Universidad de las Ciencias Informáticas

Facultad 3

Título:

Sistema para la generación del horario docente de la facultad de Ingeniería Industrial del ISPJAE

Autores:

Leordan Pérez Martínez
Luis Manuel Diaz Morejón

Tutor:

Ing. Isabel González Flores

Junio del 2014

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 _____.

Leordan Pérez Martínez

_____

Firma

Luis Manuel Díaz Morejón

_____

Firma

Tutor: Isabel González Flores

_____

Firma

Solo hay nada para aquellos que nada saben producir.

Johann Wolfang Goethe
(1749-1832) Escritor alemán

AGRADECIMIENTOS

Un agradecimiento especial a mi madre, mi padre y mi abuela, pues ellos son la razón de mi ser y mi existir, a mi tío Jorge y a su mujer Liuda, a mi tía Eulalia y su marido Alfredo y al resto de mi familia por siempre estar siempre disponibles cada vez que los necesité. A mi compañero de tesis que me ha brindado su amistad durante estos 5 años, a la tutora y a todas mis amistades que me ayudaron a superar esta dura prueba llamada universidad.

A Fidel y a la revolución por darme la oportunidad de estudiar en esta universidad y convertirme en ingeniero.

Luis Manuel

AGRADECIMIENTOS

A mis padres por su amor incondicional, por su esfuerzo y sacrificio para poder llegar hasta aquí y forjarme como profesional. A mi hermano, por darme fuerzas para seguir adelante. A toda mi familia por preocuparse y apoyarme en todo momento. A mi novia por todo su cariño, por su comprensión, por darme confianza justo cuando más lo necesitaba, a mi compañero de tesis por brindarme su amistad durante estos cinco años, a mis amigos gracias por su compañía, por estar en las buenas y en las malas, a mi tutora por haberme guiado con toda su ayuda durante todo el trabajo, a mis profesores y a todas esas personas que de una forma u otra nos dedicaron un minuto de su tiempo e hicieron posible formarme como Ingeniero en Ciencias Informáticas.

Leordan

DEDICATORIA

De Luis

A mis padres

A mis abuelos

A mis hermanos

A mis tíos

De Leordan

A mis padres

A mi hermano y a mi familia

RESUMEN

En este trabajo se propone una solución al problema de planificación del horario docente de la Facultad de Ingeniería Industrial del Instituto Superior Politécnico José Antonio Echeverría. Surge a partir de la necesidad de proporcionar una herramienta que permita la generación y disponibilidad del horario docente en un tiempo razonable, teniendo en cuenta las características y restricciones propias de la institución. La metodología de desarrollo que guía la investigación es la Programación extrema, se escogió el lenguaje y herramientas de programación para darle solución al problema de la investigación. Se realizó la propuesta de solución teniendo en cuenta las restricciones fuertes y débiles acordadas con el cliente. Finalmente se aplicaron pruebas para evaluar la solución tanto a nivel de diseño como de implementación, obteniéndose finalmente un sistema da solución al problema de la investigación.

Palabras clave: Horario docente, planificación de horarios.

TABLA DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

La planificación es una de las operaciones más importantes en la vida de un ser humano, desde sus albores se han usado varios métodos y técnicas con el fin de alcanzar una meta u objetivo determinado. Esta misma situación se aplica a nivel institucional, solo que a una escala mayor, ya que implica el manejo de numerosos recursos. Una institución que no es capaz de planificar cada una de sus actividades y recursos invertidos no logrará el éxito esperado, pues ninguna actividad se sostiene al unísono con las normas de calidad si carece de una planificación minuciosa.

La educación no está exenta a este tipo de operación, planificar y utilizar de manera efectiva los recursos involucrados en determinada actividad, a partir de distintas estrategias para diseñar y desarrollar con éxito la enseñanza, es una de las competencias docentes más importantes en las instituciones educacionales. Dichas instituciones enfrentan cada semestre al problema de la planificación de horario (Timetabling Scheduling) que consiste en la asignación de recursos escasos dentro de un número limitado de períodos de tiempo. Dentro de este contexto, existen tres tipos de problemas conocidos como: programación de horarios de evaluaciones y exámenes (Examination Timetabling), programación de horarios de clases para colegios (School Course Timetabling) y programación de horarios de clases para instituciones de educación superior o universidades (University Course Timetabling).

La planificación de un horario docente es un proceso engorroso, por lo que surge la idea de utilizar las Tecnologías de la Información y la Comunicación para su confección. Se han usado varios métodos (coloreo de grafos, recocido simulado, algoritmos genéticos, entre otros) que aplicados en un software son capaces de resolver el Problema de Planificación de Horarios Docentes (PPHD, en adelante), aunque ninguno ha logrado realizar una planificación que se adapte a todo tipo de institución, pues como plantea (Flores, y otros, 2010) trabajan sobre la base de un conjunto de características muy específicas, que dependen de cada institución, lo cual obliga a considerar restricciones muy distintas en cada caso. Actualmente existe un grupo de trabajo denominado: Working Group on Automated Timetabling (WATT), de la Universidad de Nottingham, el cual realiza la Conferencia Internacional: Practice and Theory of Automated Timetabling, donde investigadores ponen a prueba sus algoritmos para resolver problemas de prueba.

Existen distintos software con el fin de generar o asistir la creación de un horario docente, en las universidades de nuestro país también se han puesto a prueba soluciones de acuerdo a las condiciones y características propias.

La facultad de Ingeniería Industrial del Instituto Superior Politécnico José Antonio Echeverría (ISPJAE, en adelante) afronta cada semestre el problema de la planificación del horario docente. A pesar de que el programa de estudios no varía significativamente no se reutiliza el horario docente de cursos anteriores, pues la matrícula y cantidad de grupos varía de un curso a otro. Una situación similar ocurre con los profesores disponibles, en menor medida, muchos de ellos prestan servicios a la universidad y sus afectaciones también varían. Esta tarea se realiza de forma manual, implica el manejo de grandes volúmenes de datos, son numerosas las restricciones a cumplir, suele demorar más de cuatro semanas, a pesar de los años de experiencia de la planificadora, siendo un proceso lento, complejo y monótono.

Una vez que la planificadora confecciona el horario lo digitaliza mediante un asistente de planificación denominado Sistema de Automatizado de Planificación Docente (S.A.P.Doc), desarrollado en MS-DOS, que le permite identificar colisiones en el horario, sin embargo no propone alternativas de solución.

El horario no se publica en el sitio web de la facultad, en su lugar se imprime una copia a todos los profesores y jefes de grupo antes de comenzar el semestre, implicando que en reiteradas ocasiones las personas necesitadas de realizar una consulta del mismo se vean obligadas a dirigirse a quienes poseen las copias o a la planificadora.

A partir de la situación problemática, se enuncia como **problema a resolver**: ¿cómo planificar el horario docente de la facultad de Ingeniería Industrial del ISPJAE de forma que disminuya el tiempo invertido en su confección y propicie la disponibilidad de la información?

Se define como **objeto de estudio** el desarrollo de software para la planificación de horarios docentes. El **campo de acción** se enmarca en la informatización de la planificación del horario docente en la facultad de Ingeniería Industrial del ISPJAE.

El **objetivo general** de la investigación es desarrollar un sistema que permita la generación del horario docente de la facultad de Ingeniería Industrial del ISPJAE disminuyendo el tiempo invertido en su confección y propiciando la disponibilidad de la información.

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

- Elaborar el marco teórico de la investigación.

- Diseñar el sistema de generación del horario docente de la facultad de Ingeniería Industrial del ISPJAE.
- Implementar el sistema de generación del horario docente de la facultad de Ingeniería Industrial del ISPJAE.
- Valorar la efectividad de la solución propuesta.

La **idea a defender** en la investigación se define como: El desarrollo de un sistema que permita la generación del horario docente de la facultad de Ingeniería Industrial del ISPJAE disminuirá el tiempo invertido en su confección y propiciará la disponibilidad de la información.

El trabajo se encuentra estructurado en 3 capítulos:

- **Capítulo 1: Fundamentación teórica**, se presentan los principales conceptos, teorías, métodos y ejemplos de aplicaciones existentes relacionadas con el PPHD. Se realiza el análisis de la metodología de desarrollo de software a usar, así como el lenguaje de programación y las herramientas apropiadas para darle solución al problema a resolver.
- **Capítulo 2: Propuesta de solución**, se presentan las principales características de la solución al problema de la investigación: restricciones fuertes y débiles, funcionalidades del sistema y arquitectura. Se describe la solución y su eficiencia a partir del cálculo de la complejidad temporal.
- **Capítulo 3: Validación de la solución**, se presentan las pruebas hechas al software: pruebas de diseño a partir de las métricas TOC y RC y las pruebas de implementación compuestas por las pruebas de caja blanca y caja negra.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se presentan aquellos conceptos y teorías necesarias para la comprensión del PPHD, así como ejemplos de sistemas informáticos aplicados en el contexto internacional y nacional, además se fundamenta la tecnología que se utilizará para darle solución al problema de la investigación.

1.2 Definición de términos

A continuación se establece una serie de términos utilizados en la Facultad de Ingeniería Industrial del ISPJAE, los mismos facilitan la comprensión de la naturaleza del PPHD:

- **Actividad:** turno de clase de una asignatura impartido por un profesor a uno, dos o tres grupos en un local determinado.
- **Asignatura**: cada uno de los tratados o materias que se enseñan en un instituto docente, o forman un plan académico de estudios (RAE, 2013).
- **Curso**: tiempo señalado de cada año de la carrera, distribuido en dos semestres.
- **Grupo**: conjunto de estudiantes que pertenecen al mismo año e intervienen en las mismas actividades.
- **Local**: lugar destinado a desarrollar una actividad docente.
- **Profesor**: persona que ejerce o enseña una asignatura.
- **Tipo de frecuencia**: tipo de actividad impartida, puede ser conferencia, clase práctica, seminario, laboratorio o taller.

1.3 Problema de Planificación de Horarios Docentes

El PPHD consiste en programar una secuencia de clases entre profesores y estudiantes en un prefijado período de tiempo (típicamente una semana), satisfaciendo un conjunto de restricciones de varios tipos (Schaerf, 1999). En un horario docente no pueden existir actividades simultáneas en un mismo local, además un profesor o grupo tampoco deben tener actividades simultáneas, la ocurrencia de alguna de las situaciones mencionadas da lugar a las llamadas colisiones.

Las restricciones del PPHD se clasifican en dos grupos, restricciones fuertes y débiles (Lucero, 2007):

- **Restricciones fuertes u obligatorias:** son aquellas que definen la factibilidad o usabilidad del horario; para poder ser usado un horario debe satisfacer todas las restricciones fuertes.
- **Restricciones débiles o deseables:** son aquellas que definen la calidad del horario. Un horario podría violar estas restricciones, pero debería hacerlo en la menor medida posible.

Dentro de este contexto se pueden evidenciar tres tipos de problemas, cada uno con sus propias restricciones (Hernández, y otros, 2008):

- Programación de horarios de evaluaciones y exámenes (Examination Timetabling): se refiere a la programación exclusiva de un horario para las distintas evaluaciones y exámenes que pueda tener un curso, determinado por la cantidad de asignaturas, locales y el tiempo.
- Programación de horarios de clases para colegios (School Course Timetabling): está orientado a la programación de horarios para instituciones escolares de nivel medio o inferior. El problema consiste en asignar, dado un conjunto de requerimientos (asignaturas, profesores, locales, entre otros), las sesiones y recursos a los períodos de tiempo.
- Programación de horarios de clases para instituciones de educación superior o Universidades (University Course Timetabling): sigue la misma explicación anterior, la diferencia radica principalmente en torno a los estudiantes y a los profesores. Los estudiantes en los colegios reciben las mismas asignaturas, mientras que en las universidades cursan diferentes a parte de las básicas que tienen todos. Los profesores en los colegios pueden impartir todas las asignaturas, mientras que en las universidades solo un máximo de 3.

Este último es el que más se ajusta a la investigación, pues la solución está enfocada a un centro universitario, donde los profesores no imparten todas las asignaturas, además se planificará un semestre del curso en lugar del curso completo.

