Universidad de las Ciencias Informáticas

Facultad 3

Desarrollo de los módulos Reservación de locales, Cambio de turnos y Asignación de locales para exámenes finales del Sistema para la generación de horario docente de la CUJAE

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

Autores: Cesar Manuel Cruzata De la cruz

Sandy Javier Isaac Verdecia

Tutor

MSc. Isabel González Flores

La Habana, Junio del 2015

DECLARACIÓN DE AUTORÍA

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

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

Cesar Manuel Cruzata De la cruz

_____

Firma del autor

Sandy Javier Isaac Verdecia

_____

Firma del autor

MSc. Isabel González Flores

_____

Firma de la tutora

Agradecimientos

Cesar Manuel Cruzata De la cruz

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

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

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

Mil gracias a mi tutora por su paciencia y dedicación y por hacer de mí una persona mejor.

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

Gracias a todos mis amigos, en especial a Eliodanis, Beatriz, Anabel Valiente, Yiselli, Yasmani, las jimaguas Anabel y Rosabel, Carlos, Ebrik, David y todos los compañeros de mi grupo por todas sus bromas.

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

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

Sandy Javier Isaac Verdecia

Quiero agradecer ante todo a mi familia, en especial a mis padres, sin ellos no hubiera logrado graduarme

A Manolo,Yanelis, Petrona y a toda su familia que ya es como mi propia familia.

A mi Mana, a Yasmany, a mi novia, a las jimaguas y a la gente del aula por los buenos consejos.

A mi sobrina por volverme loco

A mi compañero de tesis por aguantarme todo este tiempo y soportar que me perdiera un par de semanas en Cienfuegos jejeje.

Agradecer a mí tutora, Isabel por estar arriba de nosotros todo el tiempo.

A mis colegas del cuarto que siempre están apoyándome

A todas las personas que de una forma u otra me ayudaron y me apoyaron, a todos ellos MUCHAS GRACIAS…

Dedicatoria

Cesar Manuel Cruzata De la cruz

Dedico el presente trabajo de culminación de la carrera:

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

A mis hermanos Yuliet y Roberto por estar siempre ahí cuando los necesité.

A mis amigos de la universidad y de mi barrio.

A todas esas personas que hicieron posible este momento.

Sandy Javier Isaac Verdecia

A mi mamá

A mi papá

A mi hermana

A mi novia

A mi familia completa

A mis colegas

A mí

RESUMEN

El presente trabajo propone una solución al problema del horario docente de la Facultad de Ingeniería Industrial de la CUJAE. Surge a partir de la necesidad de proporcionar una serie de funcionalidades al sistema para la generación de horarios de dicha facultad, ya que el horario generado por este sistema es estático, siendo una necesidad la incorporación de nuevos módulos que permitan realizar la reservación de locales disponibles, el cambio e intercambio de turnos de clase, además de incluir la asignación de locales disponibles para el desarrollo de exámenes finales, teniendo en cuenta las características de la CUJAE.

Para mantener una retroalimentación continua entre el equipo de desarrollo y el cliente, la metodología de desarrollo que guía la investigación es la programación extrema. Como método de solución al problema de investigación se analizan métodos y algoritmos ofrecidos por el área de inteligencia artificial, se escogen el lenguaje y las herramientas de programación para la implementación del sistema. Se realizó la propuesta de solución teniendo en cuenta las características de la CUJAE y se aplicaron las pruebas para evaluar la solución tanto a nivel de diseño como de implementación. Se obtiene un sistema capaz de minimizar los errores, el esfuerzo, el tiempo de trabajo invertido y flexible ante las necesidades de alta prioridad que puedan surgir durante el curso escolar.

Palabras claves: cambio de turnos, exámenes finales, horario docente, intercambio de turno, reservación de local.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La planificación es un proceso de vital importancia para la vida del hombre, según (RAE, 2015) consiste en desarrollar un plan general, metódicamente organizado y frecuentemente de gran amplitud, que proporciona la información necesaria para alcanzar las metas y objetivos trazados tanto social como profesionalmente, tal como el desarrollo económico, la investigación científica o el funcionamiento de una industria.

En (Franco Baquero, y otros, 2008) se plantea que el funcionamiento eficiente de una institución educativa requiere de recursos en infraestructura y medios educativos suficientes para impartir una educación de alta calidad. Estos recursos deberán ser lo suficientemente planificados y estudiados para evitar efectos negativos como una escasez o sobreoferta de ellos y afectar el deterioro en la calidad de la educación. Este último también se pone de manifiesto cuando la planificación del proceso docente no resulta ser del agrado de los profesores y alumnos implicados en ella. Cada año las universidades se enfrentan al problema de la planificación del horario docente (PPHD en adelante), consiste en la distribución de recursos y actividades docentes dentro de un limitado periodo de tiempo, conocido en la literatura como Timetabling de acuerdo a lo planteado en (Hojjat, y otros, 2003), (Burke, et al., 2007), (Cuenca Lucero, 2007), (Varona, 2012) y (Sabar, et al., 2012).

En las universidades de Cuba el proceso de confección del horario docente se organiza en turnos de clases de forma semestral, semanal y diaria, teniendo en cuenta las características, complejidad y cantidad de horas de cada asignatura a impartir por turno. Además, se debe tener en cuenta un conjunto de restricciones que deben cumplirse de forma obligatoria para garantizar la factibilidad del horario docente, conocidas como restricciones fuertes. Este proceso resulta engorroso cuando se realiza manualmente y es susceptible a errores por la presencia del factor humano, por este motivo se han aplicado las tecnologías de la información y las comunicaciones para resolver el PPHD.

La Facultad de Ingeniería Industrial de la CUJAE cuenta con un sistema que permite generar el horario docente en poco tiempo, este horario es estático y vulnerable a cambios de alta prioridad por necesidades tanto de profesores como de la propia institución. Los movimientos de turnos de clase se realizan con la previa autorización de la vicedecana docente e implican un análisis previo del efecto que pueda ocasionar, pues no se puede modificar el orden de la secuencia de actividades ni se deben violar las restricciones fuertes definidas por la institución.

En ocasiones los profesores o directivos de la Facultad necesitan saber los locales que se encuentran disponibles para realizar actividades docentes y extra-docentes. Para dar respuesta a la solicitud de reservación de un local es necesario buscar en el horario generado los locales que se encuentran disponibles en cada año y revisar el listado de reservaciones previas, esta actividad demora varios minutos y la reservación realizada se anota en una libreta o en algún otro

documento. Como constancia de haberse realizado se le entrega una nota al solicitante en el que se especifica el día, hora y local reservado.

Al finalizar el semestre se realiza la planificación de los exámenes finales, que son aprobados por el consejo de dirección docente, los dirigentes de la Federación Estudiantil Universitaria y de la Unión de Jóvenes Comunistas. El personal encargado de la planificación docente es el encargado de la asignación de los locales para los exámenes, esta actividad se realiza de forma manual pudiendo ocurrir errores como la asignación de locales con capacidad inferior a las requeridas para la realización de los exámenes de una asignatura. Además no existe un calendario en el que se pueda visualizar los exámenes de un año en específico o de todos los años, porque la información de los exámenes de cada asignatura se muestra de forma independiente en el mural de la Facultad.

Ante la problemática planteada se identifica el siguiente **problema a resolver**: ¿cómo facilitar la reservación de locales, el cambio de turnos y la asignación de locales para exámenes finales de la Facultad de Ingeniería Industrial de la CUJAE de forma que permita disminuir los errores y el tiempo invertido en estas actividades?

Objeto de estudio: Desarrollo de software en la planificación de horarios docentes.

Objetivo general: Desarrollar los módulos Reservación de locales, Cambio de turnos y Asignación de locales para exámenes finales del Sistema para la generación de horario docente de la CUJAE que permita disminuir los errores y el tiempo invertido en estas actividades.

Objetivos específicos:

- Elaborar el marco teórico de la investigación.
- Diseñar los módulos Reservación de locales, Cambio de turnos y Asignación de locales para exámenes del Sistema para la generación de horario docente de la Facultad de Ingeniería Industrial de la CUJAE.
- Implementar la solución.
- Valorar la efectividad de la solución propuesta.

Campo de acción: Informatización de la planificación del horario docente en la Facultad de Ingeniería Industrial de la CUJAE.

La **idea a defender** en la investigación se define como: El desarrollo de los módulos Reservación de locales, Cambio de turnos y Asignación de locales para exámenes finales del Sistema para la generación de horario docente de la CUJAE permitirá disminuir los errores y el tiempo invertido en estas actividades.

Tareas de la investigación:

1. Caracterización de la reservación de locales de la Facultad de Ingeniería Industrial de la CUJAE.
2. Caracterización del cambio de turnos de la Facultad de Ingeniería Industrial de la CUJAE.
3. Caracterización de la asignación de locales para exámenes finales de la Facultad de Ingeniería Industrial de la CUJAE.
4. Caracterización de los sistemas informáticos nacionales e internacionales que gestionan la reservación de locales, cambio de turnos y asignación de locales para exámenes finales de acuerdo a la metodología de desarrollo de software seleccionada.
5. Caracterización de las metodologías de desarrollo de software. Selección de la que se ajusta a la investigación.
6. Caracterización y selección de los lenguajes de programación y herramientas de desarrollo.
7. Selección de los patrones de diseño factibles para la propuesta de solución.
8. Descripción de los módulos Reservación de locales, Cambio de turnos y Asignación de locales para exámenes finales de acuerdo a la metodología de desarrollo de software seleccionada.
9. Diseño de los módulos Reservación de locales, Cambio de turnos y Asignación de locales para exámenes finales.
10. Aplicación de métricas para la validación del diseño.
11. Implementación los módulos Reservación de locales, Cambio de turnos y Asignación de locales para exámenes finales.
12. Diseño de casos de prueba.
13. Verificación de la solución propuesta: ejecución de pruebas al sistema.
14. Solución de las no conformidades detectadas.
15. Valoración de la factibilidad de la solución propuesta.

Estructura del documento

Capítulo 1: Fundamentación teórica, se reflejan los conceptos, teorías, métodos y sistemas informáticos existentes del ámbito nacional e internacional necesarias para la comprensión del problema de investigación propuesto. Además se realiza el análisis y selección de la metodología de desarrollo de software, herramientas y el lenguaje de programación para dar solución al problema de la investigación.

Capítulo 2: Propuesta de solución, se presentan las principales características de la propuesta de solución: restricciones, funcionalidades del sistema, modelo de datos, verificación del diseño de clases, arquitectura, patrones de diseño y la eficiencia de los módulos a partir del cálculo de la complejidad temporal.

Capítulo 3: Pruebas de la solución, se presenta la verificación de la solución propuesta, aplicando pruebas al software para valorar el funcionamiento del sistema implementado.

CAPÍTULO 1: Fundamentación teórica

1.1 Introducción

El capítulo muestra las teorías y conceptos relacionados con el PPHD, ejemplos de sistemas informáticos utilizados en el contexto internacional y nacional, así como la metodología de desarrollo de software, los métodos, el lenguaje de programación seleccionado para darle solución al problema de la investigación, las métricas de validación y las pruebas seleccionadas para verificar y validar el sistema.

1.2 Definición de términos

A continuación se establecen los términos utilizados en la Facultad de Ingeniería Industrial de la CUJAE que permiten una mayor comprensión 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, 2015).
- **Curso:** tiempo señalado de cada año de la carrera, distribuido en dos semestres.
- **Cambio de turno:** cambio de un turno de clases hacia otro turno en un local que se encuentre disponible.
- **Examen:** prueba que permite medir el nivel de conocimiento o habilidad del estudiante.
- **Grupo:** conjunto de estudiantes que pertenecen al mismo año e intervienen en las mismas actividades.
- **Local:** lugar destinado a desarrollar una actividad docente.
- **Locales disponibles:** Locales en los que no se ha planificado actividades docentes o de cualquier otro tipo en un turno del día determinado.
- **Profesor:** persona que enseña o imparte una asignatura.
- **Reservar local:** acción que permite reservar un local disponible para realizar una actividad que no está incluida en la planificación del horario docente.
- **Tipo de frecuencia:** tipo de actividad impartida, puede ser conferencia, clase práctica, seminario, laboratorio o taller.

1.3 Clasificación de las restricciones

Para la realizar la reservación de locales, cambio de turnos y asignación de locales para exámenes es necesario tener en cuenta diferentes restricciones relacionadas con el PPHD que se clasifican en dos categorías (Burke, y otros, 2007), (Cuenca Lucero, 2007), (Sabar, y otros, 2012) y (Guerrero, y otros, 2013):

- **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.
- **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.

1.4 Sistemas informáticos existentes

En la actualidad se han desarrollado sistemas informáticos que permiten darle solución al PPHD, generalmente desarrolladas a la medida, solamente tienen en cuenta las características y restricciones propias de la institución a la que va dirigida el sistema. Muchas no incluyen las funcionalidades relacionadas con la reservación de locales, cambio de turnos y asignación de locales para exámenes finales.

Internacionales

ascHorario: herramienta de planificación que facilita la gestión del horario de clases. Una vez generado el horario docente el sistema permite realizar cambios para varios usuarios de forma simultánea. Este sistema no permite asignar locales para exámenes finales ni la reservación de locales que se encuentren disponibles, además es un sistema privativo que solo se ejecuta en el sistema operativo Windows (asc Time Tables, 2015).

Untis 2015: sistema informático de planificación del horario docente que permite incluir los exámenes en el módulo de planificación y realizar cambios de forma manual tales como: turnos, locales y profesores sin necesidad de generar nuevamente el horario. Sin embargo no permite la reservación de locales disponibles, además de ser un sistema privativo que solo se puede ejecutar en el sistema operativo Windows (Gruber & Petters, 2015).

Los sistemas internacionales mencionados no satisfacen las necesidades actuales que presenta la Facultad de Ingeniería Industrial de la CUJAE. Sus funcionalidades incluyen parcialmente parte de los módulos que se pretenden desarrollar y no tienen en cuenta las restricciones fuertes y débiles específicas de la institución a la que va dirigida el sistema. Además, solo se ejecutan sobre el sistema operativo Windows, son sistemas privados y limitados por el pago de licencias, en (Pérez Benitez, 2014) se plantea que en nuestro país se realiza un proceso de migración a software libre, siendo una vía factible por ser un país bloqueado, lo que constituye una alternativa para superarse económicamente.

Nacionales

Los sistemas nacionales relacionadas con la generación del horario docente encontrados son: Sistema Informático para la Confección de Horarios Docentes en la Facultad de Informática de la Universidad de Cienfuegos (Echevarría Cartaya, 2009), Sistema de planificación y publicación de horarios docentes usado en la Universidad de las Ciencias Informáticas (Barrera, 2011) y el

sistema de la antigua Facultad Regional de la Universidad de las Ciencias Informáticas de Granma (Varona, 2012). Estos sistemas no incluyen las funcionalidades relacionadas con la reservación de locales, cambio de turnos y asignación de locales para exámenes finales, se desarrollaron para instancias específicas del PPHD y no se ajustan a las características y restricciones propias de la Facultad de Ingeniería Industrial de la CUJAE. El Sistema para la generación de horario docente de la CUJAE tampoco incluye las funcionalidades antes mencionadas, pero cumple con todas la restricciones fuertes de la institución, por lo que el presente trabajo propone la incorporación estas nuevas funcionalidades al sistema de la Facultad de Ingeniería Industrial de la CUJAE.

1.5 Métodos de solución

Dentro del contexto del PPHD se puede evidenciar el problema de la programación de horarios de evaluaciones y exámenes (Examination Timetabling) (Hernández, y otros, 2008), aplicándose en estos casos diferentes métodos de solución ofrecidos por el área de inteligencia artificial como la programación de restricciones, que según (Manyá, y otros, 2003) se define como el estudio de sistemas computacionales basados en restricciones. La idea de la programación por satisfacción 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. Se considera un método genérico de resolución de problemas que ha demostrado ser altamente competitivo en áreas tan diversas como inteligencia artificial, investigación operativa, bases de datos y sistemas de recuperación de la información.

Para incluir las nuevas funcionalidades al Sistema para la generación de horario docente de la CUJAE se selecciona la programación por satisfacción de restricciones, para encontrar una asignación de valores para todas las variables que no viole ninguna restricción fuerte como se plantea en (Dechter, y otros, 2007).

1.6 Metodología de desarrollo de software

Según el autor (Piattini Velthuis, 1996) las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. Rigen el proceso de desarrollo de software, con el fin de hacerlo más predecible y eficiente, tienen como objetivo principal aumentar la calidad del software que se produce en cada una de sus fases de desarrollo, de acuerdo a sus características y los objetivos que persiguen se pueden clasificar en tradicionales y ágiles (Jacobson, y otros, 2000), (Robert C, 2002), (Pressman, 2005), (Shore, y otros, 2008), (Pressman, 2005) y (Rivadeneira Molina, 2012).

