INSTITUTO SUPERIOR MINERO-METALÚRGICO
"Dr. Antonio Núñez Jiménez"
FACULTAD "METALURGIA-ELECTROMECÁNICA"

Trabajo de Diploma

para optar por el título de Ingeniero Informático

SISTEMA PLANIFICADOR DE GUARDIAS ESTUDIANTILES

Autor: Reidelbis Calderón Suárez

Tutor: Ing. Eloy R. Jiménez Iglesias

Moa, julio, 2011
Año del 52 Aniversario del Triunfo de la Revolución

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Instituto Superior Minero Metalúrgico de Moa “Dr. Antonio Núñez Jiménez” y al Departamento de Informática para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los _____ días del mes de _____ del 2011.

Reidelbis Calderón Suárez

_____
Firma autor

Ing. Eloy R. Jiménez Iglesias

_____
Firma tutor

Opinión del tutor del trabajo de diploma

Título:

Autor:

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

<Aquí el tutor debe expresar cualitativamente su opinión y medir (usando la escala: muy alta, alta, adecuada) entre otras las cualidades siguientes:

• Independencia

• Originalidad

• Creatividad

• Laboriosidad

• Responsabilidad>

<Además, debe evaluar la calidad científico-técnica del trabajo realizado (resultados y documento) y expresar su opinión sobre el valor de los resultados obtenidos (aplicación y beneficios) >

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de <nota 2-Desaprobado, 3-Aprobado, 4-Bien, 5-Excelente>. <Además, si considera que los resultados poseen valor para ser publicados, debe expresarlo también>

_____

Nombre completo del primer tutor

<Grado científico, Categoría docente

y/o investigativa>

(Si procede)

_____

Nombre completo del segundo tutor

<Grado científico, Categoría docente

y/o investigativa>

Fecha: _____

Pensamiento

"Cada persona forja su propia grandeza. Los enanos permanecerán enanos aunque se suban a los Alpes".

August von Kotzebue

Dedicatoria

Le dedico este trabajo a mi hermanita Daniela Bárbara Calderón Suárez.

Agradecimientos

A mis padres Relbis Calderón Romero y Elsa Suárez Carballoza por brindarme todo su apoyo en el transcurso de mi vida.

A mi hermana Yané Calderón Suárez.

A mi familia por el apoyo brindado en todo momento.

A mis compañeros de grupo y de la residencia estudiantil.

A todos aquellos que de una forma u otra han contribuido con mi formación como futuro profesional y a la realización de este trabajo.

Muchas gracias.

Resumen

El presente trabajo expone detalles de la elaboración de un Planificador de guardias para la realización de la misma en el Instituto Superior Minero Metalúrgico de Moa “Dr. Antonio Núñez Jiménez”, que no contaba con este servicio para sus departamentos. Contiene un análisis del estado del arte de la materia en el mundo y el estudio de la factibilidad del proyecto. Quedan expuestos los pormenores del diseño e implementación: modelos, algoritmos, diseño, clases y herramientas empleadas.

Abstract

The present work exposes detail of the manufacture of an Organizer on guard for the realization of the same thing in the superior mining institute metallurgist of Moa “Dr. Antonio Núñez Jimenez” that it did not count on this service for your departments. It contains an analysis of the state of the art of the matter in the world and the study of the feasibility of the project. Remain exposed the details of the design and implementation: models, employees algorisms, design, classes and tools.

Tabla de contenidos

Introducción **1**

Capítulo 1 Fundamentación teórica **5**

1.1 Introducción 5

1.2 Estado del arte 5

1.2.1 Turnex 5

1.2.2 Tecnohospital 6

1.3 Tendencias y tecnologías actuales 7

1.3.1 Aplicaciones Web 7

1.3.1.1 Página Web 8

1.3.1.2 Sitio Web 9

1.3.2 Servidor Web 9

1.3.2.1 Servidor HTTP Apache 10

1.3.2.2 IIS 11

1.3.2.3 Tomcat 11

1.3.3 Lenguajes de Programación 12

1.3.3.1 PHP 12

1.3.3.2 HTML 14

1.3.3.3 JavaScript 15

1.3.4 Sistemas Gestores de Bases de Datos 16

1.3.4.1 SQL 17

1.3.4.2 MySQL 18

1.3.4.3 PostgreSQL 20

1.3.5 Metodologías para el desarrollo de Sistemas Informáticos 21

1.3.5.1 Metodología RUP 22

1.3.5.2 Metodología XP 25

1.3.5.3 Metodología MSF 27

1.4 Arquitecturas para el desarrollo e implementación de software 29

1.4.1 Arquitecturas en tres capas 29

1.4.2 Patrón Arquitectónico Modelo Vistas Controladores (MVC) 31

1.5 Propuesta de solución para el sistema a desarrollar 32

1.5.1 Dreamweaver como entorno de desarrollo 33

1.5.2 Fundamentación del lenguaje de programación a utilizar 33

1.5.3 Fundamentación del SGBD a utilizar 34
1.5.4 Fundamentación de la metodología a utilizar 35
1.5.5 Fundamentación de la arquitectura a utilizar 36
1.6 Conclusiones 37
Capítulo 2 Planeación y Diseño 38
2.1 Introducción 38
2.2 Funcionalidad principal 38
2.3 Lista de reserva 39
2.4 Historias de usuario 40
2.5 Planificación de entregas 41
2.5.1 Estimación de esfuerzo por historia de usuario 42
2.5.2 Planificación de iteraciones 43
2.5.3 Plan de duración de las iteraciones 44
2.6 Tarjetas CRC 44
2.7 Conclusiones 45
Capítulo 3 Desarrollo y Pruebas 46
3.1 Introducción 46
3.2 Modelo de datos 46
3.3 Desarrollo de las iteraciones 47
3.3.1 Tareas por historias de usuario 47
3.3.2 Iteración No.1 48
3.3.3 Iteración No.2 49
3.3.4 Iteración No.3 50
3.4 Principales interfaces de la aplicación 50
3.5 Pruebas 51
3.5.1 Desarrollo dirigido por pruebas 51
3.5.2 Pruebas de aceptación 51
3.6 Conclusiones 53
Capítulo 4 Estudio de Factibilidad 54
4.1 Introducción 54
4.2 Efectos económicos 54
4.2.1 Efectos directos 55
4.2.2 Efectos indirectos 55
4.2.3 Efectos externos 55