1.4 Métodos de solución

El PPHD se puede resolver aplicando diversos enfoques de solución, tanto los ofrecidos por el área de Investigación de Operaciones como por los de Inteligencia Artificial

(Araya, 2007). Para darle solución al PPHD se han usado disímiles métodos, entre estos están:

- **Coloreo de grafos:** técnica usada para representar la asignación de horarios, donde cada vértice representa una sesión en la asignación de horarios y las aristas son creadas entre cada par de vértices de cursos que no pueden ser programados a la misma hora. La indisponibilidad y las pre-asignaciones son manejados imponiendo alguna restricción externa sobre la viabilidad de coloreo de un vértice especifico. La coloración del grafo resultante puede fácilmente volverse una asignación de horarios práctica asociando cada período con un color. El problema de colorear los vértices de un grafo con un mínimo número de colores es uno de los más difíciles de resolver entre los problemas de optimización combinatoria, debido a que pertenece a la clase de problemas NP Completo. (Pecina Federico, y otros, 2009).
- **Recocido simulado:** algoritmo de búsqueda meta-heurística para problemas de optimización global; conocido también como enfriamiento simulado, cuyo objetivo general es encontrar una buena aproximación al valor óptimo de una función en un espacio de búsqueda grande. A este valor óptimo se lo denomina "óptimo global" (Gutiérrez Andrade, y otros, 1998). Mediante el uso de este método se pueden obtener buenas soluciones para instancias del problema de planificación de horarios de tamaño medio, implicando que en última instancia llegara a ellas después de un número infinito de iteraciones, para lo cual se exigiría un alto grado de procesamiento de información (Govea, 2012).
- **Algoritmos genéticos (AG):** consisten en hallar de qué parámetros depende el problema, codificarlos en un cromosoma, y se aplican los métodos de la evolución: selección y reproducción sexual con intercambio de información y alteraciones que generan diversidad (Merelo, 2004). Este método es una buena opción para afrontar problemas de planificación de horarios aunque no carece de detractores por su carácter estocástico y tiempos elevados de ejecución.
- **Programación de restricciones:** se define como el estudio de sistemas computacionales basados en restricciones. La idea de la programación de restricciones es resolver problemas mediante la declaración de restricciones sobre el dominio del problema y consecuentemente encontrar soluciones a instancias de los problemas de dicho dominio que satisfagan todas las restricciones y, en su caso,

optimicen unos criterios determinados (Manyá, y otros, 2003). Estos pueden formalizarse como problemas de satisfacción de restricciones (Constraint Satisfaction Problems) y resolverse usando técnicas de satisfacción de restricciones. Este método genérico de resolución de problemas ha demostrado ser altamente competitivo en áreas tan diversas como inteligencia artificial, investigación operativa, bases de datos y sistemas de recuperación de la información (Manyá, y otros, 2003).

Teniendo en cuenta las características de la facultad de Ingeniería Industrial del IPSJAE y que la solución está encaminada a obtener una solución factible, sin llegar a su optimización, se opta por la programación basada en restricciones. Al estar las restricciones en la naturaleza de los problemas de planificación de horarios hace que modelar las instancias de un problema se vea muy natural, permitiendo satisfacer todas las restricciones fuertes y cumplir en la mayor medida posible las restricciones débiles.

1.5 Soluciones informáticas existentes

En función de facilitar la creación del horario en instituciones de educación superior se han creado disímiles soluciones informáticas, cada una representa una forma única de planificar el horario, convirtiéndose en un tema de gran interés por la comunidad científica. A continuación se muestran algunos los siguientes ejemplos internacionales y nacionales.

1.5.1 Internacionales

Generador de Horarios para Universidades: creado por la compañía española Peñalara Software, su objetivo fundamental es acoplar los horarios semanales de las facultades universitarias, observando las condiciones específicas y criterios de optimización propios de las enseñanzas superiores (Peñalara, 2013). Consta de varios procesos que se ejecutan en el sistema de forma independiente:

- Planificador: configura los datos iniciales.
- Motor: genera y optimiza los resultados.
- Editor: presenta y edita los resultados obtenidos.

Ventajas

- Permite exportar el horario resuelto a otras aplicaciones de gestión académica.
- Posibilita la presentación de horarios en formato PDF.

- Muestra los mensajes de validación de los horarios en forma de árbol para facilitar su manejo.
- Posee una estructura de datos formada por listas relacionadas que garantiza la integridad de la información.
- Posibilita la configuración de los conjuntos de asignaturas optativas.

Desventajas

- Es un software privativo.

KRONOWIN: creado por la empresa ADOSSIS, S.A. Sistemas Informáticos, es un sistema generador de horarios escolares que permite obtener los horarios de un centro escolar, partiendo de las asignaturas, los grupos, los profesores y las aulas. La última versión es Kronowin M-13.04, incorpora un programa opcional para la organización/generación de grupos escolares: KGRUPOS, este permite recoger los datos necesarios de la gestión de centros para asignar grupo a cada alumno (Adosis, 2013).

Ventajas

- Utiliza dos motores de generación distintos: motor convencional y motor avanzado parametrizable.
- Acelera el proceso de generación avanzada, intentando evitar el cálculo repetitivo de combinaciones previamente rechazadas por el propio algoritmo.
- Exporta e importa a distintos formatos, siendo estos configurables.

Desventajas

- Es un software privativo.

Untis Express: software generador de horarios para colegios y universidades creado por la compañía Untis Gruber & Petters, Suiza. Posee un asistente que guía al usuario a introducir los datos básicos del centro: profesores, grupos, asignaturas, grupos, aulas, instalaciones, entre otros (Petters, 2013).

Ventajas

- Permite introducir restricciones, como por ejemplo, que nunca se asigne una asignatura en un determinado horario, permitiendo así adaptar la aplicación a las necesidades concretas del planificador.
- Incorpora la ponderación pedagógica de las variables que se introduzcan y proporciona varias propuestas de horarios para el gusto del planificador.

- Permite que varias personas trabajen paralelamente con el programa en la versión "multiusuario".
- Brinda varios formatos para exportar el horario, posibilitando imprimirlo de manera individual, de sobrevista o publicarlo en una página web.

Desventajas

- Es un software privativo.

1.5.2 Nacionales

Horr: sistema de planificación y publicación de horarios docentes usado en la Universidad de las Ciencias Informáticas, está compuesto por una aplicación de escritorio (HorrPlanner) orientada a los planificadores y una aplicación web (HorrPublisher) orientada a los usuarios interesados en consultar la planificación (Barrera, 2011).

Ventajas

- Capaz de exportar a un documento Word (.doc) el horario planificado.
- Posibilidad de publicar el horario en una página web.

Desventajas

- Es un asistente, no es capaz de generar el horario.
- No verifica si la actividad que se va a realizar esta acorde al local que se le asigna.
- No presenta una buena estructura organizacional en cuanto a la clasificación de locales, actividades y asignación de profesores.

Sistema de la antigua Facultad Regional de la Universidad de las Ciencias Informáticas de Granma: el algoritmo genético se implementó en el lenguaje de programación Java acorde a las necesidades de la institución (Varona, 2012).

Ventaja

- Capaz de generar el horario acorde a las expectativas propias de la institución.

Desventajas

- Solución hecha a la medida que no responde a las necesidades de la facultad de Ingeniería Industrial.

Sistema de Gestión del Proceso Docente: herramienta web de la Universidad de Pinar del Río “Hermanos Saiz Montes de Oca”, encargada de digitalizar y controlar la planificación del horario docente.

Ventajas

- Multiplataforma.
- Posee las funcionalidades de gestionar los datos de aulas, asignaturas y profesores.
- Posee una funcionalidad para la búsqueda de turnos.

Desventaja

- Solo permite la publicación del horario, ya que la confección se realiza de forma manual.

Después de analizar las ventajas y desventajas identificadas de diferentes soluciones informáticas, se puede afirmar que ninguno cumple con los requerimientos necesarios para ser usado como solución al PPHD que presenta la facultad de Ingeniería Industrial del ISPJAE.

- Generador de Horarios para Universidades, KRONOWIN y Untis Express son privativos y su uso implica el pago del software o de una licencia para mantener el uso del mismo.
- Horr y el Sistema de Gestión del Proceso Docente son asistentes de planificación, no generan el horario.
- Los algoritmos genéticos aplicados en la antigua Facultad Regional de la Universidad de las Ciencias Informáticas de Granma están hecho a la medida y fueron implementados de acuerdo a las particularidades de este centro.

Sin embargo pueden usarse como materiales de apoyo en pos del desarrollo de la solución.

1.6 Metodología de desarrollo de software

Las metodologías de desarrollo de software según (Piattini Velthuis, 1996) están definidas como un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software, pero como tal no existe consenso alguno entre los autores sobre el concepto de metodología y por ello no existe una definición general aceptada. Sí hay un acuerdo en considerar a la metodología como un conjunto de pasos y procedimientos que deben seguirse para el desarrollo del software.

Las metodologías de desarrollo de software se pueden agrupar en dos grandes grupos (Piattini Velthuis, 1996):

- **Metodologías tradicionales o pesadas:** hacen mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, estableciendo estrictamente las actividades involucradas, los roles definidos, los artefactos que se deben producir, las herramientas y la documentación usada (RUP (Rational Unified Procces), MSF (Microsoft Solution Framework), Win-Win Spiral Model, Iconix).
- **Metodologías ágiles o ligeras:** orientadas a la generación de código con ciclos muy cortos de desarrollo manteniendo un proceso incremental, son capaces de permitir cambios en los requisitos de último momento, además el equipo de desarrollo mantiene una comunicación constante con el cliente (Extreme Programming (XP), SCRUM, Crystal Clear, Feature-Driven Development (FDD), Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD)).

No se opta por el uso de una metodología tradicional ya que en la investigación se hace mayor énfasis en la obtención del software funcional en poco tiempo, con una realimentación continua entre el equipo de desarrollo y el cliente.

1.6.1 Metodologías ágiles

SCRUM: surgió para el desarrollo de productos tecnológicos, pero también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados sistemas de software (Schwaber, 2013). Al ser una metodología de desarrollo ágil:

- Es un modo de desarrollo de carácter adaptable más que predictivo.
- Está orientado a las personas más que a los procesos.
- Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones.

Las iteraciones en SCRUM son la base del desarrollo ágil y condicionan su evolución a través de reuniones breves diarias en las que todo el equipo revisa el trabajo realizado el día anterior y el previsto para el día siguiente.

Programación extrema (Extreme Programming, XP en adelante): centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software,

promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios (Letelier Torres, y otros, 2003).

Se selecciona la metodología XP debido a que se ajusta en gran medida a las exigencias y escenarios de la investigación, pues el equipo de trabajo está formado por dos desarrolladores y se estará en constante intercambio de información con el cliente, admitiendo en la solución cambios de última hora, siendo el período de trabajo aproximadamente de 6 meses.

1.7 Lenguajes de programación

Los lenguajes de programación son aquellos que a partir de una gramática o conjunto de reglas, crean instrucciones para ser procesadas por una computadora. Estos pueden ser usados para la creación de aplicaciones capaces de controlar el comportamiento lógico y físico de las computadoras. En la actualidad existe una gran variedad de lenguajes, entre las diversas opciones disponibles se analizaron los siguientes como posibles opciones de desarrollo.

Ruby: lenguaje de programación interpretado, reflexivo y orientado a objetos. Diseñado para la productividad y la diversión del desarrollador, siguiendo los principios de una buena interfaz de usuario y su implementación oficial es distribuida bajo una licencia de software libre. Al ser orientado a objeto define como tales a todos los tipos de datos, incluidas las clases y tipos que otros lenguajes definen como primitivas, (como enteros, booleanos, y "null"). Ha sido descrito como un lenguaje de programación multiparadigma: permite programación procedural (definiendo funciones y variables fuera de las clases haciéndolas parte del objeto raíz Object) aunque el tiempo de procesamiento puede llegar a ser demasiado lento (Venners, 2003), debido a esto no se opta por su uso.

Python: en la actualidad se desarrolla como un proyecto de Código abierto. Este permite dividir el programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los

programas. Es utilizado como lenguaje de programación interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del mismo (Martelli, 2008). Python al ser un lenguaje interpretado es más lento que lenguajes compilados o de ensamblador, además conforme se crean aplicaciones más complejas es más complicado escribir el código, así como que posee baja oferta de servicios de alojamiento en la web que lo soportan debido a que es muy complejo implementar esta tecnología en web (Martín García, 2013), no será utilizado, pues la solución requiere de funcionalidades enmarcadas en un entorno web.

Java: lenguaje de programación orientado a objetos. El lenguaje en sí mismo toma mucha de su sintaxis de Lenguaje de Programación C y C++, pero tiene un modelo de objetos más simple (…). Las aplicaciones Java están típicamente compiladas en un bytecode, en el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución. Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es). Otra característica está su independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware y sistema operativo (Arnold, y otros, 1997). Debido al alto consumo de recursos que exige este lenguaje, no se opta por su uso.

Hypertext Pre-processor (inicialmente PHP Tools o Personal Home Page Tools, en lo adelante PHP) es un lenguaje interpretado, multiplataforma y diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+. Publicado bajo la licencia de PHP, la Free Software Foundation (Fundación de Software Libre) considera esta licencia como software libre. (Sæther Bakken, y otros, 2002). El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador (…). Tiene la capacidad de

expandir su potencial utilizando la enorme cantidad de módulos existentes (llamados ext's o extensiones), así como conectarse con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL (Welling, y otros, 2005).

Después del estudio de los lenguajes de programación expuestos y siguiendo los requerimientos con que debe cumplir la aplicación se opta por el uso de la programación web, usando PHP como lenguaje de programación. La posibilidad de que la comunidad universitaria con acceso a la red escolar del ISPJAE pueda consultar los resultados de la aplicación es una ventaja importante que nos ofrece este lenguaje de programación.

