Universidad de las Ciencias Informáticas

Facultad 3

Título: Sistema de gestión para la planificación y control del horario docente en la Facultad 3

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Isis Dalia García Rojas

Tutor(es): MSc. Yalice Gámez Batista

Ing. Juniel Tamayo Hernández

"Junio, 2015"

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Isis Dalia García Rojas

_____

Firma del Autor

Yalice Gámez Batista

_____

Firma del Tutor

Juniel Tamayo Hernández

_____

Firma del Tutor

RESUMEN

Las Tecnologías de la Información y las Comunicaciones (TIC) con su avance vertiginoso, ocupan un papel cada vez más determinante en el manejo de la información de cualquier institución u organismo. Dentro de ellas destacan los sistemas de gestión que son cruciales para el manejo de grandes volúmenes de datos. En la Universidad de las Ciencias Informáticas (UCI), en la Facultad 3 se imparte la especialidad de Ingeniería en Ciencias Informáticas. Para garantizar la calidad del proceso docente -educativo es preciso gestionar actividades relacionadas con el horario docente y el control de su cumplimiento. Actualmente se utiliza un sistema con muchas limitaciones que afectan el proceso y generan inconformidades en la planificación docente.

La presente investigación propone un sistema de gestión para la planificación y control docente. Para ello se estableció el estado del arte asociado a sistemas similares, que permitió fundamentar las funcionalidades del sistema, así como seleccionar la metodología de desarrollo, las tecnologías, los lenguajes y las herramientas que se ajustan a sus características. Entre las facilidades que brinda el sistema está: la planificación automática del horario docente, y llevar el control sobre el cumplimiento del mismo. Con el desarrollo de este sistema se espera que los involucrados en este proceso cuenten con una herramienta de trabajo que sea capaz de realizar una rápida planificación del horario docente y que ayude en la toma de decisiones en este ámbito.

Palabras Claves: sistemas de gestión, proceso docente-educativo, horario docente, planificación y control docente.

ABSTRACT

Information and Communications Technology (ICT) play a fundamental role in the managing of great quantities of information in any Company or Institution as they go in advance. Within these technologies, management systems are crucial to handle huge volume of data. The specialty of engineer on informatics science is given at Faculty 3 in the University of Informatics Science. To guarantee the quality of the educational process is necessary to manage activities related to timetable and its fulfillment control. Nowadays, a much more limited system is on use which affects the process and generates nonconformity in the teaching planning.

The current research is intended to propose a management system aimed at planning and controlling the teaching process. In order to carry out the investigation, the process began by setting up the state of art associated to similar systems which allowed to back up the system functional tasks and selection of the development methodology, technologies, languages and tools that fit the characteristics. Among the facilities this system offers are: timetable automatic planning and further control in the achievement thereof. With the development of this system, the people involved in the process, count on a working tool, capable of planning fast the teaching timetable and helping in making decisions at this scope.

***Keywords**: Management system, teaching process, teaching timetable, planning and teaching control.*

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) están cada vez más presentes en todas las esferas de la sociedad. Las potencialidades que ofrecen en cuanto a la velocidad de procesamiento, difusión y organización, hacen que sea cada vez más necesaria e incluso imprescindible su utilización. Su aplicación en diversas aristas del sector educativo ha causado un gran impacto, sobre todo en las instituciones académicas, donde se han obtenido resultados positivos en cuanto a dinamismo, agilidad y disponibilidad de la información, entre otros aspectos.

Precisamente, dentro de los factores claves que pueden contribuir de forma significativa a mejorar la calidad de los servicios formativos que ofrecen las universidades encontramos la asimilación y uso apropiado de las TIC. Es por ello que toda institución educativa debe poseer una cantidad suficiente de recursos en infraestructura y medios que le permita impartir una educación de alta calidad. La existencia de dichos recursos requiere de una correcta planificación y análisis, puesto que tanto su escasez como su sobreoferta influyen negativamente en calidad de la educación. El deterioro de la educación también se pone de manifiesto cuando la planificación del proceso docente no es del agrado de los profesores y alumnos implicados en ella (Baquero et al. 2008).

De ahí que, uno de los problemas recurrentes en los centros de la enseñanza superior cubana es el tema de la elaboración de los horarios docentes, por las disímiles condicionantes que inciden en dicho proceso. Los problemas de planificación de horarios consisten básicamente en fijar una serie de encuentros entre profesores y estudiantes en un periodo de tiempo predefinido, teniendo en cuenta un conjunto de restricciones de diferentes tipos. En dependencia del número de recursos implicados, la resolución de este problema de forma manual, puede tardar desde varias horas hasta incluso días [1] y no se garantiza que los resultados obtenidos sean los deseados para todas las partes involucradas.

En la Universidad de las Ciencias Informáticas (UCI), específicamente en la Facultad 3 se desarrollan un conjunto de actividades que tributan directamente al proceso docente- educativo. Una de las más importantes es la relacionada con la planificación de la docencia. Actualmente la ejecución de la planificación docente, incluso a nivel de facultad, se torna compleja. Básicamente se debe a que este proceso no está completamente automatizado, y además en él intervienen la planificadora, la vicedecana de formación y todos los profesores principales de año (cuatro en este caso), para un total de seis trabajadores. Ello implica que se consuma mucho tiempo para realizar este proceso, que la información no posea un orden consecuente de almacenamiento y en ocasiones que se pierda la información. Actualmente los profesores principales de año reciben una propuesta de horario en un libro Excel, realizan las adecuaciones pertinentes y la planificación resultante es montada por la planificadora en el sistema HORPLANNER que establece la

[1] Tomado de las entrevistas realizadas a la planificadora de la Facultad 3 de la Universidad de las Ciencias Informáticas

universidad. Esta aplicación facilita la elaboración del horario pero no cuenta con todas las funcionalidades necesarias para una adecuada planificación. Esto provoca que el trabajo del vicedecano de formación y la planificadora se vea limitado en eficiencia, ocasionando insatisfacción en aquellos que puedan estar involucrados, molestias innecesarias y pérdida de tiempo en ocasiones.

Por todo lo anteriormente argumentado se identifica como **problema a resolver** ¿Cómo mejorar la calidad de la gestión de las actividades docentes de la Facultad 3?

La investigación está enmarcada por el **objeto de estudio**: gestión de actividades docentes, incidiendo en el **campo de acción**: planificación y control del horario docente de la Facultad 3.

En correspondencia con el problema identificado y el objeto de estudio, se propone como **objetivo general:** desarrollar un sistema de gestión para la planificación y control del horario docente en la Facultad 3, que permita reducir el tiempo necesario para su diseño, teniendo en cuenta todos los requisitos y restricciones del proceso definidos por el cliente.

Para dar cumplimiento al objetivo general se plantean los siguientes **objetivos específicos**:

1. Elaborar el marco teórico de la investigación relacionado con conceptos y herramientas empleadas en la planificación de actividades.
2. Diagnosticar el estado que presenta actualmente el proceso de planificación del horario docente en la Facultad 3, para identificar características generales, deficiencias y otros elementos a tener en cuenta durante el desarrollo de la solución.
3. Desarrollar un sistema de gestión que automatice la planificación y control del horario docente de la Facultad 3.
4. Validar la propuesta, a partir de pruebas de software y de su ejecución práctica.

Para cumplir con los objetivos específicos planteados, se desarrollan las siguientes **Tareas de la investigación:**

1. Establecimiento del marco teórico relacionado con los conceptos fundamentales de la gestión de actividades docentes.
2. Análisis de herramientas empleadas en la planificación de actividades.
3. Realización de una entrevista que permita recopilar datos acerca del estado que presenta actualmente el proceso de planificación de los horarios docentes en la Facultad 3.
4. Identificación de características generales, deficiencias y otros elementos a tener en cuenta durante el desarrollo de la solución.
5. Selección de las herramientas, metodologías y tecnologías adecuadas para el desarrollo de la solución.
6. Implementación de la solución, teniendo en cuenta las herramientas, metodologías y tecnologías seleccionadas.
7. Realización de pruebas de software a la solución desarrollada.

8. Aplicación práctica de la solución con datos hipotéticos.

La investigación realizada se sustenta en la siguiente **idea a defender**: Un sistema de gestión para la planificación y control del horario docente en la Facultad 3, contribuirá a mejorar la calidad de la gestión de las actividades docentes de esta.

Para la realización de las tareas propuestas se utilizaron los siguientes **métodos científicos de investigación.**

Dentro de los **métodos teóricos** se recurre al:

- El **Histórico-lógico,** posibilita realizar un estudio sobre los antecedentes de los sistemas de gestión de la planificación, facilitando el análisis de estos sistemas para su mejor comprensión, realizar el análisis de la evolución, caracterización y revelar las insuficiencias que se manifiestan en el desarrollo de dicho proceso.
- **Modelación:** se modela el desarrollo de la solución a partir de los artefactos que establece la metodología de desarrollo y el modelo de datos de la solución propuesta
- **Sistematización:** la confluencia de diferentes disciplinas académicas aporta a la investigación variados elementos teóricos que la sustentan. En este sentido los conceptos más empleados provienen de las materias Programación e Ingeniería de software, que aportaron los elementos necesarios para la concepción, diseño, desarrollo y prueba de la solución.

Los **métodos empíricos** complementan el seguimiento a la evolución del fenómeno que se estudia, los empleados en este caso son:

- La **entrevista** a los implicados en el proceso de planificación y control docente, permite recopilar datos acerca del estado que presenta actualmente el proceso de planificación de los horarios docentes en la Facultad 3.
- **Medición:** se usa para medir la calidad de la especificación de los requisitos y el grado de ambigüedad de los mismos, además de obtener una medida de la calidad del diseño para su validación.

El presente trabajo está estructurado de la siguiente manera:

Capítulo 1: Fundamentación teórica.

Este capítulo contiene las cuestiones teóricas necesarias para la comprensión de la investigación, lo cual incluye el establecimiento del estado del arte a través del estudio de herramientas de planificación a nivel internacional y nacional. Se realiza un análisis y selección de la metodología de desarrollo en función de las necesidades del sistema y las características del equipo de desarrollo, así como el estudio de las herramientas y tecnologías a emplear en el desarrollo de la solución.

Capítulo 2: Análisis y diseño del sistema para la planificación y control del horario docente de la Facultad 3.

En este capítulo se realiza una descripción de la propuesta del sistema y sus principales funcionalidades. Se expone todo el diseño de la aplicación con los artefactos definidos por la metodología para esta fase.

Capítulo 3: Implementación y prueba del sistema para la planificación y control del horario docente de la Facultad 3.

En este capítulo se elabora la solución propuesta evidenciándose las funcionalidades del sistema de gestión para la planificación y control del horario docente en la Facultad 3. Se muestran los resultados de la validación de los requisitos y el diseño, a través de la aplicación de técnicas y métricas respectivamente. Además, se hace un resumen de los resultados obtenidos tras la aplicación de las pruebas de funcionamiento y unitarias realizadas al sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se analizan los principales conceptos, con el fin de promover el soporte teórico de la investigación. Además se realiza un análisis de los sistemas de gestión de planificación existentes a nivel internacional y nacional. De igual forma se puntualizan las principales características y funcionalidades de las tecnologías a emplear en el desarrollo de la solución propuesta.

1.1 Conceptos asociados a la planificación y control de actividades.

A continuación se exponen los principales conceptos relacionados con la planificación y control de las actividades docentes. Se destacan los conceptos de actividad docente, planificación docente y control ya que son los conceptos básicos para el desarrollo de esta investigación.

Control

Stephen Robbins define el control como "un proceso de vigilar las actividades para cerciorarse de que se desarrollan conforme se planearon y para corregir cualquier desviación evidente" (Robbins, 2005). En tanto, James Stoner añade que "el control administrativo es el proceso que garantiza que las actividades reales se ajusten a las actividades proyectadas" (Stoner et al., 1996).

Para los efectos de esta investigación se asume la conjunción de ambos enfoques. Definiéndose entonces como control a aquella acción que contribuye a medir y corregir la labor ejecutada por los trabajadores. Con el fin de lograr los propósitos de acuerdo a la planificación, y que además posibilita la comparación entre lo que se realiza y lo que se planifica.

Actividad

Según Miller, al conjunto de operaciones o tareas que le son propias a una empresa o a una persona se designan con el término de actividad; actividad empresarial, actividad docente (Miller y Spoolman 2011). Otra definición plantea que la actividad es el conjunto de acciones que se llevan a cabo para cumplir las metas. Consiste en la ejecución de ciertos procesos o tareas (mediante la utilización de los recursos humanos, materiales, técnicos, y financieros asignados a la actividad) (Gvirtz, 1998).

Se concluye a partir de estas definiciones que la actividad es el conjunto de acciones que se llevan a cabo para cumplir las metas, que consiste en la realización de tareas mediante la utilización de recursos.

Actividad docente

A partir de la definición de actividad, se define actividad docente como el conjunto de acciones planificadas llevadas a cabo por docentes y estudiantes, dentro o fuera del aula, de carácter individual o grupal, que tienen como propósito alcanzar los objetivos y finalidades de la enseñanza. Para los efectos de esta investigación se consideran como las actividades docentes las formas de organización que establece el reglamento docente metodológico: Conferencia, Clase Práctica,

Taller, Laboratorio, Seminario, Prueba Parcial, Corte de Tarea Extraclase, Corte de Trabajo de Curso, Defensa de Trabajo Extraclase y Defensa de Trabajo de Curso (Vela, 2007).

Recurso

Se denominan recursos a aquellos elementos que aportan algún tipo de beneficio a la sociedad. La palabra recurso se emplea en diversos ámbitos, pero siempre con el significado de ser medio para el logro de fines (Miller, 2011). A partir de este concepto se puede concluir que los recursos pueden ser utilizados por el hombre para realizar una actividad o como medio para lograr un objetivo.

Aunque existen varios tipos de recursos, esta investigación se centra en los siguientes:

- Recursos humanos: trabajo que aportan las personas, ya sea de forma individual o grupal, dentro de una organización.
- Recursos materiales: son los bienes tangibles que la organización puede utilizar para el logro de sus objetivos.
- Recurso tiempo: recurso no renovable, de gran valor para la gestión de las actividades, además de ser limitado y caduco por definición.

Restricción

El término restricción puede utilizarse en diferentes esferas, sin embargo en la mayoría de estas implicará lo mismo: una limitación o una reducción ya sea natural o impuesta, según corresponda, es decir algún impedimento o limitación a la hora de realizar alguna tarea (Goldratt, 2010). En correspondencia con este concepto se asume que la restricción es lo que establece límites, topes, o impide superar ciertos máximos.

Planificación

La planificación es necesaria para casi todos los aspectos de la vida y la conducta humana. Establece las bases para definir las metas correctas y después elegir los medios necesarios para alcanzarlas. Además constituye un excelente instrumento para mitigar los riesgos y afrontar las inseguridades, debido a que puede tomar en cuenta la posible ocurrencia de sucesos imprevistos.

El concepto de planificación es uno de los más abordados y constatados en la literatura. Algunos autores como Stoner conciben a la planificación como la acción de establecer metas y elegir medios para alcanzar dichas metas (Stoner et al. 1996). Dicho de esta manera la planificación es una acción matizada por el realismo, y orientada al fin en disminución de los medios para lograr las metas.

Por su parte, Barriga plantea que la planificación consiste en decidir con anticipación lo que hay que hacer, quién tiene que hacerlo, y cómo deberá hacerse (Barriga, 2009), lo cual distingue una mayor preocupación por el método a emplear para lograr dichas metas.

Por otro lado Drucker plantea: "Es el proceso de definir el curso de acción y los procedimientos requeridos para alcanzar los objetivos y metas. El plan establece lo que hay que hacer para llegar al estado final deseado" (Drucker ,2014).

Uno de los criterios más elaborados en lo concerniente a la planificación proviene de Jiménez, quien entiende a la misma como el proceso consciente de selección y desarrollo del mejor curso de acción para lograr el objetivo. Además, acota dicho autor que la planificación implica conocer el objetivo, evaluar la situación, considerar diferentes acciones que puedan realizarse y escoger la mejor. Estos elementos proponen un análisis sistémico de las acciones y/o tareas que se planifican. En la investigación se asume en concordancia con lo planteado por Jiménez, que la planificación es un proceso de toma de decisiones para alcanzar un futuro deseado, a partir de las condicionantes sociales que le imprime la situación actual, y los factores internos y externos que pueden influir en el logro de los objetivos (Castro ,1982).

Planificación docente

La definición del concepto de planificación expuesta anteriormente facilita la explicación del concepto de planificación docente. Sintéticamente puede definírsele como la distribución coherente y funcional de un conjunto de actividades destinadas al eficaz aprendizaje de los estudiantes tratando siempre de lograr el óptimo uso de los recursos disponibles, considerando los factores que limitan la planificación como son las restricciones.

1.2 Descripción actual del dominio del problema

En la Facultad 3 de la Universidad de las Ciencias Informáticas (UCI) el horario docente se confecciona para cada semestre. El mismo sufre cambios en el transcurso del período, generalmente de forma semanal. Para realizarlo se necesita una serie de documentos, los cuales son indispensables para que los profesores principales de año (PPA) realicen la propuesta de planificación. Los documentos utilizados son los siguientes: Indicaciones para la planificación docente, P1 de las asignaturas, P4 del semestre, Gráfico calendario, Planificación de las asignaturas de Práctica Profesional, Matrícula de las asignaturas optativas, Disponibilidad de locales por tipo, Carga docente, Plan anual de actividades de la Facultad. Asimismo deben considerarse las afectaciones de los profesores como días de preparación metodológica, profesores externos, madres con niños, maestrías, doctorados, etc., y las del departamento en general.

Para la elaboración de la planificación docente, cuando se tiene toda la documentación antes mencionada, la planificadora monta en el sistema la maqueta del horario en una primera propuesta. Esta propuesta se discute en los colectivos de año donde los profesores principales de año toman los acuerdos de modificaciones y luego lo concilian con la planificadora. Esta se encarga de actualizar el horario en el sistema y, con el visto bueno y aprobación de los profesores principales

de año se publica en el sitio establecido por la universidad. En caso de ser necesario, pueden realizarse adecuaciones en el horario de forma semanal, siempre que se cuente con la aprobación de los jefes de departamento y de la vicedecana de formación.

Para la realización de la planificación en la universidad se utiliza la herramienta HORRPLANNER. Este sistema es una aplicación portable que genera una planificación descrita en un fichero XML que luego se copia en un servidor central y se publica para toda la universidad a través de un sitio web. Este sistema permite la planificación no automática del horario ajustada a los P1 de cada una de las asignaturas (respetando la restricción de orden de los turnos y por semanas). Tiene en cuenta además, la coincidencia de los locales y de profesores, el gráfico calendario de la universidad con las fechas, la asignación de locales por defecto, la personalización de las asignaturas por grupo docentes, realización de reportes vía web de profesores, locales, asignaturas, entre otras facilidades que contribuyen a una mejor planificación. Sin embargo se realizó un análisis de la solución mencionada que permitió identificar aspectos nuevos o mejorables tanto tecnológicos como funcionales. Entre ellos se pueden mencionar:

1. El sistema sólo se puede ejecutar sobre el sistema operativo Windows.
2. No realiza una propuesta de horario que atienda a restricciones sino que limita el accionar de los usuarios.
3. Cuando se modifica el horario para un año académico en el momento de la publicación puede generar cambios en otro.
4. No permite la asignación de dos profesores a un mismo grupo en el sistema de profesores clases prácticas y conferencias.
5. No permite más de un grupo en los salones de conferencias.
6. No tiene en cuenta que cuando se planifica actividad en el tercer turno no se puede planificar en cuarto turno para los mismos profesores y estudiantes.
7. No se tiene en cuenta lo profesores externos para la planificación de 6to turno.
8. No permite más de 6 turnos de una asignatura lo que afecta a las asignaturas de la Práctica Profesional.
9. No permite gestionar el control del cumplimiento del horario docente.

Como parte de la constatación del estado de la planificación docente en la Facultad 3 se aplicó una encuesta a un total de dieciséis profesionales, con distintos niveles de responsabilidades y experiencia en este tema. De las dieciséis encuestas se descarta una por haberse autoevaluado con muy bajo nivel de experticia (1 de 10). El promedio de semestres de experiencia en la planificación de los encuestados es de aproximadamente once semestres y el nivel de conocimiento nueve. De los aspectos nuevos o mejorables antes mencionados se evaluaron, los correspondientes a los números del tres al nueve. Los resultados arrojados se muestran en la siguiente figura (ver Ilustración 1).