4.2.4 Intangibles 56
4.3 Beneficios y Costos Intangibles en el proyecto 56
4.4 Ficha de costo 57
4.4.1 Costos en Moneda Libremente Convertible 57
4.4.2 Costos en Moneda Nacional 58
4.5 Conclusiones 60
Conclusiones Generales 61
Recomendaciones 62
Fuentes bibliográficas 63
Glosario de términos 66
Anexos … 68
Anexo # 1 Historias de usuarios. 68
Anexo # 2 Tarjetas CRC. 74
Anexo # 3 Tarjetas de Ingeniería. 77
Anexo # 4 Pruebas de Aceptación. 86
Anexo # 5 Principales interfaces de la Aplicación. 95

Índice de Figuras

Fig. 1.1 Ciclo de vida de la Metodología RUP 23
Fig. 1.2 Metodología MSF 27
Fig. 1.3 Arquitectura en capas..... 30
Fig. 1.4 MVC 31
Fig. 1.5 Arquitectura Cliente/Servidor en tres capas. 37
Fig. 3.1 Modelo de datos..... 46
Fig. 4.1 Gráfica de la solución manual y solución automatizada 59
Interfaz principal del sistema. 95
Interfaz de la planificación de la guardia. 95
Interfaz de la gestión de usuarios..... 96
Interfaz de la realización de cambios, de fecha..... 96
Interfaz del resumen y exportación de la planificación. 97

Índice de tablas

Tabla 2.1 Personal que trabajará con el sistema 39
Tabla 2.2 Planilla de historia de usuario..... 41
Tabla 2.3 HU No.1: Autenticar usuarios 41
Tabla 2.14 Estimación de esfuerzos por historias de usuario 42
Tabla 2.15 Plan de duración de las iteraciones..... 44
Tabla 2.16 Modelo de las tarjetas CRC..... 45
Tabla 2.17 Tarjeta CRC No 1: Autenticar usuarios. 45
Tabla 3.1 Tareas por historias de usuarios 48
Tabla 3.2 Historias de usuario abordadas en la primera iteración..... 48
Tabla 3.3 Tarjeta de tarea No.1 Autenticar usuarios 49
Tabla 3.12 Historias de usuario abordadas en la segunda iteración. 49
Tabla 3.17 Historias de usuario abordadas en la tercera iteración..... 50
Tabla 3.23 Plantilla de prueba de aceptación..... 52
Tabla 3.24 Prueba para comprobar la Autenticación de usuarios. 53
Tabla 2.4 HU No.2: Configurar bases de datos..... 68
Tabla 2.5 HU No.3 Cambiar contraseña. 69
Tabla 2.6 HU No.4 Gestionar Usuarios. 69
Tabla 2.7 HU No.5 Realizar la planificación del departamento. 70
Tabla 2.8 HU No.6 Realizar cambios en la planificación..... 70
Tabla 2.9 HU No.7 Justificar estudiantes incapacitados. 71
Tabla 2.10 HU No.8 Mostrar planificación generada..... 71
Tabla 2.11 HU No.9 Mostrar estudiantes que no realizarán guardia. 72
Tabla 2.12 HU No.10 Generar el resumen de la planificación..... 72
Tabla 2.13 HU No.11 Exportar el resumen de la planificación. 73
Tabla 2.18 Tarjeta CRC No 2: Configurar la base de datos del sistema. 74
Tabla 2.19 Tarjeta CRC No 3: Cambiar contraseña. 74
Tabla 2.20 Tarjeta CRC No 4: Gestionar Usuarios. 74
Tabla 2.21 Tarjeta CRC No 5: Realizar la planificación del departamento..... 75
Tabla 2.22 Tarjeta CRC No 6: Realizar cambios en la planificación. 75
Tabla 2.23 Tarjeta CRC No 7: Justificar estudiantes incapacitados. 75
Tabla 2.24 Tarjeta CRC No 8: Mostrar planificación generada. 75

Tabla 2.25 Tarjeta CRC No 9: Mostrar estudiantes que no realizarán guardia. 76
Tabla 2.26 Tarjeta CRC No 10: Generar el resumen de la planificación. 76
Tabla 2.27 Tarjeta CRC No 11: Exportar el resumen de la planificación. 76
Tabla 3.4 Tarjeta de tarea No.2 Configurar la base de datos del sistema. 77
Tabla 3.5 Tarjeta de tarea No.3 Configurar el acceso a datos del sistema. ... 77
Tabla 3.6 Tarjeta de tarea No.4 Cambiar contraseña de la configuración del sistema. 78
Tabla 3.7 Tarjeta de tarea No.5 Cambiar contraseña. 78
Tabla 3.8 Tarjeta de tarea No.6 Insertar usuarios. 79
Tabla 3.9 Tarjeta de tarea No.7 Modificar usuarios. 79
Tabla 3.10 Tarjeta de tarea No.8 Eliminar usuarios. 80
Tabla 3.11 Tarjeta de tarea No.9 Mostrar usuarios. 80
Tabla 3.13 Tarjeta de tarea No.10 Realizar la planificación. 81
Tabla 3.14 Tarjeta de tarea No.11 Realizar cambios en la planificación de hora. 81
Tabla 3.15 Tarjeta de tarea No.12 Realizar cambios en la planificación de fecha. 82
Tabla 3.16 Tarjeta de tarea No.13 Justificar estudiantes incapacitados. 82
Tabla 3.18 Tarjeta de tarea No.14 Mostrar planificación por departamentos. 83
Tabla 3.19 Tarjeta de tarea No.15 Mostrar planificación por estudiantes. 83
Tabla 3.20 Tarjeta de tarea No.16 Mostrar estudiantes que no realizarán guardia. 84
Tabla 3.21 Tarjeta de tarea No.17 Generar el resumen de la planificación. .. 84
Tabla 3.22 Tarjeta de tarea No.18 Exportar el resumen de la planificación. .. 85
Tabla 3.25 Prueba para comprobar la configuración de las bases de datos. . 87
Tabla 3.26 Prueba para comprobar el cambio de contraseña. 87
Tabla 3.27 Prueba para verificar la gestión de usuarios. 88
Tabla 3.28 Prueba para comprobar la planificación de la guardia. 89
Tabla 3.29 Prueba para comprobar los cambios en la planificación 90
Tabla 3.30 Prueba para comprobar como se muestra la planificación. 91
Tabla 3.31 Prueba para comprobar como se muestra los estudiantes que no realizarán guardia. 92