1.8 Herramientas

Adobe Dreamweaver: herramienta diseñada para la programación web, permite al usuario utilizar la mayoría de los navegadores web instalados en su ordenador para pre visualizar las páginas web. Algunas de sus características son:

- Posibilidad de búsqueda y reemplazo de líneas de texto y código por cualquier parámetro especificado.
- Permite crear JavaScript básico sin conocimiento del código.
- Permite la conexión a Bases de Datos como MySQL y Microsoft Access, para filtrar y mostrar el contenido utilizando tecnología de script como, por ejemplo, ASP (Active Server Pages), ASP.NET, ColdFusion, JSP (Java Server Pages) y PHP sin necesidad de tener experiencia previa en programación.
- Completamiento de código.

Al ser privativa se descarta como posible herramienta a usar en la implementación de la solución.

PostgreSQL 8.0

Es un sistema gestor de base de datos (SGBD) relacional orientado a objetos y libre, publicado bajo la licencia BSD y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. La aplicación pgAdmin III es el entorno de escritorio visual que permite conectarse a las bases de datos del PostgreSQL que estén ejecutándose en cualquier plataforma. Este facilita la gestión y administración de bases de datos ya sea

mediante instrucciones SQL o con ayuda de un entorno gráfico. Dentro de las características principales del PostgreSQL se encuentran (Martinez, 2010):

- Alta concurrencia: permite que mientras un proceso está escribiendo en una tabla otros accedan a la misma.
- Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.
- Es una base de datos 100% ACID (Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español).
- Copias de seguridad en caliente (Online/hot backups).
- Múltiples métodos de autentificación.
- Acceso encriptado vía SSL.

Debido a sus características y a que es un software libre se opta por su uso.

Visual Paradigm 8.0

Herramienta para desarrollo de aplicaciones utilizando modelado UML, ideal para quienes están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Constituye una herramienta privada pero en la universidad de las ciencias informáticas, centro donde estudian los autores de la investigación, se tiene una licencia que es usada con fines educativos. Dentro de sus características se tiene:

- Modelo y código permanecen sincronizados en todo el ciclo de desarrollo.
- Interoperabilidad con modelos UML.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.

Debido a la experiencia en el uso de esta herramienta CASE en los proyectos productivos se opta por su utilización.

NetBeans IDE 8.0

Permite programar en distintos lenguajes en los cuales incluye PHP, HTML, CSS y JavaScript, además presenta una serie de características que facilitan en gran parte el desarrollo:

- Multiplataforma.
- Historial de cambios.
- Facilidades para la programación:
    - completamiento de código
    - comprobación y corrección de errores en tiempo real
    - resaltado de variables o etiquetas seleccionadas

Se opta por esta última herramienta pues es un producto libre, gratuito, sin restricciones de uso y se ajusta a las necesidades de la investigación.

1.9 Conclusiones parciales

A partir de la investigación realizada se evidencia la necesidad de implementar un sistema para la generación del horario docente de la facultad de Ingeniería Industrial del ISPJAE, pues los sistemas existentes no se ajustan a los requerimientos de la institución. Se seleccionó a XP como metodología de desarrollo de software, PHP 5 como lenguaje de programación, Visual Paradigm 8.0 para el modelado del sistema, PostgreSQL 8.0 como SGBD y el NetBeans 8.0 como herramienta para la implementación de la solución.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1 Introducción

En este capítulo se presentan los elementos fundamentales de la propuesta de solución de acuerdo a la metodología de software seleccionada: restricciones fuertes y débiles, funcionalidades del sistema, arquitectura y patrones diseño a usar en el sistema.

2.2 Restricciones del sistema

En el capítulo anterior presentaron las clasificaciones de las restricciones del PPHD, a continuación se muestran las que se tienen en cuenta en la solución, las mismas fueron conciliadas con el cliente.

Restricciones fuertes

- No puede haber clases simultáneas en un mismo local.
- No puede haber clases simultáneas asignadas a un mismo grupo.
- No puede haber clases simultáneas asignadas a un mismo profesor.
- Una asignatura no puede tener más de una frecuencia de clase un mismo día.
- No se puede asignar clases consecutivas en el tercer y cuarto turno.
- No planificar clases a profesores, grupos o en locales los días/turnos con afectaciones previamente declaradas.
- El horario debe ser compacto, no debe dejar turnos intermedios sin planificar.

Restricciones débiles

- Un grupo no puede tener clases en un local que tenga una capacidad inferior a su matrícula.
- La asignatura Educación física no puede tener clase que la preceda o la suceda.
- Después de una clase de tipo Conferencia se debe dejar, al menos, un día de diferencia con relación al próximo turno de clase.
- Los grupos deben tener clases en un mismo local en dependencia al tipo de clase.
- A los profesores que imparten clase a más de un grupo de una misma asignatura se les debe planificar clases en turnos consecutivos de un mismo día.
- Las clases de laboratorio si no pueden planificarse en la sección de clases que corresponde se planificarán en la sección contraria.

2.3 Abreviaturas del sistema

A continuación se presentan las abreviaturas utilizadas en el sistema.

Asignaturas de primer año			
Semestre 1		Semestre 2	
Matemática I	M1	Matemática II	M2
Algebra lineal y geometría analítica	AL	Economía política del capitalismo	EPC
Filosofía y sociedad	FS	Física I	F1
Introducción a la informática	IF	Química	Q
Dibujo básico	DB	Dibujo aplicado	DA
Idioma Inglés I	I1	Idioma Inglés II	I2
Historia de Cuba	H	Introducción a la ingeniería industrial	II
Introducción a la ingeniería	II	Educación física II	EF
Educación física I	EF		

Tabla 1. Abreviatura de las asignaturas de primer año

Asignaturas de segundo año			
Semestre 1		Semestre 2	
Matemática III	M3	Matemática IV	M4
Economía política de la construcción del socialismo	EPS	Teoría sociopolítica	TSP
Programación I	P1	Base de datos	BD
Física II	F2	Física III	F3
Idioma Inglés III	I3	Idioma Inglés IV	I4
Modelos probabilísticos procesos	MP	Modelos estadísticos de los procesos I	MP1
Seguridad nacional	SN	Defensa nacional	DN
Educación física III	EF	Procesos tecnológicos I	PT1
		Educación física IV	EF

Tabla 2. Abreviatura de las asignaturas de segundo año

Asignaturas de tercer año			
Semestre 1		Semestre 2	
Problemas sociales de la ciencia y la tecnología	PCT	Estudio de tiempos de trabajo	ETT
Modelos estadísticos de los procesos II	MEP	Gestión de la información	GI
Gestión económica financiara	GEF	Metodología de proyectos de investigación en ingeniería industrial	MMI
Ingeniería de métodos	IM	Procesos tecnológicos III	PT3
Ergonomía	ERG	Análisis económico	AE
Tecnología de la información	TIF	Investigación de operaciones I	IO1
Procesos tecnológicos II	PT2	Seguridad y salud en el trabajo	SST

Tabla 3. Abreviatura de las asignaturas de tercer año

Asignaturas de cuarto año			
Semestre 1		Semestre 2	
Investigación de operaciones II	IO2	Simulación de procesos	SP
Gestión organizacional	GO	Ingeniería de la calidad	IC
Gestión de recursos humanos	GRH	Logística I	L1
Gestión de procesos I	GP1	Gestión de procesos II	GP2
Pedagogía	PG	Gestión comercial	GC
Sistema de información	SI		
Procesos tecnológicos IV	PT4		

Tabla 4. Abreviatura de las asignaturas de cuarto año

Asignaturas de quinto año	
Semestre 1	
Gestión del cambio organizacional	GCO
Gestión de la calidad	GC
Logística II	L2
Distribución en planta	DP
Gestión ambiental	GA

Tabla 5. Abreviatura de las asignaturas de quinto año

Tipo de frecuencia	
Conferencia	C
Clase práctica	CP
Seminario	S
Laboratorio	L
Taller	T
Prueba inter-semestral	E

Tabla 6. Frecuencia de actividades

Locales	
Aula	A
Aula de conferencia	AC
Aula especializada	AE
Laboratorio	L
Taller	T

Tabla 7. Clasificación de los locales

2.4 Fase exploración

Según (Letelier Torres, y otros, 2003) en esta fase los clientes presentan las historias de usuario que son de interés, al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.

2.4.1 Historia de usuario

Una historia de usuario (HU, en adelante) describe una funcionalidad que realizará el sistema y que aportará valor al cliente. Deben ser escritas en un lenguaje no técnico de manera que puedan ser fácilmente comprendidas (Scott, 2009). El tratamiento de las HU es muy dinámico y flexible, en cualquier momento HU pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan

implementarla en unas semanas (Jeffries, y otros, 2001). A continuación se describe la estructura de una HU, las restantes se encuentran en el Anexo 1.

Historia de usuario	
Número: 5	**Nombre:** Gestionar profesor
Usuario: planificador	**Iteración asignada:** 1
Prioridad en negocio: alta	**Puntos estimados:** 1
Riesgo en desarrollo: medio	**Puntos reales:** 1
Descripción: El sistema debe permitir al usuario adicionar, editar, eliminar y listar los profesores, gestionar sus afectaciones, así como adicionar y eliminar una relación entre el profesor, la asignatura y el tipo de encuentro que imparte a uno o varios grupos.	
Observaciones: Cada profesor tendrá asignado una clasificación y acorde a ella puede tener o no una afectación.	

Tabla 8. HU Gestionar profesor

A continuación se explica cada uno de los datos que deben ser llenados en una HU:

- **Número:** identificador de la HU.
- **Nombre:** nombre que identifica a la HU.
- **Usuario:** involucrados en la ejecución de la HU.
- **Iteración asignada:** iteración en que se implementará la HU
- **Prioridad en el negocio:** prioridad de la HU respecto al resto de las HU, puede ser: alta, media o baja.
- **Riesgo en desarrollo:** riesgo en la implementación de la HU, puede ser: alto, medio o bajo.
- **Puntos estimados:** estima el esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, generalmente de 1 a 3 puntos.
- **Puntos reales:** resultado del esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, generalmente de 1 a 3 puntos.
- **Descripción:** descripción sintetizada de la HU.
- **Observaciones:** información de interés.

2.4.2 Personas relacionadas con el sistema

A continuación se muestran las personas relacionadas con el sistema, aquellas que de una forma u otra interactúan con el sistema.

Persona	Descripción
Usuario	Persona que accede al sistema, sin tener que autenticarse, solamente para consultar el horario docente.
Planificador	Persona que ha sido autenticada con la posibilidad de acceder las funcionalidades del sistema.

Tabla 9. Personas relacionadas con el sistema

2.4.3 Requisitos no funcionales del sistema

Los requisitos no funcionales pueden verse como las propiedades o cualidades que un producto debe tener. La metodología XP no incluye la descripción de estos requisitos en las HU, debido a que los clientes generalmente desconocen las terminologías técnicas aplicadas en la descripción de los mismos. El equipo de desarrollo es el responsable de la captura de estos requisitos a partir del intercambio con el cliente, con el objetivo de obtener una solución que cumpla con determinados estándares de calidad y con las características propias del negocio a informatizar. A continuación se presentan los requisitos no funcionales identificados:

- **Requisitos de usabilidad:** el sistema será usado por personas que tengan conocimientos básicos de informática, la interfaz debe ser amigable y fácil de operar. La tipografía debe ser uniforme, de un tamaño adecuado y con un contraste que resalte los textos.
- **Requisitos de rendimiento:** El sistema deberá tener un tiempo máximo de 10 minutos para la generación del horario y 5 segundos para cualquier operación de consulta.
- **Requisitos de portabilidad:** las herramientas podrán ser usadas bajo cualquier sistema operativo de Windows NT en adelante o cualquier distribución de Linux.
- **Requisitos de seguridad:**
    - **Autenticación:** se establecerán mecanismos de autenticación personalizada para la(s) persona(s) involucradas con la planificación del horario docente.
    - **Contraseñas:** se deben cifrar las contraseñas. La longitud mínima debe ser de 8 caracteres y debe contener combinaciones de letras minúsculas, letras mayúsculas, números y/o caracteres especiales.

- **Confidencialidad**: para acceder a las funcionalidades del sistema el usuario debe estar correctamente autenticado.
- **Integridad:** los datos almacenados en la base de datos no pueden presentar incoherencias.
- **Disponibilidad:** el sistema debe estar disponible en todo momento desde todos los puntos donde exista una máquina conectada a la red.

- **Requisitos de software:** necesidad de un navegador con soporte para HTML5 y CSS3.
- **Requisitos de hardware:**
  - Memoria RAM: 512Mb
  - Microprocesador: 1.70GHz
  - Conexión a la red escolar

2.5 Modelo de datos

El modelo de datos permite describir la estructura lógica y física de la información persistente que gestiona el sistema. Contiene las relaciones de las tablas que intervienen en la generación del horario docente, siendo la tabla “encuentro” la principal, ya que es la encargada de almacenar el horario en forma de encuentros. Se hace el uso de tablas que representan relaciones entre los elementos de la base de datos para obtener dinamismo en la aplicación, se usan llaves subrogadas para una mejor manipulación de los datos y se utilizan tablas denominadas nomencladores para almacenar datos. A continuación se muestra el diagrama.