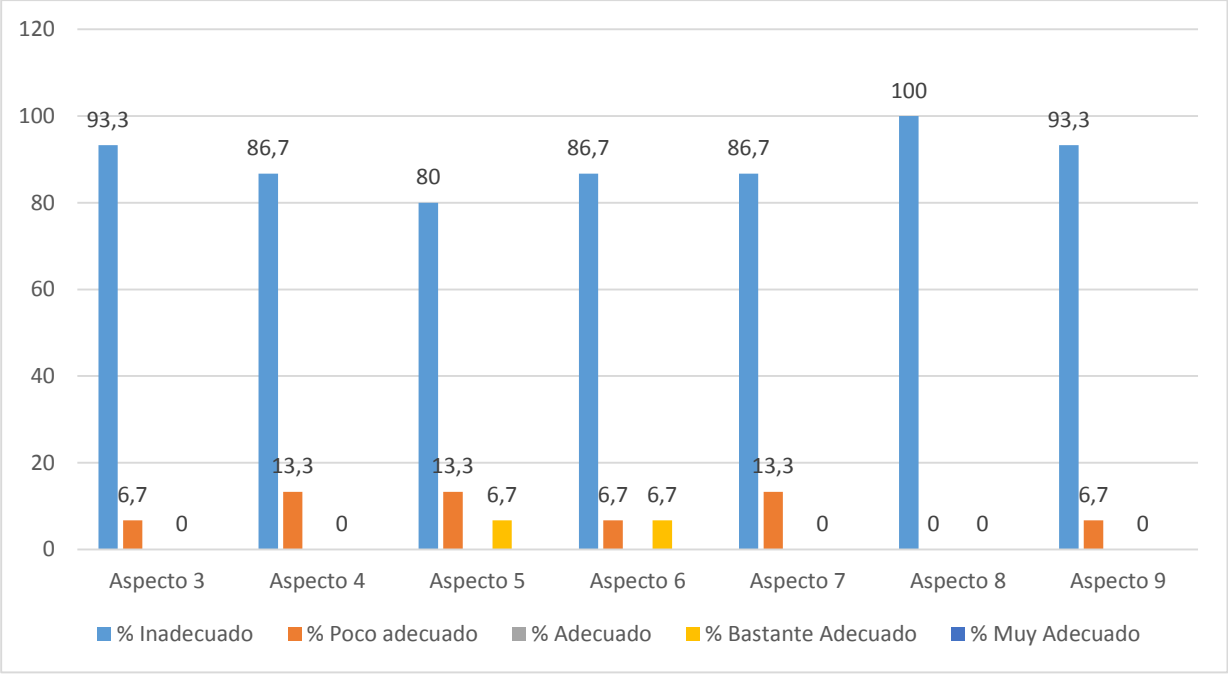

Ilustración 1: Resultado de las encuestas para la constatación del estado de la planificación docente en la Facultad 3.

Como se pudo constatar el sistema de planificación HorrPlanner presenta aspectos que no favorecen una programación adecuada de las actividades docentes. De estos aspectos los que más influyen negativamente son el tres, el ocho y el nueve, relacionados con la confiabilidad del sistema en la publicación, con la Práctica Profesional y con el control del cumplimiento del horario docente. Por lo que se evidencia la necesidad de un sistema que dé respuestas a estas inconformidades e incorpore las nuevas funcionalidades que puedan ser identificadas por el cliente (ver Anexo 1).

1.3 Sistemas automatizados existentes vinculados al proceso de planificación de actividades

1.3.1 Sistemas informáticos extranjeros

En la actualidad existen múltiples soluciones informáticas para la confección de horarios docentes. Todos tienen en común que cuentan con funcionalidades para la distribución, elaboración, edición y transferencia de horarios escolares. A nivel internacional entre los más sobresalientes se encuentran los siguientes:

GP-Untis

Este software generador de horarios es uno de los más usados internacionalmente. El “motor de GP-Untis” con el cual se genera la optimización de horarios, está basado en algoritmos heurísticos genéticos y lineales, lo cual permite que el manejo de los datos sea regido por un número limitado de restricciones que el usuario determina. Fue desarrollado por la empresa Untis, de Suiza.

El desarrollo sucesivo y continuo del algoritmo ha logrado que por más de 30 años este software presente las siguientes características (UPware Soft , 2014):

- Obtención de horarios óptimos, siendo insuperable en la calidad y rapidez de los horarios generados.
- Aplicable a todo tipo de centros educativos y orientación pedagógica según las premisas que el centro determine.
- Cálculos rapidísimos y resultados óptimos y convincentes.
- Introduce todo tipo de clases y condiciones, manejo intuitivo y estructura clara, combinados con los últimos avances de Windows.
- Versión "multiusuario" permitiendo que varios usuarios trabajen simultáneamente con el programa.

Las principales desventajas que presenta son:

- Es un software propietario, y el costo de su licencia y actualizaciones es elevado.
- El sistema sólo se puede ejecutar sobre el sistema operativo Windows.

KRONOWIN M-14

Es un software creado por la empresa ADOSSIS, S.A. creadora de sistemas informáticos. Desde su fundación ha dirigido su actividad hacia áreas determinadas dentro de los sectores de informática y electrónica. KRONOWIN es un completo generador de horarios que permite obtener los horarios óptimos de un centro escolar, sean sencillos o muy complejos, en función de la magnitud del centro.

Son diversas las ventajas que presenta este sistema y podrían servir como ayuda para la confección del sistema que se desea. Entre ellas se puede mencionar («KRONOWIN Generador de Horarios», 2015) :

- Se pueden confeccionar horarios individuales de cada grupo, profesor y aula, así como los horarios generales. Admite fijar previamente sesiones (clases), en el horario, antes de su generación. Acepta múltiples condiciones de ubicación de clases, tanto para grupos, profesores y aulas, como independientes para cada asignatura.
- Su uso es muy intuitivo debido a que tiene una ayuda que indica los pasos a seguir. Si el usuario lo desea, el programa toma decisiones, (cambios lógicos) del horario.
- Amplias posibilidades de configuración en todo el programa.
- Entorno gráfico amigable.
- Exportación e importación a distintos formatos y aplicaciones. Formatos oficiales configurables.
- Válido para horarios escolares de cualquier tipo y niveles de estudios.
- Tiene una vista gráfica previa de las modificaciones necesarias al reubicar horas de clases.

También posee algunas características que suponen desventajas que impiden su utilización:

- Es un software propietario, y el costo de su licencia y actualizaciones es elevado.

- El sistema sólo se puede ejecutar sobre el sistema operativo Windows.

Con la revisión de las funcionalidades del sistema, se identificó la necesidad de que el programa cuente con posibilidades de configuración y además tenga un entorno gráfico amigable. Esto va a permitir que el usuario trabaje de forma más cómoda y además incide en su estado anímico.

GHC 16

Gestor de Horarios para Centros de Enseñanza es un administrador muy sencillo para la realización de horarios escolares con multitud de profesores, todo ello atendiendo a preferencias e imposiciones propias. Creado por la empresa española Peñalara Software S.L., para facilitar mecanismos de interfaces de intercambio de datos y horarios con las principales aplicaciones de gestión académica. Su objetivo fundamental es ajustar los horarios escolares semanales observando todas las condiciones necesarias en cada centro de enseñanza. Sus principales características son («Gestor de Horarios para Centros de Enseñanza» 2015) :

- Posee una estructura de datos formada por listas relacionadas que garantiza la integridad de la información.
- Dispone de un generador de grupos de alumnos que ayuda a organizar las asignaturas, partiendo de las peticiones de matrícula para las combinaciones de optativas que se oferten en el curso académico.
- Puede validar la configuración del horario previamente a la generación de resultados, detectando determinadas incompatibilidades.
- Durante cada proceso de generación se facilitan estadísticas de las sesiones de clases, grupos o profesores cuyos datos han dificultado la obtención de resultados, con el fin de analizar las condiciones más conflictivas.
- Tiene una guía para usuarios de acuerdo a la versión en la que se trabaja.

El software cuenta con varias versiones. GHC 16 del año 2014, es la última versión de este experimentado programa informático, que incluye un gran número de mejorías entre las que se pueden mencionar («Gestor de Horarios para Centros de Enseñanza» 2015):

- Planificador: Posibilidad de configurar dos marcos de horarios independientes que se solapan con toda libertad. Mayor facilidad de uso para la configuración del horario.
- Motor: Mejora de la optimización general del horario. Mayor rapidez en la obtención de resultados y más capacidad para hacer que estos sean más equilibrados.
- Editor: Muestra nuevos informes acerca de las condiciones de satisfacción previamente establecida como son: resumen de los resultados por profesor, huecos, etc.

Algunas de las desventajas que presenta la herramienta son:

- Sin importar los datos de su licencia en la instalación del programa, la lista de profesores queda limitada a 10 profesores.
- Las claves son distintas para cada versión.
- El sistema sólo se puede ejecutar sobre el sistema operativo Windows.
- Es un software propietario por lo que su licencia es de un costo considerable.
- Sus versiones cuestan cada vez más en el mercado.

DocCF

Es una aplicación desarrollada por Grupo CF Developer para sistematizar y automatizar los procesos escolares, académicos y administrativos en las instituciones educativas. Como herramienta, el objetivo de DocCF, es gestionar los procesos internos optimizando la comunicación entre cargos directivos, docentes, alumnos y padres de familia para ofrecer información estadística sobre dichos procesos y facilitar la toma de decisiones en el proceso de gestión de la Institución Educativa.

Sus principales funciones («DocCF, Software de Gestión Escolar», 2015):

- Configuración
  - Flexible a cualquier sistema educativo en Latinoamérica y España.
  - Personalización: Se puede ajustar el diseño de los informes de gestión de docentes y alumnos de acuerdo a las necesidades del centro.
- Gestión de docentes
  - Ficha del Docente: Datos personales, fotografía, formación académica y perfil profesional.
  - Evaluación de docencia: Los estudiantes evalúan anónimamente a cada docente según criterios personalizados.
  - Asistente para la asignación de horarios escolares: Valida cruces, horas no asignadas y bloqueos.
  - Informes, gráficos y estadísticas para la gestión docentes.
- Gestión de alumnos
  - Ausentismo: Control de entradas y salidas de salones o espacios restringidos.
- Bitácora del alumno: Registro de incidencias y eventos del alumno en la institución.
  - Gestión de rutas de transporte escolar, programación de paradas y control de horarios.
  - Informes, gráficos y estadísticas para la gestión de alumnos.
- Auditoría
  - Información centralizada. En una sola pantalla puede conocer el estado académico, estado de cuenta, ausentismo, registro de préstamos de libros y registro de incidencias del alumno.
  - Restricción de funciones específicas del Módulo de Gestión de Docentes y Alumnos a usuarios.

- Notificaciones
  - Envío automático y/o programado de notificaciones por email a los padres de familia o acudiente cuando se registra una novedad al alumno en la bitácora, o se realiza el registro de la ausencia del alumno.

Las principales características del software, que suponen desventajas para su uso en la universidad son:

- Es software propietario.
- El sistema sólo se puede ejecutar sobre el sistema operativo Windows

ASC Horarios (Applied Software Consultants)

ASC Horarios es otro de los generadores de horarios. El mismo posibilita cubrir todos los tipos de áreas especiales. Este software presenta las siguientes características («aSc Horario de clases - horario software» 2015):

El programa de horarios de clases ofrece una definición fácil y rápida de asignaturas, clases, aulas, profesores y las horas a la semana enseñadas por cada profesor.

- Es posible juntar varios grupos de diferentes clases para una lección y también agregar varios profesores a una clase.
- El programa verifica la especificación del horario.
- Verifica en el horario ya generado si cumple todas las exigencias del centro para el cual se realizó.
- Se pueden hacer cambios manuales en el horario, al equivocarse el programa le avisa.
- El programa es capaz de generar informes mensuales y anuales de los estudiantes ausentes y de las suplencias de los profesores.

La última versión del software ASC Horarios, se comercializa en el presente año 2015, e incorpora un conjunto de mejoras tales como:

- Simplemente al introducir los requisitos el software de planificación evalúa más de 5.000.000 de posibilidades para obtener un horario bien equilibrado.
- Más rápido y sencillo que versiones anteriores.
- Cuando es necesario hacer cambios, el software es capaz de adaptarse de forma automática a todas las modificaciones.

Algunas de las desventajas que podemos mencionar sobre este software son:

- Es un software propietario.
- El sistema sólo se puede ejecutar sobre el sistema operativo Windows.
- Es solo aplicable para centros pequeños, con capacidad para cuatro grupos y seis profesores.

Pese a estas desventajas el software posee un conjunto de funcionalidades que pueden ser consideradas para la solución que se desea obtener. Estas son:

- Generación automática del horario
- Una vez generado un horario, se le pueden aplicar los ajustes manuales que se desee.
- Verificación del horario: Utiliza un algoritmo que comprueba rápidamente el horario en busca de algún conflicto, y los evita.
- El software se puede utilizar tal y como viene por defecto, o personalizar sus funciones individuales al gusto del usuario
- Es una herramienta útil para programar las sustituciones de profesores, completa con notificaciones e impresión.

El estudio de este sistema evidencia la importancia de funcionalidades como hacer cambios manuales en el horario, ubicar a varios grupos en un mismo local y asignar varios profesores a una misma clase. Además resultaría realmente útil la existencia de una funcionalidad que generase reportes sobre determinada información de forma centralizada, y dado un periodo de tiempo específico, garantizándose el control necesario sobre el cumplimiento de las actividades planificadas.

Tabla 1: Resumen comparativo de los sistemas estudiados a nivel internacional.

Parámetros	GP-Untis	ASCHorarios	KRONOWIN	GCH	DocCF
Tipo de software	Propietario	Propietario	Propietario	Propietario	Propietario
Multiplataforma	No	No	No	No	No
Generación automática del horario	Sí	Sí	Sí	Sí	Sí
Gestión de reportes	Sí	Sí	Sí	Sí	Sí
Flexibilidad	Sí	No	Sí	No	Sí

Como se observa en la Tabla 1 de forma general, todos los sistemas analizados a nivel internacional son software privativos, lo que impide modificar sus características en función de las necesidades propias de la facultad. Además en su conjunto son software que sólo se pueden ejecutar sobre el sistema operativo Windows, lo que supone un inconveniente al no encontrarse en correspondencia con la política de migración a software libre existente en el país.

1.3.2 Sistemas informáticos cubanos

A nivel nacional se tomó como referencia la UCI debido a las particularidades que presenta en la organización del proceso docente. Por estas razones se analizaron las características de los

sistemas para el horario docente desarrollados en la universidad, lo que arrojó importantes resultados. Entre las soluciones propuestas dentro de la universidad se destacan las siguientes:

"Sistema informático de gestión para actividades docentes y extra-docentes en la Facultad 3. Rol analista de sistemas.", desarrollada en 2007, la cual entre sus principales funcionalidades propone (Cuevas Brugal e Hidalgo García ,2007) :

- Gestionar Balance de Carga
- Gestionar Afectaciones de los Profesores
- Gestionar Afectaciones de la Facultad
- Gestionar Profesores
- Publicar Horario
- Actualizar Horario
- Consultar Planificaciones
- Gestionar Guardia
- Gestionar TSU

Esta investigación no concluyó con la construcción de un software utilizable, limitándose solo al análisis y diseño.

En 2008 se desarrolló la tesis titulada: **"Análisis y diseño de un sistema para la planificación automatizada del Horario Docente de la Facultad 4"** Este trabajo no concluyó la realización de un software funcional limitándose sólo al análisis y el diseño.

Las principales funcionalidades que presenta esta solución son (Ortíz Ramos y Torres Iglesias, 2008) :

- Gestionar Horario: Para confeccionar el horario se registran los siguientes datos: profesores, asignaturas, grupos, semanas, locales, balance de carga.
- Gestionar Balance de Carga: De cada balance de carga se registran los siguientes datos: asignatura, semana, frecuencias semanales, tipo de clase (clase práctica, laboratorio, conferencia), orden.

En 2009 se desarrolló la tesis: **"Sistema de Gestión para la Planificación Docente en la Facultad 5",** investigación enfocada a los procesos de la guardia docente y el cuidado de exámenes. Las características fundamentales que presenta son (Álvarez Hernández, Martínez Vázquez y Souchay Fabrega, 2009) :

- Sistema multiplataforma
- Aplicación web

Entre las principales funcionalidades de este sistema se encuentran:

- Planificar Cuidado de Exámenes
- Crear Plantilla en la Guardia
- Planificar Guardia Docente
- Mostrar listado de exámenes
- Mostrar Plantilla en la Guardia Docente
- Gestionar Grupos
- Mostrar Cuidado de Exámenes
- Mostrar Guardia Docente

En las facultades regionales, pertenecientes a la UCI, también se han realizado investigaciones en este campo como es el caso del trabajo de tesis titulado **Sistema de Gestión para la planificación docente en la Facultad Regional "Mártires de Artemisa"**, realizado en 2012. La propuesta de solución desarrollada es una aplicación de escritorio, y las funcionalidades que presenta son las siguientes (Abreu Guerra, 2012) :

- Gestionar usuario
- Gestionar Planificaciones
- Planificación de Horarios
- Planificación de Cuidados de Prueba
- Control del Registro de Asistencia
- Registros de Asistencia a Clases y Registros de Cuidados de Prueba

Tabla 2: Resumen comparativo de los sistemas estudiados a nivel nacional.

Parámetros	Facultad 3. (2007)	Facultad 4. (2008)	Facultad 5. (2009)	Mártires de Artemisa. (2012)
Tipo de software	-	-	Libre	Libre
Multiplataforma	-	-	Sí	No
Generación automática del horario	-	-	No	Sí
Gestión de reportes	-	-	Sí	Sí
Flexibilidad	-	-	Sí	Sí

Como se observa en la Tabla 2, se identificaron cuatro sistemas propuestos en trabajo de diploma, de los cuales dos sólo se limitaron al análisis y diseño, y los otros sistemas no están disponibles. A raíz del análisis de estos sistemas se pudo concluir que no existe disponibilidad de una herramienta que pueda dar respuesta al problema identificado por lo que es necesario el desarrollo de un nuevo sistema.

1.4 Metodología, herramientas y tecnologías seleccionadas para el desarrollo de la solución

La elección de la metodología, lenguajes y herramientas de desarrollo es un proceso de gran importancia, ya que de ello depende la calidad del desarrollo y del producto final. Para la realización de este proceso se tuvo en cuenta las características del equipo de desarrollo, así como las del sistema que se desea desarrollar.

Las necesidades y características del equipo de desarrollo a tener en cuenta para el desarrollo de la propuesta de solución, son:

- Se cuenta con un equipo de desarrollo pequeño, del cual el cliente es miembro.
- Es más importante crear un producto software que funcione que escribir documentación exhaustiva.
- El tiempo de desarrollo es corto (máximo 5 meses).

Los parámetros establecidos considerados para el desarrollo de la propuesta de solución y que contribuyeron al análisis de las necesidades y especificaciones del problema planteado, son los siguientes:

- Cantidad de personas que interactúan con el sistema es mínima
- Bajo nivel de acceso de los usuarios
- Dimensión de los datos conocida y limitada
- Centrado en la generación de una planificación, a partir de un algoritmo basado en restricciones
- Aplicación de escritorio
- Portable
- La complejidad del sistema está en el algoritmo (procesamiento de los datos) que genera la planificación

Además, los elementos que se obtienen de la experiencia del usuario nutren al autor de mecanismos informáticos para desarrollar una aplicación de escritorio utilizando la metodología, las herramientas y las tecnologías que se describen a continuación:

1.4.1 Metodología de desarrollo

Metodología de desarrollo de software se puede definir como un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software (Sánchez, 2004). El impacto de elegir la mejor metodología para un equipo, en un determinado proyecto, es trascendental para el éxito del producto. En la actualidad existen diversas metodologías de desarrollo de software para guiar a los equipos de trabajo en la creación de un nuevo software, que atendiendo a su complejidad son separadas en dos grupos:

- **Las metodologías tradicionales**: Este tipo de metodología requiere de una extensa documentación durante todo el ciclo de vida del proyecto, ya que procura prever todo de antemano. Suelen ser eficaces y necesarias cuando se trata de proyectos que requieren de grandes equipos de desarrollo.
- **Las metodologías ágiles:** Por definición, las metodologías ágiles son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno (Canós, 2003). Para este tipo de metodologías es más importante la capacidad de respuesta ante los cambios realizados que el seguimiento estricto de un plan. Se enfatiza en la satisfacción del cliente y promueve el trabajo en equipo.

Se resuelve seleccionar una metodología ágil de desarrollo de software. Entonces es importante definir cuál de las existentes, es la más apropiada para el desarrollo de la propuesta de solución de este trabajo.

De acuerdo a las comunidades internacionales de desarrollo de software las metodologías ágiles más populares son XP (Extreme Programming) y SCRUM. Ellas proveen un conjunto de buenas prácticas y beneficios al equipo de trabajo que las emplee. SCRUM es una forma de gestionar un equipo de manera que trabaje de forma eficiente y de tener siempre medidos los progresos. XP más bien es una metodología encaminada al desarrollo; consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto (Letelier, 2006).

Con el objetivo de aumentar los beneficios de uso de dichas metodologías, en el 2008 se desarrolló un híbrido de ambas denominada **SXP,** la cual ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permiten actualizar los procesos de software para el mejoramiento de la actividad productiva, fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo (Romero, 2011).