Tabla 3.32 Prueba para comprobar como se genera el resumen de la planificación. 93
Tabla 3.33 Prueba para comprobar como se exporta el resumen de la planificación. 94

Introducción

La informática es la disciplina que estudia el tratamiento automático de la información utilizando dispositivos electrónicos y sistemas computacionales la misma se aplica a diversas áreas, como por ejemplo: gestión de negocio, almacenamiento de información, monitorización y el control de procesos, logrando adentrarse ya sea en el ámbito económico, social o político de toda una nación. **[22]**

Hoy en día la automatización de los procesos que se realizan a mano, han posibilitado que la informática se convierta en una herramienta imprescindible para lograr el resultado esperado, siendo así que se han realizado sistemas capaces de diseñar, calcular y hasta de realizar planificaciones.

Cuando se habla de planificación se refiere a las acciones llevadas a cabo para realizar planes y proyectos de diferente índole. El proceso de planeación sigue un conjunto de pasos que se establecen inicialmente, y quienes realizan la planificación hacen uso de las diferentes expresiones y herramientas con que cuenta la planeación. La planificación ejecuta los planes desde su concepción, y si es el caso se encarga de la operación en los diferentes niveles y amplitudes de la planeación. La planificación se lleva a cabo de manera empírica en muchas situaciones cotidianas, y de manera muy seria y formal en organismos que dependen de una planificación adecuada y sistemática. La planificación se encuentra, tal como la administración, dentro de la mayoría de las actividades de las personas, instituciones y organismos de toda índole. **[21]**

En el mundo una buena planificación, ha hecho que ciertos países hayan logrado un desarrollo factible de sus economías, convirtiéndola en un instrumento inevitable para subsistir. Nuestro país no esta exceptuado de esto y las entidades han adoptado nuevas tendencias que mejoran la gestión de las organizaciones y se han visto inmersas en la automatización de las actividades que lo requieran, como vía indispensable para incrementar su eficiencia y eficacia.

Los Departamentos correspondientes al Instituto Superior Minero Metalúrgico de Moa posen una persona encargada de realizar la planificación de las guardias de los estudiantes, este proceso es llevado a cabo de forma manual utilizando en su creación tiempo y recursos, este individuo debe de tener en cuenta una serie de parámetros que determinarán la correcta planificación para que la misma sea llevada con éxito, de no ser así al distribuirse los estudiantes por posiciones las mismas quedarían mal cubiertas o no serian cubiertas en su totalidad. Además de realizar la planificación quedaría la tarea de dar a conocer con anticipación como quedo organizada la misma a los estudiantes para que si uno de ellos tuviese alguna dificultad con el turno asignado se le informe al encargado, el cual tendría que realizar los cambios pertinentes. Toda esta problemática constituye la **situación problémica** de esta investigación.

Con este trabajo se pretende dar solución a la situación anteriormente expuesta, definiéndose como **problema científico:** la no existencia de un planificador de guardias estudiantiles necesario para una mejor y más rápida organización de la misma en el Instituto Superior Minero Metalúrgico de Moa.

Entonces se impone la necesidad de desarrollar un Planificador de guardias que disponga en su función las variables necesarias para crear una correcta planificación, facilitando así todo el proceso.

El **objeto de estudio** de este trabajo es la automatización del proceso de planificación de guardias. El **campo de acción** se enmarca en la implementación de un planificador de guardias para los departamentos del Instituto Superior Minero Metalúrgico de Moa “Dr. Antonio Núñez Jiménez”.

Para dar respuesta al problema científico se plantea como **idea a defender** que: el desarrollo de una herramienta para la planificación de las guardias de los estudiantes, agilizará el proceso de esta actividad en el Instituto Superior Minero Metalúrgico de Moa “Dr. Antonio Núñez Jiménez” y por consiguiente la optimización de este proceso.

El **objetivo general** del trabajo en cuestión es analizar, diseñar e implementar un sistema planificador de guardias para la asignación de las mismas a los estudiantes del instituto.

Se plantean los siguientes **objetivos específicos**:

- Analizar el planificador de guardias.
- Diseñar el planificador de guardias.
- Implementar el planificador de guardias.
- Analizar la factibilidad del proyecto.

Para cumplir con estos objetivos y resolver la situación problémica planteada, se proponen las siguientes **tareas**:

- Entrevistas con el cliente.
- Análisis de los procesos actuales.
- Elaboración de los fundamentos teóricos de la investigación.
- Seguir cada una de las etapas de la ingeniería del software hasta la implementación de un sistema que permita la planificación de las guardias de los estudiantes.
- Estudio de la Factibilidad y Sostenibilidad.
- Prueba y Documentación del Sistema.
- Desarrollo del Manual de Usuarios.

Como **métodos de investigación científica** se utilizaron:

Métodos Teóricos:

Histórico lógico: Posibilitó el análisis histórico del proceso de gestión de información.

Métodos Empíricos:

Entrevistas: Se realizan entrevistas con el fin de precisar el problema a resolver, así como para la validación de la propuesta que se presenta.

Análisis de documentos: Se basa en la revisión a documentos utilizados en la investigación.

El presente trabajo consta de cuatro capítulos:

Capítulo 1 Fundamentación Teórica: se realiza un análisis del estado del arte en el mundo vinculado al sistema a desarrollar. Se fundamenta en varios lenguajes de programación existentes, así como en algunas metodologías, detallándose por ultimo las herramientas a utilizar en nuestra aplicación.

Capítulo 2 Planeación y Diseño: utilizándose la metodología propuesta se detallan los aspectos relacionados con la fase de Planeación y Diseño.