Metodologías tradicionales o robustas: centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto. Presentan altos costes al implementar un cambio y tienen una falta de flexibilidad en proyectos donde el entorno es volátil.

Las metodologías tradicionales se focalizan en la planificación, control del proyecto, especificación de requisitos y en el modelado (INTECO, 2009). Son eficaces y necesarias para proyectos de gran envergadura que requieren de mucho tiempo y recursos, donde la organización es de vital importancia para su desarrollo. Algunas de las metodologías robustas más conocidas son: Rational Unified Process (RUP), Microsoft Solution Framework (MSF), entre otras.

Metodologías ágiles o ligeras: son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto. La comunicación con el cliente es constante al punto de requerir un representante de él durante el desarrollo. Los proyectos son altamente colaborativos y se adaptan mejor a los cambios, al igual que las entregas constantes al cliente y la retroalimentación por parte de él. Tanto el producto como el proceso son mejorados frecuentemente (Navarro Cadavid, y otros, 2013). Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan según se describe en (INTECO, 2009). Algunas de las metodologías ágiles son: SCRUM, Extreme Programming (XP), Crystal Clear, Feature-Driven Development (FDD), Dynamic Systems Development Method (DSDM), entre otras.

A continuación se presenta una tabla comparativa entre estas dos clasificaciones de metodologías desarrollada por (INTECO, 2009)**.**

Criterios	Metodologías ágiles	Metodologías tradicionales
Cambios en los requisitos.	Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Control del proyecto.	Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
Interacción con el cliente.	El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Integrantes.	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Generar artefactos.	Pocos artefactos.	Muchos artefactos.
Roles de proyecto.	Pocos roles.	Muchos roles.
Arquitectura de software.	Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 1. Comparación entre las metodologías agiles y tradicionales

Se opta por usar una metodología ágil, ya que el equipo de desarrollo es pequeño y necesita de pocos roles, la investigación está dirigida a satisfacer al cliente mediante la entrega continua de software funcional, manteniendo una constante comunicación entre el cliente y el equipo de

desarrollo. A continuación se describen dos de las metodologías de desarrollo de software más referenciadas en la literatura.

- **SCRUM:** es un marco de trabajo basado en la teoría de control de procesos empíricos. Emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo. No es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varias técnicas y procesos (Schwaber, y otros, 2013). Según (Proyectos agiles, 2015) está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.
- **Programación extrema (XP):** es una metodología ágil para pequeños y medianos equipos, desarrollando software cuando los requerimientos son ambiguos o rápidamente cambiantes. A diferencia de los procesos tradicionales para desarrollar software, XP asume el cambio como algo natural, y que en alguna etapa de un proyecto sucede. En XP se realiza el software que el cliente solicita y necesita, en el momento que lo precisa, alentando a los programadores a responder a los requerimientos cambiantes que plantea el cliente en cualquier momento. Esto es posible porque está diseñado para adaptarse en forma inmediata a los cambios, con bajos costos asociados, en cualquier etapa del ciclo de vida (Calabria, y otros, 2003).

A continuación se presenta una tabla comparativa entre estas dos metodologías desarrollada por (Mendes Calo, y otros, 2009).

Criterios	XP	SCRUM
Fechas de entrega.	Las iteraciones de entrega son de una a tres semanas.	Las iteraciones de entrega son de dos a cuatro semanas.
Valorar la interacción del equipo por sobre el proceso.	Los miembros de XP trabajan en dúos.	Los miembros de SCRUM trabajan de forma independiente.
Valorar la respuesta al cambio.	Permite introducir cambios en la iteración en curso.	Permite la evolución y el cambio, pero no es recomendable en la iteración en curso.
Valorar el seguimiento de un plan.	Define un plan detallado para cada iteración, que puede ser modificado.	Define un plan detallado de iteraciones, no acepta cambios durante una iteración.
Valorar desarrollo de software que funcione.	Generar entregable con prueba satisfactoria e integrada con el resto de las funciones al finalizar cada iteración.	Generar entregable con prueba satisfactoria al finalizar cada iteración.

Tabla 2. Comparación entre las metodologías agiles XP y SCRUM

Debido a las características del equipo de trabajo, compuesto por dos desarrolladores, en constante colaboración con el cliente, admitiendo cambios de última hora en las iteraciones del sistema, con el objetivo de favorecer las entregas rápidas del software funcional en un periodo que no excede los seis meses y la documentación que se generara debe ser la mínima necesaria, se opta por el uso de un enfoque de desarrollo ágil, escogiéndose la Programación Extrema como metodología de desarrollo.

1.7 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar instrucciones que pueden ser llevadas a cabo por un ordenador. Puede usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión o como modo de comunicación humana. Permiten especificar de manera precisa sobre qué datos debe operar una computadora, cómo deben ser almacenados o transmitidos y qué acciones debe tomar bajo una gran cantidad de opciones posibles. Todo esto, a través de un lenguaje que intenta ser relativamente próximo al lenguaje humano o natural (Suárez, 2015). Según (Roque Cuevas, 2007) se denomina lenguaje fuente a las órdenes que escribe el programador, las cuales son traducidas al lenguaje máquina de la computadora. Cada lenguaje de programación tiene su propia gramática.

Hypertext Pre-processor (PHP): es un lenguaje de código abierto muy popular especialmente adecuado para al desarrollo de aplicaciones web dinámicas y que puede ser incrustado en HTML. El código fuente escrito en PHP es invisible al navegador y al usuario, ya que es el servidor quien se encarga de ejecutar el código y enviar su resultado HTML al navegador. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP. Presenta una extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales (PHP, 2015). Según (Suárez, 2015) presenta una mayor capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad (MySQL y PostgreSQL). Tiene la capacidad de expandir su potencial utilizando una enorme cantidad de módulos. Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.

Se opta por el uso de este lenguaje en su versión 5.5.9, debido a su potencial para el trabajo en el entorno web.

HyperText Markup Language (HTML5): provee básicamente tres características: estructura, estilo y funcionalidad, es considerado el producto de la combinación de HTML[1], CSS[2] y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML 5: HTML provee los elementos estructurales, CSS se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista, y Javascript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales (Gauchat, 2012). En (Miguel Ángel Álvarez, 2009) se explica que Incluyen novedades significativas en diversos ámbitos. No sólo se trata de incorporar nuevas etiquetas o eliminar otras, sino que supone mejoras en áreas que hasta ahora quedaban fuera del lenguaje y para las que se necesitaba utilizar otras tecnologías, entre ellas se encuentran:

- Estructura del cuerpo: la mayoría de las webs tienen un formato común, formado por elementos como cabecera, pie, navegadores, etc.
- HTML 5 permite agrupar todas estas partes de una web en nuevas etiquetas que representarán cada uno de las partes típicas de una página.
- Bases de datos locales: el navegador permitirá el uso de una base de datos local, con la que se podrá trabajar en una página web por medio del cliente y a través de un API, lo que permitirá la creación de aplicaciones web que funcionen sin necesidad de estar conectados a internet.
- Fin de las etiquetas de presentación: todas las etiquetas que tienen que ver con la presentación del documento, es decir, que modifican estilos de la página, serán eliminadas. La responsabilidad de definir el aspecto de una web correrá a cargo únicamente de CSS.

JavaScript: es un lenguaje de programación que se utiliza principalmente del lado del cliente, implementado como parte de un navegador web. Técnicamente Java Script es un lenguaje de programación interpretado por lo que no es necesario compilar el código para ejecutarlos. En otras palabras, los programas escritos con Java Script se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Además está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, estaciones de trabajo y sobre arquitecturas distintas o con sistemas operativos diversos (Eguiluz, 2015). Según (MDN, 2015) es un lenguaje de script multiplataforma que amplía las ventajas del HTML al trabajar en entornos web, aporta dinamismo y velocidad, permitiendo realizar mejoras en la interfaces de usuario. Es pequeño y ligero, está diseñado para una fácil incrustación en otros productos y aplicaciones. Es mucho más liberado que el Java ya que no tiene que declarar todas las variables, clases y métodos. No debe preocuparse si sus métodos son públicos, privados o protegidos y no

[1] HyperText Markup Language (lenguaje de marcas de hipertexto)

[2] Cascading Style Sheets (Hoja de estilos en cascada)

tiene que implementar sus interfaces. Los tipos de variables, parámetros y funciones de retorno no son explícitamente definidos.

Se opta la utilización de este lenguaje debido a las características antes mencionadas, además que se trabaja en conjunto con HTML5 para mejorar el trabajo en entornos web, permitiendo crear interfaces amigables para los usuarios del sistema.

Cascading Style Sheets (CSS3): es un lenguaje que trabaja junto a HTML para proveer estilos visuales a los elementos del documento como tamaño, color, fondo y borde (Gauchat, 2012). Ofrece nuevas posibilidades para crear impacto con sus diseños. Permite el uso de hojas de estilo más diversas para variedades de ocasiones, entre otras características que harán la aplicación más amigable para los usuarios. La novedad más importante que aporta CSS 3, de cara a los desarrolladores de webs, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos, que a menudo complicaban el código de las web. Entre las principales características de CSS 3 se encuentran: bordes redondeados, con degradados, imágenes; cajas con sombra; múltiples columnas; texto con sombra; cajas redimensiónales. También incluye mejoras en el uso de la opacidad (pudiéndose fijar a una caja, imagen o texto), selectores y personalizado de fuentes (CSS3, 2015).

1.8 Herramientas

Las herramientas garantizan la velocidad de respuesta y estabilidad de los resultados que se esperan obtener de un sistema. La combinación de diferentes herramientas permite soportar el proceso de desarrollo de software.

PhpStorm: cuenta con un editor de código rico para PHP que entiende su código y estructura profundamente, soporta PHP 5.3, 5.4, 5.5 y 5.6 para los proyectos modernos y antiguos. El entorno de desarrollo integrado proporciona finalización de código inteligente, resaltado de sintaxis, la configuración de formato de código extendido, sobre la marcha de la comprobación de errores, plegado de código, apoya las mezclas de idiomas y más. Posee gran disponibilidad de complementos, transfiriendo diversas características a marcos específicos, como finalización de código, la navegación, tipos de inferencia y otras mejoras prácticas para varios marcos de PHP (Symfony, Drupal, Magento, etc.). Utiliza el mismo entorno de desarrollo integrado en Windows, Mac OS X y Linux con su clave única (JetBrains, 2013).

Se opta por el uso de esta herramienta en su versión 8.0.2 por las características antes mencionadas, su buen rendimiento en múltiples plataformas y consumo racional del hardware de la computadora.

PostgreSQL 9.3: es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD[3]. Según (Stinson, 2001) permite que mientras un proceso escriba en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último que se ejecutó. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos. En (PostgreSQL, 2013) se plantea que utiliza un modelo cliente-servidor y multiproceso en vez de multihilo para garantizar la estabilidad del sistema. La aplicación de escritorio pgAdmin III se utiliza como interfaz gráfica para hacer más fácil la administración del usuario con la base de datos, además se puede ejecutar en varias plataformas como: Linux (todas las distribuciones recientes), Windows (Win2000 SP4 y posterior), FreeBSD, OpenBSD, NetBSD, Mac OS X, AIX, HP / UX, IRIX, Solaris, Tru64 Unix y UnixWare. Otros sistemas Unix también pueden funcionar pero no están siendo probados actualmente.

Las características más importantes a destacar en la versión 9.3, según (PostgreSQL, 2013):

- Conectores de datos foráneos modificables
- Sumas de verificación de páginas de datos
- Failover rápido
- Streaming-only remastering[4]
- Métodos constructores y extractores adicionales para JSON
- Vistas actualizables automáticas
- pg_dump en paralelo para acelerar el respaldo de bases de datos muy grandes
- Lateral JOINs
- Procesos en segundo plano definidos por el usuario

Se opta la utilización de esta herramienta debido a las características antes mencionadas.

JMeter: es una herramienta de código abierto, una aplicación de Java pura 100%, diseñada para pruebas de comportamiento funcional de la carga y medir el desempeño. Puede ser utilizado para simular una carga pesada en un servidor, grupo de servidores, red u objeto para poner a prueba su resistencia o para analizar desempeño bajo carga de diferentes tipos (Apache, 2015). Según (EJie, 2009) también ofrece la posibilidad de activar un Proxy web. Gracias a esto se puede grabar la navegación de un usuario para posteriormente usarla en la generación de una prueba.
Se opta por el uso de esta herramienta para realizar las pruebas de carga y estrés del sistema.

Apache: es un servidor HTTP de código abierto para sistemas operativos modernos, incluyendo UNIX y Windows NT. Proporciona un servidor seguro, eficiente y extensible que proporciona

[3] Berkeley Software Distribution (Distribución de Software Berkeley)

[4] Transmisión de sólo remasterización

servicios HTTP en sincronización con los estándares HTTP actuales. Mejora de almacenamiento en caché y soporta grandes cantidades de información (Apache, 2015). Presenta, entre otras características, mensajes de error altamente configurables, es flexible y de diseño modular, presenta bases de datos de autenticación y negociado de contenido, tiene una amplia aceptación en la red: en el 2005, fue el servidor HTTP más usado, siendo el servidor empleado en el 70% de los sitios web en el mundo (Bowen, 2007).

Se opta por el uso de esta herramienta ya que la arquitectura de este servidor es modular y las principales metas de su diseño son: velocidad, simplicidad, multiplataforma y facilidad de desarrollo distribuido.

Visual Paradigm for UML: es una herramienta muy completa, fácil de usar y con soporte multiplataforma. Su uso es sencillo para la creación de todo tipo de diagramas UML, para los que dispone de un número considerable de estereotipos, permitiendo mayor entendimiento de los mismos (INEI, 2015). Según lo especificado en (Visual Paradigm, 2000) permite visualizar el flujo central detallado de cada proceso mediante diagramas, posibilitando la obtención de los mismos definidos por la metodología escogida, entre ellos el diagrama de clase, el modelo de datos, etc...

Algunas características que presenta son:

- Navegación intuitiva entre la escritura del código y su visualización.
- Generador de informes en formato PDF/HTML.
- Documentación automática Ad-hoc.
- Generación de bases de datos: transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Interoperabilidad con modelos UML.
- Ingeniería inversa de bases de datos: desde sistemas gestores de bases de datos (DBMS) existentes a diagramas de Entidad-Relación.

Se opta por utilizar la herramienta CASE con la que se ha venido trabajando en el desarrollo de la aplicación existente en su versión 5.0.

Bootstrap: es un marco de trabajo desarrollado para crear interfaces de aplicaciones web y fomentar la consistencia a través de herramientas internas. Los diseños implementados con el marco son intuitivos, fáciles y ligeros, lo que propicia una mejora notable para adaptarse a casi cualquier dispositivo como: teléfonos móviles, Tablet, computadoras, etc. Utiliza CSS3 y JQuery lo que permite un mejor manejo de eventos y animaciones en la aplicación (Thornton, y otros, 2012).

Se opta por la utilización de este marco de trabajo en su versión 2.2.3 debido a las características antes mencionadas.

Mozilla Firefox: es un navegador web multiplataforma y de código abierto. Usa el motor Gecko para renderizar páginas webs, el cual implementa actuales y futuros estándares web. Firefox integra numerosas funcionalidades por defecto que se pueden ampliar gracias a un catálogo de extensiones muy largo. Ha sido nombrado rey de la velocidad en unas pruebas independientes que miden el rendimiento respecto a otros navegadores. Provee una conexión segura mediante el uso de identificadores para verificar la identidad y comprobar que la conexión es segura. Incorpora características que permiten combatir la suplantación de identidad y el software malintencionado. Se actualiza automáticamente para garantizar que se tengan las soluciones de seguridad más recientes e importantes (Mozilla, 2015).

Se opta por la utilización de esta herramienta debido a las características antes mencionadas.

1.9 Patrones de diseño

Los patrones de diseño son herramientas que proveen facilidades para crear un software reutilizable de buena calidad. Cada patrón describe un problema que ocurre repetidamente en nuestro entorno, y describe el núcleo de la solución a ese problema, de tal forma que ésta pueda ser usada un millón de veces, sin hacer el mismo trabajo dos veces (Asenjo González, y otros, 2003).

En el desarrollo de los módulos Cambio de turnos, Reservación de locales y Asignación de locales para exámenes finales los patrones de base de datos son utilizados para solucionar problemas del diseño de la base de datos, los GRASP[5] se utilizan para describir los principios fundamentales de la asignación de responsabilidades a objetos, mientras que los GoF[6] se encargan de solucionar problemas de composición de clases y objetos, creación de instancias, integración y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan para el diseño básico del sistema. A continuación se presentan los aplicados en el desarrollo de los módulos.

Patrones GRASP

- **Experto:** el patrón experto en información se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. Este permite conservar el encapsulamiento, ya que los objetos se valen de su propia información para llevar a cabo las tareas. El comportamiento se distribuye entre clases que cuentan con la información requerida, alentando con ello definiciones de clase sencillas y más cohesivas que son más fáciles de comprender y mantener. Así se brinda soporte a una alta cohesión (Buono, 2012).