Para un mayor entendimiento de los criterios anteriormente expuestos, se establece la siguiente comparación:

Tabla 3: Comparación entre las metodologías XP, SCRUM y SXP. (Elaboración propia)

Aspectos a comparar	XP	SCRUM	SXP
Prácticas de ingeniería	Establece prácticas de ingeniería como: desarrollo basado en pruebas, el enfoque en	No establece prácticas de ingeniería (Peñalver ,2008).	Establece prácticas de ingeniería como el desarrollo basado

	pruebas automatizadas, programación en parejas, diseño simple y refactorización (Peñalver, 2008).		en pruebas, metáforas, programación en parejas, diseño simple y refactorización (Romero, 2011).
Prioridad de desarrollo	Los equipos trabajan en un orden de prioridad estricta determinado por el cliente (Peñalver, 2008).	El propietario del producto SCRUM (cliente) prioriza la acumulación de productos, pero es el equipo quien determina la secuencia en la que se desarrollarán los elementos del backlog (lista de tareas identificadas por el equipo SCRUM durante la planificación del Sprint[2]) (Letelier, 2006).	Los equipos trabajan en un orden de prioridad estricta determinado por el cliente (Romero, 2011).
Tiempo de duración de una iteración	Aproximadamente 2 meses (Peñalver, 2008).	30 días (Letelier, 2006).	30 días. (Romero, 2011).
Cambios en las iteraciones	Los equipos son más susceptibles al cambio dentro de sus iteraciones, siempre y cuando el equipo no haya empezado a trabajar en una característica particular (Peñalver, 2008).	Los equipos no permiten cambios en sus Sprint. Una vez que la reunión de planificación de iteración se ha completado y se ha contraído un compromiso con la entrega de un conjunto de elementos del backlog del producto, estos se mantienen sin cambios hasta el final del Sprint (Letelier, 2006).	No es posible introducir cambios durante el Sprint[3], por lo tanto para planificar su duración hay que pensar en cuánto tiempo se puede comprometer a mantener los cambios fuera del Sprint (Peñalver, 2008).

[2] Sprint: Terminología para definir las iteraciones en la metodología SXP.
[3] Sprint: Terminología para definir las iteraciones en la metodología SXP.

Una vez establecida esta comparación puede concluirse que SXP toma los elementos particulares de SCRUM y XP, logrando mayor productividad, mejores resultados y una organización documental orientada al registro de los elementos más sustanciales del desarrollo de las soluciones sin generar una carga excesiva para el equipo de trabajo. La misma ayuda a fortalecer el trabajo en equipo, permitiendo seguir de forma clara el avance de las tareas a realizar, a partir de la inserción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la producción, aumentando así, el nivel de interés del equipo. Por tanto se concluye utilizar la metodología SXP debido a que evita generar ciertas cantidades de documentación, es una metodología ágil y por sus características es la más adecuada para proyectos de poca duración como este.

SXP consta de cuatro fases principales (Letelier, 2006):

- Planificación-Definición: Donde se establece la visión, se fijan las expectativas y se establece el aseguramiento del financiamiento del proyecto.
- Desarrollo: Es la fase donde se realiza la implementación del sistema hasta que esté listo para ser entregado.
- Entrega: Fase donde se pone en marcha el producto desarrollado y se hace la entrega al cliente.
- Mantenimiento: En esta se lleva a cabo el soporte para el cliente.

Dentro de cada una de estas fases se ejecutan numerosas actividades, tales como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las Historias de Usuario (HU), diseño, implementación, pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización continua, lo que permite mejorar el diseño cada vez que se le añada una nueva funcionalidad.

Artefactos generados por la metodología a utilizar.

Para organizar, planificar y controlar el proceso de desarrollo de software de la solución propuesta se presentan los siguientes tipos de artefactos:

- **Plantilla de Concepción del Sistema**: En la primera fase de SXP se genera la Plantilla Concepción del Sistema. En ella se especifican los aspectos generales organizativos y de concepción del sistema, su objetivo, principales involucrados y otros aspectos que permitan la posterior organización del desarrollo del proyecto.
- **Plantilla Lista de Reserva del Producto (LRP):** La LRP, es una colección organizada y priorizada de los requisitos del producto que se describen como funcionalidades que el sistema debe cumplir en su desarrollo y especifica las cualidades requeridas por el software. Además determina el orden en que se le irá dando cumplimiento a cada requisito recogido según la prioridad establecida en cada uno.

- **Plantilla de Historias de Usuarios (HU):** En la plantilla del modelo de HU del negocio se describen los actores y trabajadores del negocio. Además se presenta un diagrama de historias de usuarios del negocio que permite ver la relación entre los usuarios y las actividades que se realizan.
- **Plantilla de Tareas de Ingeniería (TI):** Las TI son un conjunto de acciones a desarrollar para resolver las HU. Permiten organizar el proceso de implementación además de posibilitar que sea conocido el grado de complejidad de cada HU teniendo en cuenta la cantidad de tareas asociadas a ella. Están dirigidas a dar cumplimiento a la implementación de las funcionalidades definidas en ellas, lo cual crea las bases para el posterior desarrollo de la solución.
- **Plantilla de Entrega:** Entre los artefactos definidos por las metodologías SXP para el flujo de trabajo Implementación, de la fase de Desarrollo, se encuentra el Plan de Entrega. En esta plantilla se recogen las iteraciones a realizar con sus características, además del orden de las HU con su planificación estimada para ser implementadas.
- **Plantilla de Estándar de Código**: El documento recoge el estándar utilizado y su explicación. No existe una plantilla definida para recoger esta información debido a que cada estándar tiene sus características específicas.
- **Plantilla de Casos de Prueba.** En esta plantilla el desarrollador escribe las pruebas realizadas según la HU seleccionada para realizar la comprobación y validar las funcionalidades del sistema y de esta forma saber si está apto para ser liberado.

1.4.2 Lenguaje de modelado UML 2.0

El Lenguaje Unificado de Modelado (UML por sus siglas en inglés), es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software orientado a objetos. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación (Scott, 1999).

El modelado es esencial en la construcción de software para:

- Especificar el comportamiento deseado del sistema.
- Comprender mejor lo que se está construyendo.
- Descubrir oportunidades de simplificación y reutilización.

Fundamentalmente permite al desarrollador visualizar los resultados de su trabajo en esquemas y diagramas estandarizados. Con UML se obtiene un estándar para escribir un "plano" del sistema, incluyendo aspectos conceptuales y concretos dentro de los que se pueden mencionar las funciones del sistema, expresiones de lenguajes de programación y esquemas de bases de datos

1.4.3 Herramienta de modelado

Las herramientas CASE (Computer Aided Software Environment) son diversas aplicaciones informáticas que se utilizan para ayudar a las actividades del proceso de desarrollo de software, como el análisis de requerimientos, el modelado de sistemas, la depuración y las pruebas (Sommerville, 2005).

CASE proporciona al ingeniero la posibilidad de automatizar actividades manuales y de mejorar su visión general de la ingeniería. Las herramientas CASE ayudan a asegurar la calidad de un producto desde su diseño antes de construirlo (Meza, 2015). Muchas son las herramientas CASE utilizadas dentro de las que se encuentran Visual Paradigm, ArgoUML y Rational Rose, por solo mencionar algunas.

Visual Paradigm es una de las herramientas UML CASE considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelente ventajas de interoperabilidad con otras aplicaciones.

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (Pressman ,2001).

Esta herramienta permite acrecentar la calidad del software, a través de la mejora de la productividad en su desarrollo y mantenimiento. 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.

Se decide utilizar la herramienta Visual Paradigm, para facilitar el uso del lenguaje de modelado UML, aprovechando las facilidades de uso y la amplia gama de utilidades que brinda. Esta herramienta permite generar todos los artefactos de la metodología seleccionada, cumpliendo con los estándares establecidos. Otra de las características por la que se resuelve usar Visual Paradigm es su disponibilidad en múltiples plataformas, ya que no obliga al usuario a desarrollar solo en el sistema operativo Windows, sino que está disponible en sistemas operativos como Windows, Linux, Unix. Además Visual Paradigm es una herramienta que permite integrarse con otras, sus modelos son editables y tiene soporte completo para los diagramas .Además de todo lo anteriormente expuesto, es la herramienta que el desarrollador domina.

1.4.4 Lenguaje de programación

Un **lenguaje de programación** es un lenguaje artificial que se utiliza para expresar programas de ordenador. Está formado por un conjunto de símbolos, palabras claves utilizables y por reglas gramaticales para construir sentencias sintáctica y semánticamente correctas (Sala, 2003).

Para la selección y fundamentación del lenguaje de programación, es necesario definir el paradigma de programación, pues están muy relacionados. Para el desarrollo de esta investigación se optó por el uso del paradigma Orientado a Objetos, teniendo en cuenta las facilidades que brinda para representar ingenierilmente el modelo del mundo real.

El paradigma orientado a objetos (OO) se refiere a un estilo de programación. La Programación Orientada a Objetos (POO) es una manera de diseñar y desarrollar software que trata de imitar la realidad tomando algunos conceptos esenciales de ella; es decir modelar la realidad del problema a través de entidades independientes pero que interactúan entre sí y cuyas fronteras no permanezcan determinadas por su instrumentación computacional sino por la naturaleza del problema. Estas entidades serán denominadas objetos por analogía con el concepto de objeto en el mundo real (Katrib, 2003).

La POO es una extensión de los lenguajes de alto nivel estructurados que trata de representar de una forma más sencilla el modelo del mundo real. La POO intenta resolver principalmente los problemas de la Ingeniería del Software como: portabilidad, reusabilidad, mantenibilidad, entre otros. Para ello se base en características claves como el encapsulamiento, la herencia, el polimorfismo, y el desarrollo orientado primero hacia el qué, y luego hacia el cómo (interfaces) (Meyer et al., 1999).

Dentro de los lenguajes de programación se decidió por el uso de un lenguaje interpretado, de forma tal que favorezca la portabilidad del sistema, así como la simplicidad en la programación que favorece la implementación de los algoritmos de planificación. De estos lenguajes los más utilizados son Java y C#. Se decide utilizar Java atendiendo a que es el lenguaje en el que el desarrollador tiene mayor experiencia. La tecnología Java está compuesta básicamente por dos elementos: el lenguaje Java y su plataforma (plataforma se refriere a la máquina virtual de Java).

Lenguaje de programación Java

Java es un lenguaje de programación orientado a objetos, y actualmente es uno de los lenguajes más usados para la programación en todo el mundo. Es un lenguaje de alto nivel, que tiene la finalidad de obtener un producto de pequeñas dimensiones, simple y portátil sobre diferentes plataformas y sistemas operativos. Es un lenguaje de desarrollo de propósito general, y como tal es válido para realizar todo tipo de aplicaciones profesionales.

La compañía Sun[4] describe el lenguaje Java como “simple, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico” (Javier García de Jalón et al. 2000). Todas estas características son importantes, sin embargo cabe destacar tres, que son las que han proporcionado tanto interés por el lenguaje: la portabilidad, el hecho de que

[4] Sun Microsystems: compañía que desarrolló el lenguaje de programación Java a principio de los años 90´s y liberó la primera versión en mayo de 1995

sea de arquitectura neutral y su simplicidad. Java ofrece toda la funcionalidad de los lenguajes potentes, pero sin las características menos usadas y más confusas de éstos.

- **Arquitectura neutral:** Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos.
- **Portabilidad:** Al ser de arquitectura neutral es altamente portable, es decir puede ser utilizado en cualquier computadora sin necesidad de ser instalado.
- **Simplicidad:** Basado en el lenguaje C++ pero donde se eliminan muchas de las características POO que se utilizan esporádicamente y que ocasionaban problemas frecuentes entre los programadores. La eliminación de causas de error y problemas de mantenimiento facilita y reduce el coste del desarrollo de software.

Es seleccionado el lenguaje Java para el desarrollo de la solución ya que se ajusta con la descripción del sistema que se quiere desarrollar y además se tiene experiencia en su uso. Además como quedo evidenciado, este lenguaje tiene la finalidad de obtener un producto de pequeñas dimensiones, simple y portátil sobre diferentes plataformas y sistemas operativos.

1.4.5 Máquina virtual de Java 8.0

La Máquina Virtual de Java (JVM, por sus siglas en inglés) constituye un elemento imprescindible para la ejecución de un software desarrollado en esta tecnología. Es una aplicación que interpreta y ejecuta programas escritos en el lenguaje de programación Java. Específicamente puede interpretar el bytecode[5] generado al compilar en Java. Lo que hace la JVM es terminar de compilar el bytecode en lenguaje máquina para que la aplicación Java pueda ser ejecutada en un dispositivo específico, este es el caso de las JVM que utilizan un compilador JIT (Just In Time).

Múltiples sitios web y aplicaciones son programados en Java y debe utilizarse una máquina virtual Java para poder ejecutarse, por lo tanto una computadora (o dispositivo electrónico) debe tenerla instalada para poder ejecutarlos. La JVM no es un hardware sino un software que se instala en la estación de trabajo o donde se quiere ejecutar una aplicación implementada en este lenguaje (Rojas, 2003)

1.4.6 Marcos de trabajo de Java

Los marcos de trabajo simplifican el desarrollo mediante la automatización de las tareas comunes, asimismo proporcionan estructura al código fuente, forzando al programador a crear código más legible y fácil de mantener. Es por ello que el uso de un marco de trabajo se hace indispensable en el desarrollo de software.

Jorge Naula plantea que, un marco de trabajo (en inglés framework), es: "… una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación" (Gutiérrez ,2006).

[5] Bytecode: es un código intermedio más abstracto que el código máquina.

Un marco de trabajo, en el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio (Ramos Fernández, 2009).

Entre los objetivos principales de un marco de trabajo se encuentran: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo, como el uso de patrones. Los marcos de trabajo utilizados para el desarrollo de la solución son:

- **Swing**: Es el marco de trabajo Java que permite crear aplicaciones de escritorio a partir de componentes comúnmente utilizados en ellas (Java, 2013). Es un conjunto de clases de Java que simplifica la construcción de aplicaciones de escritorio. Permite tener un sistema de ventanas y componentes gráficos, independiente del sistema operativo y la biblioteca de dibujo que se tenga disponible en las estaciones de trabajo clientes. En esta investigación se utiliza para el diseño y construcción de los formularios de la aplicación, debido a las ventajas que ofrece para el desarrollo de componentes que permiten la visualización y registro de los datos.
- **JPA**: Es la Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés) de persistencia desarrollada para la plataforma Java Enterprise Edition. Es un nuevo estándar de persistencia Java basado en el mapeo objeto-relacional y la utilización de características orientadas a objetos de Java. Brinda un modelo de persistencia basado en objetos comunes conocidos como POJO´s (Plain Old Java Object) para mapear las bases de datos en Java. El mapeo o relación entre entidades Java y tablas de la base de datos se realiza mediante anotaciones en las propias clases de entidad. (Yang, 2010).

1.4.7 Entorno de desarrollo integrado

Un Entorno de Desarrollo Integrado, traducido del inglés Integrated Development Environment (IDE), es un programa informático compuesto por un conjunto de herramientas de programación. Puede denominarse como un entorno de programación que ha sido tratado como un programa aplicación y que permite a los programadores escribir, compilar, depurar y ejecutar programas (Castro, Blandón y Tabares, 2009). Existen diversos IDE libres para Java entre los que destacan Eclipse, y Netbeans. Teniendo en cuenta la experiencia del equipo de desarrollo y que no existen diferencias significativas entre ellos, se decide utilizar para el desarrollo de la propuesta de solución, Netbeans.

Netbeans 8.0: Es un entorno de desarrollo gratuito y de código abierto, que permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Da soporte entre otras a la tecnología Java. Además puede instalarse en varios sistemas operativos: Windows, Linux, Mac OS, y otros.

NetBeans es también una plataforma de ejecución de aplicaciones, es decir, facilita la escritura de aplicaciones Java, proporcionando una serie de servicios comunes, que a su vez están disponibles a través del IDE. Simplifica la gestión de los proyectos con el uso de diferentes vistas, asistentes de ayuda y estructurando la visualización de manera ordenada, lo que ayuda en el trabajo diario Sus principales características son («NetBeans IDE - Java», 2015):

- Contiene editores, analizadores de código, y convertidores, los cuales hacen que se puedan actualizar las aplicaciones a utilizar.
- Los datos pueden ser visualizados desde diferentes vistas, pues permite la visión de múltiples ventanas de diferentes proyectos que hayan sido cargados y abiertos en la aplicación.
- Proporciona editores y herramientas integrales para los marcos de trabajo y las tecnologías relacionadas.

1.6.6 Sistema Gestor de Bases de Datos

El software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema de gestión de bases de datos (SGBD). El objetivo fundamental de un SGBD consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado (Mato García, 1999).

Entre los SGBD más utilizados se encuentran Oracle, PostgreSQL, MySQL, y SQLite. Al realizar un análisis de estas herramientas se decide utilizar SQLite porque se ajusta a las características del sistema de ser ligero, portable, de un volumen de datos discreto, de acceso restringido de personas y sencillo de uso. A diferencia de la mayoría de las otras bases de datos SQL, SQLite no tiene un proceso servidor independiente. SQLite lee y escribe directamente en archivos de disco ordinarios, además es simple y fácil de usar.

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad, en español). Es una biblioteca de software que implementa un sistema autónomo, sin servidor, sin necesidad de configuración, el motor transaccional de la base de datos SQL. Es el motor de base de datos SQL más utilizado en el mundo. El código fuente de SQLite es de dominio público.

Características de SQLite («SQLite Home Page», 2015):

- Transacciones son atómicas, coherentes, aisladas y durables (ACID), incluso después de caídas del sistema y fallas de energía.
- Sin configuración (ninguna configuración o administración es necesaria).

- Más rápido que otros motores populares de base de datos cliente / servidor para la mayoría de las operaciones más comunes.

Es seleccionado SQLite como sistema gestor de bases de datos para el desarrollo de la solución ya que se ajusta con la descripción del sistema que se quiere desarrollar y además es fácil de utilizar. Además como quedó evidenciado su utilización garantizará la portabilidad de la base de datos, que es una de las principales características que se desea contenga la aplicación.

1.6.7 Herramienta para la administración de Bases de Datos

Navicat Premium 11.0.6, es un Administrador de Base de datos rápido, fiable y asequible. Especialmente diseñado para simplificar la gestión de bases de datos y reducir los costes de administración. Ofrece una intuitiva y potente interfaz gráfica de gran alcance para la gestión, el desarrollo y el mantenimiento de las bases de datos. Navicat proporciona una manera más fácil de gestionar, diseñar y manipular datos en MySQL, MariaDB, SQL Server, SQLite, Oracle y PostgreSQL. Permite transferir datos de forma fácil y rápida a través de diferentes sistemas de bases de datos, o en un archivo de texto plano con el formato y codificación SQL designado. Las características de Navicat son lo suficientemente sofisticadas para ofrecer a los desarrolladores profesionales para todas sus necesidades específicas, pero fácil de aprender para los usuarios que son nuevos en el servidor de base de datos (Navicat Premium, 2015).

Es seleccionada esta herramienta para la administración de la base de datos ya que las características que posee se ajustan con la descripción del sistema que se quiere desarrollar y es fácil de utilizar. Además soporta todos los tipos de objetos de SQLite, que es el sistema gestor de bases de datos que se utiliza en la solución, y es una herramienta que se puede ejecutar sobre varios sistemas operativos como Windows y Linux.

1.5 Conclusiones del capítulo

La exposición de conceptos relacionados a la planificación de horarios docentes, permitió Una mejor comprensión de las bases teóricas de dicho proceso.

Como parte del diagnóstico del estado actual de la planificación docente de la facultad, se aplicó una encuesta a 16 trabajadores con alta experiencia en la planificación docente. Esta arrojó resultados que permitieron identificar las limitaciones de dicho proceso con el uso de la herramienta Horrplanner, y de esta forma confirmar la necesidad de una nueva herramienta.

El estudio del arte realizado posibilitó obtener las tendencias actuales de las aplicaciones y sistemas de planificación a nivel nacional e internacional. A partir del análisis realizado se pudo concluir que no existe disponibilidad de un sistema que dé respuesta al problema de la investigación por lo que se hace necesario el desarrollo de un nuevo sistema. De forma adicional tributó al enriquecimiento de las funcionalidades.

Considerando que el equipo de desarrollo es pequeño, que se cuenta con poco tiempo para entregar la propuesta de solución. Además teniendo en cuenta que el sistema que se desea obtener es un software de pequeñas dimensiones, portable, con un bajo nivel de acceso de los usuarios y que la dimensión de los datos que se manejan es conocida y limitada, se seleccionaron como metodología de desarrollo SXP y como lenguaje de programación Java. Además se escogieron las técnicas y herramientas a utilizar, quedando plasmado un análisis detallado de estas herramientas y tecnologías para lograr una mejor comprensión de cada una de ellas.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA PARA LA PLANIFICACIÓN Y CONTROL DEL HORARIO DOCENTE DE LA FACULTAD 3

En este capítulo se realiza una descripción de la solución propuesta a través de los requisitos funcionales y no funcionales, así como las particularidades del diseño. Se muestran aspectos esenciales de la arquitectura del sistema y los patrones de diseño empleados. Además se describen los artefactos para el análisis y diseño que se generan durante el flujo de trabajo.

2.1 Propuesta de solución

En vista de los objetivos que persigue el presente trabajo de diploma se propone como solución desarrollar una aplicación de escritorio que permita mejorar la gestión de información referente a la planificación y control del horario docente de la Facultad 3 de la UCI; para ello la solución propuesta debe ser capaz de:

- Generar una planificación automática del horario docente.
- Permitir que una vez generado un horario, se le puedan aplicar los ajustes manuales que sean necesarios.
- Generar reportes informativos, que permitan mantener un control del cumplimiento por parte de estudiantes y profesores del horario establecido, como ayuda a una posterior toma de decisiones.

Concepción inicial del sistema

En la primera fase de la metodología SXP se genera la plantilla Concepción del Sistema. En esta plantilla se refleja la visión general del producto a implementar, recoge los diferentes roles que intervendrán en el desarrollo del software, así como las tecnologías usadas. Este artefacto forma parte de la documentación del sistema entregada al cliente.