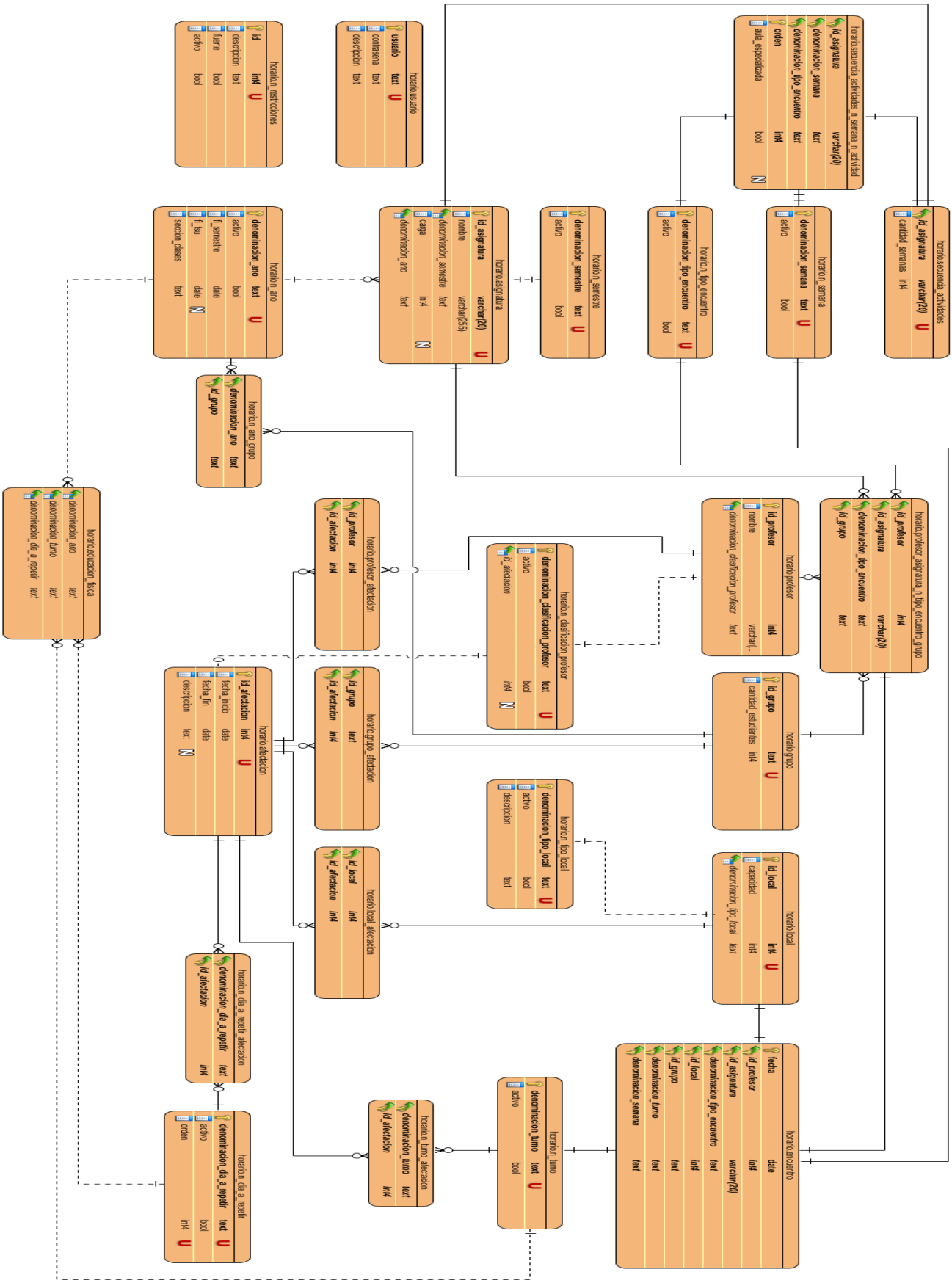

Figura 1. Diagrama Entidad-Relación

2.6 Patrones de diseño

El 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 […] que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades (Larman, 1999).

A continuación se describen los patrones generales de software para asignar responsabilidades, por su acrónimo en Inglés: GRASP (General Responsibility Asignment Software Patterns) utilizados en el diseño.

Experto: el experto en la información es la clase que tiene la información necesaria para cumplir la responsabilidad, expresa que los objetos deben hacer las cosas relacionadas con la información que poseen. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase "sencillas" y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión (Visconti, y otros, 2006). A continuación se muestra un ejemplo de cómo se manifiesta en el sistema.

Este patrón sugiere que las clases grupo, afectación y grupo_afectación tienen la responsabilidad de contestar la cantidad de estudiantes que están afectados en una fecha dada.

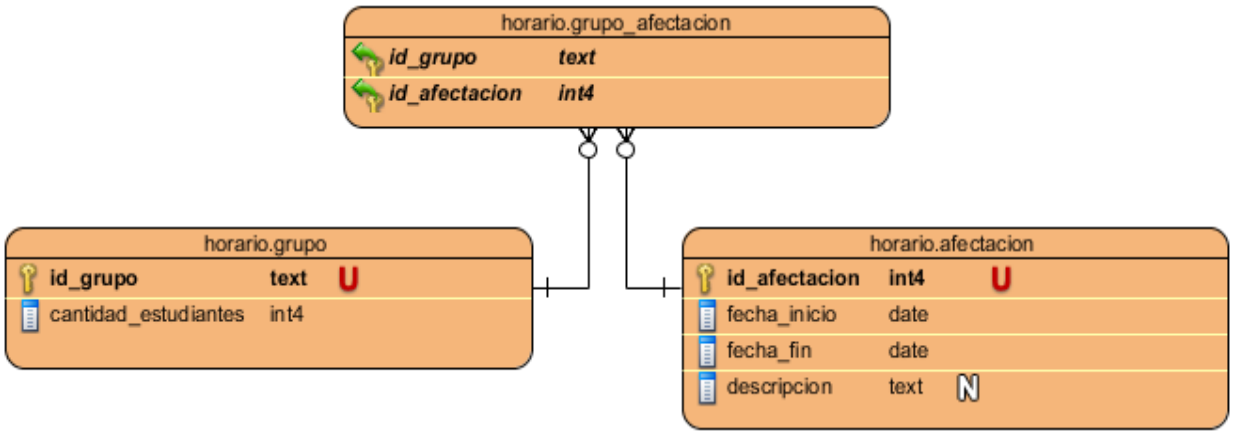

Figura 2. Uso del patrón experto

Creador: sugiere quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases. El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilizabilidad (Larman, 1999). A continuación se muestra un ejemplo de cómo se manifiesta en el sistema.

Este patrón sugiere que las clases n_dia_a_repetir, n_ano y n_turno son adecuadas para crear la instancia educacion_fisica encargada de registrar los turnos de educación física de un año determinado (educacion_fisica es un objeto agregado de n_dia_a_repetir, n_ano y n_turno).

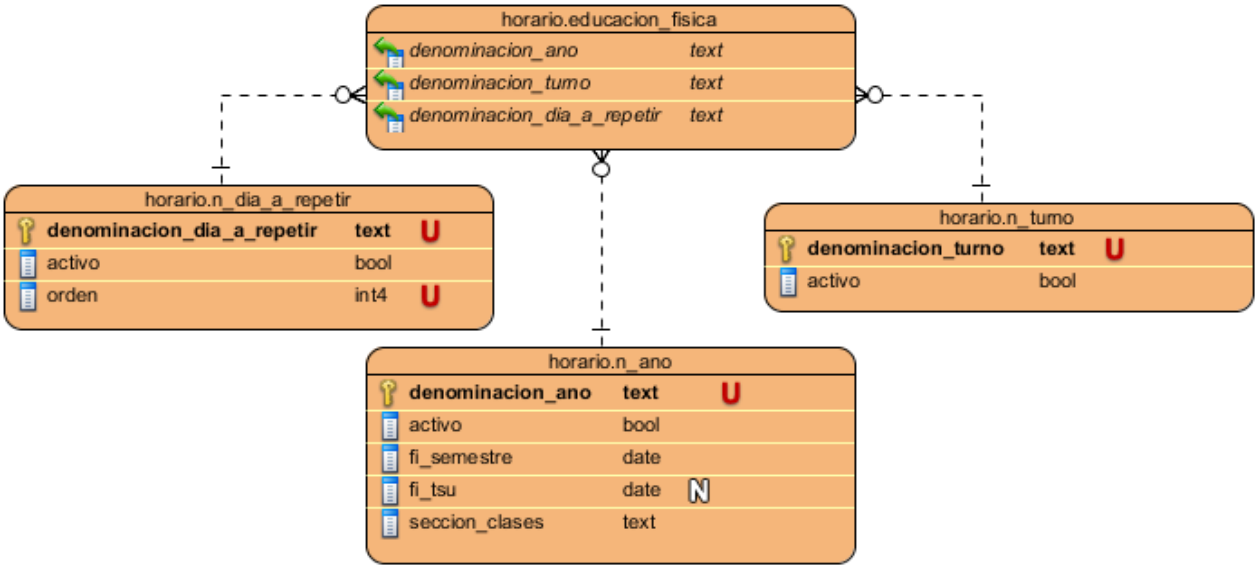

Figura 3. Uso del patrón creador

Controlador: es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación (Visconti, y otros, 2006). Los objetos externos de conexión y la capa de presentación no deberían tener la responsabilidad de llevar a cabo los eventos del sistema (Larman, 1999).

Como ejemplo de su uso en el sistema, el patrón Controlador sugiere que la clase ControladorProfesor es la encargada de atender los eventos referidos a la gestión de los datos del profesor.

2.7 Fase planificación de la entrega

En esta fase el cliente establece la prioridad de cada HU, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días (Letelier Torres, et al., 2003).

No	Historia de usuario	Estimación (semana)
1.	Autenticar usuario	⅓
2.	Gestionar año	⅓
3.	Activar o desactivar: semestres, semanas, turnos, tipo de encuentros, días, tipo de locales, restricciones fuertes y débiles.	½
4.	Gestionar clasificación del profesor	½
5.	Gestionar profesor	1
6.	Gestionar grupo	⅓
7.	Gestionar asignatura	2
8.	Gestionar local	½
9.	Gestionar fechas especiales	½
10.	Generar horario	3
11.	Visualizar el horario	1½
12.	Devolver la cantidad de restricciones débiles incumplidas.	½

Tabla 10. Plan de duración de las iteraciones

2.8 Fase iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción (Beck, 1999).

2.8.1 Plan de iteraciones

En este plan se especifica las HU que serán implementadas en cada iteración del sistema. Se concretaron tres iteraciones para la realización del sistema:

- Iteración 1: tiene como objetivo la implementación de las HU: 1, 2, 3, 4, 5 y 6 sirven como base para la generación del horario.
- Iteración 2: tiene como objetivo la implementación de las HU: 7, 8, y 9, sirven como base para la generación del horario.
- Iteración 3: tiene como objetivo la implementación de la HU: 10, encargada la creación del horario en función de los datos gestionados con anterioridad.
- Iteración 4: tiene como objetivo la implementación de la HU: 11 y 12, encargadas de mostrar el horario generado y devolver la cantidad de restricciones débiles incumplidas.

2.8.2 Plan de entregas

Iteración	Historia de usuario	Estimación (semana)	Fecha inicio - fin
1.	• Autenticar usuario • Activar o desactivar: semestres, semanas, turnos, tipo de encuentros, días, tipo de locales, restricciones fuertes y débiles • Gestionar año • Gestionar clasificación de profesor • Gestionar profesor • Gestionar grupo	3	10/03/2014 - 30/04/2014
2.	• Gestionar asignatura • Gestionar local • Gestionar fechas especiales	3	31/03/2014 - 20/04/2014
3.	• Generar horario	3	21/04/2014 - 11/05/2014
4.	• Crear filtros para visualizar el horario • Devolver la cantidad de restricciones débiles incumplidas.	2	12/05/2014 - 25/05/2014

Tabla 11. Plan de entregas

2.9 Arquitectura

El desarrollo de la solución presenta una arquitectura en capas, basada en el patrón arquitectónico Modelo-Vista-Controlador (MVC), uno de los más utilizados en las aplicaciones web, propicia el aislamiento entre el modelado del dominio, el control de los

eventos del sistema y las interfaces de la aplicación, cuya principal ventaja es que en caso de existir la necesidad de un cambio, solo se modifica la capa en cuestión y no se afecta el funcionamiento del resto del sistema.

- **Modelo:** encargado de la gestión de la base de datos. Los elementos manejados son:
  - **modelo_profesor.php:** gestiona los profesores.
  - **modelo_profesor_afectacion.php:** gestiona las afectaciones de los profesores.
  - **modelo_grupo.php:** gestiona de los grupos.
  - **modelo_grupo_afectacion.php:** gestiona las afectaciones de los grupos.
  - **modelo_local.php:** gestiona de los locales.
  - **modelo_horario.php:** gestiona las restricciones para la planificación del horario.
  - **modelo_local_afectacion.php:** gestiona las afectaciones de los locales.
  - **modelo_afectacion.php:** gestiona de manera general todas las afectaciones que maneja la aplicación.
  - **modelo_asignatura.php:** gestiona las asignaturas.
  - **modelo_clasificacion_profesor_afectacion.php:** gestiona las clasificaciones de los profesores, incluyendo sus afectaciones.
  - **modelo_profesor_grupo_asignatura_TE.php:** gestiona las relaciones entre datos de los profesores, los grupos, las asignaturas y los tipos de encuentro.
  - **modelo_filtro.php:** gestiona las peticiones de búsqueda del horario docente generado.
  - **modelo_fecha_especial.php:** gestiona las fechas especiales que intervienen en la generación del horario, dígase fin de año, semana de la victoria, entre otras.
  - **modelo_nomenclador.php:** gestiona los nomencladores.
- **Vista:** Presenta todas las interfaces de la aplicación, con las cuales el usuario gestionará visualmente los datos referentes a los profesores, locales, asignaturas, grupos, su relación, su configuración y finalmente la generación del horario. Las vistas son:
  - **vista_bienvenida.php:** se muestran los datos manejados por el sistema.
  - **vista_profesores.php:** se listan, adicionan y eliminan los profesores.