Capítulo 3 Desarrollo y Pruebas: se presenta el modelo de datos correspondiente y se realiza el desarrollo de las iteraciones a partir del desglose de las historias de usuario en tareas. Describiéndose por ultimo las pruebas realizadas y sus resultados.

Capítulo 4 Estudio de Factibilidad: se detalla mediante un estudio realizado los esfuerzos requeridos para la construcción del sistema y se valora la sostenibilidad del producto.

Capítulo 1 Fundamentación teórica

1.1 Introducción

En este capitulo se pretende realizar una explicación sobre los aspectos fundamentales que posibilitarán el desarrollo de esta aplicación. Proporcionándose detalles del estado del arte en el mundo, así como de las herramientas que se manipularán en el desarrollo de la misma y del lenguaje de programación a utilizar, brindándose además detalles importantes de la metodología a emplear.

1.2 Estado del arte

En la actualidad se ha tratado de automatizar todos los procesos que se realizan a mano o de otra forma, intensificando así el uso de los softwares. El sistema que se intenta desarrollar no es la excepción, existen aplicaciones que cumplen ciertas exigencias de empresas que solicitan diferentes tipos de servicios incluyendo este.

1.2.1 Turnex

El software **Turnex** desarrollado por la compañía Solex es un sistema para programar y planificar los turnos de los empleados en forma fácil y eficiente. Su foco principal es resolver complejas asignaciones y cambios que se presenten en esta materia, por lo que está orientado a todas aquellas empresas que trabajan con personal rotativo las 24 horas del día. La idea es que en caso de cualquier inasistencia o contratiempo de algún trabajador, se maneje una base de datos para poder asignar de forma equitativa e inteligentemente a un reemplazante. Esto, a fin de cumplir con la demanda de cobertura que se acordó: externa (con clientes) o interna (con sus propios empleados). Turnex

también permite administrar el tipo de contrato de trabajo que existe con el personal y las coberturas de servicio que se ofrecen a los clientes.
La primera versión de Turnex se implementó en el año 2001 para controlar los turnos de los más de 600 empleados de una empresa de seguridad. Turnex se trata de un software diseñado especialmente para compañías con una demanda de cobertura las 24 horas del día: guardias de seguridad, personal de aseo, casino, entre otros. La experiencia ha sido tan exitosa que hoy se encuentra incorporado en importantes empresas, gestionando diariamente los turnos de trabajo de más de 50 mil trabajadores de diferentes empresas.

Actualmente se encuentra implementado en importantes empresas como el Grupo Securitas, el Grupo Eulen, y CdR Aramark, controlando empleados de casinos, aseo y seguridad. **[3]**

1.2.2 Tecnohospital

Planificar los turnos y las guardias es una de las tareas más complicadas para el área de enfermería de un hospital. La elevada cantidad de personal crea errores de planificación, todo ello sin contar el tiempo que puede llevar generar un horario compatible. Para facilitar esta labor, existen sin embargo, herramientas de gestión de turnos como **Tecnohospital**, la cual, según su fabricante Coprava, puede ayudar a mejorar hasta un 80 por ciento la gestión de los turnos y guardias de clínicas, ambulatorios y grandes hospitales.

El sistema de Tecnohospital genera automáticamente la planificación en base a unos parámetros definidos inicialmente como área; categoría; número de horas máximas según los convenios de enfermería; número de horas máximas de guardia; o número mínimo y máximo de enfermeros necesarios por turno. Así, el sistema genera automáticamente la planificación y el responsable del área de enfermería sólo tendrá que revisar y realizar pequeños ajustes en caso de que se produzca alguna incidencia con uno de los enfermeros o enfermeras de la plantilla.

Además, la solución ofrece un portal de usuarios con servicios para los enfermeros y enfermeras. Así, cada empleado podrá consultar las guardias que tiene asignadas, el calendario laboral, los días que dispone de vacaciones, los compañeros asignados a su turno de trabajo y hasta solicitar cambios de guardias. Además, a través de una clave personal cada enfermero podrá introducir cambios y elaborar dinámicamente sus propios protocolos. **[20]**

Como se ha podido apreciar estos softwares cumplen con las expectativas que esperan varias empresas, nosotros no podemos utilizar estas aplicaciones pues a pesar de que son de tipo propietarios necesitamos una aplicación que se adecue exactamente a nuestras necesidades.

1.3 Tendencias y tecnologías actuales

Tecnología es el conjunto de saberes que permiten fabricar objetos y modificar el medio ambiente, incluyendo plantas y animales, para satisfacer las necesidades y los deseos de nuestra especie.

Históricamente las tecnologías han sido usadas para satisfacer necesidades esenciales (alimentación, vestimenta, vivienda, protección personal, relación social, comprensión del mundo natural y social), para obtener placeres corporales y estéticos (deportes, música, hedonismo en todas sus formas) y como medios para satisfacer deseos (simbolización de estatus, fabricación de armas y toda la gama de medios artificiales usados para persuadir y dominar a las personas).

1.3.1 Aplicaciones Web

Las **aplicaciones Web** son sistemas informáticos, usados por los usuarios para acceder a un servidor Web, a través de Internet o Intranet, y obtener así la información que se encuentra almacenada en él. Las aplicaciones Web deben

su popularidad a la practicidad del navegador Web como cliente ligero, además a la facilidad que poseen para actualizar y mantener aplicaciones Web sin distribuir e instalar software en miles de potenciales clientes. Las aplicaciones Web son muy utilizadas hoy en día debido a las ventajas que las mismas ofrecen. Algunas de las cuales se ven reflejadas a continuación:

Compatibilidad multiplataforma: Tienen un camino mucho más sencillo para la compatibilidad con múltiples plataformas que las aplicaciones de escritorio. Varias tecnologías incluyendo Java, Flash, ASP y Ajax permiten un desarrollo efectivo de programas soportando todos los sistemas operativos principales.

Actualización: Están siempre actualizadas con la última versión, sin necesitad de llamar la atención del usuario o interferir con sus hábitos de trabajo, con la posibilidad de que va a iniciar nuevas descargas y procedimientos de instalación.

Inmediatez de acceso: No necesitan ser descargadas, instaladas y configuradas.