2.2 Personas relacionadas con el sistema

Las personas relacionadas con el sistema son aquellas que interactúan con el mismo y obtienen resultados de los procesos desarrollados en la aplicación. Pueden inferirse luego de una correcta lectura de las HU. A continuación se relacionan las personas vinculadas con el sistema propuesto:

- **Actores del negocio**

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados (ver Tabla 4).

Tabla 4: Actores del negocio.

Actor	Descripción
Directivo (vicedecano de Formación)	Persona que obtiene o se beneficia con los resultados de una eficiente planificación de los procesos de planificación docente.

- **Trabajadores del negocio**

Un trabajador de negocio es una abstracción de una persona (o grupo de personas), una máquina o un sistema automatizado, que actúa en el negocio realizando una o varias actividades, interactuando con otros trabajadores del negocio y manipulando entidades del negocio. Representa un rol (ver Tabla 5).

Tabla 5: Trabajadores del negocio.

Trabajadores	Descripción
Planificador	Es el encargado de recopilar toda la información necesaria para confeccionar el horario, realizar la planificación del mismo y entregarlo a la vicedecana de formación.
Administrador	Es la persona encargada de llevar a cabo la administración de todo el contenido dentro de la aplicación. Gestiona todo el sistema.
Profesor Guardia Docente	Ejecuta la guardia docente emitiendo un reporte con los resultados del día a la planificadora.

2.3 Requisitos del software

Los requisitos son la descripción de los servicios y restricciones de un sistema de software. Especifican lo que el software debe hacer (sus funciones), sus propiedades esenciales y deseables (Sommerville 2005).

2.3.1 Técnicas de captura de requisitos

La ingeniería de requisitos es el conjunto de actividades implicadas en descubrir, documentar y mantener un conjunto de requisitos del software a desarrollar. La captura de los mismos es un proceso en el cual los datos son extraídos a partir de la aplicación de técnicas de captura de información (Romero, 2011).

Para la captura de requisitos se utilizan técnicas como las entrevistas, las reuniones con el cliente, los cuestionarios, la lluvia de ideas, el análisis de sistemas existentes, la arqueología de documentos y los prototipos. Para el levantamiento de requisitos del sistema se aplicaron las técnicas de análisis de sistemas existentes, entrevistas, reuniones y lluvia de ideas.

- **Análisis de sistemas existentes**: se estudiaron un total de nueve sistemas relacionados con la planificación y control de actividades docentes, con el objetivo de identificar funcionalidades que se pudieran adaptar al software que se quiere desarrollar. Entre ellas se encuentran :
  - Generar una planificación automática del horario docente.
  - Permitir realizar modificaciones al horario una vez realizada la planificación.
  - Generar reportes sobre el control de la realización de las actividades planificadas.
- **Entrevistas**: se realizaron entrevistas al cliente con el objetivo de identificar la mayor cantidad de funcionalidades posibles que dieran soporte al sistema que se quiere implementar. En las entrevistas realzadas a la planificadora (persona encargada de realizar la planificación docente), se obtuvo información referente a cómo se realiza la planificación docente, identificándose algunos requisitos funcionales y no funcionales. En el caso de los requisitos no

funcionales, se identificaron requisitos de apariencia o interfaz, ya que la planificadora planteó la posibilidad de que la aplicación tuviera una interfaz agradable.

- **Reuniones**: fueron realizados cuatro debates por parte de los involucrados con el fin de llegar a un consenso sobre las funcionalidades con las que tendría que contar el sistema a desarrollar. En un primer encuentro el equipo de desarrollo presentó al cliente una primera propuesta de las funcionalidades del sistema, de las cuales se realizaron modificaciones a doce de ellas, luego en los encuentros siguientes, se debatieron detalles del desarrollo de la aplicación.

2.3.2 Requisitos funcionales

Los requerimientos funcionales de un sistema describen las funcionalidades o los servicios que se espera que ofrezca a los usuarios finales .Estos dependen del tipo de software, del sistema que se desarrolle y de los posibles usuarios del software (Pressman, 2001). Como resultado de la aplicación de las técnicas de captura de información se obtuvo un total de 52 requisitos funcionales agrupados en 14 HU.

Los requisitos funcionales del sistema están expresados en lenguaje natural. Los mismos serán identificados con las siglas RF más el número del requisito Ej. RF#1 y el requisito. Los requisitos funcionales definidos por HU son los siguientes:

RF#1: Autenticar usuario

RF#2: Insertar usuario

RF#3: Eliminar usuario

RF#4: Modificar usuario

RF#5: Buscar usuario

RF#6: Listar usuario

RF#7: Insertar profesor

RF#8: Eliminar profesor

RF#9: Modificar profesor

RF#10: Buscar profesor

RF#11: Listar profesor

RF#12: Insertar brigada

RF#13: Eliminar brigada

RF#14: Modificar brigada

RF#15: Buscar brigada

RF#16: Listar brigada

RF#17: Insertar grupo

RF#18: Eliminar grupo

RF#19: Modificar grupo

RF#20: Buscar grupo

RF#21: Listar grupo

RF#22: Insertar local

RF#23: Eliminar local

RF#24: Modificar local

RF#25: Buscar local

RF#26: Listar local

RF#27: Insertar año

RF#28: Eliminar año

RF#29: Modificar año

RF#30: Buscar año

RF#31: Listar año

RF#32: Insertar asignatura

RF#33: Eliminar asignatura

RF#34: Modificar asignatura

RF#35: Buscar asignatura

RF#36: Listar asignatura

RF#37: Insertar turno

RF#38: Eliminar turno

RF#39: Modificar turno

RF#40: Buscar turno

RF#41: Mostrar horario docente

RF#42: Insertar departamento docente

RF#43: Eliminar departamento docente

RF#44: Modificar departamento docente

RF#45: Buscar departamento docente

RF#46: Listar departamento docente

RF#47: Insertar afectación

RF#48: Eliminar afectación

RF#49: Modificar afectación

RF#50: Buscar afectación

RF#51: Listar afectación

RF#52: Insertar incidencia

RF#53: Eliminar incidencia

RF#54: Modificar incidencia

RF#55: Mostrar reporte de incidencias

RF#56: Generar planificación docente

RF#57: Mostrar locales disponibles

2.3.3 Requisitos no funcionales

Los requisitos no funcionales (RNF) se refieren a las características o cualidades que debe tener el producto para hacerlo atractivo, usable, rápido y confiable. Un requisito no funcional es una característica ya sea del sistema, del proyecto o del servicio de soporte, que es requerida junto a la especificación del sistema pero que no se satisface añadiendo código, sino cumpliendo con esta como si de una restricción se tratara (Pressman, 2001). Para el desarrollo de la aplicación de escritorio propuesta fueron definidos ocho RNF, los que se clasificaron en requisitos no funcionales de apariencia o interfaz, confidencialidad, integridad, soporte, hardware.

Los requisitos no funcionales del sistema están expresados en lenguaje natural. Los mismos serán identificados con las siglas RNF más el número del requisito (Ej. RNF#1) y el requisito. Los requerimientos no funcionales definidos son los siguientes:

RNF#1: El sistema presentará una interfaz legible, simple de usar e interactiva.

RNF#2: La tipografía y colores serán estándares en toda la aplicación.

RNF#3: Debe estar bien definida la seguridad. Se ha establecido un sistema de roles para usuarios diferentes como la planificadora y la vicedecana de formación. Cada uno de ellos tendrá acceso sólo a ejecutar las acciones que les correspondan.

RNF#4: En caso de que falle el sistema, no se pierden los datos almacenados ya que no afecta al Gestor de Base de Datos.

RNF#5: Las excepciones que se produzcan con la utilización del sistema se mostrarán a través de mensajes.

RNF#6: El sistema debe responder las peticiones del usuario en menos de tres segundos, excepto durante la generación del horario que puede demorar hasta más de 30 minutos.

RNF#7: Solo tendrá acceso a las modificaciones o mantenimiento del sistema el informático del Departamento.

RNF#8: Se necesita una computadora con más de 512 MB de RAM, de sistema operativo Windows superior a 7, Ubuntu superior a 12.10.

2.3.4 Análisis de complejidad de los requisitos funcionales del software.

La Ingeniería de Requisitos en la actualidad cumple un papel primordial en el proceso de producción de software. Una de las tareas más importantes que realiza la misma, es la actividad de determinar la complejidad de los requisitos funcionales de software. Para ello se hace necesario contar con elementos que permitan realizar valoraciones lo más certeras posibles, que apoyen la planeación, la definición de recursos necesarios y la elaboración de cronogramas. Estos constituyen algunos de los principales mecanismos de control con los que se cuenta durante el desarrollo de un producto (Mustelier, 2013).

La complejidad del software se puede definir como el grado en que se puede entender y verificar el diseño de un sistema o componente (Kushwaha y Misra, 2006). La complejidad de un requisito funcional se refiere a una característica que describe la dificultad de diseño e implementación dentro del proceso de desarrollo de software, y se puede clasificar en alta, media y baja. Es importante para el desarrollo del software clasificar los requisitos según su complejidad, dado que permite brindar un mejor tratamiento de los mismos.

La complejidad de un requisito se puede ver afectada por un conjunto de variables que, dependiendo de la lógica del negocio, varían en cuanto al grado de afectación sobre éste (Mendoza et al., 2005). En esta investigación, para determinar la complejidad de cada requisito funcional se tuvo en cuenta una serie de criterios ya determinados en la ingeniería de software, los cuales se muestran en la siguiente tabla (ver Tabla 6):

Tabla 6: Variables que definen la complejidad en los requisitos funcionales.

Criterios de Complejidad	Descripción
Complejidad por Interfaces	El criterio interfaz se aplica a requisitos que presenten algún tipo de complejidad en su interacción con los siguientes elementos: • Humanas (formularios, informes) • Equipo (Tomógrafos, Rayos X); Ej. API, drivers • Programación (Programas externos necesarios para apoyar el producto); Ej. Adicionar un nuevo usuario, usa el LDAP) • Comunicación (Protocolos de comunicación que serán utilizados); Ej. HL7
Diferentes comportamientos	Un mismo requisito se comporta de manera diferente ante determinadas situaciones.
Formas de inicialización.	Un mismo requisito puede ser inicializado de diferentes formas.
Consultas a fuentes de almacenamientos	Los requisitos pueden presentar diversidad en la cantidad y complejidad de la interacción con la fuente de datos. • Base de Datos • Ficheros • Otro
Restricciones de validación	Complejidad de todas las validaciones que lleve un requisito.
Grado de reutilización	Complejidad de un requisito, para poder ser reutilizado por otros.
Lógica de negocio	Los requisitos pueden presentar diferentes niveles de complejidad para la implementación de la lógica de negocio que contienen; Ej. operaciones, métodos matemáticos, CRUD, etc.

Para la determinación de la complejidad de los requisitos se analizan individualmente los criterios que se recogen en la tabla (ver Tabla 6), obteniendo como resultado la clasificación del requisito en complejidad Alta, Media o Baja. La clasificación de la complejidad permite estimar el esfuerzo de implementación del requisito y contribuye a la decisión sobre la inclusión en las etapas de desarrollo del software.

Los indicadores fueron evaluados haciendo uso de una herramienta en Excel desarrollada por el programa de mejoras de CALISOFT, y se estableció una valoración final que corresponde a la complejidad de cada requisito funcional. Los resultados obtenidos fueron apropiados ya que como se muestra en el gráfico de la siguiente figura (ver Ilustración 2), el 93% de los requisitos funcionales

están dentro de las categorías media o baja, lo que demuestra que el proceso de desarrollo del sistema no será complejo.

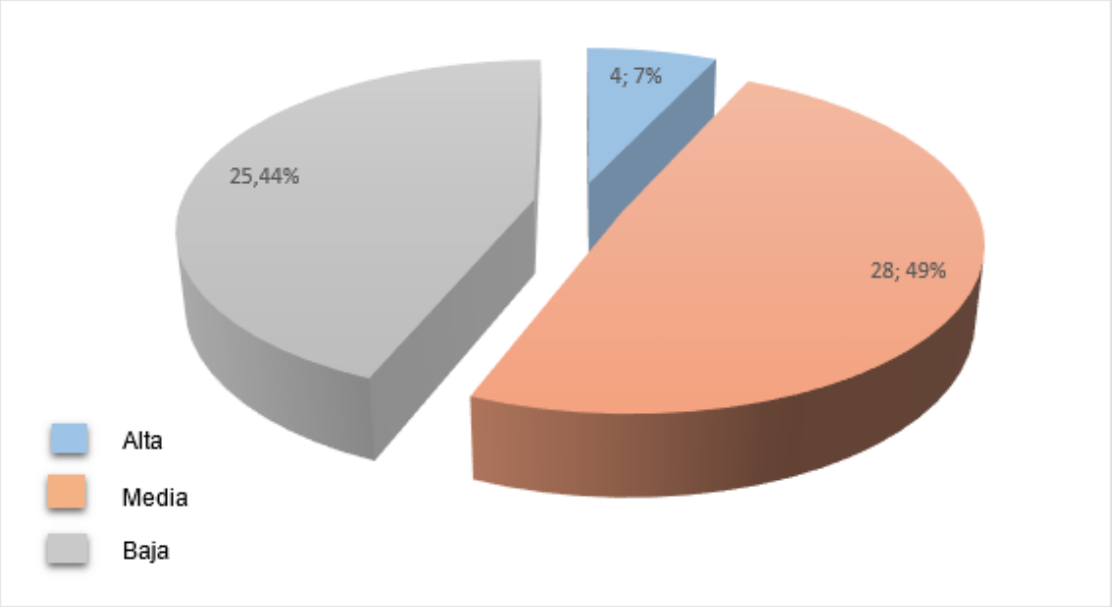

Ilustración 2: Resultado de la clasificación por complejidad de los requisitos del software.

2.3.5 Lista de Reserva del Producto (LRP)

Una de las actividades más importantes en la metodología SXP es la Lista de Reserva del Producto (LRP). La misma consta de una lista de prioridad de los requisitos funcionales y no funcionales del sistema que se va a desarrollar, el ítem de prioridad, la descripción de cada requisito, la estimación en días y por quién fue estimado. La prioridad se puede categorizar en Muy Alta, Alta, Medio y Baja.

Esta lista puede modificarse mientras avanza el desarrollo del producto. Al final del ciclo de vida del producto, la LRP debe haber asegurado que el mismo sea útil, tenga la calidad requerida a raíz de su funcionamiento, sea fácil de manipular y satisfaga la necesidad del cliente. Este artefacto forma parte de la documentación del sistema entregada al cliente. A continuación se muestra un fragmento de la LRP para lograr una mejor comprensión de la misma (ver Tabla 7).

Tabla 7: Fragmento de la LRP. Prioridad Muy Alta.

Prioridad				
Ítems *	**Requisito**	**Descripción**	**Estimación (días)**	**Estimado por:**
Requisitos Funcionales				
Prioridad Muy Alta				
RF# 1	Autenticar usuario.	Permite autenticar un usuario en la aplicación con su usuario y contraseña en el dominio UCI.	1	Programador
RF#2	Insertar usuario	Permite insertar un usuario en la aplicación con los permisos o acceso	1	Programador

		sólo a las funcionalidades que este puede ejercer.		
RF#3	Eliminar usuario	Permite eliminar un usuario de la aplicación, quitándole los permisos que tenía otorgados.	1	Programador
RF#4	Modificar usuario	Permite modificar un usuario de la aplicación con los permisos o acceso a las funcionalidades que este puede ejercer.	1	Programador
RF#41	Mostrar horario docente	Permite mostrar el horario docente, según la planificación realizada, por un criterio determinado	1	Programador
RF#52	Insertar incidencia	Permite insertar los datos de una incidencia en la aplicación.	1	Programador
RF#56	Generar planificación docente	Permite generar la planificación del horario docente	20	Programador
RF#57	Mostrar locales disponibles	Permite mostrar los locales disponibles en un rango de tiempo determinado.	1	Programador
Prioridad Alta				
RF#5	Buscar usuario	Permite buscar un usuario de la aplicación.	1	Programador
RF#6	Listar usuario	Permite listar todos los usuarios de la aplicación, mostrando los roles correspondientes a cada uno de ellos.	1	Programador
RF#7	Insertar profesor	Permitir registrar los datos de los profesores en el sistema.	1	Programador
RF#8	Eliminar profesor	Permite eliminar un profesor existente, de la base de datos modificando a su vez la matriz de afectaciones del profesor.	1	Programador
RF#9	Modificar profesor	Permite modificar los datos del profesor en la base datos.	2	Programador
RF#10	Buscar profesor	Especificando la semana, permite conocer en qué turno y con qué grupo tiene clases un profesor determinado.	1	Programador
RF#11	Listar profesor	Permite mostrar un listado de los profesores, realizando un filtrado según diferentes criterios.	1	Programador

2.4 Diseño de metáforas

Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. La práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda al diseño y métodos del sistema (Romero, 2011). La metáfora definida para el sistema a desarrollar es: la implementación de una aplicación para gestionar la planificación y control de las actividades docentes en la Facultad 3.

A partir de la definición de las funcionalidades descritas en la LRP es posible establecer el Diseño de Metáforas, mediante las HU, prototipos del sistema y las tareas ingenieriles que permiten su desarrollo siendo estas las actividades que se realizan y se describen a continuación.

2.4.1 Historias de usuarios del negocio

Las historias de usuarios (HU) es el artefacto que utiliza la metodología SXP para especificar los requisitos del software y constituyen la base para las pruebas funcionales. En la plantilla del modelo de HU del negocio se describen los actores y trabajadores del negocio, además se presenta un diagrama de HU del negocio que permite ver la relación entre los usuarios y las actividades que se realizan.

Las HU son equivalentes a los casos de uso en el proceso unificado. Básicamente una historia de usuario es una lista priorizada de requisitos o funcionalidades, descritas usando la terminología del cliente. En la siguiente tabla se puede apreciar el listado de las mismas (ver Tabla 8).

Tabla 8: Historias de Usuario.

Historias de usuario	Sprint[6]
HU1_Autenticar usuario	1
HU2_Gestionar usuario	1
HU3_Gestionar profesor	1
HU4_Gestionar brigada	1
HU5_Gestionar grupo	1
HU6_Gestionar local	1
HU7_Gestionar año	1
HU8_Gestionar asignatura	1
HU9_Gestionar turno	1
HU10_Gestionar departamento docente	1
HU11_Gestionar afectación	2
HU12_Gestionar guardia docente	3
HU13_Generar planificación docente	3
HU14_Mostrar locales disponibles	2

A continuación se ilustra la historia de usuario HU_13: Generar planificación (ver Tabla 9). Este artefacto forma parte de la documentación del sistema entregada al cliente.

[6] Sprint: Terminología para definir las iteraciones en la metodología SXP.

Tabla 9: Historia de Usuario 13: Generar planificación.

Historia de Usuario	
Número: HU_13	**Nombre Historia de Usuario**: Generar planificación docente
Modificación de Historia de Usuario Número: 1	
Usuario: Isis Dalia García Rojas	**Iteración Asignada**: Sprint 3.
Prioridad en Negocio: Muy Alto	**Puntos Estimados**: 15 días
Riesgo del Desarrollo: Alto	**Puntos Reales**: 20 días
Descripción: Genera el horario docente para todas las asignaturas de un semestre. El horario generado debe cumplir con las restricciones establecidas a nivel institucional, formando los encuentros entre grupos, profesores, asignaturas y locales.	
Observaciones: El usuario debe haber introducido los datos a la aplicación, que incluye los grupos, los profesores, los locales, el semestre, las asignaturas y las afectaciones.	
• RF#56	

2.4.2 Tareas de la Ingeniería

Las TI es uno de los artefactos que genera la metodología SXP, en el que se recogen las tareas por HU a realizar. Luego de la descripción de las HU del sistema, se hace necesaria la definición de un grupo de tareas ingenieriles dirigidas a dar cumplimiento a la implementación de las funcionalidades definidas en ellas, lo cual sienta las bases para el posterior desarrollo de la solución. Por ejemplo, las tareas relacionadas con la HU_13: Generar planificación, son las siguientes (ver Tablas 10 y 11):

Tabla 10: T_26. Diseñar las interfaces requeridas para la funcionalidad: Generar planificación.

Tareas de la Ingeniería	
Número Tarea: T_26	**Número Historia de Usuario**: HU_13.
Nombre Tarea: Diseñar las interfaces requeridas para la funcionalidad: Generar planificación docente	
Tipo Tarea: Desarrollo.	**Puntos Estimados**: 3 días
Fecha Inicio: 10-6-2015	**Fecha Fin**: 12-6-2015
Programador Responsable: Isis Dalia García Rojas	
Descripción Se diseñan las interfaces necesarias para la funcionalidad: Generar planificación.	

Tabla 11: T_27.Implementación de la funcionalidad: Generar planificación

Tareas de la Ingeniería	
Número Tarea: T_27	**Número Historia de Usuario**: HU_13.
Nombre Tarea: Implementación de la funcionalidad: Generar planificación docente	
Tipo Tarea: Desarrollo.	**Puntos Estimados**: 10 días
Fecha Inicio: 5-6-2015	**Fecha Fin**: 25-6-2015
Programador Responsable: Isis Dalia García Rojas	
Descripción: Se crean las clases y algoritmos necesarios para la implementación de la funcionalidad: generar planificación.	

El resto de las tablas que describen las TI están recogidas en el artefacto Plantilla de Tareas de la Ingeniería, que forma parte de la documentación del sistema entregada al usuario.