[5] General Responsibility Assignment Software Patterns (Patrones generales de software para asignación de responsabilidades)

[6] Gang of Four (La banda de los cuatro)

- **Creador:** el patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización (Larman, 1999) y (Visconti, et al., 2004).
- **Controlador:** el patrón controlador es un objeto que no pertenece a la interfaz de usuario, es un manejador artificial de todos los eventos del sistema que define el método de su operación. A menudo genera un bajo acoplamiento y permite un mayor potencial de los componentes reutilizables (Visconti, y otros, 2004). En (Larman, 1999) se plantea que los objetos externos de conexión (los objetos ventana y pequeñas aplicaciones) y la capa de presentación no deberían tener la responsabilidad de llevar a cabo los eventos del sistema.
- **Bajo acoplamiento:** consiste en tener las clases lo menos relacionadas entre sí, para que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión en el resto de las clases. Esta característica permite potenciar la reutilización y disminuye la dependencia entre las clases (USM.cl, 2015).
- **Alta cohesión:** en la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (Larman, 1999) y (Visconti, et al., 2004).

Patrones GoF:

- **Fachada:** la clase definida que ofrece una interfaz común con un conjunto heterogéneo de interfaces. Las interfaces heterogéneas pueden ser un conjunto de funciones, un esquema, un grupo de otras clases o un subsistema.
- **Decorador:** permite añadir responsabilidades extra a objetos concretos de manera dinámica. Proporciona una alternativa flexible a la herencia estática para extender funcionalidad. Evita que las clases más altas en la jerarquía estén demasiado cargadas de funcionalidad y sean complejas (Sánchez, 2010).

Patrones de base de datos

Los patrones de diseño de bases de datos son plantillas que han sido evaluadas como las responsables de resolver un problema. Constituyen la base para la búsqueda de soluciones a problemas comunes en el proceso de diseño de las mismas (Blaha, 2010).

Se utiliza el patrón de base de datos llaves subrogadas, que consiste en asignar una llave o identificador único para cada entidad cuyo único requisito es almacenar un valor numérico único

para cada fila de la tabla, actuando como una clave sustituta, de forma totalmente independiente a los datos de negocio (GuilleSQL, 2011).

1.10 Validación de los requisitos

La validación de requisitos sirve para demostrar que éstos realmente definen el sistema que el cliente desea. Asegura que los requerimientos están completos, son exactos y consistentes. Debe garantizar que lo descrito es lo que el cliente pretende ver en el producto final. Esta validación es importante porque la detección de errores durante el proceso de análisis de requerimientos reduce mucho los costos. Si se detecta un cambio en los requerimientos una vez que el sistema está hecho, los costos son muy altos, ya que significa volver a cambiar el diseño, modificar la implementación del sistema y probarlo nuevamente (Gómez Fuentes, 2011). Para llevar a cabo este proceso, se aplicaron las siguientes técnicas de validación de requisitos:

- ✓ **Revisión de requisitos:** Se realizan reuniones para localizar errores en el documento. Donde se agregaron requisitos y se modificaron otros.
- ✓ **Generación de casos de prueba de aceptación*:*** para validar los requisitos funcionales de la solución, se diseñan casos de pruebas de aceptación para cada una de las Historias de Usuario.

1.11 Métricas de requisitos

Las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo de software y los proyectos de mantenimiento (Briand, 1996) y pueden ser utilizadas por profesionales e investigadores para tomar mejores decisiones (Pfleeger, 1997). A continuación se presentan las métricas a tener en cuenta para la validación de los requisitos del sistema.

Métrica de estabilidad

El objetivo de esta métrica es medir la estabilidad de los requerimientos para asegurar su adecuación antes de pasar al próximo flujo de trabajo. Se considera que los requerimientos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación. La estabilidad de los requerimientos se calcula como (Alcantara Rabí, y otros, 2010):

$$\boldsymbol{ETR = \left[\frac{RT - RM}{RT}\right] * 100}$$

- ERT: valor de la estabilidad de los requerimientos.
- RT: total de los requerimientos definidos.
- RM: número de requerimientos modificados, que se obtienen como la sumatoria de los requerimientos insertados, modificados y eliminados.

Esta métrica ofrece valores entre 0 y 100. El mejor valor de ETR es el más cercano a 100 ya que mostrará que no se están realizando cambios sobre los requisitos, son estables y por tanto es confiable trabajar el diseño sobre ellos.

Métrica de especificidad

Según (Pressman, 2005) el objetivo de esta métrica es cuantificar la especificidad o la ausencia de ambigüedad en la definición de los requisitos. Para calcular esta métrica deben contarse los requisitos que tuvieron igual interpretación por los revisores y compararlos con el total de requisitos definidos. La misma se calcula como:

$$Q_1 = \frac{n_{ui}}{n_r}$$

nui: número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

nr: cantidad de requisitos en una especificación y comprende la suma de los requisitos funcionales y no funcionales. Cuanto más cerca de 1 esté el valor de Q1, menor será la ambigüedad de la especificación.

Grado de validación de los requisitos

Los requisitos deben ser posibles de validar. La validación de los requisitos se realiza en consenso del equipo de desarrollo al contrastar lo que desea el cliente con la posibilidad real de implementarlo. El grado de validación de los requisitos mide la corrección en la definición de los requisitos (Alcantara Rabí, y otros, 2010).

$$V_R = \frac{n_c}{n_c + n_{nv}}$$

VR: grado de validación de los requisitos.

nc: número de requisitos que se han validado como correctos.

nnv: número de requisitos no validados aún.

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo de esta métrica es el más cercano a 1 e indica un alto nivel de corrección en la definición de los requisitos.

1.12 Validación del diseño

Para evaluar la calidad del diseño se opta por el uso de las métricas orientada a clases: Tamaño operacional de clase (TOC) y Relaciones entre clases (RC), para la validación del diseño del sistema.

TOC: está dada por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad (CUFM, 2012):

- **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.

A continuación se muestra en la siguiente tabla la aplicación de la métrica TOC, donde CP se refiere a la cantidad de procedimientos:

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

Tabla 3. Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC

RC: está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad (EcuRed, 2013):

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

A continuación se muestra en la siguiente tabla la aplicación de la métrica RC, donde CRU se refiere a la cantidad de relaciones de uso.

Atributo de calidad	Categoría	Criterio
Acoplamiento	Baja	1
	Media	2
	Alta	>2
Complejidad de mantenimiento	Baja	CRU<= Promedio
	Media	Promedio<=CRU<=2* Promedio
	Alta	CRU> 2* Promedio
Reutilización	Baja	CRU>2* Promedio
	Media	Promedio<=CRU<= 2* Promedio
	Alta	CRU<= Promedio
Cantidad de pruebas	Baja	CRU<= Promedio
	Media	Promedio<=CRU<=2* Promedio
	Alta	CRU> 2* Promedio

Tabla 4. Rangos de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC

1.13 Pruebas

La (IEEE, 2015) 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 descubrir defectos en el software y demostrar que el mismo que satisface los requerimientos establecidos con el cliente.

Según se explica en (Pressman, 2005) la estrategia de prueba a seguir posee un enfoque incremental, inicia con la prueba de unidades individuales del programa, pasa a pruebas diseñadas para facilitar la integración de las unidades, y culmina con las pruebas que se realizan sobre el sistema construido. A continuación se presentan los tipos de pruebas a realizar al sistema.

Pruebas unitarias

Se concentran en el esfuerzo de verificación de la unidad más pequeña del diseño del software: el componente o módulo de software. Tomando como guía la descripción del diseño a nivel de componentes, se prueban importantes caminos de control para descubrir errores dentro de los límites de los componentes (Pressman, 2005).

Prueba de caja blanca

En (Pressman, 2005) se plantea que la prueba de caja blanca del software se basa en un examen cercano al detalle procedimental. Se prueban las rutas lógicas del software y la colaboración entre componentes, al proporcionar casos de pruebas que ejerciten conjuntos específicos de

condiciones, bucles o ambos. Al emplear los métodos de prueba de caja blanca el ingeniero del software podrá derivar casos de prueba que:

- Garanticen que todas las rutas independientes dentro del módulo se han ejercido al menos una vez.
- Ejerciten los lados verdadero y falso de todas las decisiones lógicas.
- Ejecuten todos los bucles en sus límites y dentro de sus límites operacionales.
- Ejerciten estructuras de datos internos para asegurar su validez.

Para ejecutar este tipo de pruebas según (Pressman, 2005) se utilizará la **técnica de flujo básico:** permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Para aplicar esta prueba se han definido en una serie de pasos a seguir que a continuación se describen:

- **Notación del grafo de flujo:** usando el código como base, se realiza la representación del grafo de flujo (grafo del programa) mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo: los nodos que representa una o más sentencias procedimentales, las aristas representan flujo de control y son análogas a las flechas del diagrama de flujo y las regiones, áreas delimitadas por aristas y nodos.
- **Complejidad ciclomática:** es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa y propone que se realice al menos un caso de prueba por cada camino independiente, de manera que se asegure la ejecución de cada sentencia como mínimo una vez. La complejidad ciclomática V (G), se calcula de tres formas:
    - Número de regiones del grafo de flujo.
    - V (G) = A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
    - V (G) = P+ 1, donde P es el número de nodos predicado contenidos en el grafo de flujo G.

  Determinar un conjunto básico de caminos linealmente independientes: el valor de V (G) coincide con el número de caminos linealmente independientes de la estructura de control del programa.

  Obtención de casos de prueba: se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico.

Prueba de caja negra

Las pruebas de caja negra se concentran en las HU del sistema, 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:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en la estructura de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Según (Pressman, 2005) para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

- **Técnica de la partición de equivalencia:** divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Técnica del análisis de valores límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Técnica de grafos de causa-efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Según se establece en (Pressman, 2005) dentro del método de caja negra la técnica de la partición de equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. Se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de prueba que hay que desarrollar.

Pruebas de integración

Según (Pressman, 2005) la prueba de integración es una técnica sistemática para construir la arquitectura del software mientras, al mismo tiempo, se aplican pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar componentes a los que se le aplicó una prueba de unidad y construir una estructura de programa que determine el diseño.

En integración incremental el programa se construye y se prueba en pequeños incrementos, en los cuales resulta más fácil aislar y corregir los errores. A continuación se presentan diferentes estrategias de integración incremental.

- **Integración descendente:** es un enfoque incremental para la construcción de la arquitectura del software. Los módulos se integran al descender por la jerarquía de control, empezando con el módulo de control principal. Los módulos subordinados al módulo de

control principal se incorporan en la estructura de una de dos maneras: primero en profundidad o primero en anchura.

- **Integración ascendente:** empieza la construcción y prueba con módulos atómicos. Debido a que los módulos se integran de abajo hacia arriba, siempre están disponibles al procesamiento requerido para los módulos subordinados a un determinado nivel y se elimina la necesidad de resguardo.

Para realizar la prueba de integración a los módulos Reservación de locales, Cambio de turnos y Asignación de locales para exámenes finales se decide emplear la estrategia de integración incremental ascendente.

Prueba de carga y estrés

Las pruebas de resistencia o estrés están diseñadas para enfrentar a los programas con situaciones anormales, es decir, ejecutan un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales. Evalúan el comportamiento del sistema en casos extremos de uso y situaciones como múltiples instancias del sistema, diversas peticiones de elementos al mismo tiempo o la utilización del sistema en redes saturadas de manera que produzcan inestabilidad o procesamiento incorrecto (Pressman, 2002).

Prueba de aceptación

Este tipo de pruebas se desprenden directamente de las especificaciones de los requerimientos del usuario, verificando que el software realiza lo especificado (reglas de negocio). Se describe un escenario de ejecución o uso del sistema desde la perspectiva del usuario, teniendo en cuenta requisitos funcionales o no funcionales. Cada uno de los requisitos puede tener una o más pruebas de aceptación (Díaz, y otros, 2009).

1.14 Conclusiones parciales

A partir de la investigación realizada se evidencia la necesidad de incorporar al sistema existente en la Facultad de Ingeniaría Industrial de la CUJAE los módulos Cambio de turnos, Reservación de locales y Asignación de locales para exámenes finales.

Se opta por un enfoque de desarrollo ágil, seleccionándose a XP como metodología de desarrollo de software. Los lenguajes y herramientas de desarrollo seleccionadas contribuyen al desarrollo de los módulos mencionados. Las métricas y los tipos de pruebas seleccionados permiten identificar deficiencias en el diseño y en la implementación de las funcionalidades del sistema, con el objetivo de solventarlas y obtener un sistema que contribuya a solucionar el problema de la investigación.

CAPÍTULO 2: Propuesta de solución

2.1 Introducción

En este capítulo se describen los aspectos fundamentales de la solución propuesta de acuerdo a la metodología de software seleccionada, teniendo en cuenta las funcionalidades del sistema, la arquitectura, los patrones de diseño y las restricciones fuertes y débiles que se tienen en cuenta en el sistema.

2.2 Descripción del negocio actual

El horario docente de la Facultad de Ingeniería Industrial de la CUJAE afronta el problema de ser vulnerable a cambios de última hora. La ejecución de un cambio en el horario docente conlleva a un análisis de las posibles repercusiones que pueda traer consigo, ya que los cambios atentan contra el orden de la secuencia de actividades e implican que se tenga que rehacer el horario. Al realizar un cambio se debe tener en cuenta las restricciones fuertes y débiles establecidas por la institución.

A continuación se muestran las restricciones que se tendrán en cuenta en la solución, son las utilizadas para el desarrollo del Sistema para la generación de horario docente de la Facultad de Ingeniería Industrial de la CUJAE.

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.

Restricciones débiles

- Un grupo no puede tener clases en un local que tenga una capacidad inferior a su matrícula.
- El horario debe ser compacto, no debe dejar turnos intermedios sin planificar.
- 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 se deben planificar en la sección de clases que corresponde.

Algunos de los problemas detectados en la realización de cambios en el horario son los siguientes: la reservación de un local por más de un profesor en el mismo día y turno, el intercambio de turnos de clases sin tener en cuenta la capacidad de los locales, la asignación de un local en el que se tienen actividades docentes planificadas, entre otras. Estos cambios actualmente se realizan de forma manual, propiciando la ocurrencia de errores por el factor humano, pues son numerosos los datos y restricciones que se deben tener en cuenta.

2.3 Abreviaturas del sistema

Asignaturas de primer año			
Semestre1		**Semestre2**	
Matemática 1	M1	Matemática 2	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 industria	II
Introducción a la ingeniería	II	Educación física II	EF
Educación física I	EF		

Tabla 5. Abreviatura de las asignaturas de primer año

Asignaturas de segundo año			
Semestre1		**Semestre2**	
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 6. Abreviatura de las asignaturas de segundo año

Asignaturas de tercer año			
Semestre1		**Semestre2**	
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 7. Abreviatura de las asignaturas de tercer año

Asignaturas de cuarto año			
Semestre1		**Semestre2**	
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 8. Abreviatura de las asignaturas de cuarto año

Asignaturas de quinto año	
Semestre1	
Gestión del cambio organizacional	GCO
Gestión de la calidad	GC
Logística II	L2
Distribución en planta	DP
Gestión ambiental	GA

Tabla 9. 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 10. Frecuencia de actividades

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

Tabla 11. Clasificación de locales

2.4 Fase I: Exploración

Según (Letelier Torres, 2003) los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema.

2.4.1 Historia de usuario

La historias de usuario (HU) sirven para registrar los requerimientos de los clientes y son utilizadas para crear las pruebas de aceptación y realizar la estimación de cada una de las iteraciones durante la fase de planificación como se plantea en (Calabria, y otros, 2003). Describen brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales en base a lo que se estima necesario para el sistema. El tratamiento de las HU es muy dinámico y flexible. Cada una es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas, así se asume en (Grupo ISSI, 2003). A

continuación se describe la HU encargada de listar los locales disponibles, las restantes se encuentran en el Anexo 1.

Historia de usuario	
Número:1	**Nombre:** Listar locales disponibles
Usuario: planificador(a)	**Iteración asignada:** 1
Prioridad en negocio: alta	**Puntos estimados:** 1
Riesgo en desarrollo: medio	**Puntos reales:** ½
Descripción: Una vez generado el horario el usuario podrá visualizar los locales no han sido utilizados en las actividades docentes establecidas en el horario de acuerdo a los criterios de búsqueda seleccionados: • fecha y local, puede seleccionar un local específico o todos. (no pongo la opción de no seleccionar ningún local porque si no selecciona uno se muestran todos).	
Observaciones: • Por cada local se debe mostrar el turno que está disponible. • Se mostrarán solamente los locales en los cuales no se desarrollen talleres (porque solamente se utilizan en turnos y días específicos durante el semestre). • No se mostrarán los locales disponibles en los días feriados, festivos, no laborables ni afectados.	