Menos Bugs (errores): Con aplicaciones basadas en Web, todos los usuarios utilizan la misma versión, y los errores pueden ser corregidos tan pronto como son descubiertos. **[24]**

1.3.1.1 Página Web

Una **página Web** es una fuente de información adaptada para la World Wide Web (WWW) y accesible mediante un navegador de Internet. Esta información se presenta generalmente en formato HTML y puede contener hiperenlaces a otras páginas Web, constituyendo la *red* enlazada de la World Wide Web.

Las páginas Web pueden ser cargadas de un ordenador o computador local o remoto, llamado Servidor Web, el cual servirá de HOST. El servidor Web puede

restringir las páginas a una red privada, por ejemplo, una intranet, o puede publicar las páginas en el World Wide Web. Las páginas Web son solicitadas y transferidas de los servidores usando el Protocolo de Transferencia de Hipertexto (HTTP - Hypertext Transfer Protocol). La acción del Servidor HOST de guardar la página Web, se denomina "HOSTING".

Las páginas Web pueden consistir en archivos de texto estático, o se pueden leer una serie de archivos con código que instruya al servidor como construir el HTML para cada página que es solicitada, a esto se le conoce como Página Web Dinámica. **[14]**

1.3.1.2 Sitio Web

Un **sitio Web** (en ingles: website) es un conjunto de páginas Web, típicamente comunes a un dominio de Internet o subdominio en la World Wide Web en Internet. Todos los sitios Web públicamente accesibles constituyen una gigantesca "World Wide Web" de información.

A las páginas de un sitio Web se accede desde una URL raíz común llamada portada, que normalmente reside en el mismo servidor físico. Las URLs organizan las páginas en una jerarquía, aunque los hiperenlaces entre ellas controlan como el lector percibe la estructura general y como el tráfico Web fluye entre las diferentes partes de los sitios.

Algunos sitios Web requieren una suscripción para acceder a algunos o todos sus contenidos. Ejemplos de sitios con suscripción incluyen muchos sitios de noticias, sitios de juegos, foros, servicios de correo electrónico basados en Web y sitios que proporcionan datos de bolsa en tiempo real. **[13]**

1.3.2 Servidor Web

Un **servidor Web** es un programa que implementa el *protocolo HTTP* (*hypertext transfer protocol*). Este protocolo está diseñado para transferir lo que

llamamos hipertextos, páginas Web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

Es un programa que se ejecuta continuamente en un ordenador (también se emplea el termino para referirse al ordenador que lo ejecuta), manteniéndose a la espera de peticiones por parte de un cliente (un navegador de Internet) y que responde a estas peticiones adecuadamente, mediante una *página Web* que se exhibirá en el navegador o mostrando el respectivo mensaje si se detecta algún error.

Instalar un servidor Web en nuestro PC nos permitirá, entre otras cosas, poder montar nuestra propia página Web sin necesidad de contratar hosting, probar nuestros desarrollos vía local, acceder a los archivos de nuestro equipo desde una PC remoto. **[23]**

1.3.2.1 Servidor HTTP Apache

El **servidor HTTP Apache** es un software (libre) servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero mas tarde fue reescrito por completo. Su nombre se debe a que Behelendorf eligió ese nombre porque quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la ultima en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de Internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en ingles, *a patchy server* (un servidor "emparchado").

Ventajas

- Modular
- Open source

- Multiplataforma
- Extensible
- Popular (fácil conseguir ayuda/soporte)
- Gratuito **[11]**

1.3.2.2 IIS

Internet Information Services, **IIS**, es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS.

Este servicio convierte a un ordenador en un servidor de Internet o Intranet es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas Web tanto local como remotamente (servidor Web). **[27]**

1.3.2.3 Tomcat

Tomcat es un servidor Web con soporte de servlets y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor Web Apache.

Tomcat puede funcionar como servidor Web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor Web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java. **[26]**

1.3.3 Lenguajes de Programación

Un **lenguaje de programación** es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Aunque muchas veces se usa lenguaje de programación y lenguaje informático como si fuesen sinónimos, no tiene por que ser así, ya que los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como, por ejemplo, el HTML (lenguaje para el marcado de páginas Web).

Un lenguaje de programación permite a uno o más programadores especificar de *manera precisa*: sobre que datos una computadora debe operar, como deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar *relativamente* próximo al lenguaje humano o natural, tal como sucede con el lenguaje Léxico. Una característica relevante de los lenguajes de programación es precisamente que más de un programador puedan tener un conjunto común de instrucciones que puedan ser comprendidas entre ellos para realizar la construcción del programa de forma colaborativa. **[9]**

1.3.3.1 PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas Web dinámicas. Es usado principalmente en interpretación del lado del servidor (Server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación

de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

PHP es un acrónimo recursivo que significa *PHP Hypertext Pre-processor* (inicialmente PHP Tools, o, *P*ersonal *H*ome *P*age Tools). Fue creado originalmente por Rasmus Lerdof en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

Visión general

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

Aunque todo en su diseño está orientado a facilitar la creación de página Web, es posible crear aplicaciones con una interfaz gráfica para el usuario, utilizando la extensión PHP-Qt o PHP-GTK. También puede ser usado desde la línea de comandos, de la misma manera como Perl o Python pueden hacerlo, a esta versión de PHP se la llama PHP CLI (*Command Line Interface*).

Cuando el cliente hace una petición al servidor para que le envíe una página Web, el servidor ejecuta el interprete de PHP. Este procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Mediante extensiones es también posible la generación de archivos PDF, Flash, así como imágenes en diferentes formatos.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX (y de ese tipo, como Linux o Mac OS X) y Windows, y puede interactuar con los servidores de Web más populares ya que existe en versión CGI, modulo para Apache, e ISAPI.

PHP es una alternativa a las tecnologías de Microsoft ASP y ASP.NET (que utiliza C#/VB.NET como lenguajes), a ColdFusion de la compañía Adobe (antes Macromedia), a JSP/Java de Sun Microsystems, y a CGI/Perl. Aunque su creación y desarrollo se da en el Ámbito de los sistemas libres, bajo la licencia GNU, existe además un IDE (entorno de desarrollo integrado) comercial llamado Zend Studio. Recientemente, CodeGear (la división de lenguajes de programación de Borland) ha sacado al mercado un entorno integrado de desarrollo para PHP, denominado Delphi for PHP, Existe un módulo para Eclipse uno de los IDE más populares.