2.5 Arquitectura

La Arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones (Clements, 1996). Según la IEEE Estándar 1471-2000: “la arquitectura del software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución” (Fernández, 2006).

2.5.1 Estilo arquitectónico en capas

Un estilo arquitectural puede entenderse como un conjunto de principios que definen a alto nivel un aspecto de la aplicación. Un estilo arquitectural está definido por un conjunto de componentes, un conjunto de conexiones entre dichos componentes y un conjunto de restricciones sobre cómo se comunican dos componentes cualesquiera conectados (Llorente et al., 2010).

Estos estilos arquitectónicos constituyen herramientas básicas a la hora de definir la estructura de un sistema. También tienen una serie de directivas para organizar los componentes del mismo con el objetivo de facilitar la tarea del diseño de la aplicación. Indican además cómo se va a dividir en partes y cómo será la interacción entre estas.

Arquitectura en capas

El estilo arquitectural en capas se define como una organización jerárquica, donde cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior (Garlan y Shaw ,1994).

Una característica importante de esta arquitectura es que las funcionalidades de cada una de las capas van a estar separadas de las de otras de manera clara, donde cada una sólo contendrá funcionalidades relacionales con las tareas que le corresponde, proporcionando un bajo acoplamiento. El esquema de n-capas presenta diversas ventajas, entre ellas:

- Facilita la reutilización de código y la estandarización.
- Independencia de la interfaz del cliente y la arquitectura de datos.
- Mejores posibilidades de balancear la carga.

El desarrollo de la solución define una arquitectura basada en cuatro capas: (Capa de Presentación, la Capa de Negocio, la Capa de Acceso a Datos y la Capa de Datos). En la Ilustración 3 se muestra la arquitectura de la aplicación.

- **Capa de presentación** (denominada también "capa de usuario"): contiene los componentes con los que el usuario va a interactuar. Presenta el sistema al usuario, le comunica la información y captura su información, realiza un filtrado previo para comprobar que no hay errores de formato. Esta capa se comunica únicamente con la capa de negocio.
- **Capa de negocio:** es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio), pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.
- **Capa de acceso a datos:** Contiene las entidades y repositorios que mapea JPA como marco de trabajo para la comunicación con el servidor de datos. Gestiona las peticiones de la capa de negocio consultando la base de datos y retornando los datos que se recuperan al gestor correspondiente. Es importante destacar que las entidades del dominio se encuentran en esta capa pero son accesibles además desde las capas superiores.
- **Capa de datos:** es donde residen los datos. En esta capa se encuentra el gestor de base de datos que realiza todo el almacenamiento de datos, recibe solicitudes de almacenamiento o recuperación de información, desde la capa de negocio.
- **Recursos:** Los recursos que se utilizan en el desarrollo de la aplicación son entre otros: una biblioteca para el acceso y la creación de archivos de bases de datos SQLite en Java, las imágenes y otras bibliotecas necesarias para la implementación de la solución.

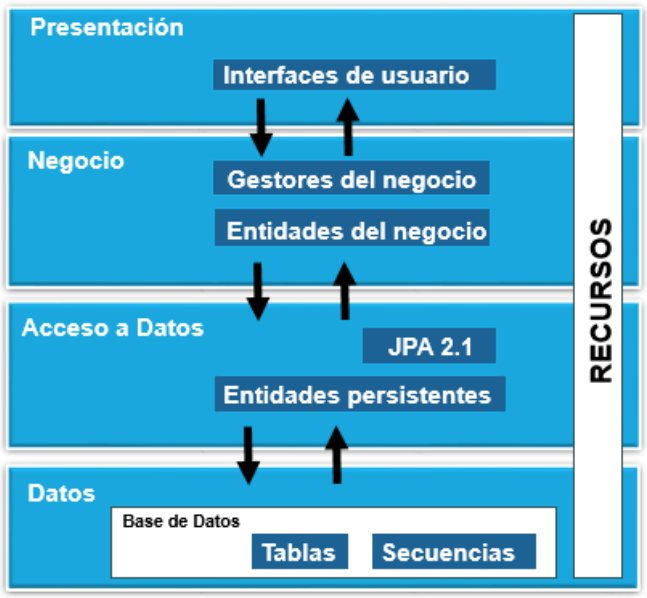

Ilustración 3: Arquitectura en cuatro capas. (Elaboración propia)

Se seleccionó este estilo arquitectónico, además de lo antes argumentado, porque permite una clara separación de las funciones de control de la interfaz y presentación de datos con la lógica de la aplicación. Además se tuvo en cuenta la principal ventaja que presenta y es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.

2.5.2 Patrones de diseño

Los patrones de diseño constituyen la base para la búsqueda de soluciones a problemas comunes en el desarrollo del software, ya que brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. En términos generales, un patrón es una pareja de problema/solución, con un nombre, y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas (Larman, 1999).

Son varios los patrones de diseño que existen y que pueden ser utilizados indistintamente en uno u otro modelo. Sin embargo, es casi imposible realizar un modelo o diseño eficiente sin hacer uso de los patrones generales de asignación de responsabilidades o patrones GRASP, (por sus siglas en inglés General Responsibility Assignment Software Patterns). Estos patrones forman un conjunto de cinco propuestas que se basan en describir los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman, 1999). Seguidamente se exponen los cinco patrones que satisfacen el conjunto de los GRASP.

Patrones de asignación de responsabilidades (GRASP)

- **Experto:** se utiliza con frecuencia en la asignación de responsabilidades. Es un principio de guía básico que se utiliza continuamente en el diseño de objetos. El Experto expresa la "intuición" común de que los objetos hacen las cosas relacionadas con la información que tienen. Este patrón se ve reflejado en las entidades persistentes debido a que cada una de ellas es experta, pues contienen la información necesaria para cumplir con las responsabilidades que le fueron asignadas. A continuación se muestra la clase Asignatura la cual es experta en información referente a una asignatura (ver Ilustración 4).

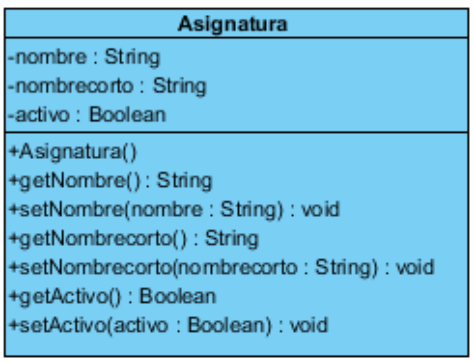

Ilustración 4: Empleo del patrón Experto.

- **Bajo Acoplamiento:** guía la asignación de responsabilidades de manera que el acoplamiento permanezca bajo, lo que significa dar soporte a una escasa dependencia y a

un aumento de la reutilización. La solución que brinda el patrón consiste en asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases posibles. Con la aplicación de la métrica TOC quedará confirmado su utilización en el desarrollo de la aplicación.

- **Alta Cohesión:** guía la asignación de responsabilidades relacionadas con un número relativamente pequeño o manejable. Se basa en cómo mantener la complejidad dentro de límites manejables. Para ello propone asignar a las clases las responsabilidades que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad.
  El patrón se evidencia en cada clase que realiza una labor única dentro del sistema y colabora con las otras para llevar a cabo una tarea, básicamente en las clases que forman parte de la capa de acceso a datos.
- **Controlador:** guía la asignación de responsabilidades relacionadas con la gestión de un evento de entrada, o sea, define quién debería encargarse de atender un evento del sistema. La solución consiste en asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase controladora (Larman, 1999). Este patrón se ve reflejado en todas las clases controladoras, las cuales contienen todos los métodos necesarios para controlar las actividades que se realicen.
- **Creador:** guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica de este patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Es el encargado de crear las instancias de los objetos para cada una de las funcionalidades de la aplicación. El patrón se evidencia en las clases controladoras que, para cada uno de las funcionalidades de la aplicación, son las encargadas de crear las instancias de los objetos que manejan, favoreciendo así la reutilización y el bajo acoplamiento.

Patrón de diseño de bases de datos

- Llaves subrogadas: El uso de este patrón es bastante generalizado en la base de datos pues la mayoría de las entidades contienen una llave única entera autoincremental, lo que permite reducir el costo de las búsquedas en la base de datos. El uso de este patrón constituye una protección ante los cambios, debido a que la lógica del negocio no está en las llaves y evita la contención (bloqueo) de la base de datos, gracias a la rapidez de los mecanismos de generación que provee el sistema (ver Ilustración 5).

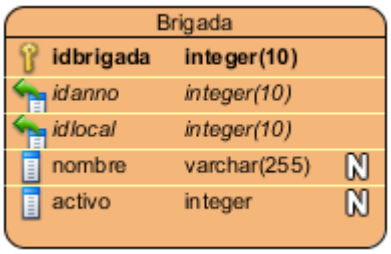

Ilustración 5: Uso del patrón Llaves Subrogadas en la entidad Brigada.

2.6 Modelo de datos

El modelo de datos es un conjunto de conceptos que sirven para describir la estructura, semántica y las relaciones existentes entre las entidades de una base de datos importantes para el funcionamiento del negocio, y muestra los datos que serán contenidos en el sistema (Silberschatz et al., 2002). En la siguiente figura se muestra el modelo de datos de la solución (ver Ilustración 6).

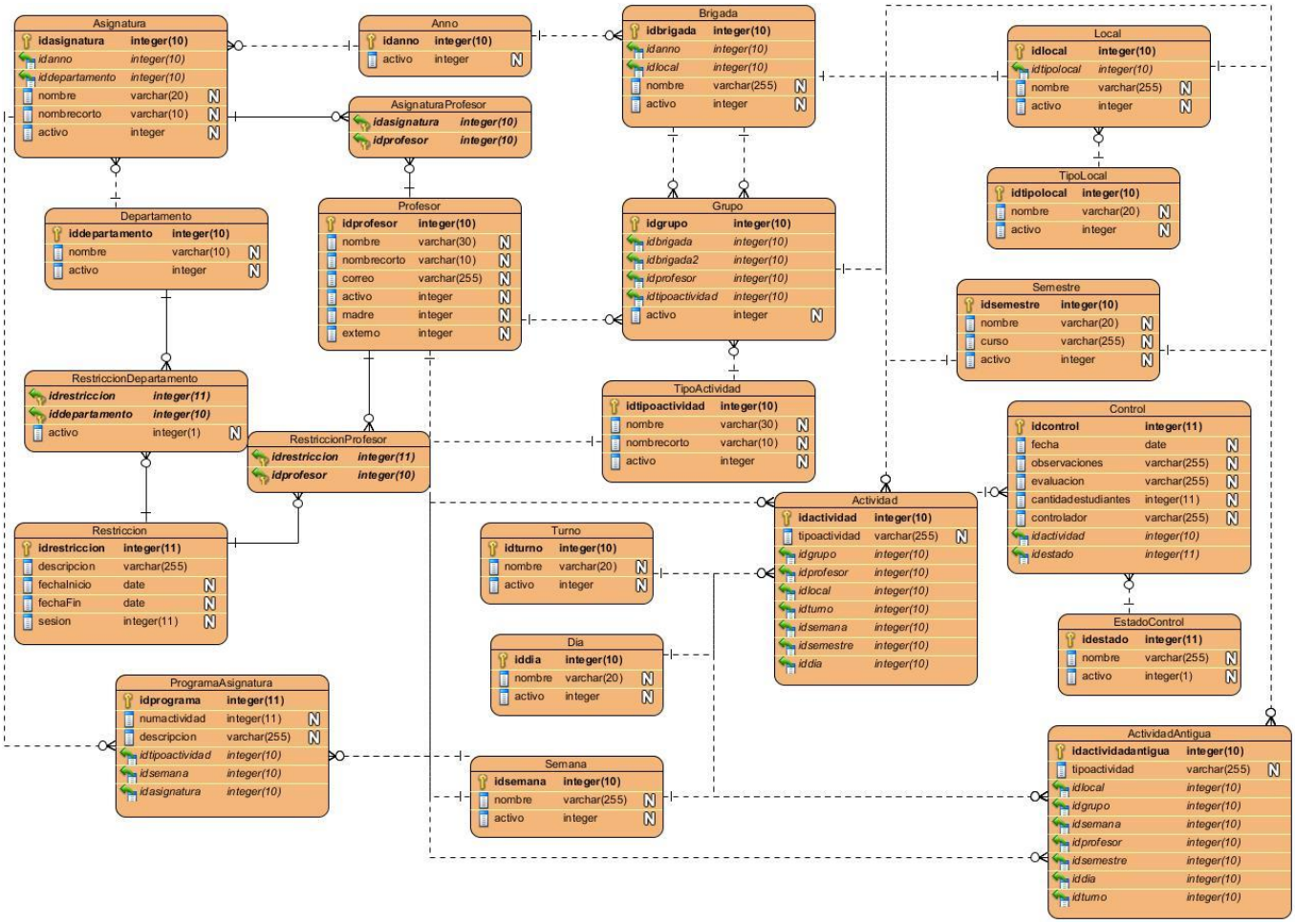

Ilustración 6: Modelo de datos

El modelo de datos obtenido cuenta con un total de 16 tablas donde seis de ellas son nomencladores encargados de gestionar conceptos específicos del negocio ya predefinidos. Los datos que contienen tablas como: Programa Asignatura, Profesor, Asignaturas, Local, Departamento, entre otras. Estas tablas en su mayoría contienen un gran volumen de datos, los que fueron introducidos manualmente por el autor del presente trabajo, con el propósito de hacer posible el funcionamiento de la herramienta propuesta.

2.7 Conclusiones del capítulo

Con el presente, quedaron definidos los elementos fundamentales para la construcción de la aplicación. Como parte de la fase de Planificación-Definición se realizó una descripción detallada de los principales procesos del negocio y se presentó la propuesta de solución que sugiere la investigación, basada en el problema y el estado del arte establecido en el capítulo 1.

El flujo de trabajo generó la plantilla de Concepción Inicial del Sistema, la cual reúne los aspectos más significativos de la aplicación a desarrollar, entre los que se encuentran la misión y visión del sistema, así como las herramientas seleccionadas.

Como parte de la realización del segundo flujo (Captura de Requisitos), se aplicaron técnicas de ingeniería de requisitos para su captura. Fueron definidos un total de 57 RF y 8 RNF, estos últimos fueron descritos y agrupados por categorías. Además, se generó el artefacto LRP, en el cual se agruparon los RF según la estimación de su complejidad por parte del desarrollador.

Se realizó un estudio sobre todas las HU definidas como punto de partida para el diseño y realización de los prototipos del sistema. Lo que se concretó con la elaboración de las Plantillas de HU, que aglutinan su descripción, junto al prototipo y los requisitos a los que responde la HU. A partir de las HU se definieron las TI orientadas a guiar el posterior desarrollo de la solución y sentar las bases para las restantes fases de la metodología.

Luego se realizó un análisis sobre la arquitectura, definiéndose una arquitectura en capas para garantizar la seguridad, el tratamiento de excepciones, entre otros aspectos. Posteriormente se analizó el uso de los patrones de diseño utilizados durante el desarrollo de la aplicación. Por último, para describir la estructura y las relaciones existen entre las entidades de la base de datos se realizó el Modelo de Datos de la aplicación.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA PARA LA PLANIFICACIÓN Y CONTROL DEL HORARIO DOCENTE DE LA FACULTAD 3.

En la realización de este capítulo se describe el proceso de implementación de la aplicación, después de haber definido las funcionalidades requeridas. Se generan los principales artefactos definidos por la metodología, correspondientes a la fase de Desarrollo, en sus dos flujos de trabajo (Implementación y Prueba). Además, se muestran los resultados obtenidos de la aplicación de las métricas y técnicas empleadas para validar los requisitos y el diseño de la aplicación, así como los resultados alcanzados luego de la realización las pruebas de funcionalidad y unitarias.

3.1 Implementación

La implementación constituye una de las fases más importantes del desarrollo de software. En ella se toman como punto de partida los resultados obtenidos en el diseño, implementándose el sistema en términos de componentes como ficheros de código binario, código fuente, scripts y ejecutables. Su importancia reside en que se obtiene como consecuencia un sistema ejecutable, siendo esto uno de los principales objetivos en el desarrollo de software (Sommerville, 2005).

3.1.1 Plan de entrega

Entre los artefactos definidos por las metodologías SXP para el flujo de trabajo Implementación, de la fase de Desarrollo, se encuentra el Plan de entrega; donde se describen cada una de las iteraciones en que se divide la implementación del sistema, especificando en cada caso la duración aproximada y el orden a realizar las HU correspondientes.

El plan de entrega del sistema, cuenta con tres iteraciones para su implementación, donde previo a cada iteración se realiza una reunión de planificación de iteración para determinar en qué funcionalidad del producto trabajará el equipo, teniendo como resultado el Plan de entrega que se muestra a continuación (ver Tabla 12).

Tabla 12: Plan de Entrega

Entrega	Descripción de la iteración	Orden de la HU a implementar	Duración total
Iteración 1	En esta iteración se desarrollarán las HU de mayor importancia en cuanto a la funcionalidad que describen cada una, lo que se ve reflejado en la alta prioridad de negocio	HU1: Autenticar usuario	50 Días
		HU_2: Gestionar usuario	
		HU_3: Gestionar profesor	
		HU_4: Gestionar brigada	
		HU_5: Gestionar grupo	
		HU_6: Gestionar local	
		HU_7: Gestionar año	
		HU_8: Gestionar asignatura	
		HU_9: Gestionar turno.	
		HU_10: Gestionar departamento docente.	

Iteración 2	En esta iteración se desarrollarán aquellas funcionalidades del sistema con menos prioridad, es decir, las HU que presentan prioridad media o baja. Por otra parte en esta iteración se corregirán los errores encontrados en la iteración anterior, obteniéndose una nueva versión del sistema.	HU_11: Gestionar afectación	10 Días
		HU_14: Mostrar locales disponibles	
Iteración 3	En esta iteración se realizarán las HU que se consideren opcionales y que no inciden críticamente en la lógica de la aplicación. Una vez concluida esta iteración el sistema se encontrará completamente concluido y listo para su explotación	HU_13: Generar planificación	25 Días
		HU_12: Gestionar guardia docente	

3.1.2 Estándar de codificación empleado

Los estándares de programación son muy útiles por muchas razones, entre las que destacan (Fine, 2004):

- Facilitan el mantenimiento de una aplicación. Dicho mantenimiento constituye el 80% del coste del ciclo de vida de la aplicación.
- Permite que cualquier programador entienda y pueda mantener la aplicación. En muy raras ocasiones una misma aplicación es mantenida por su autor original.
- Los estándares de programación mejoran la legibilidad del código, al mismo tiempo que permiten entenderlo rápidamente.

Con el objetivo de lograr una estandarización en la programación de la aplicación de escritorio se decide aplicar el estándar de codificación CamelCase, específicamente la variante lowerCamelCase. Este facilitará la lectura, comprensión y mantenimiento del código. A continuación se describen a grandes rasgos las convenciones de nomenclatura. Las convenciones de nomenclatura quedan recogidas en el artefacto Plantilla de Estándar de código, que forma parte de la documentación del sistema que se brinda al cliente en la fase de Entrega para su posterior mantenimiento.

General:

- Se exceptúan el uso de las tildes y la letra ñ, la que será sustituida por nn.
- En todo momento se utilizarán nombres que sean claros, concretos y libres de ambigüedades. Ejemplo: "idAsignatura" y no solamente "id".
- El nombre de todas las variables y métodos comenzarán con letra minúscula y si este está compuesto por varias palabras, todas las palabras internas que lo componen comienzan con mayúscula. Ejemplo: “buscarProfesorAsignatura()”.

Identación:

- El contenido siempre se identará con la tecla “Tab”, nunca utilizando espacios en blanco.

Comentarios en las funciones:

- Todas las funciones deben tener un comentario, antes de su declaración, explicando qué hacen. Ningún programador debería tener que analizar el código de una función para conocer su utilidad. Tanto el nombre de la función, como el comentario que acompañe deben bastar para ello.

Clases:

- El nombre de la clase comenzará con mayúscula si este está compuesto por varias palabras, todas las palabras internas que lo componen comienzan con mayúscula también. Ejemplo: “Actividad”, “TipoActividad”.
- Los nombres de las clases deben ser descriptivos y simples. Usar palabras completas, evitar acrónimos y abreviaturas, a no ser que la abreviatura sea mucho más conocida que el nombre completo, como las abreviaturas URL o HTML.

Nombre de variables:

- Los nombres deben ser descriptivos y concisos.
- No se utilizarán nombres de variables que puedan ser ambiguos.
- No usar ni grandes frases ni pequeñas abreviaciones para las variables.
- Se procurará evitar dar nombres sin sentido a variables temporales. Por ejemplo: temp, i, tmp. Las variables booleanas deben tener nombres que sugieran respuestas o contenidos de tipo Sí/No, por ejemplo: “esBaja”.
- Los nombres de las variables booleanas deben ser positivos, por ejemplo: “esBaja”.

Una vez definidos el plan de entregas y el estándar de codificación a utilizar, se desarrollan cada una de las tres iteraciones cumpliendo con el tiempo estimado.

3.2 Validación de los requisitos.

Luego de terminada la implementación del sistema, se procede a la verificación de los requisitos y el diseño propuesto, proceso que se describe a continuación. Las métricas del software aportan una manera cuantitativa de medir los atributos internos del producto, permitiendo por tanto al ingeniero, valorar la calidad durante la construcción del sistema. Con este objetivo se le aplica la métrica a los requisitos del software, para determinar la especificidad de los mismos. A continuación se muestra el resultado de la aplicación de estas métricas.

