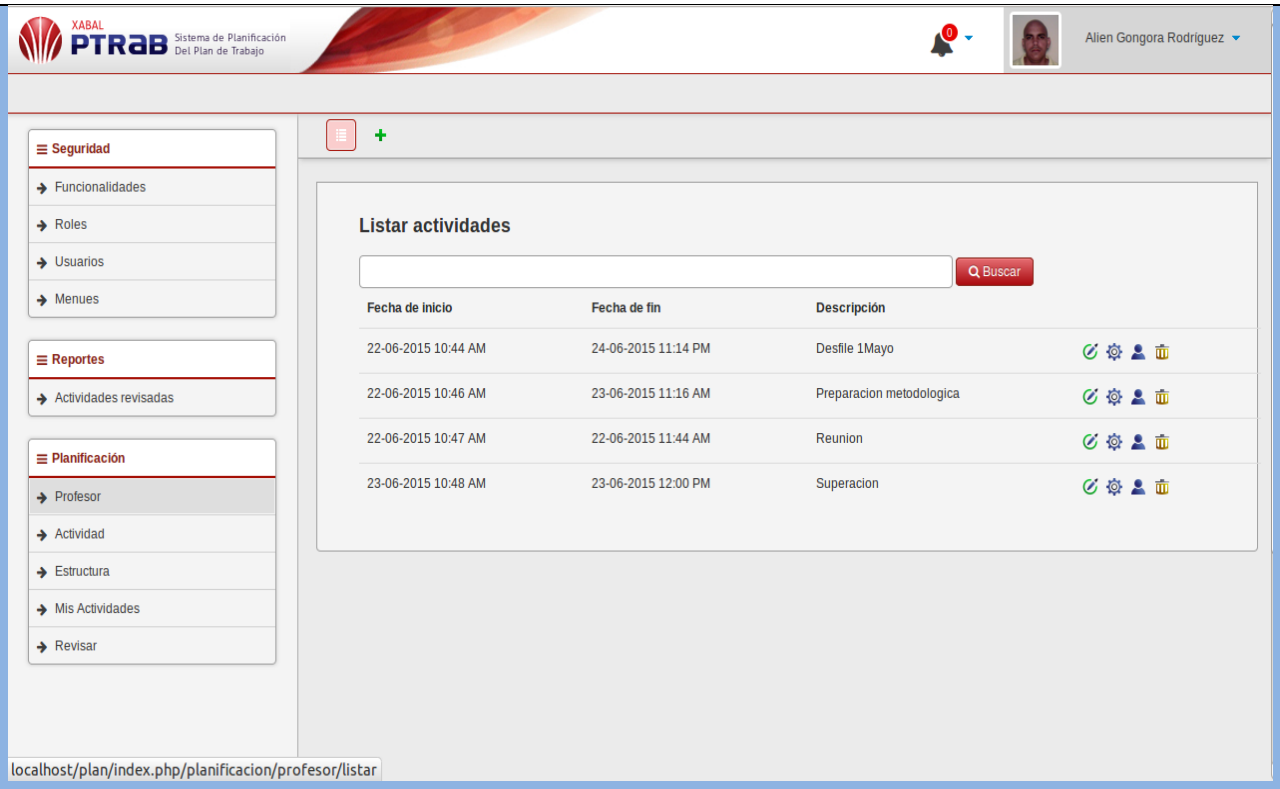
Prototipo Autenticarse en el sistema

	Campos	Tipos de datos	Reglas o restricciones
	Usuario	texto	Campo obligatorio, admite cualquier cadena de caracteres

	Contraseña	texto	Campo Obligatorio, admite cualquier cadena de caracteres
	Observaciones:		