Tabla 12. HU Listar locales disponibles

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 (ver siguiente epígrafe: Personas relacionadas con el sistema)
- **Iteración asignada:** iteración en que se implementará la HU
- **Prioridad en el negocio:** prioridad de la HU con respecto al resto de las HU (alta, media o baja)
- **Riesgo en desarrollo:** riesgo en la implementación de la HU (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 uno a tres puntos.
- **Puntos reales:** resultado del esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, generalmente de uno a tres puntos.
- **Descripción:** descripción sintetizada de la HU.
- **Observaciones:** información adicional.

2.4.2 Personas relacionadas con el sistema

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

Persona	Descripción
Usuario:	Persona que accede al sistema, sin tener que autenticarse, para consultar el horario docente y los exámenes finales.
Planificador(a):	Persona que ha sido autenticada con la posibilidad de acceder las funcionalidades del sistema.

Tabla 13. Personas asociadas al sistema

2.4.3 Requisitos no funcionales del sistema

Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación como restricciones en el diseño o estándares de calidad. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. No son parte de la razón fundamental del producto pero si son necesarios para hacer funcionar el producto de la manera deseada según (Ecured, 2015). La metodología XP no incluye la descripción de estos requisitos en las HU, debido a que los clientes generalmente no están familiarizados con las terminologías técnicas que se utilizan para describir las mismas, siendo el equipo de desarrollo el encargado de capturar estos requisitos a partir del intercambio de información con el cliente. A continuación se presentan los requisitos no funcionales definidos:

Requisito de usabilidad: el sistema debe ser usado por personas que tengan conocimientos básicos de informática y deben alcanzar su máxima productividad en una semana.

Requisitos de fiabilidad:

- Las transacciones deben demorar, como promedio, entre uno y seis segundos aproximadamente, en dependencia de la cantidad de estudiantes accediendo al sistema simultáneamente. Se debe tener en cuenta los requisitos de hardware requeridos en la estación de trabajo y en el servidor.
- Son requeridos los siguientes recursos tecnológicos.
  Estación de trabajo:
    - Software: Navegador con soporte para HTML 5 y CSS 3.
    - Hardware**:**
      256Mb de memoria RAM
      Procesador Pentium IV
      Conexión de red

  Servidor:
    - Hardware**:**

512Mb de memoria RAM
Procesador de Pentium IV
Disco duro de 80Gb

Requisito de soporte: se mantendrá un sistema de codificación estándar siguiendo las normas establecidas.

Requisitos de interfaz: el sistema debe ofrecer una interfaz amigable y fácil de operar, la tipografía debe ser uniforme, de un tamaño adecuado y con un contraste que resalte los textos. Se debe mantener la línea de diseño establecida, con la uniformidad y representatividad de la solución. Todas las interfaces deben estar en idioma español.

Requisitos de portabilidad: las herramientas podrán ser usadas bajo cualquier sistema operativo de Windows NT en adelante o cualquier distribución de Linux.

Requisitos de disponibilidad: el sistema debe estar disponible las 24 horas del día, los 365 días del año desde todos los puntos donde exista una máquina conectada a la red universitaria.

Requisitos de integración:

- Se debe integrar el módulo Reservar locales al módulo Generar horario, de forma que al realizar una reservación los locales no se encuentren afectados por un encuentro[7].
- Se debe integrar el modulo Cambio de turnos al módulo Generar horario, de forma que el cambio realizado quede reflejado en el horario docente generado.
- Se debe integrar al módulo Asignaturas el módulo Asignación de locales para exámenes finales, de forma que se muestre la planificación de los exámenes en el sistema.

Requisitos de seguridad:

Autenticación:

- Se establecerán mecanismos de autenticación personalizada para la(s) persona(s) involucrada(s) con la planificación del horario docente.
- Para acceder a las funcionalidades del sistema el usuario debe estar correctamente autenticado.

Contraseñas:

- Se deben cifrar las contraseñas. La longitud mínima debe ser de ocho caracteres y debe contener combinaciones de letras minúsculas, letras mayúsculas, números y/o caracteres especiales.

[7] Turno de clase planificado por el sistema generador de horarios de la CUJAE

2.5 Fase II: planificación de la entrega

El objetivo de esta fase es el de llegar a un acuerdo entre los clientes y los programadores respecto a cuáles serán las HU a ser implementadas durante cada iteración y establecer cuál va a ser el contenido de la primera entrega. Los programadores estiman cuanto tiempo y esfuerzo requiere cada HU y se establece el cronograma. La duración del calendario para la primera entrega no suele superar los dos meses. Si se hace una buena preparación durante la fase de exploración esta actividad no suele llevar más de dos días (Calabria, y otros, 2003).

No	Historia de usuario	Estimación (semana)
1	Listar locales disponibles	½
2	Reservar local disponible	1
3	Listar locales reservados	½
4	Modificar reservación	½
5	Eliminar reservación	½
6	Seleccionar tipo de cambio de turnos	⅓
7	Listar locales con turno disponibles	1
8	Cambio de turnos de clase	1
9	Listar turnos para intercambiar	⅓
10	Intercambio de turnos de clase	⅓
11	Listar locales para exámenes finales	⅓
12	Asignar locales para exámenes finales manualmente	⅓
13	Asignar locales para exámenes finales automáticamente	1
14	Replanificar asignación de locales para exámenes finales	1
15	Eliminar asignación de locales para exámenes finales	⅓
16	Listar exámenes finales	½
17	Intercambiar de exámenes finales	½
18	Listar planificación de exámenes finales	½
19	Exportar servicios web del horario	½

Tabla 14. Plan de duración de las iteraciones

2.6 Fase III: Iteraciones

Esta fase incluye varias iteraciones del sistema antes de la primera entrega. El calendario es dividido en un número iteraciones de tal manera que cada iteración tome de una a cuatro semanas de implementación. En la primera iteración se crea un sistema que abarca los aspectos más importantes de la arquitectura global, esto se logra seleccionando las HU que hagan referencia a la construcción de la estructura de todo el sistema. El cliente decide que HU van a ser implementadas para cada iteración. Además, se realizan las pruebas funcionales, realizados por el cliente, al final de cada iteración. Al final de la última iteración el sistema está listo para ser puesto en producción según (Calabria, y otros, 2003).

2.6.1 Plan de iteraciones

En el plan de iteraciones se especifican las HU a implementar en cada iteración del sistema, estableciéndose cuatro iteraciones para la realización del sistema en coordinación con el cliente:

- Iteración 1: tiene como objetivo la implementación de las HU: 1, 2, 3, 4 y 5 que son las HU relacionadas con la implementación del módulo Reservación de locales.
- Iteración 2: tiene como objetivo la implementación de las HU: 6, 7, 8, 9 y 10 que son las HU relacionadas con la implementación del módulo Cambio de turnos.
- Iteración 3: tiene como objetivo la implementación de la HU: 11, 12, 13, 14 y 15 que son las HU relacionadas con la implementación del módulo Asignación de locales para exámenes finales.
- Iteración 4: tiene como objetivo la implementación de la HU: 16, 17, 18 y 19 que son las HU relacionadas con la implementación del módulo Asignación de locales para exámenes finales.

2.6.2 Plan de entregas

No	Historia de usuario	Estimación (semana)	Fecha (inicio-fin)
1	Listar locales disponibles Reservar local disponible Listar locales reservados Modificar reservación Eliminar reservación	3	16/02/2015- 08/03/2015
2	Seleccionar tipo de cambio de turnos Listar locales con turno disponibles Cambio de turnos de clase Listar turnos para intercambiar	3	09/03/2015- 29/03/2015

	Intercambio de turnos de clase		
3	Listar locales para exámenes finales Asignar locales para exámenes finales manualmente Asignar locales para exámenes finales automáticamente Replanificar asignación de locales para exámenes finales Eliminar asignación de locales para exámenes finales	3	09/04/2015-30/04/2015
4	Listar exámenes finales Intercambiar exámenes finales Listar planificación de exámenes finales Exportar servicios web de horario	2	01/05/2015-15/05/2015

Tabla 15. Plan de entregas

2.6.3 Modelo de datos

El modelo de datos es un conjunto de conceptos, reglas y convenciones que permiten describir y manipular los datos de la parcela de un cierto mundo real que deseamos almacenar en la base de datos (Piattini, y otros, 1999).

El sistema generador de horario de la CUJAE utiliza un modelo de datos para la generación del horario docente. Utiliza llaves subrogadas para una mejor manipulación de los datos y tablas nomencladoras para almacenar información. La tabla *encuentro* es la principal, ya que es la encargada de almacenar el horario en forma de encuentros. Este modelo será utilizado para desarrollar los módulos Cambio de turnos, Reservación de locales y Asignación de locales para exámenes finales. Para facilitar el desarrollo de los módulos Reservación de locales y Asignación de locales para exámenes finales se añaden las tablas *horario.local_horario_reservar_local*, *n_examen, horario.examenes* y *horario.reserva_local*.

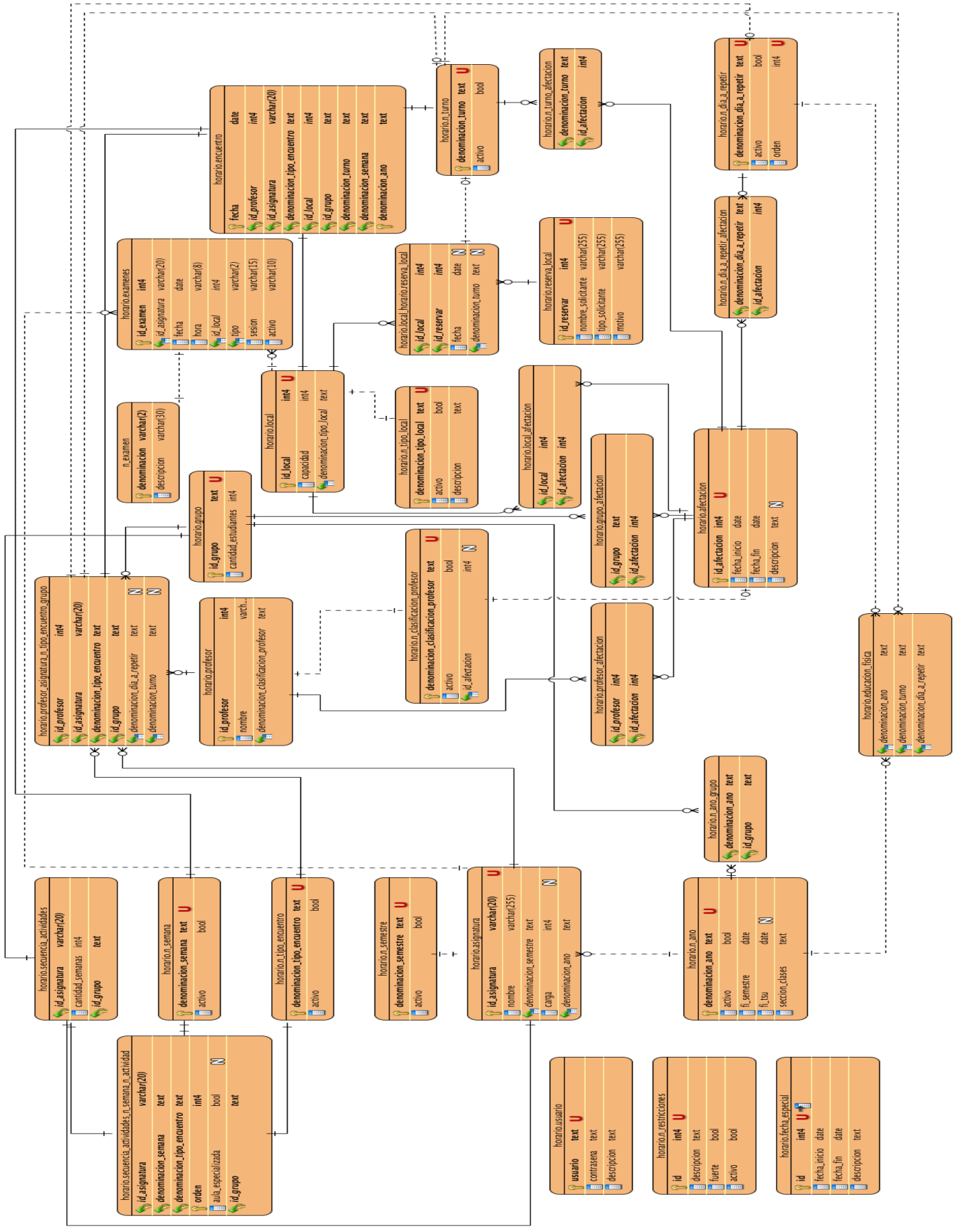

Figura 1. Diagrama Entidad Relación [elaboración propia]

2.6.4 Tarjetas Clase - Responsabilidad - Colaborador

Las tarjetas Clase - Responsabilidad - Colaborador (CRC) representan una entidad del sistema, a la cual asignar responsabilidades y colaboraciones. El formato físico de las tarjetas CRC facilita la interacción entre los participantes del proyecto, en sesiones en las que se aplican técnicas de grupos como tormenta de ideas o juego de roles, y se ejecutan escenarios a partir la de especificación de requisitos, historias de usuarios o casos de uso. De esta forma, van surgiendo las entidades del sistema junto con sus responsabilidades y colaboraciones (Casas, y otros, 2009).

En la metodología XP el proceso de diseño es iterativo. Las tarjetas no se crean en un mismo tiempo, se van desarrollando en la medida que se ejecuten las iteraciones del sistema y se le añaden responsabilidades y colaboradores según sea necesario. A continuación se muestran las clases de diseño identificadas por el equipo de desarrollo, así como un ejemplo de las tarjetas CRC, las restantes se encuentran en el Anexo 2 al final del documento.

Las tarjetas CRC están compuestas por tres campos, clase, responsabilidades y colaboradores:

- **Clase:** nombre de la clase que se está modelando.
- **Responsabilidades:** las responsabilidades de una clase son las acciones que conocen y realizan, sus atributos y métodos.
- **Colaboradores:** los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

A continuación se realiza la descripción de la estructura de la tarjeta CRC de la clase modelo_reservar.

Clase ModeloReservar	
Responsabilidad	**Colaboración**
insertarReservarLocal() insertarLocalReservarLocal() lastInsertId() eliminarReservacion() modificarReservacion()	Entidad Reservar Controladora Reservar Local

Tabla 16. Tarjeta CRC: ModeloReservar

2.6.5 Arquitectura

Para darle continuidad al sistema generador de horario de la CUJAE se mantendrá la arquitectura en capas, basada en el patrón arquitectónico Modelo-Vista-Controlador (MVC), el cual según (Fernández Romero, y otros, 2012) es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por

separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo.

Para desarrollar la propuesta de solución se modelo la arquitectura del sistema. Se determinó una capa perpendicular para las entidades. De forma horizontal se tienen los módulos: Cambio de turnos, Reservación de locales y Asignación de locales para exámenes finales que están asociados con las funcionalidades propias del sistema, en este caso son los que van a cumplir con los requisitos funcionales y que a su vez interactúan con los módulos presentes en el sistema: Generar horario, Locales y Asignaturas. Para reflejar los requisitos no funcionales se ubicaron en una capa vertical los componentes: Seguridad, Librería SOAP (incluida en la arquitectura), Configuraciones, Validaciones (se le incluyeron nuevas funcionalidades) y GenerarPDF.

Los servicios web no se encuentran en la capa de los módulos antes mencionados debido a que no representan una funcionalidad propia del sistema, estos fueron incluidos por iniciativa del equipo de trabajo para permitir la interoperabilidad entre diferentes sistemas independientemente del lenguaje o la plataforma donde esté montado el sistema.

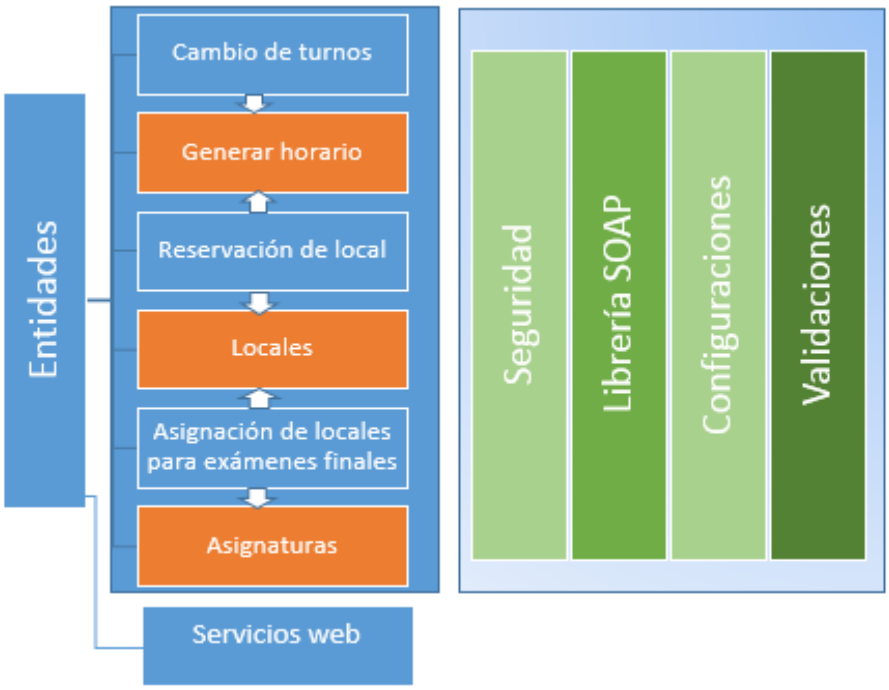

Figura 2. Arquitectura del sistema [elaboración propia]

A continuación se presentan las restricciones de diseño de la arquitectura:

- Las clases controladoras se deben nombrar utilizando la nomenclatura controladora como prefijo seguido de un guión bajo y una palabra.
- Las clases vistas se deben nombrar utilizando la nomenclatura vista como prefijo seguido de un guión bajo y una palabra.