1. **Métrica para calidad de especificación de los requisitos**

Una de las métricas propuestas por la metodología de desarrollo seleccionada es la métrica de Calidad de la especificación (CE), la cual tiene como objetivo demostrar que la definición de los requisitos cumple con lo que quiere el usuario y que los errores no se propaguen a las fases siguientes, es decir, permite evaluar cuán entendibles y precisos son los requisitos. Para determinar la especificidad de los requisitos identificados, se procedió a comprobar la consistencia de la interpretación del equipo de revisores, constituido por el cliente y el equipo de desarrollo.

Para la realización de este proceso primero se calcula el total de requisitos de la especificación como se muestra a continuación:

NR = NRF+ NRNF .

NR= 57 + 8 .

NR= 65

Donde:

NR: Número de requisitos de especificación

NRF: Número de requisitos funcionales

NRNF: Número de requisitos no funcionales.

Para determinar la Especificidad de los Requisitos (ER), o ausencia de ambigüedad, se sugiere una métrica basada en la consistencia de la interpretación de los revisores para cada requisito. La misma no es más que el resultado de la división entre el total de requisitos con igual interpretación por parte de los revisores y la cantidad total de requisitos:

ER= NUI / NR

Donde NUI es el número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas. Mientras más cerca de 1 esté el valor de ER, menor será la ambigüedad.

Con el objetivo de obtener la menor ambigüedad posible de los requisitos, de modo que satisfagan las necesidades de los clientes, se llevaron a cabo un total de dos revisiones a cada uno de los requisitos por parte del equipo de desarrollo y por el cliente. En la primera, el mismo quedó satisfecho con el trabajo efectuado por el equipo de desarrollo, sin embargo se añadieron detalles

a doce de los requisitos funcionales para su perfeccionamiento. De un total de 52 requisitos funcionales, los revisores tuvieron la misma interpretación para 40 de ellos.

ER= 40 / 52

ER= 0,76

En la segunda revisión, ya corregidos los errores identificados en la primera, se añadieron cinco requisitos funcionales. Esta vez los revisores tuvieron la misma interpretación para el total de requisitos.

ER= 57 / 57

ER= 1

En la siguiente figura se muestra una gráfica resumen con los resultados obtenidos (ver Ilustración 7).

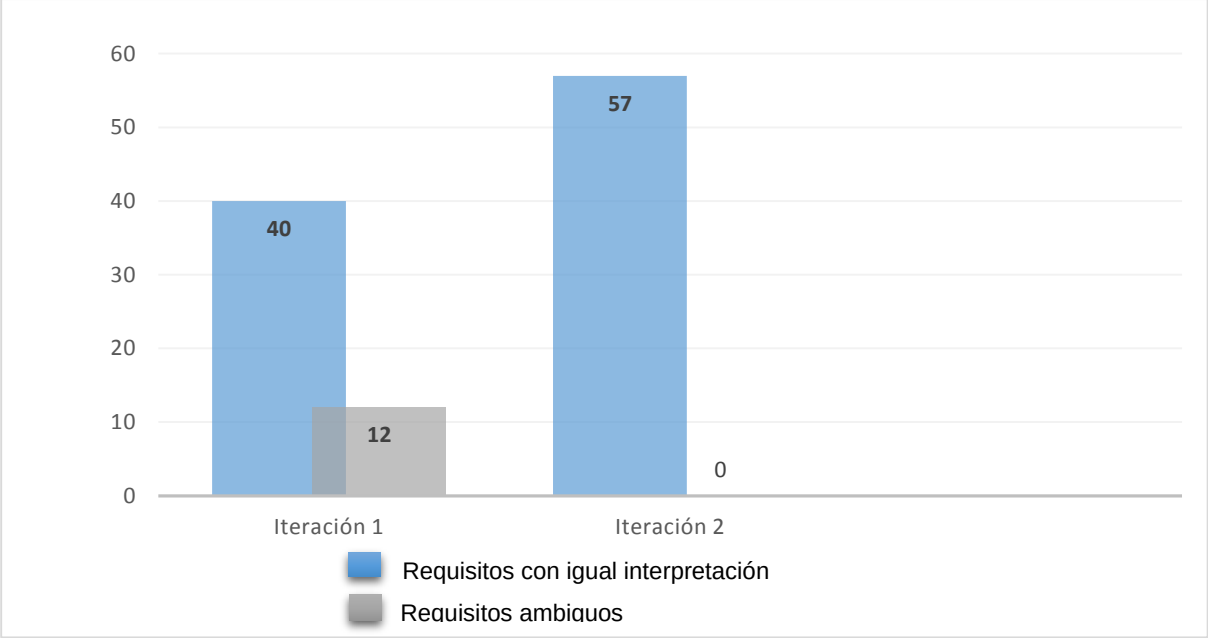

Ilustración 7: Gráfica de comparación entre la primera y la segunda revisión en la especificidad.

Luego de realizadas las dos revisiones, a partir de los resultados obtenidos, se puede concluir que el nivel de ambigüedad en los requisitos disminuyó, lográndose el nivel de especificidad requerida y el entendimiento total por parte de los revisores. Se considera que la definición de los requisitos fue satisfactoria.

3.2.1 Matriz de trazabilidad

La matriz de trazabilidad es una técnica que tiene como objetivo verificar si todas las partes del producto desarrollado se pueden identificar o relacionar con cada uno de los requisitos definidos en la fase de planificación-definición, es decir, si a partir de cualquier elemento del software se puede llegar hasta los requisitos o viceversa.

Mediante la matriz de trazabilidad se pudo comprobar que los requisitos definidos en el sistema eran los mismos que estaban reflejados en las HU. A continuación se presenta un fragmento de la tabla correspondiente a la matriz de trazabilidad del sistema (Ver Tabla 13).

Tabla 13: Matriz de trazabilidad

Requisitos / Historias de Usuario	Autenticar usuario	Gestionar usuario	Gestionar profesor	Gestionar brigada	Gestionar grupo	Gestionar local	Gestionar año	Gestionar asignatura	Gestionar turno	Gestionar departamento docente	Gestionar afectación	Gestionar guardia docente	Generar planificación
1	x												
2		x											
3		X											
4		X											
5		x											
6		X											
7			X										
8			X										
9			X										
10			X										
11			x										
12				X									
13				X									
14				X									
15				X									
16				x									
17					X								
18					X								
19					X								

3.3 Validación del diseño de la aplicación

Para comprobar la calidad y fiabilidad del diseño del sistema propuesto, se emplearon las métricas basadas en clases: Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC).

3.3.1 Tamaño Operacional de Clases (TOC)

La métrica TOC permite medir la responsabilidad, la complejidad de implementación y la reutilización de las clases del diseño. Es importante destacar que para esta métrica, la responsabilidad y la complejidad son inversamente proporcionales a la reutilización, por lo que a mayor responsabilidad y complejidad de implementación de una clase, menor será su nivel de reutilización. El tamaño operacional de una clase se puede determinar empleando medidas para saber el número total de operaciones que contiene.

Para una aplicación satisfactoria de esta métrica, es necesario conocer los atributos a medir (responsabilidad, complejidad, reutilización), y las categorías (alta, media, baja) en las que se puede clasificar cada clase según los resultados obtenidos luego de aplicar el criterio de clasificación. En la siguiente tabla se resumen los atributos, las categorías y los criterios que se definen para la aplicación de esta métrica (ver Tabla 14).

Tabla 14: Atributos, categorías y criterios definidos para la aplicación de la métrica TOC.

Atributos	Categoría	Criterio
Responsabilidad	Baja	< =Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio.
Complejidad implementación	Baja	< = Promedio.
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Reutilización	Baja	> 2* Promedio.
	Media	Entre Promedio y 2* Promedio
	Alta	<= Promedio

La variable promedio definida en la tabla anterior, representa el promedio de las clases. La misma se obtiene a partir de la división de la cantidad de procedimientos de las clases entre el total de clases. A continuación se muestra el resultado de la aplicación de la métrica TOC a las 22 clases más significativas dentro de los procesos principales del sistema informático.

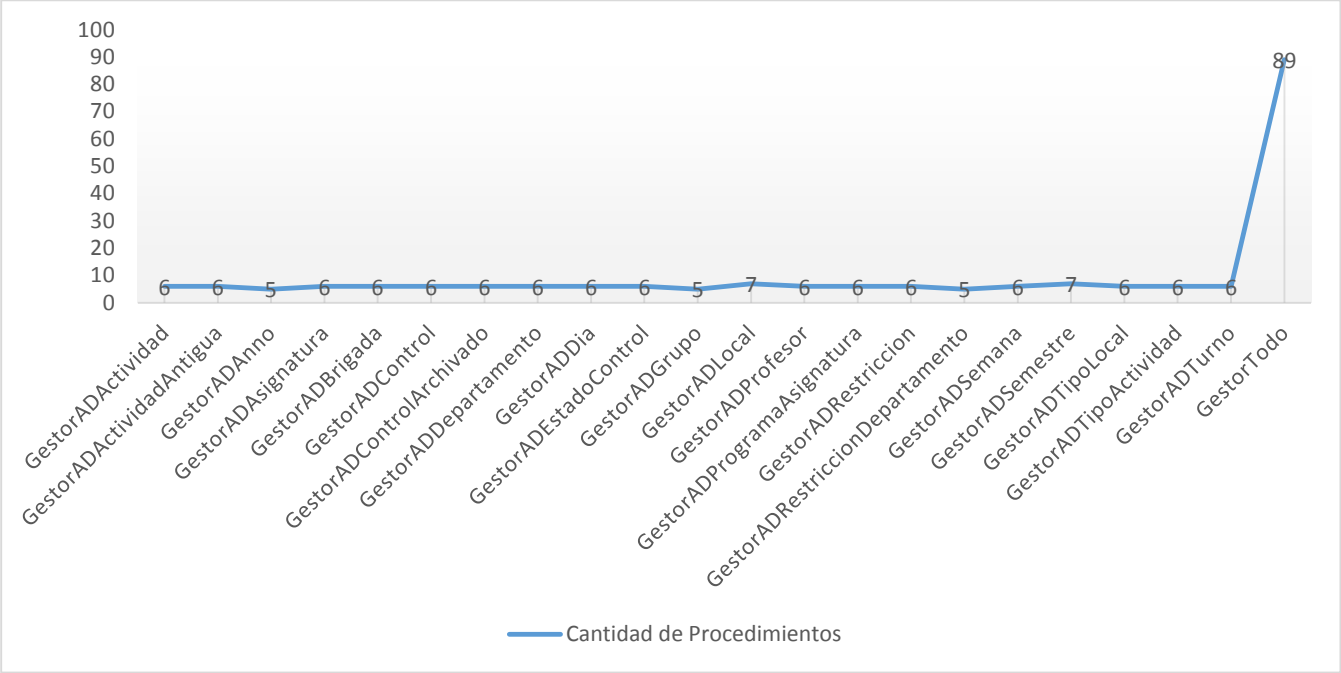

Ilustración 8: Cantidad de procedimientos por clase.

Aplicando como umbral un promedio de 10 procedimientos aproximadamente (214 procedimientos en total / 22 clases), se obtienen los valores de los atributos de calidad: responsabilidad de las clases, complejidad al implementar las mismas, así como sus niveles de reutilización, los cuales se resumen en la siguiente tabla (ver Tabla 15).

Tabla 15: Valores de las variables de calidad: Responsabilidad, Complejidad y Reutilización.

	Responsabilidad	Complejidad	Reutilización
Nivel	Cantidad de clases		
Bajo	21	21	1
Medio	0	0	0
Alto	1	1	21

En la siguiente figura se muestran los resultados obtenidos en porciento (%) durante la aplicación de la métrica en cuanto a la responsabilidad, la complejidad y la reutilización de los procedimientos de las clases en el sistema (ver Ilustración 9).

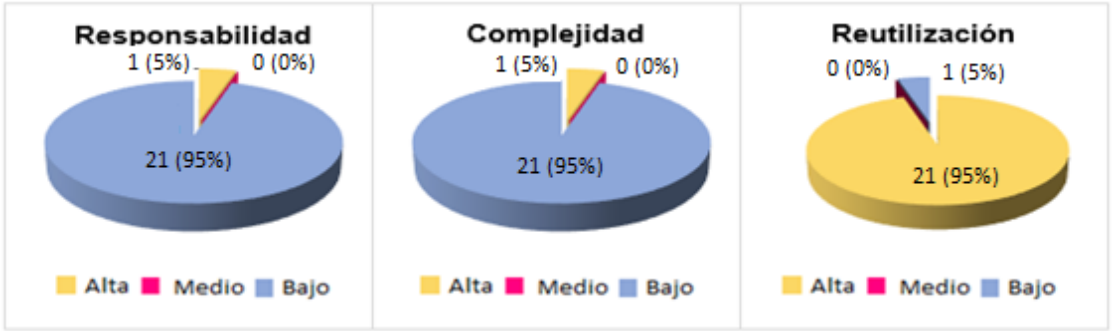

Ilustración 9: Resultados en porciento (%) de las variables: Responsabilidad, Complejidad y Reutilización.

Luego de aplicada la métrica TOC se observa, que las clases del diseño de la aplicación no se encuentran sobrecargadas en cuanto a responsabilidades, y el nivel de complejidad de las mismas es bajo, lo que favorece en gran medida su reutilización.

3.3.2 Relaciones entre Clases (RC)

La métrica Relaciones entre Clases (RC) está determinada por el número de relaciones de uso de una clase con otra, y evalúa los siguientes atributos de calidad:

- **Acoplamiento:** un aumento del RC implica un aumento del acoplamiento de la clase.
- **Complejidad de mantenimiento:** un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- **Reutilización:** un aumento del RC implica una disminución en el grado de reutilización de la clase.
- **Cantidad de pruebas:** un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Para una aplicación satisfactoria de esta métrica es necesario conocer los atributos a medir (acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas), y las categorías (alta, media, baja) en las que se puede clasificar cada clase según los resultados obtenidos, luego de aplicar el criterio de clasificación. En la siguiente tabla se recogen los criterios y categorías de evaluación para los atributos de calidad (ver Tabla 6).

Tabla 16: Atributos, categorías y criterios definidos para la aplicación de la métrica RC.

Atributos	Categoría	Criterio
Acoplamiento	Ninguno	0
	Baja	2
	Media	=Promedio
	Alta	>Promedio
Complejidad Mantenimiento	Baja	< = Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Reutilización	Baja	> 2* Promedio.
	Media	Entre Promedio y 2* Promedio
	Alta	<= Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio

Luego de aplicar la métrica RC, se obtuvieron los siguientes resultados (ver Ilustraciones 10 y 11):

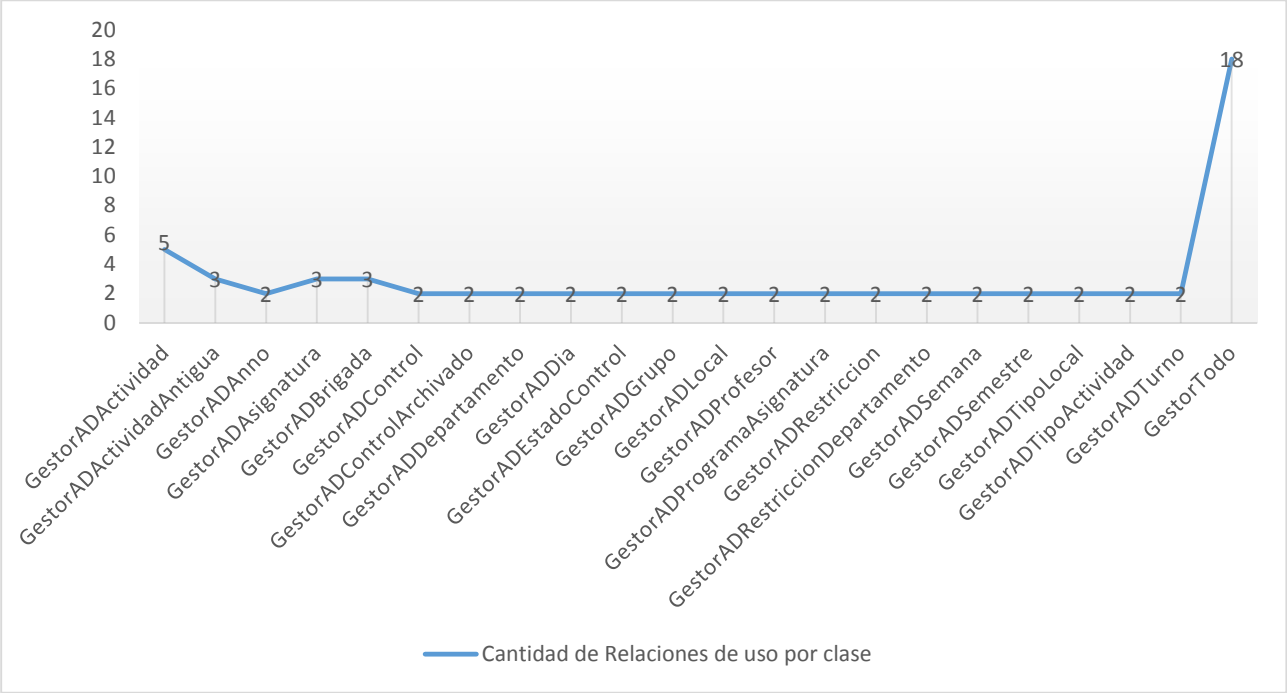

Ilustración 10: Cantidad de relaciones de uso por clases.

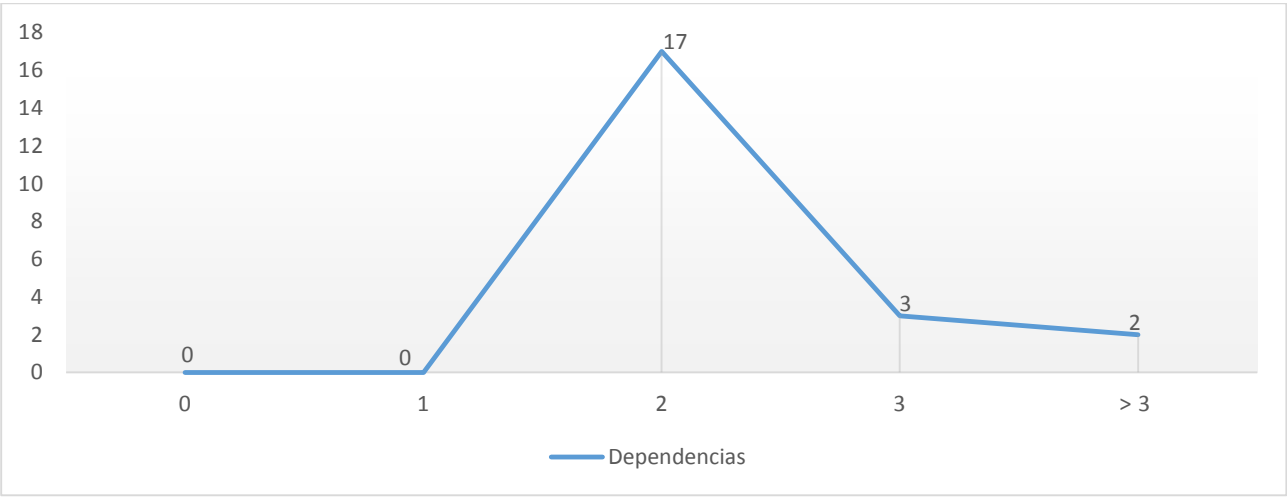

Ilustración 11: Resumen de relaciones de uso.

Los resultados mostrados en la ilustraciones 10 y 11, demuestran que el nivel de relaciones entre clases es bajo. Luego de calcular los valores de las variables de calidad: acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas, aplicando como umbral un promedio de tres asociaciones de uso por clase; se obtuvieron los siguientes resultados: (ver Tabla 17e Ilustración 12)

Tabla 17: Valores de las variables de calidad: Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas.

	Acoplamiento	Complejidad de Mantenimiento	Reutilización	Cantidad de pruebas
Nivel	Cantidad de clases			
Ninguno	0	-	-	-
Bajo	17	20	1	20

Medio	3	1	1	1
Alto	2	1	20	1

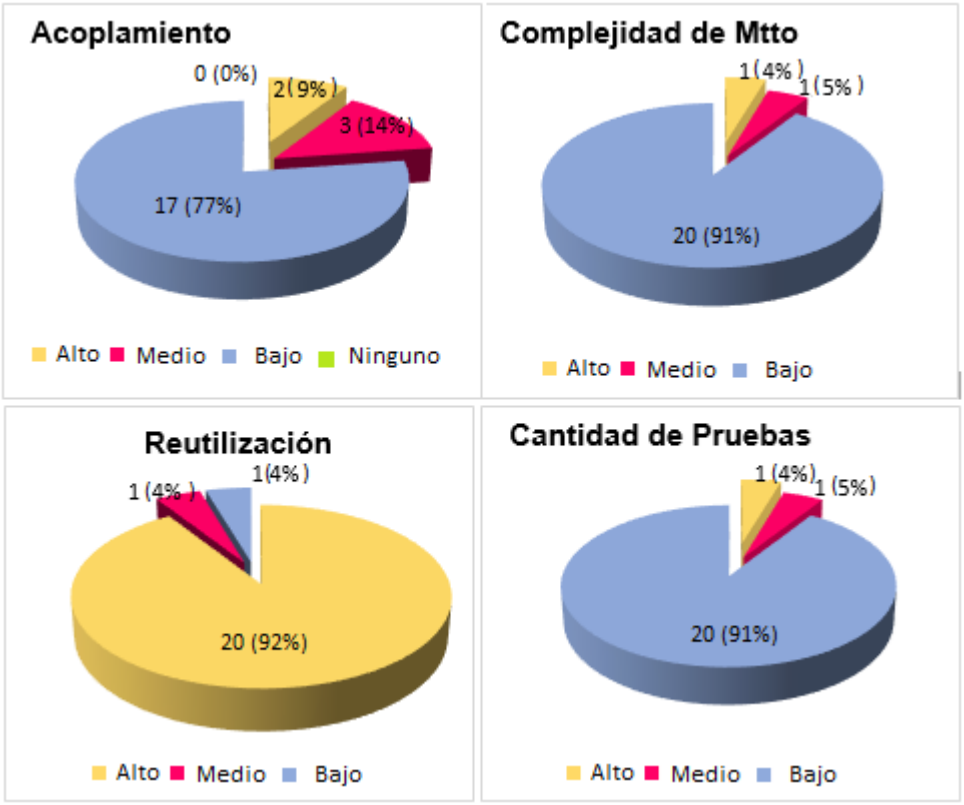

Ilustración 12: Resultado de las variables Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas.