TABLA 20:DESCRIPCIÓN DE FUNCIONALIDAD: LISTAR ACTIVIDADES

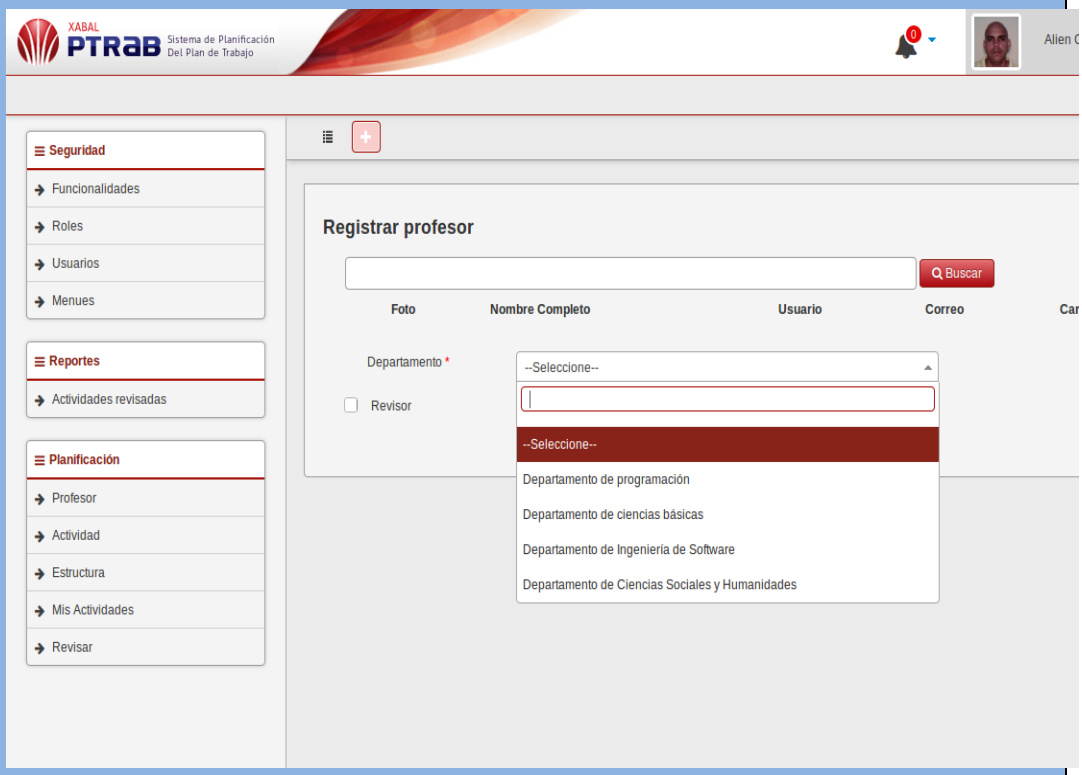
No	Nombre	Descripción	Complejidad	Prioridad para el cliente
RF6	Listar actividades		Alta	Alta
	Prototipo			



Prototipo Listar actividades

Campos	Tipos de datos	Reglas o restricciones
Buscar	texto	Campo obligatorio/ Admite cadena de caracteres que sean solo letras, números y espacios
Observaciones:		

TABLA 21: DESCRIPCIÓN DE FUNCIONALIDAD: REGISTRAR PROFESOR

No	Nombre		Descripción	Complejidad	Prioridad para el cliente
RF11	Registrar profesor			Alta	Alta
		Prototipo			
				
		Prototipo Registrar profesor			

	Campos		Tipos de datos	Reglas o restricciones
	Buscar		texto	Admite cadena de caracteres que sean solo letras, números y espacios.
	Departamento		No procede	Campo obligatorio
	Revisar		No procede	No procede
	Observaciones:			

Tarjetas CRC (Contenido, Responsabilidad, Colaboración).

TABLA 22: TARJETA ESTRUCTURA

TARJETA CRC	
Clase: Estructura	
Responsabilidades:	Colaboración:
Permite cargar árbol de estructura Permite modificar datos de una determinada estructura Permite eliminar estructura Permite registrar una estructura Permite obtener una cantidad de estructuras dado parámetros	

TABLA 23:TARJETA ROL

TARJETA CRC	
Clase: roll	
Responsabilidades:	Colaboración:
Permite registrar un usuario Permite eliminar un usuario	

Permite modificar un usuario Permite encriptar la contraseña del usuario para mayor seguridad.	