Usos de PHP

Los principales usos del PHP son los siguientes:

- Programación de páginas Web dinámicas, habitualmente en combinación con el motor de base de datos MySQL, aunque cuenta con soporte nativo para otros motores, incluyendo el estándar ODBC, lo que amplía en gran medida sus posibilidades de conexión.
- Programación en consola, al estilo de Perl o Shell scripting.
- Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y Qt/GTK+, lo que permite desarrollar aplicaciones de escritorio en los sistemas operativos en los que está soportado. **[7]**

1.3.3.2 HTML

HTML, siglas de Hypertext Markup Language (*Lenguaje de Marcas de Hipertexto*), es el lenguaje de marcado predominante para la construcción de páginas Web. Es usado para describir la estructura y el contenido en forma de

texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un *script* (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores Web y otros procesadores de HTML.

HTML también es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores). Por convención, los archivos de formato HTML usan la extensión .htm o .html. **[8]**

1.3.3.3 JavaScript

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

Al igual que Java, JavaScript es un lenguaje orientado a objetos propiamente dicho, ya que dispone de Herencia, si bien esta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas Web. Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del DOM.

El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, que es la que desarrolló los primeros navegadores Web comerciales. Aparecía por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.

Tradicionalmente, se venia utilizando en páginas Web HTML, para realizar tareas y operaciones en el marco de la aplicación únicamente cliente, sin acceso a funciones del servidor. JavaScript se ejecuta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Los autores inicialmente lo llamaron Mocha y más tarde LiveScript pero fue rebautizado como JavaScript en un anuncio conjunto entre Sun Microsystems y Netscape, el 4 de diciembre de 1995. **[12]**

1.3.4 Sistemas Gestores de Bases de Datos

Los **sistemas de gestión de base de datos** (SGBD); (en ingles: *Database management system*, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Propósito

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de datos.

Objetivos

Existen distintos objetivos que deben cumplir los SGBD:

- Abstracción de la información.
- Independencia.
- Redundancia mínima
- Consistencia.
- Seguridad.
- Integridad.
- Respaldo y recuperación.

- Control de la concurrencia.
- Tiempo de respuesta.

Ventajas

1. Facilidad de manejo de grandes volúmenes de información.
2. Gran velocidad de ejecución de las consultas.
3. Independencia del tratamiento de información.
4. Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
5. No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
6. Integridad referencial el terminar los registros.

Inconvenientes

1. El costo de actualización del hardware y software son muy elevados.
2. El costo (salario o remuneración) del administrador de la base de datos es grande.
3. El mal diseño de esta puede originar problemas a futuro.
4. Un mal adiestramiento a los usuarios puede originar problemas a futuro.
5. Si no se encuentra un manual del sistema no se podrían hacer relaciones con facilidad.
6. Generan campos vacíos en exceso.
7. El mal diseño de seguridad genera problemas en esta. **[15]**

1.3.4.1 SQL

El Lenguaje de consulta estructurado (**SQL** Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar, de una forma sencilla, información de interés de una base de datos, así como también hacer cambios sobre la misma. Es un lenguaje de cuarta generación (4GL).

Orígenes y evolución

Los orígenes del SQL están ligados a los de las bases de datos relacionales. En 1970 E. F. Codd propone el modelo relacional y asociado a este un sublenguaje de acceso a los datos basado en el cálculo de predicados. Basándose en estas ideas, los laboratorios de IBM definen el lenguaje SEQUEL (Structured English Query Language) que más tarde sería ampliamente implementado por el SGBD experimental System R, desarrollado en 1977 también por IBM. Sin embargo, fue Oracle quien lo introdujo por primera vez en 1979 en un programa comercial.

El SEQUEL terminaría siendo el predecesor de SQL, siendo éste una versión evolucionada del primero. El SQL pasa a ser el lenguaje por excelencia de los diversos SGBD relacionales surgidos en los años siguientes y es por fin estandarizado en 1986 por el ANSI, dando lugar a la primera versión estándar de este lenguaje, el SQL-86 o SQL1. Al año siguiente este estándar es también adoptado por la ISO.

Características generales

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos.

Es un lenguaje declarativo de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizasen un lenguaje de bajo nivel orientado a regro. **[19]**

1.3.4.2 MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Al contrario que proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder

del autor individual, MySQL es propiedad y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

Aplicaciones

MySQL es muy utilizado en aplicaciones Web como MediaWiki, Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación Web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones Web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

Características

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas Web con contenido dinámico, justamente por su simplicidad.

Poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.

- Búsqueda e indexación de campos de texto. **[18]**

1.3.4.3 PostgreSQL

PostgreSQL es un servidor de base de datos relacional orientada a objetos de software libre, liberado bajo la licencia BSD.

Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*).

Características:

Alta concurrencia: Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en ingles) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. 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.

Amplia variedad de tipos nativos

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indizables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

Historia

PostgreSQL ha tenido una larga evolución, comenzando con el proyecto Ingres en la Universidad de Berkeley. Este proyecto, liderado por Michael Stonebraker, fue uno de los primeros intentos en implementar un motor de base de datos relacional. Después de haber trabajado un largo tiempo en *Ingres* y de haber tenido una experiencia comercial con el mismo, Michael decidió volver a la Universidad para trabajar en un nuevo proyecto sobre la experiencia de Ingres, dicho proyecto fue llamado post-ingres o simplemente POSTGRES.