La aplicación de esta métrica de software demostró que existe un bajo acoplamiento entre las diferentes clases de la solución propuesta, que la complejidad de mantenimiento también es baja y la reutilización alta. Se puede mencionar como otro de los resultados satisfactorios que el atributo cantidad de pruebas es bajo en un 91 %.

En sentido general, los resultados obtenidos al aplicar las métricas TOC y RC demuestran que el diseño de la aplicación no es complejo ya que las clases presentan bajo acoplamiento, una alta cohesión y un alto grado de reutilización.

3.4 Pruebas

Las pruebas son un instrumento adecuado para verificar el estado de la calidad de un producto en las cuales un sistema o componente es ejecutado bajo condiciones o requisitos especificados (Pressman, 2001). Una de las principales fortalezas de la metodología de desarrollo SXP es el proceso de pruebas, el cual permite asegurar el éxito del producto al realizarse de manera continua, proporcionando la obtención de un producto de mayor calidad donde los errores son detectados en un corto plazo de tiempo y se corrigen de una manera más sencilla.

Las pruebas no se realizan con el objetivo de prevenir errores, sino de detectarlos. Se efectúan sobre el trabajo realizado, y se deben ejecutar con la intención de descubrir la mayor cantidad de errores posible. A la aplicación se le realizarán pruebas unitarias y de funcionalidad. Para las pruebas de unidad el método que será utilizado es el de Caja Blanca, con la técnica de Camino Básico. Mientras que para las pruebas de funcionalidad será aplicado el método de Caja Negra, con la técnica de Partición de Equivalencia. A continuación se muestran los resultados de la aplicación de dichas pruebas.

3.4.1 Pruebas unitarias

Las pruebas de caja blanca del software se basan en un examen cercano al detalle procedimental. Se prueban las rutas lógicas del software y la colaboración entre componentes, al proporcionar casos de prueba que ejerciten conjuntos específicos de condiciones, bucles o ambos (Pressman, 2001).

Con el objetivo de comprobar que cada sentencia de código se ejecuta al menos una vez, se realizaron pruebas al código de las funcionalidades más complejas desde el punto de vista de la programación. Para ello se empleó el método de caja blanca con la técnica del Camino Básico; a partir de la obtención de la medida de la complejidad de un procedimiento o algoritmo, y la obtención de un conjunto básico de caminos de ejecución de este, los cuales son utilizados para obtener los casos de prueba. Seguidamente se muestra el proceso de pruebas realizado a la funcionalidad GenerarCalendarioSemestral, la cual pertenece al gestor del negocio GestorTodo. A continuación en la Ilustración 13 se muestra este método.

```
public void GenerarCalendarioSemestral(String[][] tabla, ENSemestre semestre) throws ParseException {
    String[][] calendario;
    SimpleDateFormat formateador = new SimpleDateFormat("yyyy-MM-dd 00:00:00");
    Date fecha;
    int filas = tabla.length;
    calendario = new String[filas][16];
    for (int i = 0; i < filas; i++) {
        fecha = formateador.parse(tabla[i][1]);
        calendario[i][0] = formateador.format(fecha);
        for (int j = 1; j < 16; j++) {
            Calendar calendar = Calendar.getInstance();
            calendar.setTime(fecha);
            calendar.add(Calendar.DAY_OF_YEAR, 7);
            fecha = calendar.getTime();
            while (ExisteRestriccion(fecha, tabla[i][0])) {
                calendar = Calendar.getInstance();
                calendar.setTime(fecha);
                calendar.add(Calendar.DAY_OF_YEAR, 7);
                fecha = calendar.getTime();
            }
            calendario[i][j]=formateador.format(fecha);
        }
    }
    semestre.setCalendario(calendario);
}
```

Ilustración 13: Método GenerarCalendarioSemestral

Para obtener los casos de prueba a partir de la técnica seleccionada se debe construir el grafo de flujo correspondiente al código de la funcionalidad. Luego se determina la complejidad ciclomática

V (G) del grafo resultante, la cual es un indicador del número de caminos independientes que existen en un grafo. En otras palabras, es cualquier camino dentro del código que introduce por lo menos un nuevo conjunto de sentencias de proceso o una nueva condición (del Pino et al., 2007). Esta métrica fue propuesta por Thomas McCabe[7] en 1976 y permite tomar la temperatura del código respecto su nivel de mantenibilidad, la probabilidad de incluir fallos, o el esfuerzo necesario para poder probar todos sus caminos. Es decir, cuanto más compleja sea la lógica de un código, más difícil será de entender, mantener y probar. La complejidad ciclomática puede ser calculada de tres formas:

1. V (G) = a - n + 2, siendo *a* el número de arcos o aristas del grafo y *n* el número de nodos.
2. V (G) = r, siendo *r* el número de regiones cerradas del grafo.
3. V (G) = c + 1, siendo *c* el número de nodos de condición.

Valores de referencia: Thomas McCabe, establece en sus trabajos los siguientes valores de referencia:

- <= 10, métodos sencillos, sin mucho riesgo.
- > 10, <= 20, métodos medianamente complejos, con riesgo moderado.
- > 20, <= 50, métodos complejos, con alto riesgo.
- > 50, métodos inestables, de altísimo riesgo.

Utilizando el código de la Ilustración 13 se realizó la representación del grafo de flujo, el cual está compuesto por los siguientes elementos (ver Ilustración 14):

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- Regiones: son las áreas delimitadas por aristas y nodos.

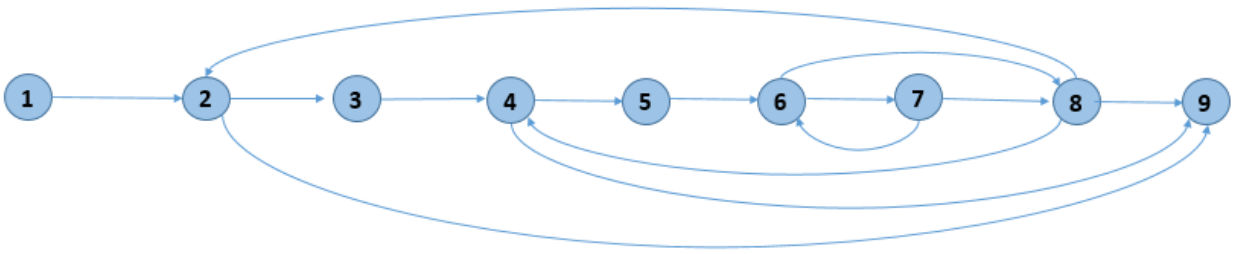

Ilustración 14: Grafo del camino básico del método GenerarCalendarioSemestral

[7] Thomas J. McCabe. A Complexity Measure. 308-320. http://dblp.uni-trier.de/db/journals/tse/tse2.html#McCabe76

A partir de esta representación, realizando los cálculos correspondientes se obtiene para cualquiera de las variantes el siguiente resultado:

V (G) = a - n + 2 V (G) = r V (G) = c + 1

V (G) = 14-9+2 V (G) = 7 V (G) = 6+1

V (G) =7

Por lo que el conjunto de caminos básico sería:

- Camino básico 1: 1-2-3-4-5-6-7-8-9
- Camino básico 2: 1-2-9
- Camino básico 3: 1-2-3-4-9
- Camino básico 4: 1-2-3-4-5-6-8-9
- Camino básico 5: 1-2-3-4-5-6-7-6-8-9
- Camino básico 6: 1-2-3-4-5-6-7-8-2-9
- Camino básico 7: 1-2-3-4-5-6-7-8-4-9

Luego se definen los casos de prueba para cada uno de los caminos básicos obtenidos. A continuación se muestra el resultado de las pruebas aplicadas al camino básico 1 (Ver tabla 18). Los resultados de las pruebas aplicadas al resto de los caminos básicos están reflejados en el artefacto Plantilla de Casos de Prueba.

Tabla 18: Caso de prueba para el camino básico 1.

Descripción: Se genera el calendario del semestre para todos los años, teniendo en cuenta las afectaciones.		
Condición de ejecución	Entrada	Resultados esperados
No debe existir ningún semestre activo.	-Se debe establecer la fecha de inicio del nuevo semestre, para cada año. -Se debe gestionar las afectaciones, que existen para ese semestre.	El sistema establece y muestra a través de una tabla, las fechas de inicio de cada una de las semanas del semestre correspondiente para cada año.
Resultado: Satisfactorio.		

3.4.2 Pruebas de funcionalidad

Las pruebas de caja negra son las que se aplican a la interfaz del software. Una prueba de este tipo se realiza si se conoce la función específica para la que se diseñó el producto, y se aplican con el

objetivo de probar que cada función es plenamente operacional, mientras se buscan los errores de cada función (Pressman, 2001).

Las pruebas de caja negra, realizadas para comprobar que las funcionalidades de la herramienta responden a las necesidades del cliente, fueron desarrolladas por el grupo de calidad de CEGEL. El desarrollo de las pruebas se realizó en un total de tres iteraciones. A continuación en la Ilustración 15 se refleja el resultado de aplicar las mismas, teniendo en cuenta los parámetros de interfaz, funcionalidad, redacción y recomendación.

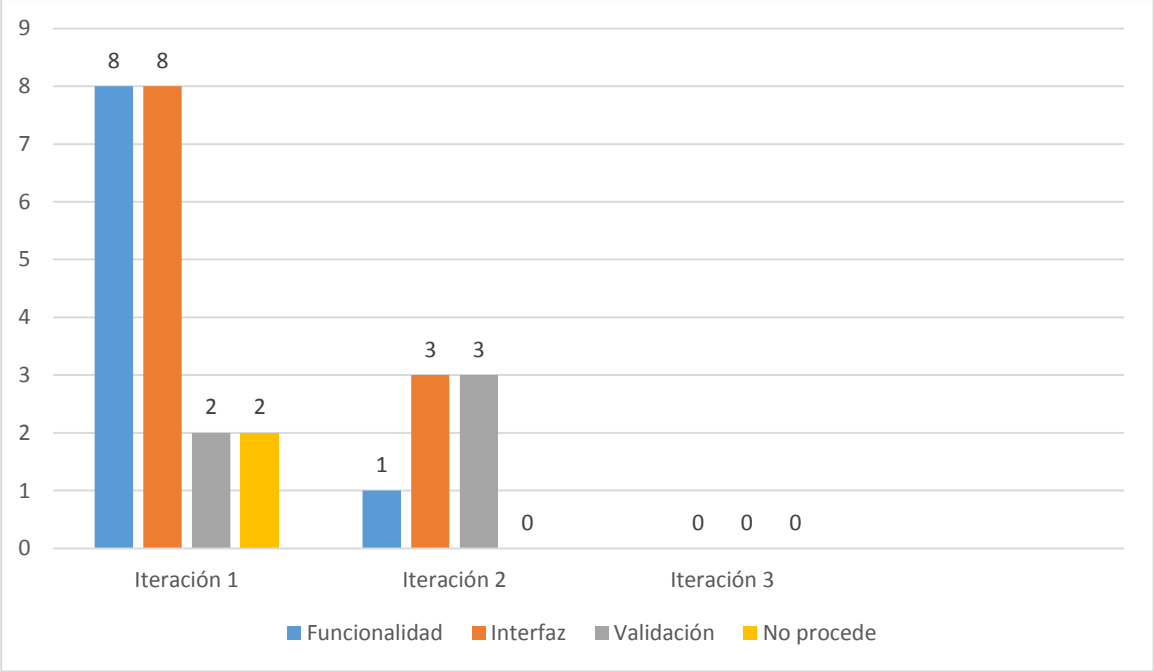

Ilustración 15: No conformidades detectadas en la herramienta a través de las pruebas de caja negra.

En una primera iteración se detectaron un total de 20 No Conformidades (NC), predominando entre ellas las de los tipos de funcionalidad y de interfaz. De estas NC, se resolvieron 18 y 2 no procedieron. En una segunda iteración se encontraron 7 no conformidades, esta vez prevaleciendo las de interfaz y validación. Finalmente en una tercera iteración, se obtuvieron resultados satisfactorios debido a que no se encontraron NC, emitiéndose por parte del grupo de calidad de CEGEL el Acta de liberación interna de productos de software como evidencia (ver Anexo 2).

3.5 Validación de la investigación

La herramienta desarrollada facilita en gran medida los procesos de planificación y control de las actividades docentes que se desarrollan en la Facultad 3. A continuación se muestra una tabla resumen donde se evidencia dicha afirmación. En la misma se realiza una comparación entre la herramienta actual y la propuesta en esta investigación, basada en los principales aspectos que dificultan la planificación en la herramienta existente.

Tabla 19: Comparación entre la herramienta actual y la propuesta.

Antes	Después
Cuando se modifica el horario para un año académico en el momento de la publicación puede generar cambios en otro.	Al sistema garantizar que la planificación tenga menos errores, reduce la necesidad de la planificación sistemática, y con ello la probabilidad de ocurrir errores a la hora de la publicación.
No permite la asignación de dos profesores a un mismo grupo en el sistema de profesores clases prácticas y conferencias.	Permite la asignación de dos profesores a un mismo grupo en el sistema de profesores clases prácticas y conferencias.
No permite más de un grupo en los salones de conferencias.	Permite agrupar al menos dos brigadas en los salones de conferencia.
No tiene en cuenta que cuando se planifica actividad en el tercer turno no se puede planificar en cuarto turno para los mismos profesores y estudiantes.	Para la realización de la planificación se tiene en cuenta que cuando se planifica actividad en el tercer turno no se puede planificar en cuarto turno para los mismos profesores y estudiantes.
No se tiene en cuenta lo profesores externos para la planificación de 6to turno.	Es una restricción incluida en el sistema y que se cumple siempre que sea posible en la planificación
No permite gestionar el control del cumplimiento del horario docente.	Permite gestionar el control del cumplimiento del horario docente.
Sistema que sólo se ejecuta sobre el sistema operativo Windows.	El sistema es multiplataforma.
No implementa seguridad de la información	Limita el acceso a las diferentes funcionalidades, ya que los usuarios de la aplicación acceden a las mismas en dependencia de su rol.

De los aspectos antes mencionados existe una mejora significativa en seis de ellos, por lo que se concluye que de forma general estas ventajas de la propuesta representan un gran apoyo para la realización de la planificación en la Facultad 3 de la UCI.

Para evaluar el impacto de la propuesta de solución se aplicó una encuesta a un total de dieciséis profesionales, con distintos niveles de responsabilidades y experiencia en este tema. De las dieciséis encuestas se descarta una por haberse autoevaluado con muy bajo nivel de experticia (1 de 10). El promedio de semestres de experiencia en la planificación de los encuestados es de aproximadamente once semestres y el nivel de conocimiento nueve. Los indicadores definidos para evaluar el impacto del sistema en la planificación, dando por hecho que resuelve las limitaciones antes mencionadas, fueron:

1. Menor tiempo de planificación
2. Mayor calidad de la planificación
3. Menor esfuerzo de los involucrados en la planificación
4. Mayor satisfacción de estudiantes y profesores
5. Mejor control de las incidencias en el cumplimiento del horario
6. Mejor gestión y planificación del horario

Los resultados arrojados se muestran en la siguiente figura (ver Ilustración 16).

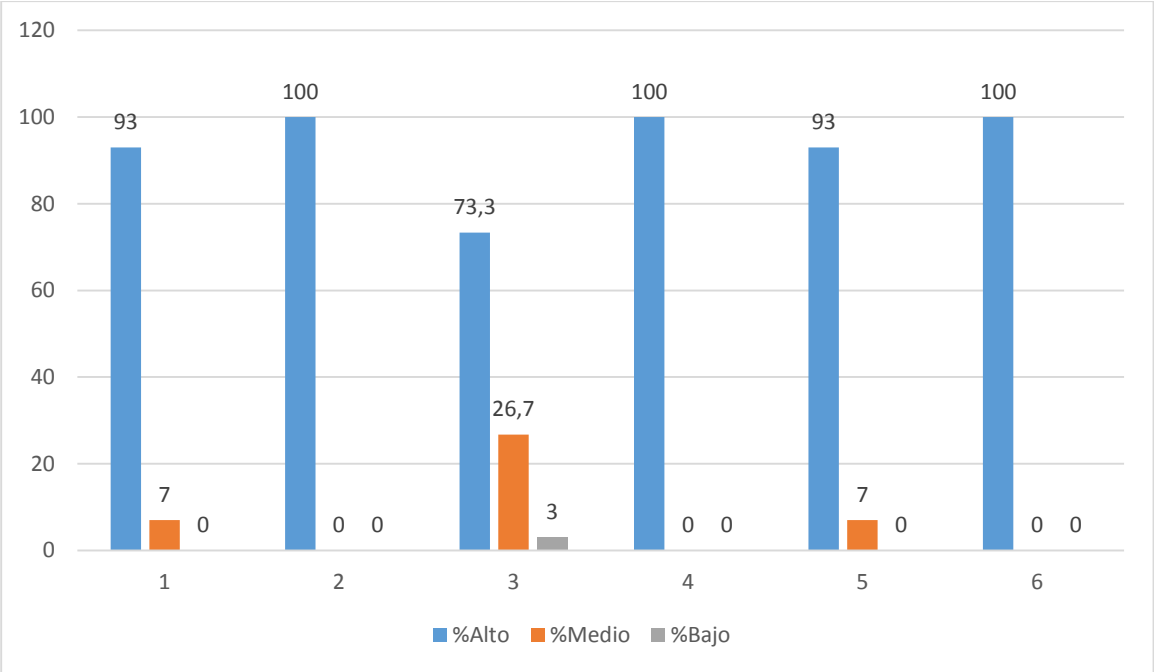

Ilustración 16: Resultados de la encuesta aplicada para valorar el impacto de la herramienta

Como se pudo constatar, se prevé que la explotación de la herramienta de planificación desarrollada favorecerá con un impacto alto el proceso de planificación de las actividades docentes. De esta forma queda evidenciado el cumplimiento del objetivo general propuesto de desarrollar un sistema de gestión para la planificación y control del horario docente en la Facultad 3, que permita reducir el tiempo necesario para su diseño, teniendo en cuenta todos los requisitos y restricciones del proceso.

3.6 Conclusiones del capítulo

A lo largo del capítulo se abordaron los elementos asociados a las fases de desarrollo siguiendo la metodología seleccionada. Los resultados obtenidos fueron:

- Se definió el Plan de entregas con un total de tres iteraciones, las cuales se desarrollaron de forma satisfactoria utilizando el estándar de codificación lowerCamelCase.

- Para la verificación de los requisitos se utilizaron las métricas de Especificación de requisitos, en la cual luego de dos revisiones se lograron eliminar las ambigüedades identificadas.

- Se aplicaron las métricas TOC y RC para validar el diseño, las cuales permitieron medir la responsabilidad, la complejidad de implementación, el acoplamiento y la reutilización de las clases, arrojando como resultado que las mismas no se encuentran sobrecargadas en cuanto a responsabilidades, y que presentan un bajo nivel de complejidad, así como un bajo acoplamiento, lo que favorece en gran medida una alta reutilización de estas.
- Las pruebas realizadas para validar las historias de usuario permitió comprobar que los flujos para cada una de ellas se desarrollaban de la forma en que estaban definidos.
- De forma general, con las pruebas y validaciones realizadas se puede concluir que la herramienta desarrollada cumple con las especificaciones y requisitos definidos por los clientes en la etapa de concepción del sistema.
- A partir de la valoración de quince expertos con once semestres de experiencia en la planificación de actividades y nivel de conocimiento en el tema nueve, se pudo constatar que el impacto de la solución en menor tiempo de planificación, mayor calidad, menor esfuerzo de los involucrados, y mejor gestión y planificación del horario, es alto.

De esta forma se da por cumplido satisfactoriamente el objetivo general propuesto en esta investigación.