- Las clases modelos se deben nombrar utilizando la nomenclatura Modelo como prefijo seguido de un guión bajo y una palabra.
- Las clases controladoras, vistas, modelo, entidades se deben ubicar físicamente en la carpetas controladora, vistas, modelos y entidades respectivamente. Ver figura 3.

Figura 3. Ubicación de las clases controladoras, vistas, modelo y entidades [elaboración propia]

En la arquitectura del sistema se hace uso del patrón arquitectónico MVC. A este patrón arquitectónico de software se le añaden las clases necesarias para desarrollar los módulos Cambio de turnos, Reservación de locales y Asignación de locales para exámenes finales.

Modelo: se encarga de las operaciones sobre la base de datos. Los elementos manejados son:

- ***modelo_reservar.php*:** gestiona las reservaciones de los locales.
- ***modelo_cambiar_turno.php*:** gestiona los cambios de turnos.
- ***modelo_intercambiar_grupos.php*:** gestiona los intercambios de turnos.
- ***modelo_examen.php*:** gestiona la planificación de exámenes finales.
- ***reservar.php*:** crear una instancia de reservar
- ***exmanen.php*:** crear una instancia de examen
- ***functions.php***: modela los servicios web

Se hizo uso de las siguientes clases modelo para la implementación de los módulos: Cambio de turnos, Reservación de locales y Asignación de locales para exámenes finales.

- ***modelo_profesor.php*:** gestiona los profesores.
- ***modelo_profesor_afectacion.php*:** gestiona las afectaciones de los profesores.
- ***modelo_grupo.php*:** gestiona de los grupos.
- ***modelo_local.php*:** gestiona de los locales.
- ***modelo_horario.php*:** gestiona las restricciones para la planificación del horario.
- ***modelo_asignatura.php*:** gestiona las asignaturas.
- ***modelo_filtro.php*:** gestiona las peticiones de búsqueda del horario docente generado.
- ***modelo_nomenclador.php*:** gestiona los nomencladores.

Vista: aquí se presentan todas las interfaces con las cuales el usuario va a interactuar directamente. Las vistas definidas son:

- ***vista_asociar_local_solicitante.php*:** se asocia un local a un solicitante para realizar la reservación del local.
- ***vista_actualizar_locales_reservados.php*:** se modifica la reservación echa para un local determinado.
- ***vista_cambiar_turno.php*:** se listan los turnos por locales y se selecciona algún turno.
- ***vista_editar_locales_reservados.php*:** se listan los locales reservados.
- ***vista_intercambiar1.php*:** se listan los turnos de una semana y profesor especificado.
- ***vista_intercambiar2.php*:** se listan los turnos de una semana y profesor especificado.
- ***vista_nuevo_turno.php*:** se listan los turnos de un profesor especificado.
- ***vista_reservar_locales.php*:** se listan los locales de una fecha y un profesor especificado.
- ***vista_examenes_asignaturas.php*:** se listan los locales disponibles por sección de clase.
- ***vista_servicios.php*:** se describen los servicios web

Controlador: facilita y controla la comunicación entre las vistas y los modelos.

- ***controladora_reservar_local.php*:** facilita y controla las peticiones que se le hacen a la modelo desde las vistas vista_asociar_local_solicitante, vista_actualizar_locales_reservados y vista_editar_locales_reservados.
- ***controlador_cambiar_turno.php*:** facilita y controla las peticiones que se hacen a la modelo desde la vista vista_nuevo_turno.
- ***controlador_intercambiar_turno.php*:** facilita y controla las peticiones que se le hacen a la modelo desde la vista vista_intercambiar2.
- ***controlador_examen*:** facilita y controla las peticiones que se le hacen a la modelo desde la vista vista_examenes_asignaturas.

2.6.6 Patrones de diseño

A continuación se muestra la aplicación de los patrones de diseño en la propuesta de solución.

Patrones GRASP

- **Experto:** otorga la responsabilidad de insertar un local para reservar a la clase modelo *ModeloReservar*, pero como ésta no contiene los datos de la reservación le delega la responsabilidad a la clase *Reservar* que es la experta informando acerca de los datos de la reservación y ésta se encarga de devolver la información que necesita la modelo para realizar la inserción de la reservaciones de locales. Se representa en la figura 4.

- **Creador:** asigna la responsabilidad de crear un objeto de la clase *Reservar* a la clase modelo *ModeloMatriz*, por ello el patrón sugiere que dicha modelo es la idónea para asumir la responsabilidad de crear la instancia de *Reservar*. Se representa en la figura 4.
- **Controlador:** la clase controladora *controlador_reservar* le asigna la responsabilidad de realizar la operación de *insertarResevarLocal* a la clase modelo *ModeloReservar*, este evento es sugerido mediante dicho patrón. Se representa en la figura 4.
- **Bajo acoplamiento:** la clase modelo *ModeloMatriz* es la responsable de crear la instancia de la clase *Reservar* para evitar que al realizar esta operación en varias clases específicas cree dependencias. Esto posibilita que en caso de producirse algún tipo de cambio en la clase *Reservar* solo se deba realizar mantenimiento a la clase modelo *ModeloMatriz* y la clase modelo *ModeloReservar* no sufra modificación. Se representa en la figura 4.
- **Alta cohesión:** la *ModeloReservar* se enfoca en realizar la inserción de la reservación como responsabilidad de ella y la *ModeloMatriz* de crear la instancia de la clase *Reservar* que puede ser utilizada por la *ModeloReservar*, esta relación evita saturar de responsabilidades a una sola clase modelo. Se representa en la figura 4.

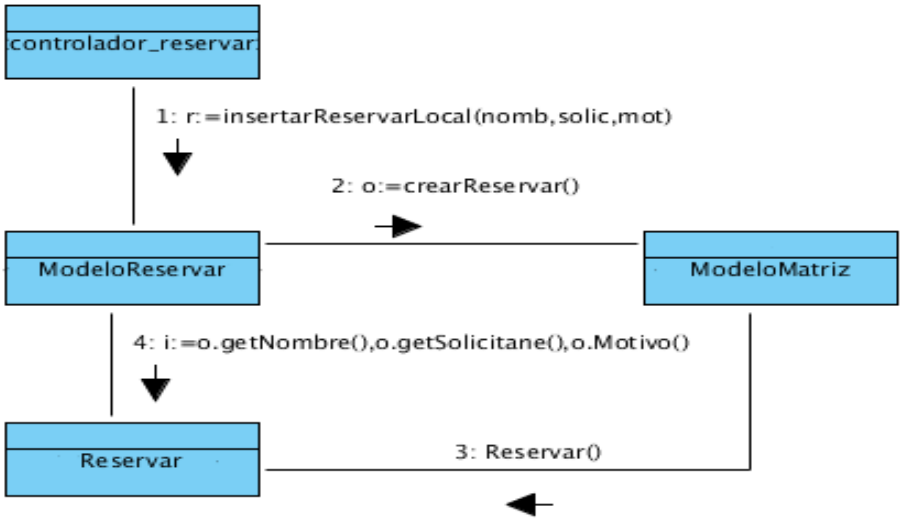

Figura 4. Representación de los patrones GRASP y GoF [elaboración propia]

Patrones GoF

- **Fachada:** la controladora *controlador_reservar* representa el cliente a la cual la modelo *ModeloReservar* le encapsula un conjunto de complejidades del negocio que la controladora no conoce, constituyendo la modelo una fachada de la controladora. Se representa en la figura 4.
- **Decorador:** se evidencia en el sistema mediante el archivo nombrado *pie_de_pagina.php* que contiene el diseño del pie de página que se usa en todas las páginas del sistema, para evitar tener que repetirlo en cada una. Se representa en la siguiente figura.

```
<?php include('estructura/encabezado.php') ?>
<?php include('estructura/comun.php') ?>

<a class="brand">Intercambiar turno</a>

<?php include('estructura/comun2.php') ?>
<?php include('estructura/incluir.php');

?>
```

Figura 5. Uso del patrón decorador [elaboración propia]

Patrón de base de datos: llave subrogada

En la siguiente figura se muestra un ejemplo del patrón llave subrogada, donde a la tabla de datos *horario.examenes* se le asigna un identificador único.

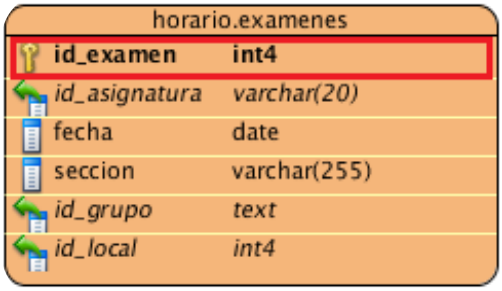

Figura 6. Uso del patrón Llave subrogada [elaboración propia]

2.6.7 Resultado de las métricas

Los requisitos del software son la base de las medidas de calidad. En la disciplina de requisitos se tuvo en cuenta la métrica para medir su estabilidad, especificidad y grado de validación.

Aplicación de la métrica Estabilidad de requisitos:

Teniendo en cuenta que se identificaron un total de 19 requisitos funcionales, de los cuales dos resultaron modificados (uno por modificación, uno por inserción) se calcula:

ETR = [(19 - 2) / 19 * 100]= 89,47

Como resultado se obtuvo un valor de 89,47. Dicha cifra demuestra que no se han realizado cambios significativos sobre los requisitos, son estables y, por tanto, es confiable el análisis y diseño sobre ellos.

Aplicación de la métrica Especificidad de los requisitos:

La especificidad de los requisitos se calcula como:

Q1 = nui /nr = 19 /27 = 0,70

Como resultado de la aplicación de la métrica se obtuvo un valor de 0,70. Al estar el valor cerca de 1 se estima que los requisitos son entendibles, específicos y que no poseen ambigüedades.

Aplicación de la métrica Grado de validación:

El grado de validación de los requisitos se calcula:

Q3 = nc / (nc + nnv) = 19 /(19+ 0) = 1

La aplicación de esta métrica dio como resultado 1, por lo tanto se concluye que se realizó una correcta definición de los requisitos.

2.6.8 Verificación del diseño

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

TOC: permite medir los atributos de calidad responsabilidad, complejidad de implementación y reutilización de las clases del diseño. La responsabilidad y la complejidad son inversamente proporcionales a la reutilización, por lo que a mayor responsabilidad y complejidad de implementación de una clase, menor será su nivel de reutilización. 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
ModeloProfesorAfectacion	11	Baja	Baja	Alta
ModeloProfesor	12	Baja	Baja	Alta
ModeloNomenclador	59	Alta	Alta	Baja
ModeloLocal	17	Baja	Baja	Media
ModeloHorario	43	Alta	Alta	Baja
ModeloFiltro	7	Baja	Baja	Alta
ModeloReservar	6	Baja	Baja	Alta
ModeloIntercambiarGrupo	4	Baja	Baja	Alta
ModeloCambiarTurno	4	Baja	Baja	Alta
ModeloExamen	9	Baja	Baja	Alta

Tabla 17. Evaluación de las clases del sistema mediante la métrica TOC

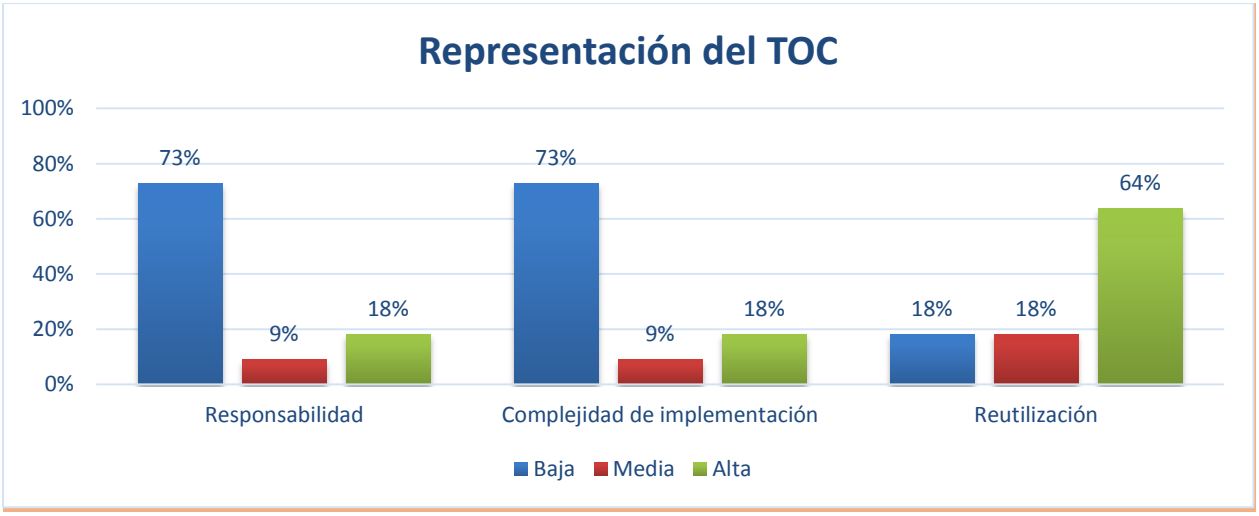

Figura 7. Representación de los resultados de la métrica TOC [elaboración propia]

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 es satisfactorio.

RC: está dada por el número de relaciones de uso de una clase con otra. Permite evaluar los atributos de calidad: acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas de unidad necesarias de cada clase, teniendo en cuenta las relaciones existentes entre ellas. 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
examen	1	Bajo	Baja	Alta	Baja
asignatura	1	Bajo	Baja	Alta	Baja
profesor	2	Medio	Baja	Alta	Baja
grupo	1	Bajo	Baja	Alta	Baja
local	3	Alto	Baja	Alta	Baja
encuentro	4	Alto	Media	Media	Media
afectacion	2	Medio	Baja	Alta	Baja
reservar_local	1	Bajo	Baja	Alta	Baja

Tabla 18. Evaluación de las clases del sistema mediante la métrica RC

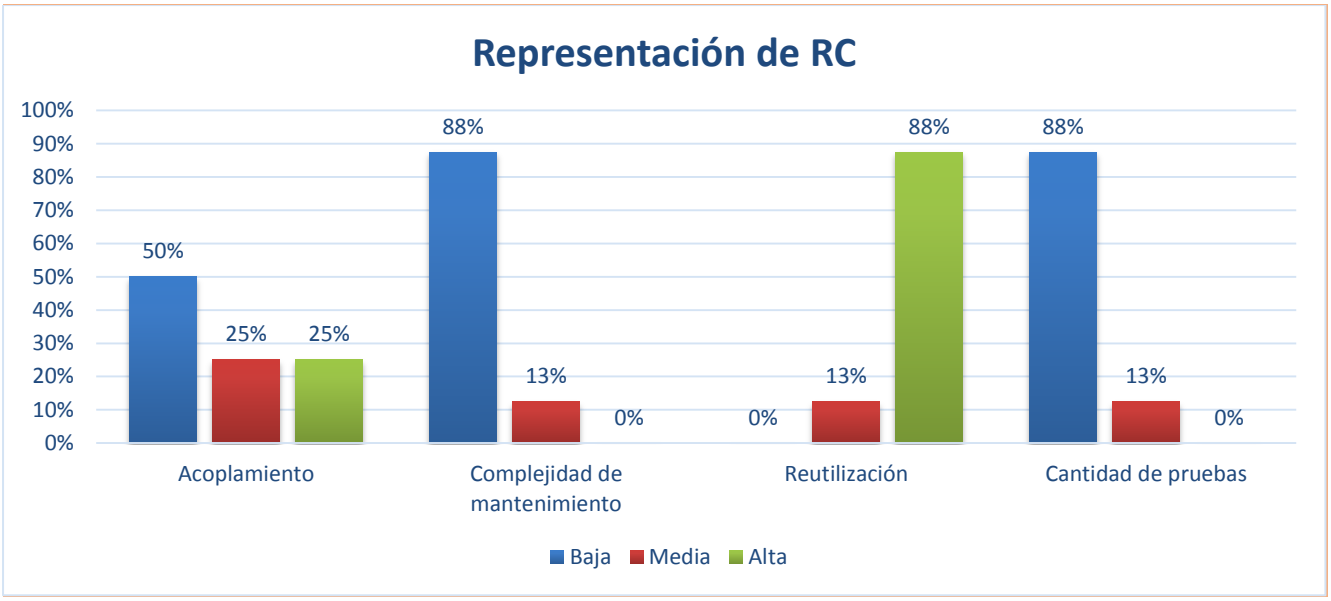

Figura 8. Representación de los resultados de la métrica RC [elaboración propia]

Como resultado de la aplicación de la métrica RC se evidencia que las clases del sistema poseen un bajo acoplamiento, baja complejidad de mantenimiento, alta reutilización y una baja cantidad de pruebas, por lo que de forma general, se valoran como buenos los resultados.