- **vista_profesor_especificacion.php:** se modifica el profesor seleccionado y se listan y gestionan las afectaciones de dicho profesor.
- **vista_secuencia_actividades.php:** se modifica la secuencia de actividades de la asignatura seleccionada.
- **vista_grupos.php:** se listan, adicionan y eliminan los grupos.
- **vista_grupo_especificacion.php:** se modifica el grupo seleccionado y se listan y gestionan las afectaciones de dicho grupo.
- **vista_locales.php:** se listan, adicionan y eliminan los locales.
- **vista_local_especificacion.php:** se modifica el local seleccionado y se listan y gestionan las afectaciones de dicho local.
- **vista_asignatura [1, 2, 3, 4, 5].php:** se listan, adicionan y eliminan las asignaturas pertenecientes al año seleccionado.
- **vista_horario.php:** se lista el horario generado a partir de distintos criterios de búsqueda.
- **vista_planificar_EF.php:** se listan y gestionan los turnos de educación física del año seleccionado.
- **vista_configuracion.php:** lista y activa/desactiva los semestres.
- **vista_configuracion_semanas.php:** lista y activa/desactiva las semanas.
- **vista_configuracion_turnos.php:** lista y activa/desactiva los turnos.
- **vista_configuracion_encuentros.php:** lista y activa/desactiva los encuentros.
- **vista_configuracion_dias.php:** lista y activa/desactiva los días.
- **vista_configuracion_fe.php:** lista y activa/desactiva las fechas especiales.
- **vista_configuracion_anos.php:** lista y modifica los años.
- **vista_configuracion_locales.php:** lista y activa/desactiva los locales.
- **vista_configuracion_afectacion.php:** lista y gestiona los tipos de profesores.
- **vista_configuracion_afectacion_clasificacion_profesor.php:** gestiona la afectación que puede estar asociada al tipo de profesor seleccionado.
- **vista_configuracion_usuario.php:** lista y gestiona los usuarios.
- **vista_configuracion_restricciones.php:** lista y activa/desactiva las restricciones fuertes y débiles.

- **vista_generar_horario.php:** lista los datos faltantes para la generación del horario, lista las restricciones débiles incumplidas después de la generación del horario docente y genera el horario docente.

- **Controlador:** Facilita y controla la comunicación entre las vistas y los modelos.
  - **controlador_profesor.php:** facilita y controla las peticiones realizadas a los modelos desde vista_profesores y vista_profesor_especificacion.
  - **controlador_asignatura.php:** facilita y controla las peticiones realizadas a los modelos desde vista_asignatura [1, 2, 3, 4,5] y vista_planificar_EF.
  - **controlador_grupo.php:** facilita y controla las peticiones realizadas a los modelos desde vista_grupos y vista_grupo_especificacion.
  - **controlador_local.php:** facilita y controla las peticiones realizadas a los modelos desde vista_locales y vista_local_especificacion.
  - **controlador_nomenclador.php:** facilita y controla las peticiones realizadas a los modelos desde vista_configuracion, vista_configuracion_semanas, vista_configuracion_turnos, vista_configuracion_encuentros, vista_configuracion_dias, vista_configuracion_anos, vista_configuracion_locales, vista_configuracion_afectacion, vista_configuracion_afectacion_clasificacion_profesor, vista_configuracion_usuario y vista_configuracion_restricciones.
  - **controlador_profesor_grupo_asignatura_TE.php:** facilita y controla las peticiones realizadas a los modelos desde vista_profesor-grupo-asignatura.
  - **controlador_generar_horario.php:** facilita y controla las peticiones realizadas a los modelos desde vista_generar_horario.

2.10 Estándar de codificación

Nombres de las funciones: contienen caracteres alfanuméricos, deben comenzar con minúsculas y cuando tienen más de una palabra la primera letra de estas debe capitalizarse.

Nombres de las variables: deben ser descriptivos y concisos, deben comenzar con minúsculas y en caso de contener más de una palabra esta debe ser separada por guion bajo (_).

Llamadas a funciones: se llamarán sin utilizar espacios entre el nombre de la función, el paréntesis de apertura y el primer parámetro, tampoco entre el último parámetro, el

paréntesis de cierre y el punto y coma; los espacios se usaran entre las comas y cada uno de los parámetros.

A continuación se muestra un ejemplo de cómo se aplica el estándar de codificación definido:

```
public function existeEncuentro($lista_encuentros, $encuentro) {
    $valido = 1;
    $no_valido = 0;
    for ($i = 0; $i < count($lista_encuentros); $i++)
        if ($lista_encuentros[$i] == $encuentro)
            return $valido;
    return $no_valido;
}
```

Figura 4. Ejemplo de estándar de codificación

2.11 Descripción de la solución

El sistema propuesto constituye una herramienta para la generación del horario docente de la facultad de Ingeniería Industrial del ISPJAE. Incluye dos tipos de usuarios: los que podrán acceder al sistema solamente para consultar el horario docente y los que tendrán acceso a las funcionalidades del mismo, estos últimos deberán autenticarse. La estructura del sistema para los usuarios autenticados es la siguiente:

- En la barra superior se muestran las siguientes opciones: revisar horario, ayuda (ayuda del sistema), contacto (datos de los autores del sistema). Además se muestra el nombre de la página activa, el usuario que se encuentra registrado y la opción de cerrar sesión.
- En el menú izquierdo se muestran las opciones correspondientes a la gestión de los datos necesarios para la generación del horario docente: grupos, asignaturas, profesores, locales y la configuración del sistema. Además se muestra la opción correspondiente a la generación del horario docente.
- En el área de trabajo de la interfaz inicial se muestran los datos manejados por el sistema: semestre activo, años activos, cantidad de profesores, grupos, locales, asignaturas, entre otros. También es donde se muestran los formularios asociados a las opciones del menú izquierdo.

Figura 5. Interfaz de inicio de un usuario autenticado

El sistema permite la gestión de las afectaciones de profesores, locales y grupos. Los profesores pueden ser clasificados de acuerdo a las necesidades de la institución, cada clasificación puede poseer una afectación que heredarán los profesores asociados a la misma, permitiendo el registro de la misma afectación a varios profesores.

Para la implementación del sistema se tiene en cuenta un conjunto de restricciones específicas de la institución. La cantidad de semanas a planificar depende de la cantidad de semanas que contenga la secuencia de actividades de cada año.

A partir de los datos insertados se puede generar el horario docente desde la opción del menú izquierdo correspondiente, si existen problemas en los datos de entrada se listan y el usuario debe decidir si continúa o no la operación, de lo contrario el sistema procede a generar el horario docente. Finalmente se obtiene una versión factible del horario docente en función de los datos de entrada y se muestra el por ciento de cumplimiento de las restricciones débiles.

El horario docente se mostrará de acuerdo a los criterios de búsqueda seleccionados (semana, año, grupo, profesor, local) y podrá ser descargado en formato PDF. A continuación se muestra el horario generado de la semana 11 del grupo I11 de primer año.

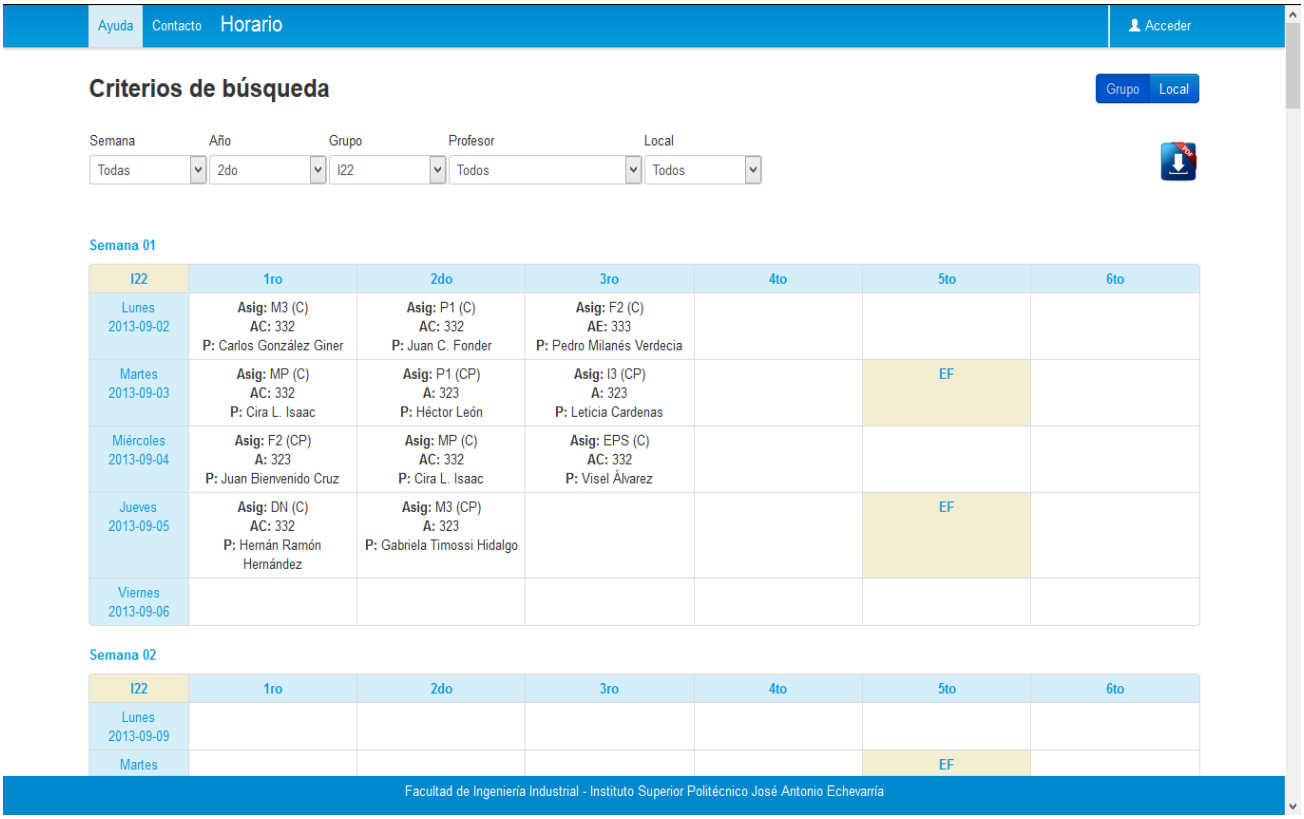

Figura 6. Ejemplo de horario generado

2.12 Complejidad y tiempo de respuesta de la solución

Según (Ortiz Grandel, 2013) la eficiencia suele medirse en términos de consumo de recursos:

- Temporales: tiempo empleado por el algoritmo para ejecutarse y proporcionar un resultado a partir de los datos de entrada (complejidad temporal).
- Espaciales: cantidad de memoria requerida (suma total del espacio que ocupan las variables del algoritmo) antes, durante y después de su ejecución (complejidad espacial).

La complejidad temporal o tiempo de ejecución de un programa se mide en función de T(N). Esta función se puede calcular físicamente ejecutando el programa acompañados de un reloj, o calcularse directamente sobre el código, contando las instrucciones a ser ejecutadas y multiplicando por el tiempo requerido por cada instrucción (López, 2010).

A partir del cálculo directo sobre el algoritmo que converge a la solución del sistema, la función T(N) posee una cota superior de O (n^k) $k > 3$.

En correspondencia a los elementos ajenos del propio algoritmo, su tiempo de ejecución viene dado por los datos siguientes:

Prestaciones de estaciones de trabajo	Año	Tiempo de respuesta
Planificadora: • Memoria RAM: 1Gb • Microprocesador: Celeron R CPU 1.70GHz	2do	1 minuto 3 segundos
	Todos	4 minutos 1 segundos
• Memoria RAM: 1Gb • Microprocesador: DualCore 2.4GHz	2do	48 segundos
	Todos	3 minutos 34 segundos
• Memoria RAM: 4Gb • Microprocesador: Core I5 2.4Ghz	2do	46 segundos
	Todos	3 minutos 23 segundos

Tabla 12. Tiempos de respuesta del algoritmo.

Para el cálculo de la complejidad espacial se actúa de la misma manera, solo que en lugar de contar instrucciones, se cuenta la cantidad de piezas de memoria dinámica que se utilizan. Sin embargo, su estudio suele tener menos interés, porque el tiempo es un recurso mucho más valioso que el espacio.