El proyecto post-ingres pretendía resolver los problemas con el modelo de base de datos relacional que habían sido aclarados a comienzos de los años 1980. El principal de estos problemas era la incapacidad del modelo relacional de comprender "tipos", es decir, combinaciones de datos simples que conforman una única unidad. Actualmente estos son llamados objetos. Se esforzaron en introducir la menor cantidad posible de funcionalidades para completar el soporte de tipos. Estas funcionalidades incluían la habilidad de definir tipos, pero también la habilidad de describir relaciones - las cuales hasta ese momento eran ampliamente utilizadas pero mantenidas completamente por el usuario. En POSTGRES la base de datos «comprendía» las relaciones y podía obtener información de tablas relacionadas utilizando *reglas*. **[10]**

1.3.5 Metodologías para el desarrollo de Sistemas Informáticos

La rama de la **metodología**, dentro de la ingeniería de software, se encarga de elaborar estrategias de desarrollo de software que promuevan prácticas adoptativas en vez de predictivas; centradas en las personas o los equipos,

orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente. **[4]**

1.3.5.1 Metodología RUP

El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como **RUP**) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. También se conoce por este nombre al software desarrollado por Rational, hoy propiedad de IBM, el cual incluye información entrelazada de diversos artefactos y descripciones de las diversas actividades. Está incluido en el Rational Method Composer (RMC), que permite la personalización de acuerdo a necesidades. Originalmente se diseñó un proceso genérico y de dominio público, el Proceso Unificado, y una especificación más detallada, el Rational Unified Process, que se vendiera como producto independiente.

Principios de desarrollo en Metodología RUP

El RUP está basado en 5 principios clave que son:

Adaptar el proceso: El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.

Balancear prioridades: Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.

Demostrar valor iterativamente: Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

Elevar el nivel de abstracción: Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o esquemas (frameworks) por nombrar algunos. Esto previene a los ingenieros de software ir directamente de los requisitos a la codificación de software a la medida del cliente. Un nivel alto de abstracción también permite discusiones sobre diversos niveles arquitectónicos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.

Enfocarse en la calidad: El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

Ciclo de vida en Metodología RUP

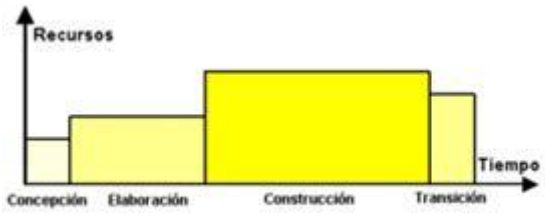

Fig. 1.1 Ciclo de vida de la Metodología RUP

Un típico perfil de proyecto mostrando el tamaño relativo de las cuatro fases. El ciclo de vida RUP es una implementación del Desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones. El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminar cada una de ellos,

estos a la vez se dividen en fases que finalizan con un hito donde se debe tomar una decisión importante:

- Concepción: se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos.
- Elaboración: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.
- Construcción: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.
- Transición: se Instala el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.
- Mantenimiento: una vez instalado el producto, el usuario realiza requerimientos de ajuste, esto se hace de acuerdo a solicitudes generadas como consecuencia del interactuar con el producto.

Características Metodología RUP

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software El RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso). **[5]**

1.3.5.2 Metodología XP

La programación extrema o extreme Programming (XP) es un enfoque de la ingeniería de software formulado por Kent Beck y De Jean, Extreme Programming Explained: Embrace Change (1999). Es la más destacada de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

Características

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación. Véase, por ejemplo, las herramientas de prueba JUnit orientada a Java, DUnit orientada a Delphi y NUnit para la plataforma.NET. Estas dos últimas inspiradas en JUnit.
- Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.
- Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.

- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo. La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Mientras más simple es el sistema, menos tendrá que comunicar sobre este, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores

¿Qué es lo que propone XP?

- Empieza en pequeño y añade funcionalidad con retroalimentación continua
- El manejo del cambio se convierte en parte sustantiva del proceso
- El costo del cambio no depende de la fase o etapa
- No introduce funcionalidades antes que sean necesarias
- El cliente o el usuario se convierte en miembro del equipo

Derechos del Cliente

- Decidir que se implementa
- Saber el estado real y el progreso del proyecto

- Añadir, cambiar o quitar requerimientos en cualquier momento
- Obtener lo máximo de cada semana de trabajo
- Obtener un sistema funcionando cada 3 o 4 meses

Derechos del Desarrollador

- Decidir como se implementan los procesos
- Crear el sistema con la mejor calidad posible
- Pedir al cliente en cualquier momento aclaraciones de los requerimientos
- Estimar el esfuerzo para implementar el sistema
- Cambiar los requerimientos en base a nuevos descubrimientos

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores
- La simplicidad, al desarrollar y codificar los módulos del sistema
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales. **[5]**

1.3.5.3 Metodología MSF

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF (Microsoft Solution Framework) se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Fig. 1.2 Metodología MSF

Características:

- ***Adaptable:*** es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- ***Escalable:*** puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- ***Flexible:*** es utilizada en el ambiente de desarrollo de cualquier cliente.
- ***Tecnología Agnóstica:*** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

- ***Modelo de Arquitectura del Proyecto:*** Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.
- ***Modelo de Equipo:*** Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles.
- ***Modelo de Proceso:*** Diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de equipo.
- ***Modelo de Gestión del Riesgo:*** Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.

- ***Modelo de Diseño del Proceso:*** Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.
- ***Modelo de Aplicación:*** Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores. **[4]**

1.4 Arquitecturas para el desarrollo e implementación de software

1.4.1 Arquitecturas en tres capas

Arquitectura en capas: es donde se define como organizar el modelo de diseño a través de capas, que pueden estar físicamente distribuidas, lo que quiere decir que los componentes de una capa solo puede hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la comprensión y la organización de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son consistentes de ningún detalle o interfaz de las superiores.

La programación por capas es un estilo de programación en el que el objetivo primordial es separar la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación del usuario.

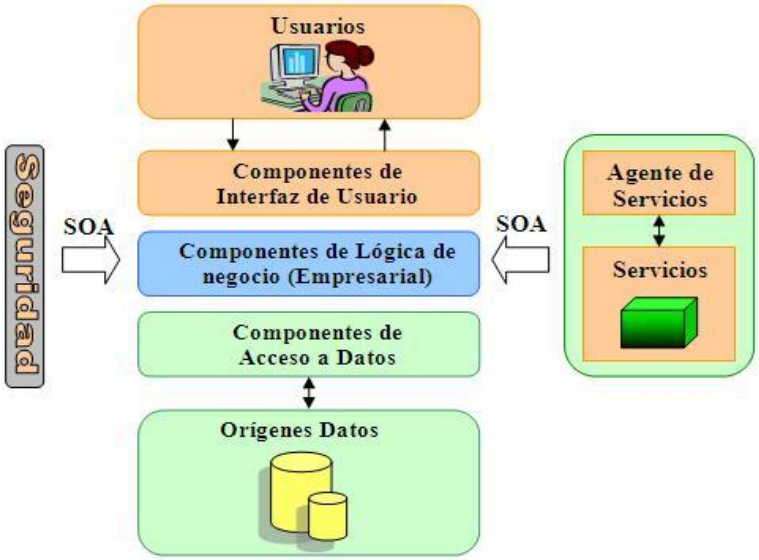

Fig. 1.3 Arquitectura en capas.