CONCLUSIONES GENERALES

Con la realización del presente trabajo de diploma se arriban a las siguientes conclusiones:

- A partir de una encuesta realizada a 16 profesionales con elevada experiencia en la planificación docente, se pudo constatar la necesidad de un sistema de gestión para la planificación y el control docente.
- Se realizó un análisis crítico de las herramientas de planificación docente a nivel internacional, así como de propuestas realizadas en la UCI, lo que permitió concluir que no existe ninguna que dé respuesta a las necesidades planteadas en la investigación. Es por ello que se decide la implementación de un nuevo sistema que incluya los elementos positivos de dichas herramientas.
- Teniendo en cuenta las características del sistema y del equipo de desarrollo se seleccionaron como metodología de desarrollo SXP, como lenguaje de programación Java, como IDE Netbeans, SQLite como sistema gestor de bases de datos y Navicat Premium como herramienta de administración de bases de datos.
- Las métricas aplicadas, las pruebas unitarias y de funcionalidad, verificaron el correcto funcionamiento del sistema, la calidad de la solución y el cumplimiento de las expectativas del cliente.
- A partir de la valoración de quince expertos con once semestres de experiencia en la planificación de actividades y nivel de conocimiento en el tema nueve, se pudo constatar que el impacto de la solución en menor tiempo de planificación, mayor calidad, menor esfuerzo de los involucrados, y mejor gestión y planificación del horario, es alto.

De esta forma se da por cumplido satisfactoriamente el objetivo general propuesto en esta investigación.

RECOMENDACIONES

El autor de este trabajo recomienda:

- Profundizar en el estudio, de algoritmos clásicos utilizados en la resolución de problemas de horario, con el objetivo de añadir a la solución propuesta una mayor flexibilidad en la manipulación de las restricciones involucradas.
- Incorporar técnicas de Inteligencia Artificial que permitan incorporar mecanismos de aprendizaje para lograr una planificación más eficiente.
- Mantener constante retroalimentación de los usuarios, de forma que se añadan mejoras funcionales enfocadas al beneficio de los mismos, y que enriquezcan y perfeccionen la aplicación.

BIBLIOGRAFÍA

1. Acosta Henríquez, G.F. y otros 2008. Utilización de herramientas de código abierto para la generación de reportes en Java. Revista tecnológica. (2008), 1 (1), 24-26 [en línea], [Consulta: 3 junio 2015]. Disponible en: http://redicces.org.sv/jspui/handle/10972/522.

2. aSc Horario de clases - horario software. [en línea] 2015. [Consulta: 3 junio 2015]. Disponible en: http://www.asctimetables.com/timetables_es.html.

3. Baquero, F., Fredy, J., Ocampo, T., Mirledy, E., Rendón, G. y Alfonso, R. 2008. Course timetabling problem resolved using Tabu Search. Ingeniería y Desarrollo, no. 24, pp. 149-175. ISSN 0122-3461.

4. Barriga, L. 2009. La Planificación. Breve Introducción en: http://www. geocities. com/WallStreet/District/7921/Planification. html. Bishwapriya Sanya. Planning as anticipation of resistance en: http: www. seweb. uci. edu/ppd/ufiles/Sanyal% 20Resistance. pdf [en línea], [Consulta: 3 junio 2015]. Disponible en: http://www.geocities.ws/franklin.marcano/planificacion/t1/link1.pdf.

5. Brugal, L.C. e Hidalgo García, C.Y. 2007. Sistema informático de gestión para actividades docentes y extradocentes en la facultad 3. Rol Analista de Sistemas. [en línea], [Consulta: 3 junio 2015]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_0236_07.

6. Canós, J., Letelier, P. y Penadés, M.C. 2003. Metodologías Ágiles en el desarrollo de Software. Universidad Politécnica de Valencia, Valencia [en línea], [Consulta: 3 junio 2015]. Disponible en: http://www.willydev.net/Willydev_old/Root/descargas/prev/TodoAgil.Pdf.

7. Castro, C.A.C., Blandón, G.E.T. y Tabares, R. de J.B. 2009. Método y Entorno Integrado de Desarrollo para el Aprendizaje en Lógica de Programación Orientada por Objetos. Revista Iberoamericana de Sistemas, Cibernética e In$\neg$ formática.[En línea] [en línea], vol. 6, no. 2. [Consulta: 3 junio 2015]. Disponible en: http://www.iiisci.org/journal/CV$/risci/pdfs/GX926UI.pdf.

8. Castro, W.J. 1982. Introducción al estudio de la teoría administrativa [en línea]. S.l.: Fondo de Cultura Económica. [Consulta: 3 junio 2015]. Disponible en: http://bases.bireme.br/cgi-bin/wxislind.exe/iah/online/?IsisScript=iah/iah.xis&src=google&base=LILACS&lang=p&nextAction=lnk&exprSearch=373116&indexSearch=ID.

9. Clements, P.C. 1996. A survey of architecture description languages. Proceedings of the 8th international workshop on software specification and design [en línea]. S.l.: IEEE Computer Society, pp. 16. [Consulta: 3 junio 2015]. Disponible en: http://dl.acm.org/citation.cfm?id=858261.

10. de la Torre Llorente, C., Castro, U.Z., Nelson, J.C., Barros, M.A.R., Pacios, C.M., Hierro, F.C. y Mesa, I.G. 2010. Guía de arquitectura N-Capas orientada al dominio con. NET 4.0. S.l.: Krasis Press.

11. del Pino, J.C.R., Roca, M.D., Figueroa, Z.H. y Domínguez, J.D.G. 2007. Hacia la evaluación continua automática de prácticas de programación. Actas de las XIII Jornadas de Enseñanza universitaria de la Informática [en línea], [Consulta: 4 junio 2015]. Disponible en: http://www2.dis.ulpgc.es/~mdiaz/EvalContAPP.pdf.

12. Desarrolladores de Java 2013. definición-Java (lenguaje de programación). [en línea], [Consulta: 3 junio 2015]. Disponible en: http://diccionario.sensagent.com/Java%20(lenguaje%20de%20programaci%C3%B3n)/es-es/.

13. DocCF, Software de Gestión Escolar. Grupo CF Developer | DocCF, Software de Gestión Escolar [en línea] 2015. [Consulta: 3 junio 2015]. Disponible en: http://www.grupocfdeveloper.com/.

14. Drucker, P. 2014. La gerencia efectiva. S.l.: Sudamericana.

15. Fernández, L.F. 2006. Arquitectura de Software. Software Guru, vol. 2, no. 3, pp. 40–45.

16. Fine, L.H. 2004. Seguridad en centros de cómputo. México, Editorial Trillas [en línea], [Consulta: 3 junio 2015]. Disponible en: http://dspace.ucbscz.edu.bo/dspace/handle/123456789/3485.

17. Fowler, M. by Scott, K. 1999. UML gota a gota [en línea]. S.l.: Pearson Educación. [Consulta: 3 junio 2015]. Disponible en: http://books.google.com/books?hl=es&lr=&id=AL0YkFeaHwIC&oi=fnd&pg=PT14&dq=Lenguaje+Unificado+de+Modelado+(UML)&ots=Fv_RK4chRr&sig=dA1NnRLP3-8pQue9Dv1XyCmCiAk.

18. Garlan, D. y Shaw, M. 1994. An introduction to software architecture. [en línea], [Consulta: 3 junio 2015]. Disponible en: http://repository.cmu.edu/compsci/724/?utm_source=repository.cmu.edu%2Fcompsci%2F724&utm_medium=PDF&utm_campaign=PDFCoverPages.

19. Gestor de Horarios para Centros de Enseñanza. Softonic [en línea] 2015. [Consulta: 3 junio 2015]. Disponible en: http://gestor-de-horarios-para-centros-de-ensenanza.softonic.com/.

20. Goldrate, E. 2010. La meta [en línea]. S.l.: Ediciones Granica SA. [Consulta: 3 junio 2015]. Disponible en: http://books.google.com/books?hl=es&lr=&id=DKBYLMwp07AC&oi=fnd&pg=PA9&dq=Goldratt+la+meta&ots=BTsG9wcxBs&sig=nVn6sWOewfHZM6mNQkz0q8DQSQs.

21. G. Tyler Miller y Scott Spoolman 2011. Living in the Environment: Principles, Connections, and Solutions. [en línea]. S.l.: Belmont Books. [Consulta: 3 junio 2015]. Disponible en: http://scholar.google.com/scholar?hl=es&q=Living+in+the+Environment%3A+Principles%2C+Connections%2C+and+Solutions+Miller&btnG=&lr.

22. Guerra, L.A. 2012. Sistema de Gestión para la planificación docente en la Facultad Regional Mártires de Artemisa. [en línea], [Consulta: 3 junio 2015]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_06167_12.

23. Gutiérrez, J.J. 2006. ¿Qué es un framework web? [en línea]. S.l.: Recuperado el. [Consulta: 3 junio 2015]. Disponible en: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

24. Gvirtz, S. y Palamidessi, M. 1998. El ABC de la tarea docente: currículum y enseñanza [en línea]. S.l.: Aique. [Consulta: 3 junio 2015]. Disponible en: http://www.terras.edu.ar/biblioteca/35/35GVIRTZ-Silvina-PALAMIDESSI-Mariano-Segunda-parte-Cap-6-La-planificacion.pdf.

25. Hernández, D.Á., Martínez Vázquez, E. y Souchay Fabrega, D. 2009. Sistema de Gestión para la Planificación Docente en la Facultad 5. [en línea], [Consulta: 3 junio 2015]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_2597_09.

26. Javier García de Jalón, José Ignacio Rodríguez, Iñigo Mingo, Aitor Imaz, Alfonso Brazález, Alberto Larzabal, Jesús Calleja y Jon García 2000. Aprenda Java como si estuviera en primero. [en línea]. S.l.: s.n. [Consulta: 3 junio 2015]. Disponible en: http://eva.uci.cu/file.php/136/IP/Bibliografia/Complementaria_espanol/Aprenda_Java_como_si_estuviera_en_primero.pdf.

27. Katrib, M. 2003. Programación Orientada a Objeto en C++. [en línea]. S.l.: s.n. [Consulta: 3 junio 2015]. Disponible en: http://eva.uci.cu/file.php/136/IP/Bibliografia/Complementaria_espanol/Miguel_Katrib_Programacion_Orientada_a_Objetos_en_CPP/CAPI1.pdf.

28. KRONOWIN Generador de Horarios. Softonic [en línea] 2015. [Consulta: 3 junio 2015]. Disponible en: http://kronowin-generador-de-horarios.softonic.com/.

29. Kushwaha, D.S. y Misra, A.K. 2006. A complexity measure based on information contained in the software. Proceedings of the 5th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems [en línea]. S.l.: World Scientific and Engineering Academy and Society (WSEAS), pp. 187–195. [Consulta: 3 junio 2015]. Disponible en: http://www.wseas.us/e-library/conferences/2006madrid/papers/512-321.pdf.

30. Larman, C. 1999. UML y Patrones [en línea]. S.l.: Pearson. [Consulta: 3 junio 2015]. Disponible en: http://gravepa.com/granaino/biblioteca/aprende/UNED%20-%20Grado%20Inform%C3%A1tica%20-/Extras%20%26%20Ediciones%20Antiguas/Una%20introduccion%20al%20analisis%20y%20diseno%20orientado%20a%20objetos%20y%20al%20proceso%20unificado.pdf.

31. Letelier, P. 2006. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [en línea], [Consulta: 3 junio 2015]. Disponible en: http://www.cyta.com.ar/ta0502/b_v5n2a1.htm.

32. Mato García, R.M. 1999. Diseño de Bases de Datos. Editorial Pueblo y Educación, La Habana, Cuba, octubre del, vol. 99, pp. 51.

33. Mendoza, L., Pérez, M., Grimán, A. y otros 2005. Prototipo de modelo sistémico de calidad (MOSCA) del software. Computación y Sistemas, vol. 8, no. 3, pp. 196–217.

34. Meyer, B., Sánchez, S., Mora, M.K. y Bermejo, R.G. 1999. Construcción de software orientado a objetos [en línea]. S.l.: Prentice Hall. [Consulta: 3 junio 2015]. Disponible en: http://www.sidalc.net/cgi-bin/wxis.exe/?IsisScript=AGRIUAN.xis&method=post&formato=2&cantidad=1&expresion=mfn=013927.

35. Miller, G. y Spoolman, S. 2011. Living in the environment: principles, connections, and solutions [en línea]. S.l.: Cengage Learning. [Consulta: 3 junio 2015]. Disponible en: http://books.google.com/books?hl=es&lr=&id=QcQ8AAAAQBAJ&oi=fnd&pg=PP1&dq=Living+in+the+Environment:+Principles,+Connections,+and+Solutions&ots=yTEkSYHpsG&sig=Gi1IZBMcfDtvmEYv_pzvCrPCUkw.

36. Mustelier, D., 2013. Variables que Definen la Complejidad de los Requisitos Funcionales del Software. [en línea]. 2013. S.l.: s.n. [Consulta: 3 junio 2015]. Disponible en: http://fundacioniai.org/raccis/v3n2/n5a4.pdf.

37. Navicat Premium | Conecta múltiples bases de datos en una única interfaz gráfica de usuario. [en línea] 2015. [Consulta: 3 junio 2015]. Disponible en: http://www.navicat.com/es/products/navicat-premium.

38. NetBeans IDE - Java. [en línea] 2015. [Consulta: 3 junio 2015]. Disponible en: https://netbeans.org/features/java/.

39. Pressman, R.S. 2001. Pressman [en línea]. S.l.: s.n. [Consulta: 3 junio 2015]. Disponible en: http://eva.uci.cu/file.php/158/Documentos/Bibliografia_general/Textos_Basicos/Ediciones_del_Pressman/Pressman_7ma_edicion/04_Chapter_1._Software_and_Software_Engineering.pdf.

40. Ramos Fernández, V. 2009. Desarrollo de un marco de trabajo para Java/J2EE: plataforma Puzzle. [en línea], [Consulta: 3 junio 2015]. Disponible en: http://e-archivo.uc3m.es/handle/10016/6231.

41. Ramos, M.O. y Torres Iglesias, Y. 2008. Análisis y Diseño de un sistema para la planificación automatizada del Horario Docente de la facultad 4. [en línea], [Consulta: 3 junio 2015]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_1488_08.

42. Rojas, S.G. y Díaz, L.O. 2003. Java a Tope: J2me (java 2 Micro Edition). [en línea]. S.l.: Sergio Gálvez Rojas. [Consulta: 3 junio 2015]. Disponible en: http://books.google.com/books?hl=es&lr=&id=USaAQ0hHQWIC&oi=fnd&pg=PR5&dq=M%C3%A1quina+virtual+de+java+8.0&ots=7xL0THfycj&sig=e2ls4ybo-cxGTBXb9TUXxkX2SrI.

43. Romero, G.M.P. 2011. SXP, metodología de desarrollo de software. Serie Científica [en línea], vol. 4, no. 11. [Consulta: 3 junio 2015]. Disponible en: http://publicaciones.uci.cu/index.php/SC/article/view/429.

44. Sala, J.J.R. 2003. Introducción a la programación.pdf [en línea]. S.l.: s.n. [Consulta: 3 junio 2015]. Disponible en: http://eva.uci.cu/file.php/136/IP/Bibliografia/Complementaria_espanol/Introduccion_a_la_programacion.pdf.

45. Sánchez, M.A.M. 2004. Metodologías De Desarrollo De Software [en línea]. S.l.: Jun. [Consulta: 3 junio 2015]. Disponible en: http://www.willydev.net/insitecreation/v1.0/Descargas/cualmetodologia.pdf.

46. Silberschatz, A., Korth, H.F., Sudarshan, S., Pérez, F.S., Cordero, A.G. y Fernández, J.C. 2002. Fundamentos de bases de datos [en línea]. S.l.: McGraw-Hill. [Consulta: 3 junio 2015]. Disponible en: http://www.sidalc.net/cgi-bin/wxis.exe/?IsisScript=SIDINA.xis&method=post&formato=2&cantidad=1&expresion=mfn=003016.

47. Sommerville, I. 2005. Sommerville_Parte_I_Vision_General.pdf [en línea]. 7ma. S.l.: s.n. [Consulta: 3 junio 2015]. Disponible en: http://eva.uci.cu/file.php/158/Documentos/Bibliografia_general/Textos_Complementarios/Ediciones_del_Sommerville/Sommerville_7ma_edicion/Sommerville_Parte_I_Vision_General.pdf.

48. SQLite Home Page. [en línea] 2015. [Consulta: 3 junio 2015]. Disponible en: https://sqlite.org/.

49. Stoner J.A.F., Freeman, R.E., Gilbert, D.R., Sacristan, P.M. y otros 1996. Administración [en línea]. S.l.: Pearson Educación. [Consulta: 1 junio 2015]. Disponible en: http://books.google.com/books?hl=es&lr=&id=eWOvsi2iY8C&oi=fnd&pg=PR13&dq=James+Administraci%C3%B3n.+6a+Ed.&ots=TWku4ZuhbL&sig=GeBTx1hPFkKU8erQLwOQdfpGX1M.

50. UPware Soft - Casa de desarrollo de Software - School Pack Infinite - School Pack - Campus Pack - SCI - GPUntis. UPware Soft [en línea] 2014. [Consulta: 3 junio 2015]. Disponible en: http://www.upwaresoft.com.

51. Vela, 2007. Resolución no. 210 del 2007 con sus adecuaciones resaltadas para la UCI. 2007. S.l.: s.n.

52. Yang, D. 2010. Java persistence with JPA [en línea]. S.l.: Outskirts Press. [Consulta: 3 junio 2015]. Disponible en: http://dl.acm.org/citation.cfm?id=1841782.

ANEXOS

Anexo 1: Encuesta realizada para la constatación del estado de la planificación docente en la Facultad 3.

Estimado(a) compañero(a), la presente encuesta forma parte de la aplicación de la Consulta a Expertos con el objetivo de obtener sus consideraciones sobre un sistema para la planificación y control del horario docente en la Facultad 3.

Se agradece su valiosa y calificada colaboración. Se le asegura que sus criterios contribuirán al perfeccionamiento de la propuesta y a la validación teórica de la presente investigación.

1. Información sobre los expertos

Nombre y Apellidos: _____

Graduado de: _____

Fecha de graduación: _____

Categoría Docente: Instructor__ Asistente__ Prof. Auxiliar__ Prof. Titular__

Título académico y/o grado científico: Máster___ Doctor___

Puesto de trabajo actual: _____

Semestres de experiencia en la planificación docente: _____

2. ¿Qué conocimientos usted posee acerca de la planificación del horario docente? _____

Indicación: Indique con un valor en la siguiente escala creciente del 1 (ninguno) al 10 (máximo) el valor que corresponda al grado de experiencia que usted tiene sobre el tema.

3. Le pedimos su criterio sobre los siguientes elementos que se corresponden con las características del sistema de planificación HORR PLANNER. Esto permitirá determinar la necesidad de un nuevo sistema.

Indicación: Marque con una equis (x) el criterio que más se ajusta al suyo para cada elemento de acuerdo a la propuesta.

Leyenda para los criterios: I – Inadecuado; PA – Poco Adecuado; A – Adecuado; BA – Bastante Adecuado; MA – Muy Adecuado.

#	Elementos a valorar del sistema	Criterios				
		I	PA	A	BA	MA
1	En el sistema actual se modifica el horario para un año académico en el momento de la publicación puede generar cambios en otro					
2	No permite la asignación de dos profesores a un mismo grupo en el sistema de profesores clases prácticas y conferencias.					
3	No permite más de un grupo en los salones de conferencias.					
4	No tiene en cuenta que cuando se planifica actividad en el tercer turno no se puede planificar en cuarto turno para los mismos profesores y estudiantes.					
5	No se tiene en cuenta las afectaciones de los profesores externos ni de las madres.					
6	No permite más de 6 turnos de una asignatura lo que afecta a las asignaturas de la Práctica Profesional.					
7	No permite gestionar el control del cumplimiento del horario docente.					

Otras valoraciones que usted desee aportar sobre la presente investigación serán de gran valor para su enriquecimiento y se le invita a realizarlas a continuación: _____

_____

4. Le pedimos su criterio sobre el impacto del uso de un sistema de planificación que resuelva estas limitaciones.

Indicación: Marque con una equis (x) el criterio que más se ajusta al suyo para cada elemento de acuerdo a la propuesta.

Leyenda para los criterios: B- Bajo; M- Medio; A – Alto.

#	Elementos a valorar el impacto	B	M	A
1	Menor tiempo de planificación			
2	Mayor calidad de la planificación			
3	Menor esfuerzo de los involucrados en la planificación			
4	Mayor satisfacción de estudiantes y profesores			
5	Mejor control de las incidencias en el cumplimiento del horario			
6	Mejor gestión y planificación del horario			

Otras valoraciones que usted desee aportar sobre el impacto de la propuesta serán de gran valor para su enriquecimiento y se le invita a realizarlas a continuación: _____

_____

Anexo 2: Acta de Liberación Interna de Productos de Software.

UCI Universidad de las Ciencias Informáticas | FACULTAD # 3
CENTRO DE GOBIERNO ELECTRÓNICO

CEGEL

Acta de Liberación Interna de Productos Software

Fecha de emisión del acta: 22/06/2015

Emitida a favor de: Sistema de gestión para la planificación y control del horario docente de la Facultad 3.

Datos del producto

Artefacto	Versión	Estado final	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación
App: *Sistema de gestión para la planificación y control del horario docente de la Facultad 3.*	*1.0*	*0*	*3*	*Evaluación dinámica* *Pruebas de Funcionalidad*	*22/06/2015*

UCI Universidad de las Ciencias Informáticas
Centro de Gobierno Electrónico

Ing. Yordanis Garcia Leiva
Especialista del Grupo Calidad CEGEL

Isis Dalia García Rojas
Autor

1