2.7 Estándar de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. El código fuente debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, se establece un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento según se explica en (Microsoft, 2015). A continuación se muestra el estándar de codificación utilizado para el desarrollo del sistema.

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 conocerSemana($fecha)
{
    $semana = 01;
    $fecha_actual = $this->conocerFechaInicialCiclo();
    while ($fecha_actual != $fecha) {
        if ($this->conocerDiaSemanaFecha($fecha_actual) != 'Domingo') {
            $fecha_actual = $this->sumarDiasAFecha($fecha_actual, 1);
        } else {
            $fecha_actual = $this->sumarDiasAFecha($fecha_actual, 1);
            $semana++;
        }
    }
    if ($semana >= 1 && $semana <= 9) {
        return '0' . $semana;
    } else {
        return $semana;
    }
}
```

Figura 9. Ejemplo de estándar de codificación [elaboración propia]

2.8 Descripción de la solución

La solución propuesta está constituida por tres módulos que se integran al Sistema para la generación del horario docente de la Facultad de Ingeniería Industrial de la CUJAE, se puede acceder a las funcionalidades a través de opciones que se encuentran en el menú izquierdo que se describen a continuación.

- **Reservación de locales:** el módulo permite al usuario reservar un local en un turno de clase que no esté afectado, ni ocupado en otra actividad docente ni reservado. El usuario debe indicar el día en que se desea realizar la reservación, si selecciona un local específico se muestran los turnos que ese local tiene disponible el día indicado, de lo contrario se listan los locales que tienen algún turno disponible el día especificado. Después de indicar el día, local y turno se registra el tipo de solicitante (profesor, estudiante u otro), el nombre del solicitante y una breve descripción del motivo de la reservación. Desde la interfaz inicial de este módulo se visualiza la cantidad de reservaciones realizadas, al seleccionar la opción: Locales reservados se listan las reservaciones registradas, que pueden ser modificadas o eliminadas.
- **Mover turno:** el módulo está constituido por dos funcionalidades: cambiar e intercambiar turno, ambos permiten realizar cambios de forma manual en el horario generado. Para el cambio de turnos es necesario seleccionar un día que tenga un local disponible en un turno

de clase determinado. El usuario debe indicar el día en que se debe mover el turno, si selecciona un local específico se muestran los turnos que ese local tiene disponible el día indicado; un local está disponible si en un turno determinado no está afectado, ni ocupado en otra actividad docente ni reservado. Si no se seleccione un local se listan los locales que tienen algún turno disponible el día especificado, para que el usuario indique el turno y el local en que se desea realizar el movimiento. El próximo paso consiste en especificar el turno de clases que se desea mover, para ello se debe indicar el profesor involucrado en el cambio y el turno en cuestión. Para facilitar la selección se listan los turnos que debe impartir el profesor a partir dl día actual, de cada uno se muestra: la fecha, semana, turno, asignatura, tipo de encuentro, local y grupo. A continuación se muestran los problemas que puede ocasionar el movimiento de turno indicado teniendo en cuenta las restricciones fuertes y débiles, para que el usuario decida si desea realizar la operación o no. Si no existen violaciones de las restricciones establecidas se indica que el cambio se puede realizar satisfactoriamente y el usuario debe decidir si desea realizar o no la operación. El intercambio de turnos funciona de forma similar, con la diferencia de que en este caso se deben especificar los dos turnos de clase que se desean intercambiar.

- **Asignación de locales para exámenes finales:** el módulo permite asignar manual o automáticamente locales para exámenes finales o de revalorización de una asignatura. Para la asignación manual el usuario debe especificar la asignatura y el día del examen, si selecciona un local específico se muestran los turnos que ese local tiene disponible el día indicado, de lo contrario se listan los locales que tienen algún turno disponible en la sesión de clase del año en que se imparte la asignatura en el día especificado. Para la asignación de locales de forma automática se debe especificar los tipos de locales que se desean utilizar, se tiene en cuenta que los locales seleccionados no estén afectados, ni reservados ni ocupados en otras actividades docentes o por otros exámenes. Además, se escogen los locales que con la menor capacidad que permita cubrir la necesidad de matrícula del examen. Las asignaciones realizadas pueden ser modificadas o eliminadas antes de seleccionar la opción: Finalizar asignación. Después de finalizar la asignación de exámenes se puede ejecutar la funcionalidad relacionada con el intercambio de exámenes, la misma requiere que el usuario seleccione las asignaturas y los tipos de exámenes que se desean intercambiar.

Se incluye la posibilidad de obtener los servicios web del horario docente y los exámenes finales. Desde la opción: Servicios, ubicada en el área superior del sistema sin necesidad de registrarse en el sistema y pueden ser consumidos por cualquier persona de la universidad. Se muestran a través

de la interfaz SOAP[8] y se describen en un documento WSDL[9], teniendo en cuenta los requisitos funcionales necesarios para establecer una comunicación con los servicios web.

A continuación se muestran algunas imágenes de los módulos descritos

Figura 10. Ejemplo de turnos libres para reservar un local [elaboración propia]

Figura 11. Cambio de turnos para un turno disponible [elaboración propia]

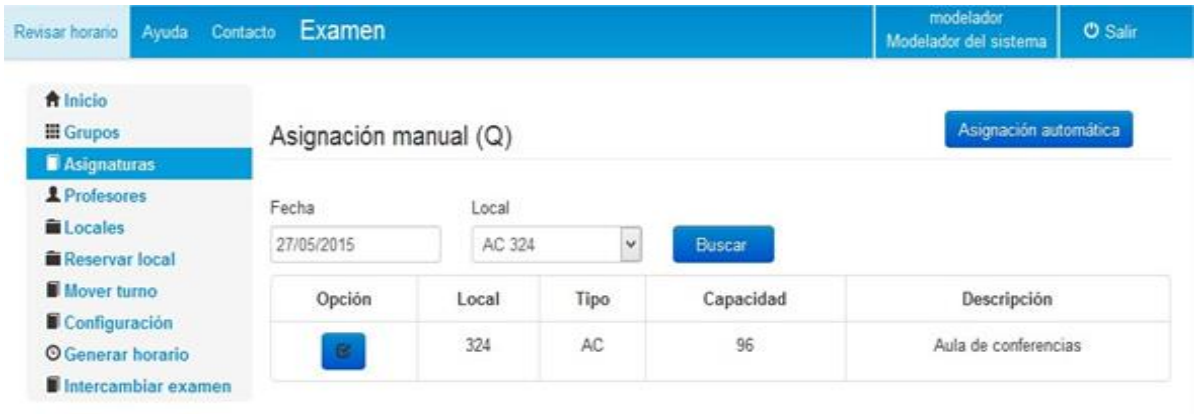

Figura 12. Listar local disponible para un examen [elaboración propia]

[8] Simple Object Access Protocol (Protocolo simple de acceso a objetos**)**

[9] Web Services Description Language (Lenguaje de descripción de servicios web)

2.9 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).

Funcionalidades	Complejidad temporal
Asignación de locales para exámenes finales	cota superior de O (n^4)
Reservación de locales	cota superior de O (n^4)
Cambio de turno	cota superior de O (n^4)
Intercambiar turno	cota superior de O (4n)

Tabla 19. Complejidad temporal

El sistema se instaló en un servidor con 3Gb de memoria RAM y un microprocesador Core 2 Duo, CPU 2.20GHz y se ejecutó en diferentes estaciones de trabajo, cuyo tiempo de respuesta se muestra en la siguiente tabla.

Prestaciones de estaciones de trabajo	Funcionalidades	Tiempo de respuesta
• Memoria RAM: 1Gb • Microprocesador: Pentium 4 CPU 3.0GHz	Asignación de locales para exámenes finales	5.1 segundo
	Reservación de locales	3.0 segundo
	Cambio de turno	2.5 segundo
	Intercambiar turno	4.2 segundo
• Memoria RAM: 2Gb • Microprocesador: Core 2 Duo CPU 2.20GHz	Asignación de locales para exámenes finales	4.5 segundo
	Reservación de locales	2.36 segundo
	Cambio de turno	1.98 segundo
	Intercambiar turno	3.87 segundo

• Memoria RAM: 2Gb • Microprocesador: Core I3 CPU 3.30Ghz	Asignación de locales para exámenes finales	4,13 segundo
	Reservación de locales	840 milisegundo
	Cambio de turno	931 milisegundo
	Intercambiar turno	791 milisegundo

Tabla 20. 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.

2.10 Conclusiones parciales

En este capítulo se describió la propuesta de solución del problema de la investigación. Las HU establecidas por el cliente permitieron identificar las funcionalidades de los módulos a desarrollar. La prioridad de para el negocio de cada una permitió realizar la estimación del esfuerzo para su desarrollo en cuatro iteraciones. La creación de tarjetas CRC permitieron añadir responsabilidades y colaboración a las clases. La aplicación de los patrones y el uso de un mismo estándar de codificación facilitaron la reutilización del código y una mejora en el nivel de complejidad de las clases. El uso del modelo arquitectónico definido permitió analizar el sistema desde distintos puntos de vista y la aplicación de la metodología XP posibilitó reducir el tiempo de desarrollo estimado ante cambios de último momento. Además el cálculo de la complejidad temporal permitió comprobar la eficiencia de los métodos más complejos de cada uno de los módulos, obteniéndose resultados satisfactorios.

CAPÍTULO 3: Pruebas de la solución

3.1 Introducción

En este capítulo se realiza la validación de la solución propuesta mediante las pruebas de caja blanca, caja negra y pruebas de aceptación, así como el cumplimiento del problema de la investigación.

3.2 Fase IV: Producción

En esta fase el producto se pone en producción y se realizan todas las pruebas al sistema. Este momento en donde se afinan los detalles del sistema debido a que se tiene un gran conocimiento del diseño y además, se dispone del hardware en donde se va a correr el sistema. Durante esta fase se debe ir más despacio a medida que es desarrollando el software. Esto no significa que el desarrollo se detenga pero si es de considerar, que el riesgo se vuelve más importante a medida que los cambios afectan la planificación de entrega del proyecto (Pressman, 2005).

3.2.1 Resultados de la prueba de caja blanca

A continuación se muestra el resultado de aplicar la prueba de caja blanca mediante la técnica del camino básico al método *obtenerLocalExamen*, quien se encarga de asignar los locales para los exámenes finales o las revalorizaciones de un año determinado, de acuerdo a la cantidad de estudiantes del año y la capacidad de los locales disponibles.