Historias de usuario

TABLA 24: HU-ELIMINAR ACTIVIDAD

HISTORIA DE USUARIO	
Número: 2	Nombre: Eliminar actividad.
Usuario: Trabajadores (profesores del departamento).	
Prioridad en Negocio: Alta.	Iteración Asignada: 1
Riesgo en Desarrollo: Medico.	Puntos Estimados: 2
Descripción: Los usuarios en dependencia de su roll como profesores del departamento, tendrán opciones de eliminar una determinada actividad, suprimiendo la actividad del sistema.	
Observaciones: Existen profesores que no tienen permisos para la eliminación de una actividad.	

TABLA 25: MODIFICAR ACTIVIDAD

HISTORIA DE USUARIO	
Número: 4	Nombre: Modificar actividad.
Usuario: Trabajadores (profesores del departamento).	
Prioridad en Negocio: Alta.	Iteración Asignada: 1
Riesgo en Desarrollo: Medio.	Puntos Estimados: 2
Descripción: Los usuarios en dependencia de su roll como profesores del departamento, tendrán opciones de modificar una determinada actividad, variando los campos de fecha de inicio, fecha de fin, descripción de la actividad y la fecha de creación de la misma.	
Observaciones: Existen profesores que no tienen permisos para la eliminación de una actividad.	

TABLA 26: HU-MODIFICAR PROFESORES

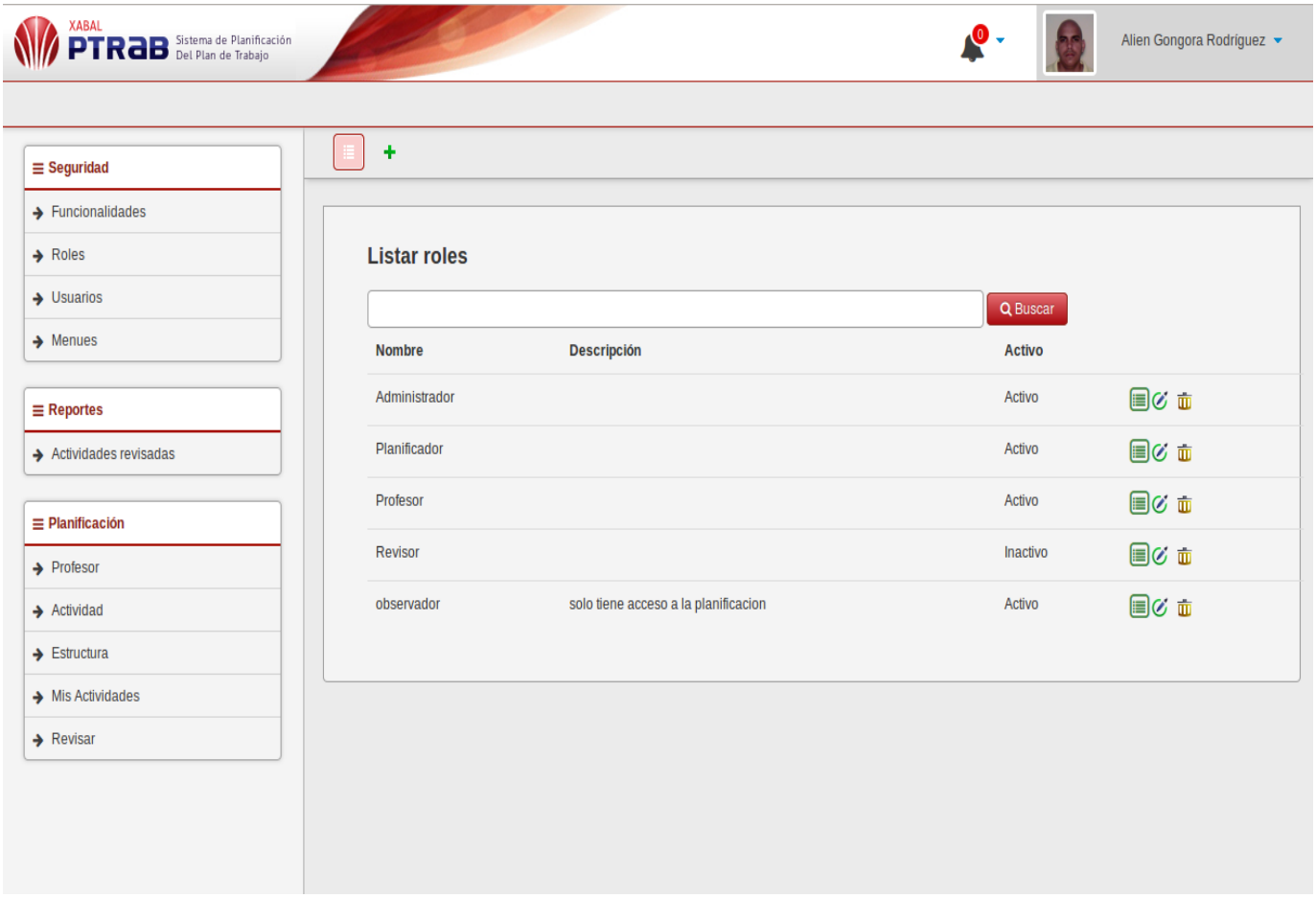
HISTORIA DE USUARIO	
Numero: 14	Nombre: Modificar profesores.
Usuario: Trabajadores (profesores del departamento).	
Prioridad en Negation: Alta.	Iteración Asignada: 2
Riesgo en Desarrollo: Medio.	Puntos Estimados: 0.5