2.13 Conclusiones parciales

En este capítulo se describió la propuesta de solución para la creación del Sistema de generación del horario de la facultad de Ingeniería Industrial del ISPJAE. Se presentaron las restricciones fuertes y débiles involucradas en la solución, acordadas con el cliente. Se describieron las HU permitiendo representar las funcionalidades del sistema, se estableció la prioridad de las mismas para el negocio, correspondiendo a una estimación del esfuerzo de desarrollo, también se especificaron cuáles van a ser implementadas en cada iteración, determinándose las fechas de entrega de cada una y por consecuente la fecha final de entrega del sistema. Se definió la estructura de la base de datos, así como la arquitectura y los patrones de diseños usados en la solución. Por último se calculó la complejidad temporal del algoritmo encargado de la generación del horario docente para medir su eficiencia en diferentes estaciones de trabajo.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN

3.1 Introducción

En este capítulo se realiza la validación de la solución propuesta. Se evalúa tanto a nivel de diseño como de implementación para comprobar el correcto funcionamiento del sistema.

3.2 Validación del diseño

Las métricas de software son una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. Pressman plantea en (Pressman, 2005) que los ingenieros de software usan las métricas del producto como apoyo para construir software de mayor calidad, aunque no suelen ser absolutas, proporcionan una manera sistemática de evaluar la calidad a partir de un conjunto de reglas definidas con claridad.

Para la evaluación de la calidad del diseño propuesto se hará uso de las métricas Tamaño operacional de clase (TOC) y Relaciones entre clases (RC) propuestas por Lorenz y Kidd (Lorenz, y otros, 1994).

TOC: Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad (EcuRed, 2014):

- Responsabilidad: Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- Complejidad de implementación: Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- Reutilización: Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Atributo de calidad	Categoría	Criterio
Responsabilidad	Baja	< =Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Complejidad de implementación	Baja	< =Promedio
	Media	Entre Promedio y 2* Promedio

	Alta	> 2* Promedio
Reutilización	Baja	> 2*Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	<= Promedio

Tabla 13. Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC

A continuación se muestran las clases del sistema evaluadas en los atributos de calidad mencionados.

Clase	Cantidad de procedimientos	Responsabilidad	Complejidad de implementación	Reutilización
ModeloAsignatura	30	Media	Media	Media
ModeloAfectacion	10	Baja	Baja	Alta
ModeloClasificacionProfesorAfectacion	12	Baja	Baja	Alta
ModeloProfesorGrupoAsignaturaTE	18	Media	Media	Media
ModeloProfesorAfectacion	11	Baja	Baja	Alta
ModeloProfesor	12	Baja	Baja	Alta
ModeloNomenclador	57	Alta	Alta	Baja
ModeloLocalAfectacion	9	Baja	Baja	Alta
ModeloLocal	9	Baja	Baja	Alta
ModeloHorario	43	Alta	Alta	Baja
ModeloGrupoAfectacion	9	Baja	Baja	Alta
ModeloGrupo	10	Baja	Baja	Alta
ModeloFiltro	3	Baja	Baja	Alta
ModeloFechaEspecial	6	Baja	Baja	Alta

Tabla 14. Evaluación de las clases del sistema mediante la métrica TOC

La gráfica que corresponde a los resultados obtenidos se muestra a continuación.

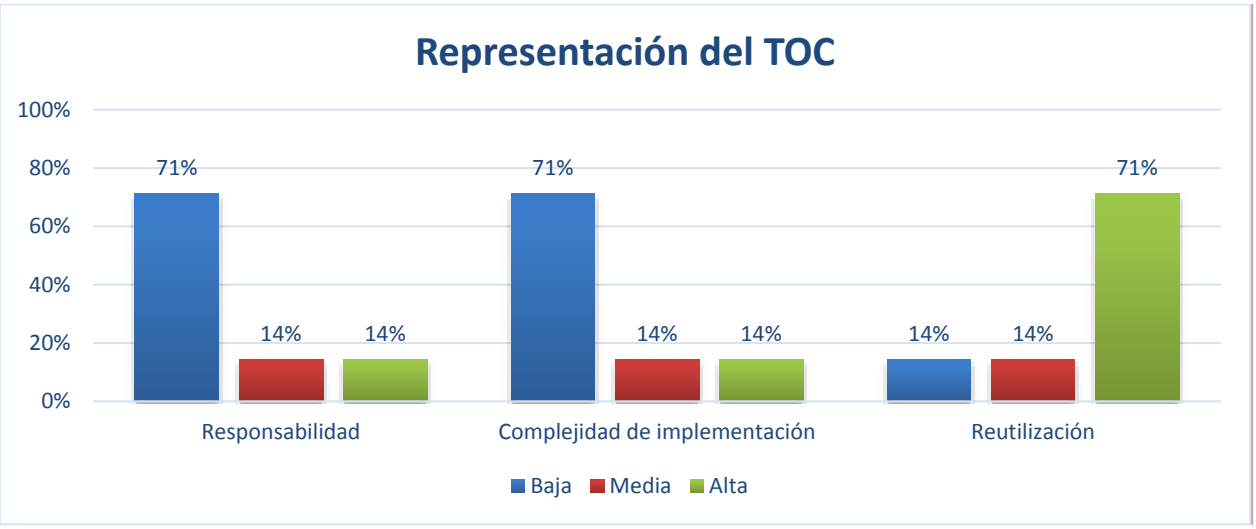

Figura 7. Representación de los resultados de la métrica TOC

Como resultado de la aplicación de la métrica TOC se evidencia que las clases del sistema poseen una baja responsabilidad, baja complejidad de implementación y alta reutilización por lo que el diseño de las clases en cuanto a cantidad de funcionalidades por cada una es bueno.

RC: Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad (EcuRed, 2014):

- 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.

Atributo de calidad	Categoría	Criterio
Acoplamiento	Ninguno	0
	Baja	1
	Media	2
	Alta	>2

Complejidad de 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

Tabla 15. Rangos de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC

A continuación se muestran las clases del sistema evaluadas en los atributos de calidad mencionados.

Clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad de mantenimiento	Reutilización	Cantidad de pruebas
secuencia_actividades_n_semana_n_actividad	3	Alto	Baja	Alta	Baja
asignatura	4	Alto	Media	Media	Media
profesor_asignatura_n_tipo_encuentro_grupo	7	Alto	Media	Media	Media
profesor	3	Alto	Baja	Alta	Baja
grupo	4	Alto	Media	Media	Media
local	3	Alto	Baja	Alta	Baja
encuentro	4	Alto	Media	Media	Media
profesor_afectacion	2	Medio	Baja	Alta	Baja
local_afectacion	2	Medio	Baja	Alta	Baja
grupo_afectacion	2	Medio	Baja	Alta	Baja
afectacion	6	Alto	Media	Media	Media

educacion_fisica	3	Alto	Baja	Alta	Baja

Tabla 16. Evaluación de las clases del sistema mediante la métrica RC

La gráfica que corresponde a los resultados obtenidos se muestra a continuación.

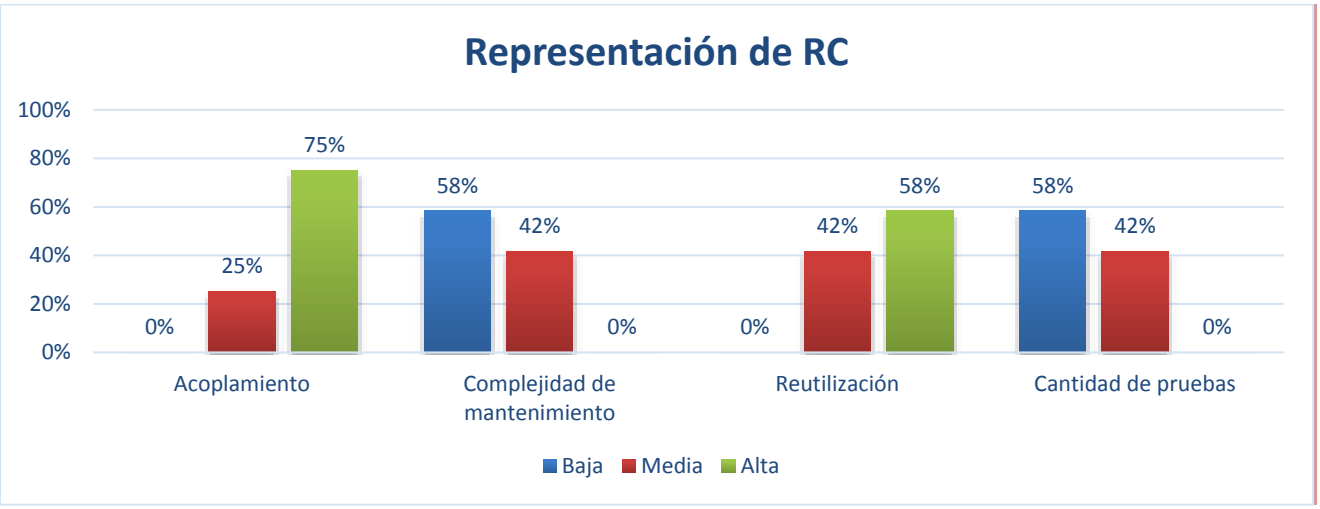

Figura 8. Representación de los resultados de la métrica RC

Los resultados obtenidos en los atributos de calidad: complejidad de mantenimiento, reutilización y cantidad de pruebas son satisfactorios de acuerdo a lo planteado en la métrica RC, no siendo así para el acoplamiento entre las clases que obtuvo un valor alto debido a las necesidades del negocio a desarrollar, donde la alta relación entre las clases facilita el manejo de los datos. De forma general se valoran como buenos los resultados.

3.3 Pruebas de software

La IEEE define las pruebas de software como “una actividad en la que un sistema o un componente es ejecutado bajo condiciones especificadas, los resultados son observados o registrados, y una evaluación es realizada de un aspecto del sistema o componente”. Se realizan para para descubrir defectos en el software y demostrar el mismo que satisface los requerimientos establecidos con el cliente.

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente (Beck, y otros, 2000). Se realizarán pruebas a nivel de unidad y sistema a

través del método de caja blanca y caja negra, para revisar código de la aplicación y la correcta implementación de las funcionalidades del sistema respectivamente.

3.3.1 Prueba de caja blanca

En (Pressman, 2005) se plantea que al emplear métodos de caja blanca, el ingeniero de software podrá derivar casos de prueba que garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez.

La prueba de la ruta básica es una técnica de prueba de caja blanca y se utilizará para realizar las pruebas unitarias a la solución propuesta. Esta técnica permite que se obtenga una medida de complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivador para ejercitar el conjunto básico deben garantizar que se ejecuta cada instrucción del programa por lo menos una vez durante la prueba (Pressman, 2005). A continuación se muestra el método validarEncuentro en el cual se valida el cumplimiento de las principales restricciones fuertes.

```
public function validarEncuentro($fecha, $id_grupo, $id_profesor, $id_asignatura, $turno) {
    $encuentro_valido = 0;
    if ($this->modeloNomenclador->conocerTurnoActivo($turno)[0][1]) {
        if ($this->comprobarGrupo($fecha, $id_grupo, $turno)) {
            if ($this->comprobarProfesor($fecha, $id_profesor, $turno)) {
                if ($this->comprobarAsignatura($fecha, $id_asignatura, $id_grupo)) {
                    if ($this->comprobarTurno($fecha, $id_grupo, $id_asignatura, $turno)) {
                        $encuentro_valido = 1;
                    }
                }
            }
        }
    }
    return $encuentro_valido;
}
```

Figura 9. Método para validar un encuentro

A continuación se muestra el grafo dirigido de flujo del algoritmo **validarEncuentro:**

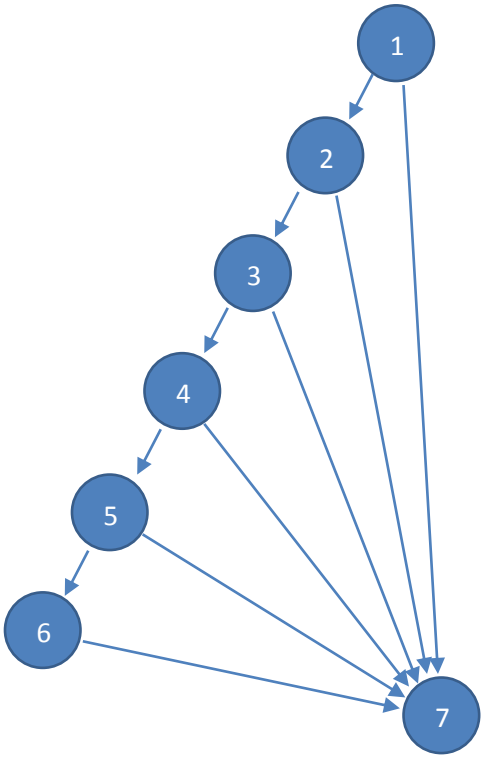

Figura 10. Grafo de flujo

Según (Pressman, 2005) la complejidad ciclomática es una métrica de software que proporciona una medida cuantitativa de la complejidad lógica de un programa. Cuando se emplea en el contexto del método de prueba de la ruta básica, el valor calculado mediante la complejidad ciclomática define el número de rutas independientes en el conjunto básico de un programa, y proporciona un límite superior para el número de pruebas que deben aplicarse para asegurar que todas las instrucciones se hayan ejecutado por lo menos una vez. La complejidad ciclomática, V(G), se calcula de una de tres maneras:

1. El número de regiones corresponde a la complejidad ciclomática.
2. V(G) = E-N+2, donde E es el número de aristas y N el número de nodos.
3. V(G) = P+1, donde P es el número de nodos predicado (son aquellos de los que salen varios caminos).