```
     */
    public function obtenerLocalExamen($anno, $fecha, $tipo_locales)
    {
        $fecha = str_replace('/', '-', $fecha);
        $fecha = date("Y-m-d", strtotime($fecha));
        $sesion = $this->modelo_nomenclador->obtenerAno($anno)[0][4];
        $turnos = $this->obtenerTurnosActivosXSeccion($sesion);
        $capacidad = 0;
        $cantidad = 0;
        $resultado = null;
        $locales = $this->modelo_local->obtenerListaLocalesOrdenados();
        $grupos = $this->modelo_grupo->obtenerCantidadXGrupos($anno);
        $locales = $this->filtroDeLocales($locales, $tipo_locales, $turnos, $fecha, $sesion);
        for ($i = 0; $i < count($grupos); $i++) {
            $cantidad += $grupos[$i][1];
        }
        $capacidad += $locales[0][1];
        if (count($locales) > 0) {
            $resultado[] = $locales[0];
            for ($j = 1; $j < count($locales) + 1; $j++) {
                if ($capacidad < $cantidad) {
                    if ($j != count($locales)) {
                        $resultado[] = $locales[$j];
                        $capacidad += $locales[$j][1];
                    }
                } else {
                    return $resultado;
                }
            }
        }
        return 0;
    }
```

Figura 13. Método obtenerLocalExamen [elaboración propia]

En la siguiente imagen se muestra el grafo dirigido del flujo asociado al algoritmo *obtenerLocalExamen*.

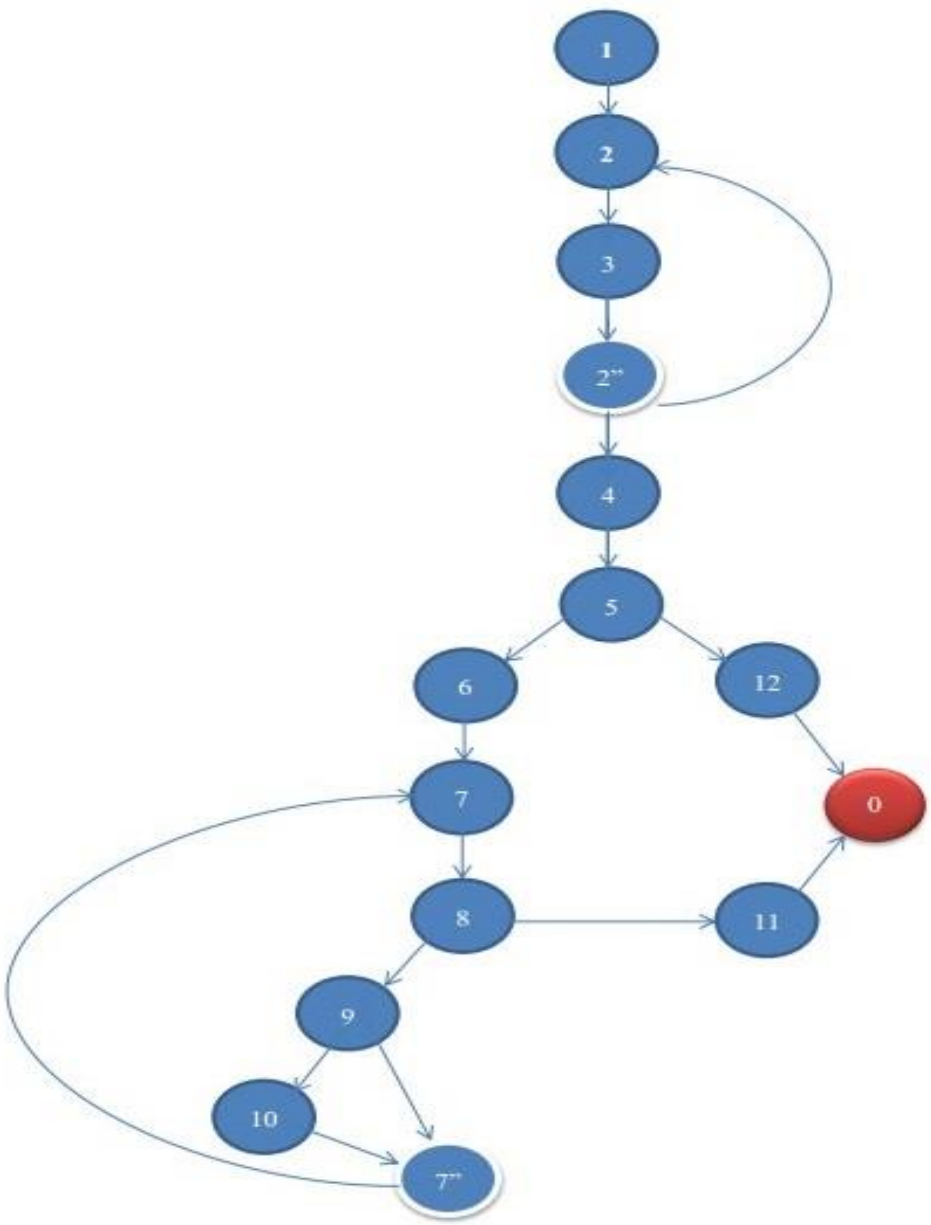

Figura 14. Grafo del flujo [elaboración propia]

Tomando como referencia al grafo dirigido del flujo se procede a calcular la complejidad ciclomática empleando las tres variantes propuestas por (Pressman, 2005).

Existen cinco regiones.

V (G) = 18 aristas -15 nodos +2=5

V (G) = 4 predicado + 1=5

La complejidad ciclomática es igual a cinco, lo que significa que existen cinco posibles caminos linealmente independientes y representa el mínimo número de casos de prueba para el algoritmo, a continuación se muestran los caminos existentes.

Numero	Camino
1	1-2-3-2”-4-5-12-0
2	1-2-3-2”-4-5-6-7-8-11-0
3	1-2-3-2”-4-5-6-7-8-9-7”-8-11-0
4	1-2-3-2”-4-5-6-7-8-9-10-7”-8-11-0
5	1-2-3-2”-4-5-6-7-8-9-7”-8-9-10-7”-8-11-0

Tabla 21. Caminos por donde el flujo puede circular

3.2.2 Casos de prueba para caja blanca

El próximo paso es ejecutar los casos de pruebas para cada camino y se compara con los resultados esperados, verificando que las instrucciones se ejecuten por lo menos una vez. A continuación se muestra el caso de prueba para el camino básico 2, en el Anexo 3 se muestran los restantes casos de prueba.

Caso de prueba para el camino básico 2	
Descripción	Comprueba si la cantidad de estudiantes de un año no excede la capacidad de los locales.
Condiciones de ejecución	• Que la fecha esté dentro del rango del curso escolar. • Que el año exista y esté activo. • Que el tipo de local exista y esté activo.
Entrada	$anno , $fecha, $tipo_locales
Resultado esperado	Devuelve un listado de los locales disponibles para el año especificado.

Tabla 22. Caso de prueba para el camino básico 2

3.2.3 Casos de prueba para caja negra

Los casos de prueba representan los datos que se utilizarán como entrada para ejecutar el software a probar. Determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Cada técnica de pruebas proporciona unos criterios distintos para generar estos casos o datos de prueba. Por lo tanto, durante la tarea de generación de casos de prueba, se han de confeccionar los distintos casos de prueba según la técnica o técnicas

identificadas previamente. La generación de cada caso de prueba debe ir acompañada del resultado que ha de producir el software al ejecutar dicho caso (Juristo, y otros, 2006). A continuación se muestran algunos ejemplos de casos de prueba desarrollados utilizando la técnica de partición equivalente, el resto de los casos de pruebas se encuentran en el Anexo 4.

Caso de prueba de caja negra	
Código:2	**No. Historia de Usuario:2**
Nombre: Reservar local disponibles	
Descripción: El usuario puede seleccionar un local que esté disponible en una fecha y turno determinado para utilizarlo en cualquier tipo de actividad de interés para la Facultad. Después de realizar la reservación se debe mostrar un mensaje informativo indicando que la reservación fue realizada con éxito.	
Condiciones de Ejecución: El usuario debe estar autenticado en el sistema.	
Entrada/Pasos de Ejecución: • Seleccionar la opción “Reservar local” del menú izquierdo. • Seleccionar la fecha. • Seleccionar el tipo de local es opcional. • Se muestra el listado de turnos disponibles para los criterios de búsqueda especificados. • Seleccionar el turno en que se encuentre libre un local. • Llenar los campos: tipo de solicitante, nombre del solicitante y motivo. • Seleccionar la opción Adicionar. • Mostrar la confirmación de la operación al usuario.	
Resultado esperado: se debe mostrar al usuario un mensaje informativo para indicar que la reservación fue realizada con éxito.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 23. Caso de prueba para la HU Reservar local disponible

3.2.4 Resultados de la prueba de caja negra

Las pruebas de caja negra fueron ejecutadas por un especialista del equipo de calidad del centro CEGEL en dos iteraciones, se detectaron ocho no conformidades. En la primera iteración se detectaron un total de ocho no conformidades dentro de las cuales: una de aplicación, seis de diseño y una de ortografía que fueron resueltas de forma rápida. En la segunda iteración no se detectaron no conformidades emitiéndose el acta de liberación que se muestra en el Anexo 6.

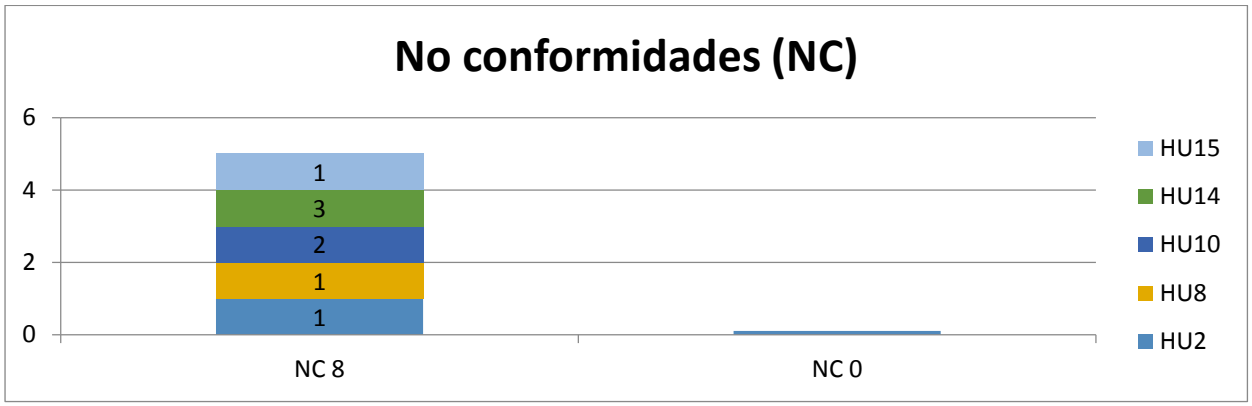

Figura 15. Gráfica resultado de aplicar las pruebas de caja negra [elaboración propia]

3.2.5 Prueba de integración

A nivel de integración se utilizó la estrategia incremental de forma ascendente, a continuación se describen los pasos para realizar la integración:

- Se realizaron pruebas unitarias al módulo Reservación de locales para verificar su correcto funcionamiento.
- Se realizó la integración del módulo Reservación de locales con el módulo Locales, comprobando que los turnos afectados de un local no estén disponibles para ser reservados.
- Se realizó la integración del módulo Reservación de locales con el módulo Generar horario, comprobando que los turnos ocupados por un encuentro de clases no se encuentren disponibles para ser reservados.
- Se identificó un total de una no conformidad para el módulo Reservación de locales.
- Se realizaron pruebas unitarias al módulo Cambio de turnos para verificar su correcto funcionamiento.
- Se realizó la integración del módulo Cambio de turnos con el módulo Locales, comprobando que los turnos afectados de un local no puedan ser seleccionados para realizar un cambio de turnos.

- Se realizó la integración del módulo Cambio de turnos con el módulo Generar horario, comprobando que los turnos ocupados por un encuentro de clases no puedan ser seleccionados para realizar un cambio de turnos.
- Se identificó un total de tres no conformidad para el módulo Cambiar de turno.
- Se realizaron pruebas unitarias al módulo Asignación de locales para exámenes finales para verificar su correcto funcionamiento.
- Se realizó la integración del módulo Asignación de locales para exámenes finales con el módulo Asignaturas, comprobando que no se pueda planificar un examen para una asignatura que no exista.
- Se realizó la integración del módulo Asignación de locales para exámenes finales con el módulo Locales, comprobando que no se pueda planificar un examen en un local que se encuentre afectado.
- Se identificó un total de cuatro no conformidad para el módulo Asignación de locales para exámenes finales
- Se realizó la integración de los módulos Cambiar de turno, Reservación de locales y Asignación de locales para exámenes finales con el módulo Configuración, comprobando que no se muestren los locales y los turnos que se encuentren desactivados.
- Se realizó la integración del módulo Servicios web a los módulos Generar horario y Asignación de locales para exámenes finales, comprobando que solo se exporten los servicios relacionados con el horario generado y los exámenes planificados.

3.2.6 Resultado de las pruebas de carga y estrés

A nivel de sistema se realizaron las pruebas de carga y estrés con el propósito de verificar el comportamiento del sistema ante altas cantidades de peticiones, lo que permite conocer el tiempo de respuesta por transacción JMeter en su versión 2.9.

Entorno de pruebas

Se utilizaron dos ordenadores, uno servidor y el otro cliente, los cuales cumplen con las siguientes características:

PC cliente

- Microprocesador Intel Pentium 4 CPU 3.0GHz
- Memoria RAM de 2Gb
- Sistema operativo Linux Mint 17

PC servidor

- Microprocesador Intel Core 2 Duo CPU 2.20GHz
- Memoria RAM de 2Gb
- Sistema operativo Linux Mint 17

Las pruebas fueron realizadas sobre la funcionalidad del sistema, accediendo a cada una 500 usuarios simultáneamente, arrojando un error de 0.6% para 2000 conexiones de forma concurrente, en la siguiente figura se muestran los resultados.

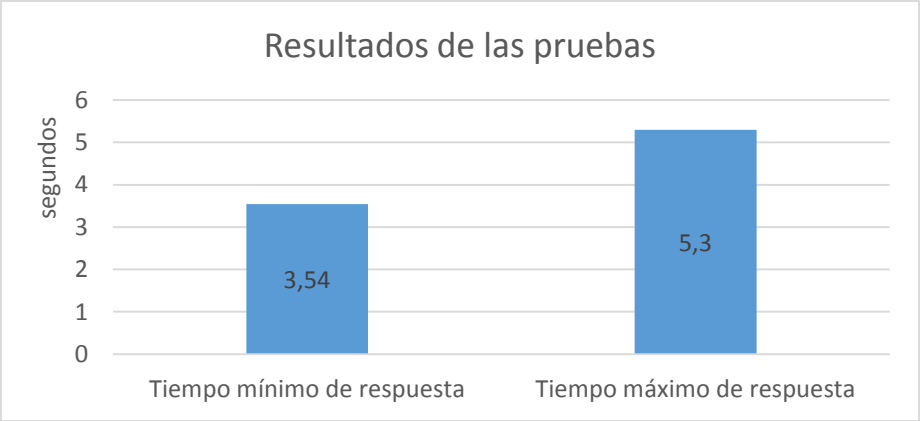

Figura 16. Resultado de aplicar las pruebas de carga y estrés [elaboración propia]

Las pruebas realizadas se consideran exitosas, teniendo en cuenta el entorno de trabajo donde fueron realizadas y los resultados obtenidos. Para 2000 conexiones de forma concurrentes los tiempos de respuestas fueron menores de seis segundos.

3.2.7 Pruebas de aceptación

A continuación se muestran los casos de pruebas diseñados para las historias de usuario Reservar local disponible y Asignar locales para exámenes finales automáticamente, debido a que son parte esencial de las funcionalidades requeridas para el desarrollo exitoso del sistema. Los demás casos de prueba se describen en el Anexo 5.

Caso de prueba de aceptación	
Código: CPA-02	**No. Historia de usuario: 2**
Nombre: Reservar local disponible	
Descripción: El usuario puede seleccionar un local que esté disponible y reservarlo para realizar cualquier tipo de actividad de interés para la Facultad.	
Condiciones de Ejecución: El usuario debe estar autenticado en el sistema.	
Entrada/Pasos de Ejecución: El usuario debe seleccionar la fecha y el local en que desea realizar la reservación. Luego debe registrar los datos de la solicitud.	

Resultado esperado: se le debe mostrar al usuario un mensaje informativo para indicar que la reservación fue realizada con éxito.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 24. Caso de prueba de aceptación para la HU Reservar local disponible

Caso de prueba de aceptación	
Código: CPA-08	**No. Historia de Usuario: 8**
Nombre: Asignar locales para exámenes finales automáticamente	
Descripción: El usuario podrá seleccionar el o los tipos de locales que desea utilizar para el examen. El sistema asigna los locales de acuerdo a la cantidad de estudiantes del curso y la capacidad de los locales.	
Condiciones de Ejecución: El usuario debe estar autenticado en el sistema.	
Entrada/Pasos de Ejecución: El usuario selecciona la asignatura a la que desea planificar el examen y la opción: Asignación automática. Luego debe registrar la fecha, hora, tipo de examen y tipo de local donde se desee planificar los exámenes finales y selecciona la opción Asignar.	
Resultado esperado: El sistema debe mostrar una notificación de confirmación de la acción.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 25. Caso de prueba de aceptación para la HU Cambio de turnos de clase

3.2.8 Validación de la propuesta de solución

Como parte de las actividades que se desarrollaron, se realizó una comparación en cuanto al comportamiento de las variables de la investigación errores y tiempo. Los resultados obtenidos fueron los siguientes.

- **Variable errores:**

	Antes	Después
Cambio de turnos	Horario rígido, no permite que se puedan realizar cambios manuales ante cambios de última hora, por lo	Se pueden realizar cambios manuales del horario generado. Se muestran los locales con turnos

	que pueden ocurrir errores como: se modifica el orden de la secuencia de actividades y se violen las restricciones fuertes	disponibles en un día determinado para realizar el cambio de turnos y permite informar las restricciones fuertes y débiles que se violan al realizar el cambio o intercambio de turnos.
Reservación de locales	No permite mostrar los locales con turnos disponibles en un día determinado para todos los años, por lo que pueden ocurrir errores como: que existan clases simultáneas en un mismo local y que el local se encuentre afectado.	Permite mostrar los locales con turnos disponibles en un día determinado para todos los años que pueden ser reservados para otras actividades, así como la confirmación de la operación.
Asignación de locales para exámenes finales	No permite la asignación de locales para exámenes finales, por lo que pueden ocurrir errores como: locales con capacidad inferior a las requeridas para la realización de los exámenes de una asignatura.	Permite la asignación de locales para exámenes finales teniendo en cuenta que los locales seleccionados no estén afectados, ni reservados ni ocupados en otras actividades docentes o por otros exámenes.

Tabla 26. Validación para la variable: errores

- **Variable tiempo**

	Antes	Después
Cambio de turnos	De forma manual esta actividad demora seis minutos para el cambio de turnos y 12 minutos para el intercambio de turnos aproximadamente.	Demora 1.08 segundos aproximadamente para el cambio de turnos y 2.95 segundos aproximadamente para el intercambio de turnos.

Reservación de locales	De forma manual esta actividad demora cuatro minutos aproximadamente.	Demora 2.07 segundos para listar los locales en lo que se puede realizar la reservación.
Asignación de locales para exámenes finales	De forma manual esta actividad demora cuatro minutos aproximadamente para la asignación de locales de un examen final.	Demora 4.57 segundos aproximadamente.

Tabla 27. Validación para la variable: tiempo

3.3 Fase V: Mantenimiento

Mientras la primera versión se encuentra en producción el proyecto XP debe agregar nuevas funcionalidades, al mismo tiempo que se mantiene el sistema corriendo. Por lo general se necesita un esfuerzo extra de los programadores para satisfacer los requerimientos del cliente. Por este motivo la velocidad de desarrollo suele disminuir una vez que el sistema es puesto en producción. A raíz de esto se requiere incorporar nuevos integrantes y cambiar la estructura del equipo (Calabria, y otros, 2003).

3.4 Fase VI: Muerte

Esta última fase se acerca una vez que el cliente no tiene ninguna HU a ser implementada. Esta es la etapa en la cual no hay más cambios en la arquitectura, el diseño o el código y se realiza la documentación correspondiente. Esta fase aparece también, cuando el sistema no da los resultados deseados o se vuelve demasiado caro para seguir siendo desarrollado (Pressman, 2005).

3.5 Conclusiones parciales

En este capítulo se describieron las pruebas efectuadas a las funcionalidades del sistema. Las pruebas de caja blanca efectuadas a los métodos principales del sistema mediante la técnica “camino básico” permitieron un estudio de la operatividad del sistema, esto se refiere a que todos los caminos se ejecutan al menos una vez. Las pruebas de aceptación permitieron verificar que el software realiza lo especificado en las reglas del negocio. Las pruebas de caja negra aplicadas a las HU permitieron detectar un conjunto de no conformidades que fueron resueltas en dos iteraciones, obteniéndose la solución esperada por el cliente.