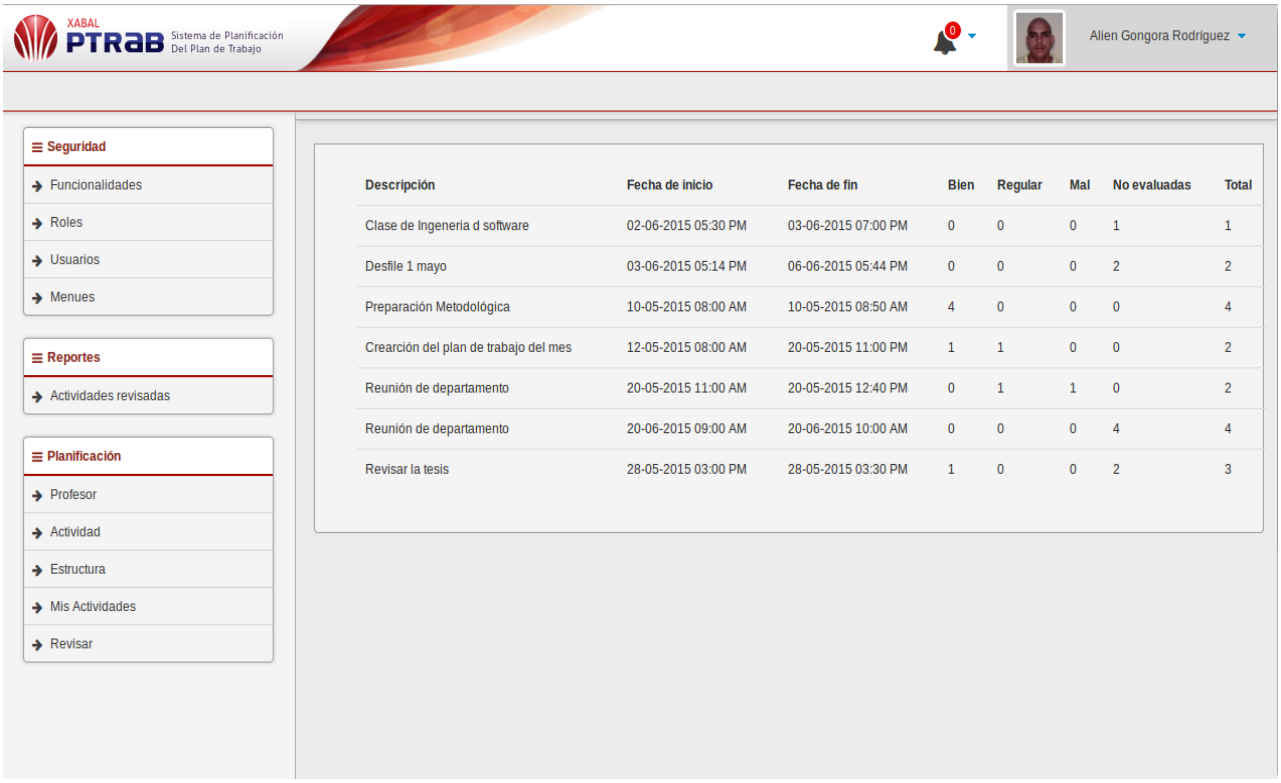
Descripción: Los usuarios en dependencia de su roll como profesores del departamento, tendrán opciones de modifica los datos de un determinado profesor, teniendo en cuenta los datos del profesor , nombre_completo, correo, usuario, credencial(solapin), foto y se define el departamento.
Observaciones: Existen profesores que no tienen permisos para la modificación de los datos de un determinado profesor.