Tomando como referencia la figura anterior, se calculó la complejidad ciclomática empleando los tres algoritmos indicados para demostrar que la respuesta es la misma.

1. Existen 6 regiones.
2. V(G) = 11 aristas - 7 nodos + 2 = 6
3. V(G) = 5 nodos predicado + 1 = 6

La complejidad ciclomática es igual a 6, esto significa que existen 6 posibles caminos linealmente independientes y representa el mínimo número de casos de prueba para el procedimiento tratado. A continuación se muestran los caminos existentes:

Número	Caminos
1	1-7
2	1-2-7
3	1-2-3-7
4	1-2-3-4-7
5	1-2-3-4-5-7
6	1-2-3-4-5-6-7

Tabla 17. Caminos por donde el flujo puede circular

El próximo paso es ejecutar los casos de pruebas para cada camino y se compara con los resultados esperados, teniendo en cuenta que las instrucciones se hayan ejecutado por lo menos una vez. A continuación se muestra el caso de prueba para el camino básico 1, en el Anexo 2 se muestran los restantes casos de prueba.

Caso de prueba para el camino básico 1	
Descripción	Se comprueba si el tipo de turno pasado por parámetro está disponible para ser usado.
Condiciones de ejecución	Que el tipo de actividad exista y esté activo.
Entrada	$turno
Resultado esperado	Que continúe con la ejecución del resto del código.

Tabla 18. Caso de prueba para el camino básico 1

3.3.2 Prueba de caja negra

Las pruebas de caja negra se concentran en las HU del software, permiten al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todas las HU de un programa. Según (Pressman, 2005) tratan de encontrar errores en las siguientes categorías:

1. Funciones incorrectas o faltantes
2. Errores de interfaz

3. Errores en estructuras de datos o en accesos a bases de datos externas
4. Errores de comportamiento o desempeño
5. Errores de inicialización y término

3.3.3 Casos de prueba

Un caso de prueba incluye un conjunto de entradas, condiciones de ejecución y resultados esperados para conseguir un objetivo particular o condición de prueba por ejemplo verificar el cumplimiento de un requisito específico.

El objetivo principal del diseño de casos de prueba consiste en derivar un conjunto de pruebas que tenga la mayor probabilidad de descubrir errores en el software (Pressman, 2005).

A continuación se muestran algunos ejemplos de casos de prueba para la aplicación:

Caso de prueba de caja negra	
Código: 4	**HU:** 5
Nombre: Adicionar profesor.	
Descripción: Permite al usuario adicionar un profesor.	
Condiciones de ejecución: • La persona debe estar autenticada en el sistema. • Debe existir como mínimo una clasificación de profesores.	
Entrada/Pasos de ejecución: • Seleccionar la opción “Profesores” del menú izquierdo. • Seleccionar el botón “Adicionar” de la parte superior derecha. • Escribir en la ventana flotante mostrada el nombre que identifica al nuevo profesor y seleccionar su clasificación. • Seleccionar el botón “Adicionar”.	
Resultados esperados: El sistema debe adicionar al nuevo profesor y listarlo en la tabla correspondiente.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 19. Caso de prueba para la HU Gestionar profesor

Caso de prueba de caja negra	
Código: 2	**HU:** 2
Nombre: Modificar año.	
Descripción: Permite modificar las fechas de inicio de semestre, del trabajo socialmente útil (TSU, en adelante) para cada año y la sesión del día en que se le planificarán clases.	
Condiciones de ejecución: • La persona debe estar autenticada en el sistema. • El año debe estar habilitado.	
Entrada/Pasos de ejecución: • Seleccionar la opción “Editar” ( ) que se muestra en la columna “Opciones” de la tabla “Años escolares”. • Escribir una fecha de inicio del semestre. • Escribir una fecha de inicio del TSU (opcional). • Seleccionar la sesión en la que se le planificará las actividades. • Seleccionar el botón “Modificar”.	
Resultados esperados: El sistema muestra los elementos modificados en la tabla “Años escolares”.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 20. Caso de prueba para la HU Gestionar año.

El resto de los casos de pruebas se encuentran en el Anexo 3.

3.3.4 Resultados de la prueba de caja negra

Se llevaron a cabo 4 iteraciones en las que se detectaron 31 no conformidades (NC, en adelante) que fueron resueltas de forma rápida. En la primera iteración se detectaron un total de 15 NC dentro de las cuales 5 fueron de aplicación, 9 de validaciones y 1 de ortografía, en la segunda se detectaron 11 de las cuales 6 fueron de aplicación y 5 de validaciones, en la tercera se detectaron 5 de las cuales 2 de la aplicación y 3 de validaciones y en la cuarta no se detectaron NC.

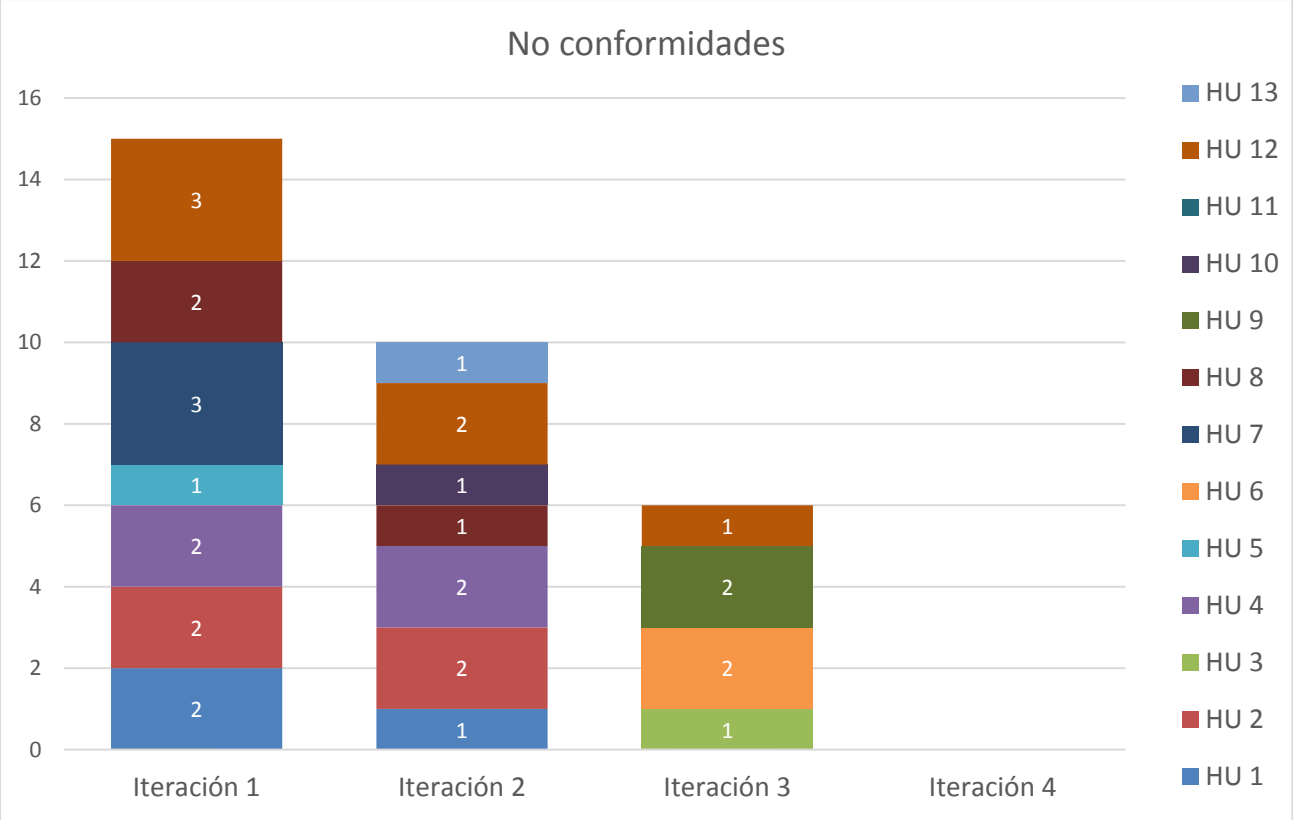

Figura 11. Gráfica resultado de aplicar las pruebas de caja negra

Conclusiones parciales

En este capítulo se describieron las pruebas efectuadas al diseño y las funcionalidades del sistema propuesto. Al aplicar las métricas TOC y RC se pudo comprobar el adecuado diseño de las clases. Las pruebas de caja blanca efectuadas a los métodos del sistema mediante la técnica “ruta básica” arrojaron los resultados esperados en cada caso. Las pruebas de caja negra aplicadas a las HU permitieron detectar un conjunto de NC que fueron resueltas en cuatro iteraciones, obteniéndose la solución esperada por el cliente.

CONCLUSIONES GENERALES

La investigación realizada evidenció la necesidad de desarrollar un sistema que sea capaz de generar un horario docente y que cumpla con las características y restricciones propias de la Facultad de Ingeniería Industrial del ISPJAE, ya que las soluciones existentes no se ajustan a esta realidad o son privativas.

Se realizó el diseño e implementación de un sistema que permite la generación del horario docente de la facultad de Ingeniería Industrial del ISPJAE en un tiempo aproximado de cuatro minutos, de forma manual esta actividad suele demorar más de cuatro semanas. El horario docente generado estará disponible desde el sitio web de la institución, además podrá ser descargado, en formato PDF.

Se realizó la validación de la solución evidenciándose el cumplimento de las funcionalidades acordadas con el cliente.

Como resultado de la realización de este trabajo se desarrolló el "Sistema para la generación del horario docente de la Facultad de Ingeniería Industrial del ISPJAE" para contribuir a disminuir el tiempo invertido por la planificadora en esta actividad y propiciar la disponibilidad del horario docente a la comunidad universitaria.

RECOMENDACIONES

Se recomienda:

- Incluir algoritmos de optimización para mejorar la solución factible obtenida
- Incluir las siguientes funcionalidades:
    - Cambio de restricciones débiles por fuertes y viceversa.
    - Intercambios de actividades una vez generado el horario.
    - Planificación de las asignaturas optativas.
- Valorar la factibilidad de extender la solución a las demás facultades del IPSJAE que comparten características y restricciones similares a las de la facultad de Ingeniería Industrial.

BIBLIOGRAFÍA

64, Informática. Informática 64. *Informática 64.* [En línea] [Citado el: 17 de Noviembre de 2013.] http://www.informatica64.com/foca.aspx.

Adosis, S.A Sistemas Informáticos. 2013. *KRONOWIN.* [En línea] 20 de 10 de 2013. http://www.adossis.es.

Alvarez, Miguel Angel. 2013. *Manual de CSS.* 2013.

Araya, Juan Enrique Molina. 2007. *Algoritmos Evolutivos para la resolución de un problema de tipoTimetabling.* 2007.

Arnold, Ken y Gosling, James. 1997. *Lenguaje de programación Java.* Madrid : s.n., 1997.

Barber, Federico y Salido y Miguel A. 2003. *Introducción a la Programación de Restricciones.* [En línea] 2003. [Citado el: 26 de abril de 2014.] http://users.dsic.upv.es/~msalido/publications.htm.

Barrera, MSc. David Silva. 2011. Sistema de Planificación y publicación de horarios docentes UCI (Horr). *Manual de usuario.* Universidad de las Ciencias Informáticas, Habana, Cuba : s.n., 2011.

BAUTISTA, JOSE M. 2013. Ingenieria de Software. *Ingenieria de Software.* [En línea] 29 de Octubre de 2013. http://ingenieriadesoftware.mex.tl/images/18149/PROGRAMACIÓN EXTREMA.pdf.

Beck, Kent. 1999. *Extreme Programming Explained.* s.l. : Pearson Education, 1999. ISBN: 0201616416.

—. 2002. *Una explicación de la Programación extrema: aceptar el cambio.* 2002. 8478290559.

Beck, Kent y Fowler, Martin. 2000. *Planning Extreme Programming.* 2000. 0201710919.

Biblioteca Nacional de Nueva Zelanda. 2003. Metadata Extraction Tool. *Metadata Extraction Tool.* [En línea] 2003. [Citado el: 12 de Diciembre de 2013.] http://www.natlib.govt.nz/services/get-advice/digital-libraries/metadata-extraction-tool.

BioMed Central Ltd unless otherwise stated. 2014. Source Code For Biology aand Medicine. *Source Code For Biology aand Medicine.* [En línea] 2014. [Citado el: 24 de Abril de 2014.] http://www.scfbm.org/content/7/1/7/abstrac. 1751-0473.