CONCLUSIONES GENERALES

La investigación evidenció la necesidad de incorporar las nuevas funcionalidades al Sistema para la generación de horario docente de la Facultad de Ingeniería Industrial de la CUJAE, ya que los sistemas estudiados son software privados o no se ajustan a las características y restricciones propias de la CUJAE.

El diseño e implementación de los módulos Cambio de turnos, Reservación de locales y Asignación de locales para exámenes finales permitió disminuir los errores y el tiempo invertido en estas actividades.

Las pruebas realizadas permitieron comprobar el adecuado diseño del sistema y el cumplimento de los requisitos acordados con el cliente.

La incorporación de los módulos Cambio de turnos, Reservación de locales y Asignación de locales para exámenes finales al Sistema para la generación de horario docente de la CUJAE contribuye a disminuir el tiempo empleado por la planificadora para realizar estas actividades, las cuales estaban propensas a errores debido a que se realizaban de forma manual. El sistema presenta una solución factible a la rigidez del horario docente generado, ya que al digitalizar estas funcionalidades se evita que se viole en la menor medida posible el orden de la secuencia de actividades y las restricciones fuertes establecidas por la institución.

RECOMENDACIONES

- Incluir la funcionalidad cambio de restricciones débiles por fuertes y viceversa.
- Valorar la factibilidad de extender la solución a las demás Facultades de la CUJAE que comparten características y restricciones similares a las de la Facultad de Ingeniería Industrial.
- Adaptar la solución informática para su visualización en dispositivos móviles.

BIBLIOGRAFÍA

Alcantara Rabí, Dayanis Elvia y Hernández Luque, Eylín. 2010. Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas de Información Geográfica. *Revista vinculado.* [En línea] 4 de Enero de 2010. [Citado el: 1 de Mayo de 2015.] http://vinculando.org/articulos/sociedad_america_latina/propuesta_guia_de_medidas_para_evaluacion_sistemas_informacion.html.

Apache. 2015. Apache Software Foundation. *Apache Software Foundation.* [En línea] Apache Software Foundation, 2015. [Citado el: 5 de 4 de 2015.] link: http://jmeter.apache.org/.

—. 2015. The Apache Software Foundation. *The Apache Software Foundation.* [En línea] The Apache Software Foundation, 2015. [Citado el: 3 de 4 de 2015.] www.apache.org.

asc Time Tables. 2015. asc Time Tables. *asc Time Tables.* [En línea] © 2015 aSc Applied Software Consultants, 2015. [Citado el: 2 de 3 de 2015.] http://www.asctimetables.com/.

Asenjo González, Diego Andrés y Ríos Peña, Alejandro. 2003. *Patrones de Diseño.* Habana : s.n., 2003.

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

Blaha, Michael. 2010. *Patterns of Data Modeling.* Reino Unido : CRC Press, 2010.

Bowen, Rich. 2007. *What's New in Apache Web Server 2.2?* s.l. : s.l. : O'Reilly Media, Inc., 2007. 2007.

Briand, L., S. Morasca. 1996. *Property-Based Software Engineering Measurement.* s.l. : IEEE Transactions on Software Engineering, 1996. 22(1): 68-86..

Buono, Enrique. 2012. Patrones fundamentales en el Diseño Orientado a Objetos (DOO). *Red Colaborativa Postgrado UCV.* [En línea] 2012. [Citado el: 25 de 3 de 2015.] http://kuainasi.ciens.ucv.ve/red_educativa/blogs/42.

Burke, Edmund Kieran, y otros. 2007. *A graph-based hyper-heuristic for educational timetabling problems.* s.l. : European Journal of Operational Research, 2007. Vol. 176, pp. 177–192.

Calabria, Luis y Píriz, Pablo. 2003. *Metodología XP.* Universidad ORT Uruguay : Editorial: Universidad ORT, 2003.

—. 2003. *Metodología XP.* Universidad ORT Uruguay : s.n., 2003.

Casas, Sandra y Reinaga, Héctor. 2009. *Aspectos Tempranos: un enfoque basado en Tarjetas CRC.* Argentina : Sandra Casas y Héctor Reinaga, 2009.

CSS3. 2015. CSS3.com. *CSS3.com.* [En línea] CSS3.com, Mayo de 2015. [Citado el: 22 de Febrero de 2015.] http://www.css3.com/.

Cuenca Lucero, Fredy. 2007. Técnicas de inteligencia artificial aplicadas a la confección de horarios. *Revista digital de la Facultad de Ingeniería de Sistemas, Universidad de Lima.* [En línea] 2007. [Citado el: 16 de Abril de 2014.] http://www.ulima.edu.pe/pregrado/ingenieria-de-sistemas/interfases-ndeg-2-2007. Nº 2, pp. 35-50.

CUFM. 2012. Fundamento del Diseño de Software. *Unidad 5. Estándares de Diseño.* [En línea] IngSoftwareII-CUFM , 2012. [Citado el: 18 de 4 de 2015.] https://cufmingsoftware.wordpress.com/estandares-de-diseno/.

Dechter, Rina y Mateescu, Robert. 2007. *AND/OR search spaces for graphical models.Artificial intelligence.* University of California, Irvine : Elsevier, 2007. Vol. 171, No. 2, pp. 73-106..

Díaz, Javier, y otros. 2009. *Herramientas open source para testing de aplicaciones Web. Evaluación y usos.* Universidad de La Plata. Buenos Aires. Argentina : s.n., 2009.

Echevarría Cartaya, Yuviny. 2009. *Sistema Informático para la Confección de Horarios Docentes en la Facultad de Informática de la Universidad de Cienfuegos.* Universidad de Cienfuegos : s.n., 2009.

Ecured. 2015. Ecured. *Requisitos no funcionales.* [En línea] 2015. [Citado el: 27 de 3 de 2015.] http://www.ecured.cu/index.php/Requisitos_no_funcionales.

EcuRed. 2013. EcuRed. *EcuRed.* [En línea] 2013. [Citado el: 6 de Abril de 2015.] http://www.ecured.cu/index.php/Métrica_de_diseño.

Eguiluz, Javier . 2015. LIBROSWEB. *LIBROSWEB.* [En línea] 2015. [Citado el: 1 de 3 de 2015.] http://librosweb.es/libro/javascript/..

EJie. 2009. *JMeter.* Sociedad Informatica del Gobierno Vasco : EJie, 2009.

Fernández Romero, Yenisleidy y Díaz, González, Yanette. 2012. REVISTA DIGITAL DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES. *Patrón Modelo-Vista-Controlador.* [En línea] 2012. [Citado el: 25 de 3 de 2015.] http://revistatelematica.cujae.edu.cu/index.php/tele/article/viewFile/15/10.

Franco Baquero, John Fredy , Toro Ocampo, Eliana Mirledy y Gallego Rendón, Ramón Alfonso. 2008. Problema de asignación óptima de salones resuelto con Búsqueda Tabú. *Scielo.* [En línea] 2008. [Citado el: 14 de Enero de 2015.] http://www.scielo.org.co/scielo.php?pid=S0122-34612008000200011&script=sci_arttext. ISSN 2145-9371.

Gauchat, Juan Diego. 2012. *El gran libro de HTML5, CSS3 y JavaScript.* Barcelona : MARCOMBO, 2012.

—. 2012. *El gran libro de HTML5, CSS3 y Javascript.* Barcelona, España : Marcombo, 2012. ISBN: 978-84-267-1770-2.

Gómez Fuentes, María del Carmen . 2011. *MATERIAL DIDÁCTICO: ANÁLISIS DE REQUERIMIENTOS.* Universidad Autonoma Metropolitana: Mexico : Casa abierta al tiempo, 2011. IBSN: 978-607-477-442-9.

Gruber & Petters. 2015. Untis. *Untis.* [En línea] Untis gruber & petters, 2015. [Citado el: 2 de 3 de 2015.] http://www2.grupet.at/espanol/produkte/untis/aktuelle_version.php.

Grupo ISSI. 2003. *Metodologías Ágiles en el Desarrollo de Software.* Alicante : s.n., 2003.

Guerrero, Álvaro, Suárez, Víctor F y Castrillón, Omar D. 2013. Scielo. *Programación de Horarios Escolares basados en Ritmos Cognitivos usando un Algoritmo Genético de Clasificación No-dominada, NSGA-II.* [En línea] 2013. [Citado el: 5 de Diciembre de 2014.] http://www.scielo.cl/pdf/infotec/v24n1/art12.pdf. Vol. 24, No. 1.

GuilleSQL. 2011. GuilleSQL. *GuilleSQL.* [En línea] GuilleSQL, 2011. [Citado el: 24 de Marzo de 2015.] http://www.guillesql.es/Articulos/Claves_Subrogadas_Slowly_Changing_Dimension_SCD_Tipo_2.aspx.

Hernández, Rodrigo, Miranda, Jaime 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.

Hojjat, Adeli, See y Karim, Asim. 2003. *Construction Scheduling, Cost Optimization and Management.* Lahore University of Management Sciences : Editorial Board, 2003. p.54.

IEEE. 2015. IEEE. *IEEE.* [En línea] 2015. [Citado el: 24 de 3 de 2015.] http://www.computer.org/cms/Computer.org/Computer.org/s2esc/s2esc_pols/SP-06_Vocabulary_Objectives.htm.

INEI. 2015. academia. *Herramientas Case.El mejor soporte para el proceso de desarrollo de software.* [En línea] 2015. [Citado el: 12 de 4 de 2015.] http://www.academia.edu/4513393/Libro_HERRAMIENTAS_CASE.

INTECO. 2009. *INGENIERÍA DEL SOFTWARE:METODOLOGÍAS Y CICLOS DE VIDA.* España : INTECO, 2009. ISBN: 9788478290963.

Jacobson, I y Booch, G & Rumbaugh, J. 2000. *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación. S.A., 2000.

JetBrains. 2013. JetBrains s.r.o. *JetBrains.* [En línea] 2013. [Citado el: 3 de 12 de 2014.] referencia: https://www.jetbrains.com/phpstorm/features/.

Juristo, Natalia, Moreno, Ana M y Vegas, Sira. 2006. *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* Facultad de Informática de Madrid : s.n., 2006. 12.0.

Larman, Craig. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : PRENTICE-HALL, 1999.

Letelier Torres, Patricio y Otros. 2003. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Universidad Politécnica de Valencia (UPV) : 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.* New Jersey : Englewood Cliffs, 1994.

Manyá, Felip y Gomes, Carla. 2003. *Técnicas de resolución de problemas de satisfacción de restricciones.* La Universidad de Lérida : Grupo Planeta, 2003. 1137-3601.

MDN. 2015. Mozilla Developer Network. *Mozilla Developer Network.* [En línea] © 2005-2015 Mozilla Developer Network and individual contributors, 2015. [Citado el: 23 de 2 de 2015.] https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Obsolete_Pages/Guía_JavaScript_1.5/Concepto_de_JavaScript#toc.

Mendes Calo, Karla, Estevez, Elsa y Fillottrani, Pablo. 2009. *Un Framework para Evaluación de Metodologías Ágiles.* Argentina.Universidad Nacional del Sur : EdiUNS, 2009.

Microsoft. 2015. Microsoft Developer Network. *Microsoft Developer Network.* [En línea] Microsoft, 2015. [Citado el: 4 de 5 de 2015.] https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

Miguel Ángel Álvarez. 2009. desarrolloweb.com. *desarrolloweb.com.* [En línea] 14 de Octubre de 2009. [Citado el: 22 de Febrero de 2015.] http://www.desarrolloweb.com/articulos/que-es-html5.html.

Mozilla. 2015. Mozilla Foundation. *Mozilla Foundation.* [En línea] Mozilla, 2015. [Citado el: 5 de 4 de 2015.] https://www.mozilla.org/es-ES/firefox/new/.

Navarro Cadavid, Andrés y Fernández Martínez, Juan Daniel & Morales Vélez, Jonathan. 2013. *Revisión de metodologías ágiles para el desarrollo de software.* Colombia : Editorial:"Universidad ICESI", 2013. Vol. 11, No. 2.

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

Pérez Benitez, Alfredo & Sánchez Ortiz, Susana & León Ramos, Dayana. 2014. *Portal Cubano para la Migración a Software Libre.* Cuba : Grupo Unicornios, 2014.

Pfleeger, S. L. 1997. *Assessing Software Measurement.* s.l. : IEEE Software, 1997. March/April: 25-26.

PHP. 2015. PHP. *PHP.* [En línea] The PHP Group, 2015. [Citado el: 8 de 4 de 2015.] http://php.net/manual/es/security.intro.php.

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

Piattini, Miguel A y E, Marcos. 1999. *Diseño de base de Datos Relacionales. Ra-Ma.* 1999.

PostgreSQL. 2013. The PostgreSQL Global Development Group. *PostgreSQL.* [En línea] 9 de 9 de 2013. [Citado el: 4 de 12 de 2014.] http://www.postgresql.org/docs/current/static/supported-platforms.html.

—. 2013. The PostgreSQL Global Development Group. *PostgreSQL.* [En línea] 2013. [Citado el: 4 de 12 de 2014.] http://www.postgresql.org/about/press/presskit93/es/.

Pressman, Roger. 2005. Estrategía de prueba 6ta edición Cap 13. *Ingeniería del Software: Un Enfoque Práctico.* Mexico : MCGRAW-HILL / INTERAMERICANA DE MEXICO, 2005.

—. 2002. *Ingenieria del Software, Un Enfoque Practico.* Mexico : McGraw-Hill, 2002. ISBN: 8448132149.

—. 2005. Métricas del producto para el sotfware 6ta edición Cap 15. *Ingeniería del Software: Un Enfoque Práctico.* Mexico : MCGRAW-HILL / INTERAMERICANA DE MEXICO, 2005.

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

—. 2005. *Ingeniería del Software. Un enfoque práctico 6ta edición.* Nueva York : McGraw-Hill, 2005. 0072853182.

Pressman, Roger S. 2005. *Ingeniería del Software. Un enfoque práctico.* s.l. : McGraw-Hill, 6ta edición, 2005. ISBN: 9701054733.

—. 2005. *Ingeniería del Software. Un enfoque práctico:Capitulo 13. Estrategias de pruebas.* USA : McGraw-Hil, 6ma edición, 2005. ISBN 978-0-07-337597-7.

Proyectos agiles. 2015. proyectosagiles.org. *proyectosagiles.org.* [En línea] Creative Commons by-sa, 2015. [Citado el: 22 de 5 de 2015.] http://www.proyectosagiles.org/que-es-scrum.

RAE. 2015. RAE. *RAE.* [En línea] RAE, 2015. [Citado el: 8 de 2 de 2015.] http://lema.rae.es..

—. 2015. Real Academia Española. [En línea] 2015. [Citado el: 16 de 3 de 2015.] http://lema.rae.es/drae/srv/search?key=planificación.

Rivadeneira Molina, Silvia Gabriela. 2012. Revista de informes científicos y técnicos de la Universidad Nacional de la Patagonia Austral. *Metodologías ágiles enfocadas al modelado de requerimientos.* [En línea] Mayo de 2012. [Citado el: 13 de Enero de 2015.] http://ict.unpa.edu.ar/files/ICT-UNPA-57-2013.pdf. ISSN: 1852 - 4516.

Robert C, Martin. 2002. *Agile Software Development, Principles, Patterns, and Practices.* s.l. : Prentice Hall, 1st edition, 2002. ISBN-10: 0135974445, ISBN-13: 978-0135974445.

Roque Cuevas, Elena García. 2007. *Principios básicos de Informática.* Madrid, España : Dykinson, 2007, 2007. ISBN: 978-84-9849-098-5.

Sabar, Nasser R, y otros. 2012. *A honey-bee mating optimization algorithm for educational timetabling problems.* Amsterdam : European Journal of Operational Research, 2012. Vols. 216, pp 533-543. ISSN 0377-2217, ZDB-ID 2430034. Vol. 216, pp. 533-543.

Sánchez, Esther Guerra. 2010. Esther Guerra. *Escuela Politécnica Superior.* [En línea] 2010. [Citado el: 2 de Mayo de 2015.] http://arantxa.ii.uam.es/~eguerra/docencia/0809/10%20Decorator.pdf.

Schwaber, Ken y Sutherland, Jeff. 2013. *La Guía de Scrum.* s.l. : PLANETA, 2013. ISBN: 9788408135326.

Shore, James y Warden, Shane. 2008. *The art of agile development.* USA : O'Reilly Media, 2008. 1ra edición, ISBN-13: 978-0-596-52767-9, ISBN-10: 0-596-52767-5.

Stinson, Barry. 2001. *PostgreSQL: Essential Reference.* s.l. : s.l. : Sams Publishing, 2001. ISBN: 0752064711216.

Suárez, María Lorena. 2015. Competencias en TIC: Colección de Fasículos Digitales. *Cuaderno 1: Introducción a la programación y sus lenguajes.* [En línea] 2015. [Citado el: 5 de 4 de 2015.] http://competenciastic.educ.ar/pdf/lenguajes_de_programacion_1.pdf.

Suárez, María Lorena. 2015. Competencias en TIC: Colección Fascículos Digitales. *Cuaderno 4: Lenguajes del lado del servidor y lenguajes del lado del usuario.* [En línea] 2015. [Citado el: 17 de 4 de 2015.] http://escritorioalumnos.educ.ar/datos/recursos/lenguajes_de_programacion_4.pdf.

Thornton, Jacob y Otto, Mark. 2012. Bootstrap. *Bootstrap.* [En línea] 2012. [Citado el: 8 de 12 de 2014.] http://getbootstrap.com/2.3.2/index.html.

USM.cl. 2015. UNIVERSIDAD TÉCNICA FEDERICO SANTA MARIA. [En línea] Sitio web administrado por la Dirección General de Comunicaciones, 2015. [Citado el: 19 de 5 de 2015.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08.

Varona, Karel Rodríguez. 2012. *Aplicación de algoritmos genéticos en la generación automática de horarios docentes en la Facultad Regional de Granma.* Manzanillo : Revista Cubana de Ciencias Informáticas, 2012.

Visconti, Marcello y Astudillo, Hernán. 2004. *Fundamento de Ingeniería de Software.* Chile : USM, 2004.

Visual Paradigm. 2000. Visual Paradigm International. *Visual Paradigm.* [En línea] 2000. [Citado el: 4 de 12 de 2014.] http://www.visual-paradigm.com.