TABLA 27: ASOCIAR MENÚ A ROL

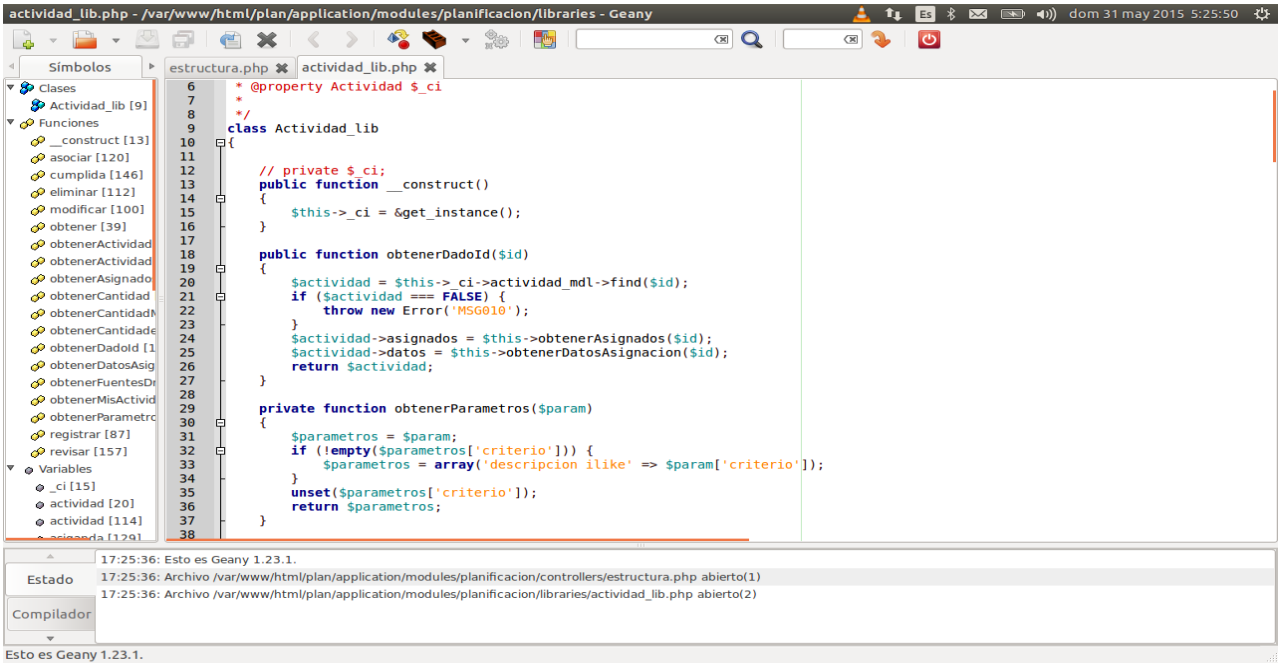
HISTORIA DE USUARIO	
Número: 38	Nombre: Asociar menú a rol.
Usuario: Trabajadores (profesores del departamento).	
Prioridad en Negocio: Alta.	Iteración Asignada: 1
Riesgo en Desarrollo: Alta.	Puntos Estimados: 2
Descripción: Se definen los menuces a los que los usuario con este rol tendrán acceso.	
Observaciones:.	