Boehm, Barry W. 1979. *GUIDELINES FOR VERIFYING AND VALIDATING SOFTWARE REQUIREMENTS AND DESIGN SPECIFICATIONS.* Redondo Beach, CA, USA : s.n., 1979.

Carmona, Marcel Curbelo. 2013. *Herramienta web para el control al horario docente en la.* Pinar del Río : s.n., 2013.

Catalunya, Fundación para la Universidad Oberta de. EduKanda. Recursos formativos en red. *EduKanda. Recursos formativos en red.* [En línea] [Citado el: 19 de Febrero de 2014.] http://www.edukanda.es/mediatecaweb/data/zip/614/PID_00154130/web/main/m1/v2_2_1.html..

CIBERTEC. *Pruebas de Software.* España : s.n.

Ciencias de la Información. **Cedeño, Dunia Maria Colomé, Rodríguez, Mirurgia Ávila y Sentí, Vivian Estrada.** s.l. : Revista Científica del Instituto de la Información Científica y Tecnológica.

Deitel, H. M. y Deitel, P. J. 1990. *C++ como programar.* Ciudad de México : s.n., 1990.

Doctrine. Doctrine Query Language. *Doctrine Query Language.* [En línea] [Citado el: 5 de Diciembre de 2013.] http://docs.doctrine-project.org/en/2.1/reference/dql-doctrine-query-language.html.

—. Object Relational Mapper. *Object Relational Mapper.* [En línea] [Citado el: 3 de Diciembre de 2013.] http://www.doctrine-project.org/projects/orm.html.

EcuRed. 2014. EcuRed. *Métrica de diseño.* [En línea] 2014. [Citado el: 26 de mayo de 2014.] http://www.ecured.cu/index.php/Métrica_de_diseño.

Eguíluz Pérez, Javier. 2009. *Introducción a JavaScript.* 2009.

Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A. 2009. *Metodologías Tradicionales vs. Metodologías Ágiles.* s.l. : Universidad Técnica Particular de Loja, Escuela de Ciencias Informáticas, 2009.

findbestopensource.com. 2010. Best Open Source. *Best Open Source.* [En línea] 2010. [Citado el: 24 de Abril de 2014.] http://www.findbestopensource.com/product-lapdftext.

Flores, Pedro , y otros. 2010. *Experimentos con Algoritmos Genéticos para resolver un problema real de Programación Maestros-Horarios-Cursos.* 2010.

Fornaris, Maite Sánchez y Rabí, Dayana Alcantara. 2010. *Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas de Información Geográfica.* 2010.

Foundation, The Apache Software. Configuration Files. *Configuration Files.* [En línea] [Citado el: 25 de Noviembre de 2013.] http://httpd.apache.org/docs/current/configuring.html.

—. Licenses. *Licenses.* [En línea] [Citado el: 24 de Noviembre de 2013.] http://www.apache.org/licenses/.

—. What is the Apache HTTP Server Project? *What is the Apache HTTP Server Project?* [En línea] [Citado el: 24 de Noviembre de 2013.] http://httpd.apache.org/ABOUT_APACHE.html.

Govea, Blanca A. Vargas. 2012. http://blancavg.com/. *http://blancavg.com/.* [En línea] 31 de Agosto de 2012. [Citado el: 6 de Junio de 2014.] http://blancavg.com/tc3023ic/ic8.pdf.

Gutiérrez Andrade, Miguel Ángel, de los Cobos Silva, Sergio Gerardo y Pérez Slavador, Blanca Rosa. 1998. Universidad Autónoma Metropolitana. *OPTIMIZACION CON RECOCIDO SIMULADO PARA EL PROBLEMA DE CONJUNTO INDEPENDIENTE.* [En línea] Junio de 1998. [Citado el: 12 de Diciembre de 2013.] http://www.azc.uam.mx/publicaciones/enlinea2/3-2.htm.

Gutiérrez JJ, Escalona MJ, Mejías M, Torres J. 2006. *Pruebas del Sistema en Programación Extrema.* s.l. : Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla, 2006.

Hernández, Rodrigo, P., Jaime Miranda y Rey, Pablo A. 2008. *Programación de Horarios de Clases y Asignación de Salas para la Facultad de Ingeniería de la Universidad Diego Portales Mediante un Enfoque de Programación Entera.* Chile : s.n., 2008.

HUMBERTO. 2007. *LIBRO DE BIBLIOTECAS.* 2007.

Jeffries, Ron, Anderson, Ann y Hendri, Chet . 2001. *Extreme Programming Installed.* s.l. : Addison-Wesley Professional, 2001.

Joskowicz, José. 2008. *Reglas y Prácticas en eXtreme Programming.* España : s.n., 2008.

JUnit. 2012. JUnit.org. *JUnit.org.* [En línea] 2012. [Citado el: 17 de Noviembre de 2013.] http://www.junit.org.

Kruchten, Philippe. 2008. rational.com. *rational.com.* [En línea] 2008. http://www.rational.com.

Larman, Craig. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos y al proceso unificado.* México : PRENTICE HAL, 1999.

—. 1999. *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos.* Prentice Hall,México : Pearson, 1999. ISBN:970-17-0261-1.

Larman, Craig y Hall, Prentice. 2003. *El modelo de diseño. UML y Patrones 2ª Edición.* 2003.

Letelier Torres, Patricio, y otros. 2003. *Metodologías Ágiles en el Desarrollo de Software.* Alicante : Grupo ISSI, 2003.

López, Rolf Pinto. 2010. Algoritmos y Estructura de Datos I. *Algoritmos y Estructura de Datos I.* Arequipa, Perú : s.n., 2010.

Lorenz, Mark y Kidd, Jeff. 1994. *Object-Oriented Software Metrics.* Nueva Jersey : Englewood Cliffs, 1994.

Lucero, Fredy Cuenca. 2007. *Técnicas de Inteligencia Artificial aplicadas a la confección de horarios.* Lima : s.n., 2007. Vol. 2.

Madrid, Universidad Carlos III. 2014. Metadatos Recuperación y Acceso a la Información. *Metadatos Recuperación y Acceso a la Información.* [En línea] 25 de Febrero de 2014. http://www.metadatos-xmlrdf.com/metadatos/dublin-core.

Manyá, Felip y Gomes, Carla. 2003. *Técnicas de resolución de problemas de satisfacción de restricciones.* 2003. 1137-3601.

Martelli, Alex. 2008. *Python. Guía de referencia.* España : ANAYA MULTIMEDIA, 2008.

Martín García, Elena. 2013. *Desarrollo de una aplicación web para la creación de exámenes de opción múltiple.* Madrid : s.n., 2013.

Martinez, Rafael. 2010. www.postgresql.org.es. *www.postgresql.org.es.* [En línea] 2 de Octubre de 2010. [Citado el: 3 de Junio de 2014.] http://www.postgresql.org.es/sobre_postgresql.

Mendeley. infobiblio.es. *infobiblio.es.* [En línea] Mendeley. [Citado el: 24 de Noviembre de 2014.] http://www.infobiblio.es/tutorial-de-mendeley.

Merelo, Juan Julián. 2004. RedHur. Investigación en Optimización Heurística. *Investigación en Optimización Heurística.* [En línea] 29 de Enero de 2004. [Citado el: 2013 de Diciembre de 13.] http://www.redheur.org/sites/default/files/metodos/GA01.pdf.

Miguel Angel Alvarez. 2009. DesarrolloWeb.com. *DesarrolloWeb.com.* [En línea] 14 de Octubre de 2009. [Citado el: 15 de 12 de 2013.] http://www.desarrolloweb.com/articulos/que-es-html5.html.

Murphey, Rebecca. *Fundamentos de JQuery.*

OAI-PMH: Protocolo para la transmisión de contenidos en Internet. **Barrueco, José Manuel.** Barcelona, España : Biblioteca de Ciencias Sociales. Universidad de Valencia.

Ortiz Grandel, Adolfo. 2013. *Complejidad Espacial.* Santo Domingo : s.n., 2013.

Pecina Federico, I.S.C. Alonso y Cruz Reyes, M.C. Laura. 2009. *El Problema de la Programación de Horarios con Coloreo de Grafos.* México : s.n., 2009.

Peñalara. 2013. penalara.com. *penalara.com.* [En línea] 21 de 10 de 2013. https://www.penalara.com.

Pérez, Chantal. Estudio de Lenguística del Español. Explotación de los córpora textuales informátizados para la creación de bases de datos terminológicas basadas en el conocimiento. *Estudio de Lenguística del Español. Explotación de los córpora textuales informátizados para la creación de bases de datos terminológicas basadas en el conocimiento.* [En línea] Universidad Autónoma de Barcelona. [Citado el: 14 de Febrero de 2014.] http://elies.rediris.es-elies18/23.html.

Petters, Untis Gruber &. 2013. Untis Express Horarios en tiempo record. *Untis Express Horarios en tiempo record.* [En línea] 21 de 10 de 2013. http://www.programahorario.com.

—. 2013. Untis Programa de Horarios. *Untis Programa de Horarios.* [En línea] 20 de 10 de 2013. http://www.grupet.at.

PHP. What is PHP? *What is PHP?* [En línea] [Citado el: 4 de Diciembre de 2013.] http://php.net/.

Piattini Velthuis, Mario G. 1996. *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión Rama.* Madrid : RA-MA EDITORIAL, 1996.

Piattini, M. G., y otros. 1996. *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión.* 1996.

Piattini, Mario. 2007. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* Madrid : RA-MA, 2007. 9788478977765.

Piñeiro, Antonio de la Rosa y Senso, José A. 2003. *El concepto de Metadato. Algo más que descripcioón de recursos electrónicos.* s.l. : Universidad de Granada, 2003.

PostgresSQL. PostgreSQL: About. *PostgreSQL: About.* [En línea] [Citado el: 5 de Diciembre de 2013.] http://www.postgresql.org/about/.

Potencer, Fabien y Weaver, Ryan. 2013. *Symfony 2, el libro oficial.* s.l. : Creative Commons Atribución - Compartir igual (CC BY-SA) 3.0, 2013.

Pressman, Roger S. 2005. *Ingeniería del Software. Un enfoque práctico.* 2005. Sexta edición, 0072853182.

Puig de la Bellacasa, Jorge. 2014. openaccess.uoc.edu. *Desarrollo de una aplicación web multiplataforma según el patrón Modelo Vista Controlador.* [En línea] 5 de Enero de 2014. [Citado el: 5 de Mayo de 2014.] http://hdl.handle.net/10609/28501.

RAE. 2013. RAE. *RAE.* [En línea] 23 de Noviembre de 2013. [Citado el: 23 de Noviembre de 2013.] http://lema.rae.es.

Raghavan, S., Zelesnik, G. y Ford, G. 1994. *Lecture Notes on Requirements Elicitation.* s.l. : Software Engineering Institute, Carnegie Mellon University, 1994. Educational Materials CMU/SEI–94–EM–10.

Real Academia de la Lengua Española. 2001. *Diccionario de la Lengua Esppañola.* Madrid : s.n., 2001.

Sæther Bakken, Stig, y otros. 2002. *Manual de PHP.* 2002.

Schaerf, Andrea. 1999. TEI Central Macedonia. *A Survey of Automated Timetabling.* [En línea] 1999. [Citado el: 26 de abril de 2014.] http://www.teicm.gr/arximidis/pdf/kazarlis/PE1/1d.pdf. ISSN: 0269-2821.

Schwaber, Ken. 2013. *Scrum Manager.* 2013.

Scott, Kendall. 2009. *El Proceso Unificado Explicado.* s.l. : Pearson Education, 2009.

Somerville, Ian. 2002. *Ingeniería de Software.* s.l. : Pearson Educación, 2002.

—. 2005. *Ingeniería del software.* Madrid : Pearson Education S.A., 2005. ISBN:84-7829-074-5.

SyGMe: Sistema para la Gestión de Metadatos Geográficos. **Padrón, Liset García y Sosa, Alejandro Leandro. 2013.** La Habana : s.n., 2013.

Testa, Patricia y Ceriotto, Paula . *Descripción de Objetos Digitales:Metadatos.* s.l. : Universidad Nacional de Cuyo Centro Universitario.

Toro, Amador Durán y Jiménez, Beatriz Bernárdez. 2000. *Metodología para la Elicitación de Requisitos de Sistemas Software.* Sevilla : s.n., 2000.

Varona, Karel Rodríguez. 2012. *Aplicación de algoritmos genéticos en la generación automática de horarios docentes en la Facultad Regional de Granma.* Granma : s.n., 2012.

Venners, Bill. 2003. Artima Developer. *La filosofía de Ruby, una conversación con Yukihiro Matsumoto, Parte I.* [En línea] 29 de Septiembre de 2003. [Citado el: 15 de 12 de 2013.] http://www.artima.com/intv/ruby4.html.

vicomtech. vicomtech. *vicomtech.* [En línea] [Citado el: 24 de Febrero de 2014.] http://www.vicomtech.org/t4/e11/procesamiento-del-lenguaje-natural.

Visconti, Marcello y Astudillo, Hernán. 2006. *Fundamentos de Ingeniería de Software.* 2006.

Welling, Luke y Thomson, Laura. 2005. *Desarrollo web con php y mysql php 5 y mysqul 4.1 y 5: disco compacto.* Madrid : s.n., 2005.