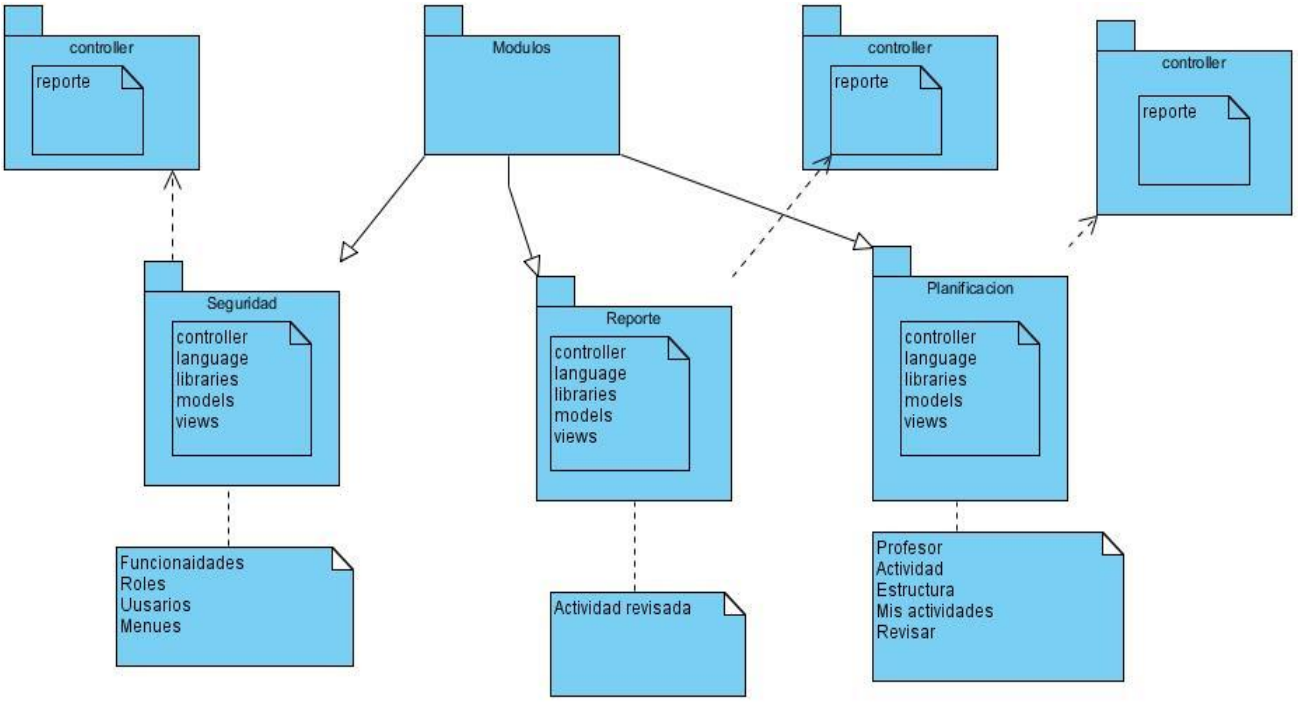
Imágenes


XABAL
PTRaB
Sistema de Planificación
Del Plan de Trabajo
Alien Gongora Rodríguez
Seguridad
Funcionalidades
Roles
Usuarios
Menues
Reportes
Actividades revisadas
Planificación
Profesor
Actividad
Estructura
Mis Actividades
Revisar
Listar roles
Buscar
Nombre
Descripción
Activo
Administrador
Activo
Planificador
Activo
Profesor
Activo
Revisor
Inactivo
observador
solo tiene acceso a la planificacion
Activo


XABAL
PTRaB
Sistema de Planificación
Del Plan de Trabajo
Alien Gongora Rodriguez
Seguridad
Funcionalidades
Roles
Usuarios
Menues
Reportes
Actividades revisadas
Planificación
Profesor
Actividad
Estructura
Mis Actividades
Revisar
Descripción
Fecha de inicio
Fecha de fin
Bien
Regular
Mal
No evaluadas
Total
Clase de Ingeneria d software
02-06-2015 05:30 PM
03-06-2015 07:00 PM
0 0 0 1 1
Desfile 1 mayo
03-06-2015 05:14 PM
06-06-2015 05:44 PM
0 0 0 2 2
Preparación Metodológica
10-05-2015 08:00 AM
10-05-2015 08:50 AM
4 0 0 0 4
Crearción del plan de trabajo del mes
12-05-2015 08:00 AM
20-05-2015 11:00 PM
1 1 0 0 2
Reunión de departamento
20-05-2015 11:00 AM
20-05-2015 12:40 PM
0 1 1 0 2
Reunión de departamento
20-06-2015 09:00 AM
20-06-2015 10:00 AM
0 0 0 4 4
Revisar la tesis
28-05-2015 03:00 PM
28-05-2015 03:30 PM
1 0 0 2 3


actividad_lib.php - /var/www/html/plan/application/modules/planificacion/libraries - Geany
dom 31 may 2015 5:25:50
Símbolos
estructura.php
actividad_lib.php
Clases
Actividad_lib [9]
Funciones
__construct [13]
asociar [120]
cumplida [146]
eliminar [112]
modificar [100]
obtener [39]
registrar [87]
revisar [157]
Variables
_ci [15]
actividad [20]
actividad [114]
* @property Actividad $_ci
*
*/
class Actividad_lib
{
// private $_ci;
public function __construct()
{
$this->_ci = &get_instance();
}
public function obtenerDadoId($id)
{
$actividad = $this->_ci->actividad_mdl->find($id);
if ($actividad === FALSE) {
throw new Error('MSG010');
}
$actividad->asignados = $this->obtenerAsignados($id);
$actividad->datos = $this->obtenerDatosAsignacion($id);
return $actividad;
}
private function obtenerParametros($param)
{
$parametros = $param;
if (!empty($parametros['criterio'])) {
$parametros = array('descripcion ilike' => $param['criterio']);
}
unset($parametros['criterio']);
return $parametros;
}
Estado
Compilador
17:25:36: Esto es Geany 1.23.1.
17:25:36: Archivo /var/www/html/plan/application/modules/planificacion/controllers/estructura.php abierto(1)
17:25:36: Archivo /var/www/html/plan/application/modules/planificacion/libraries/actividad_lib.php abierto(2)
Esto es Geany 1.23.1.


actividad_lib.php - /var/www/html/plan/application/modules/planificacion/libraries - Geany
dom 31 may 2015 5:25:50
Símbolos
estructura.php
actividad_lib.php
Clases
Actividad_lib [9]
Funciones
__construct [13]
asociar [120]
cumplida [146]
eliminar [112]
modificar [100]
obtener [39]
Variables
_ci [15]
actividad [20]
actividad [114]
6 * @property Actividad $_ci
7 *
8 */
9 class Actividad_lib
10 {
11
12 // private $_ci;
13 public function __construct()
14 {
15 $this->_ci = &get_instance();
16 }
17
18 public function obtenerDadoId($id)
19 {
20 $actividad = $this->_ci->actividad_mdl->find($id);
21 if ($actividad === FALSE) {
22 throw new Error('MSG010');
23 }
24 $actividad->asignados = $this->obtenerAsignados($id);
25 $actividad->datos = $this->obtenerDatosAsignacion($id);
26 return $actividad;
27 }
28
29 private function obtenerParametros($param)
30 {
31 $parametros = $param;
32 if (!empty($parametros['criterio'])) {
33 $parametros = array('descripcion ilike' => $param['criterio']);
34 }
35 unset($parametros['criterio']);
36 return $parametros;
37 }
38
Estado
Compilador
17:25:36: Esto es Geany 1.23.1.
17:25:36: Archivo /var/www/html/plan/application/modules/planificacion/controllers/estructura.php abierto(1)
17:25:36: Archivo /var/www/html/plan/application/modules/planificacion/libraries/actividad_lib.php abierto(2)
Esto es Geany 1.23.1.


controller
reporte
Modulos
controller
reporte
controller
reporte
Seguridad
controller
language
libraries
models
views
Reporte
controller
language
libraries
models
views
Planificacion
controller
language
libraries
models
views
Funcionaidades
Roles
Uusarios
Menues
Actividad revisada
Profesor
Actividad
Estructura
Mis actividades
Revisar