Capas o niveles

- ***Capa de presentación o interface:*** es la capa que le permite al usuario interactuar con el sistema, captura y le comunica la información al mismo, dando un mínimo de proceso (realiza un filtrado previo para asegurarse que no haya errores de formato). Esta capa se comunica únicamente con la de de negocio.
- ***Capa de negocio:*** es donde reside los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitarle al gestor de bases de datos almacenar o recuperar datos de él.
- ***Capa de datos:*** es donde residen los datos y es la encargada de acceder ellos. Está formada por uno o más gestores de bases de datos que se encargan de realizar el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un mismo ordenador aunque no es lo típico. Lo más usual es que haya una multitud de ordenadores donde reside la capa de interface (son los clientes de la arquitectura cliente/servidor). Las capas de negocios y de datos pueden residir en un mismo ordenador, y si el crecimiento de las necesidades lo aconseja, pueden dividirse en dos o más ordenadores. Así, si el tamaño o complejidad de las base de datos aumenta, pueden dividirse en varios ordenadores los cuales recibirán las peticiones del ordenador donde resida la capa de negocio. Si por el contrario, la complejidad fuese en la capa de negocio lo que obligase a la separación, esta lógica del negocio podría residir en uno o más ordenadores que realizarían las solicitudes a una única base de datos. **[1]**

1.4.2 Patrón Arquitectónico Modelo Vistas Controladores (MVC)

Es un patrón de diseño de arquitectura de software que separa los datos de una aplicación la interfaz de usuario**,** y la lógica de control en tres componentes distintos.

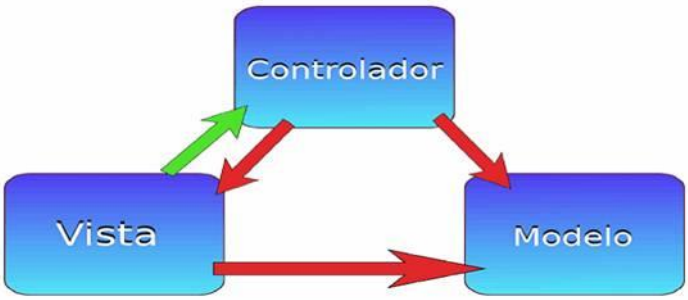

Fig. 1.4 MVC

- **Modelo:** Representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.

- **Vista:** Presenta el modelo en un formato adecuado, como en una página Web que le permite al usuario interactuar con ella, usualmente un elemento de interfaz de usuario.
- **Controlador:** Responde a eventos, usualmente acciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura **MVC** separa la lógica de negocio (el modelo) y la presentación (la vista) logrando un mantenimiento más rápido y sencillo de las aplicaciones. Ejemplo, para el caso de la Web, si se fuera a mostrar una misma aplicación en un navegador estándar, como en un navegador de un dispositivo móvil, sólo es necesario crear una vista nueva por cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (Aplicación de escritorio, HTTP, consola de comandos, e-mail, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. **[1]**

1.5 Propuesta de solución para el sistema a desarrollar

Desarrollar una aplicación Web que permita la Planificación de las guardias de los estudiantes de una manera más factible, permitiéndonos en su organización realizar cambios de acuerdo a situaciones que presenten los mismos, logrando en su conjunto un sistema capaz de manipular más eficientemente el servicio que se desarrolla de forma manual. Al este sistema implementarse mediante una aplicación Web nos permitirá interactuar con el mismo desde cualquier maquina accediendo a el mediante la red (utilizando un navegador), facilitando así su uso.

1.5.1 Dreamweaver como entorno de desarrollo

Dreamweaver, sin lugar a dudas, es unas de las herramientas más utilizadas por los webmasters para el trabajo con aplicaciones visuales en este caso en el diseño e implementación de páginas Web. Se adapta increíblemente a las necesidades de todo tipo de profesional de diseño Web, tanto para lo que prefieren programar el código directamente en el editor de texto como para los que gustan del ambiente visual. Se trata de un editor de texto especialmente diseñado para trabajar con documentos Web como HTML, PHP, ASP, JavaScript, entre otros.

1.5.2 Fundamentación del lenguaje de programación a utilizar

PHP (Personal Home Page) es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo Web y puede ser embebido dentro de código HTML. Generalmente se ejecuta en un servidor Web, tomando el código en PHP como su entrada y creando páginas Web como salida. Puede ser desplegado en la mayoría de los servidores Web y en casi todos los sistemas operativos y plataformas sin costo alguno

Ventajas de utilizar PHP

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial (*http://www.php.net/manual/es/),* 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.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones. **[7]**

1.5.3 Fundamentación del SGBD a utilizar

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) que ha sido desarrollado de varias formas desde la década de 1980.

- El proyecto PostgreSQL sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto.
- PostgreSQL es ampliamente considerado como una de las alternativas de sistema de bases de datos de código abierto.

Ventajas:

- ***Inhalación ilimitada:*** Con PostgreSQL, nadie puede demandarlo por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.
- ***Soporte:*** Además de nuestras ofertas de soporte, tenemos una importante comunidad de profesionales y entusiastas de PostgreSQL de los que su compañía puede obtener beneficios y contribuir.
- ***Ahorros considerables en costos de operación:*** PostgreSQL ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que otros productos, conservando todas las características, estabilidad y rendimiento.
- ***Estabilidad y Confiabilidad Legendarias:*** PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Ni una sola vez. Simplemente funciona.

- ***Extensible:*** El código fuente está disponible para todos sin costo. Si su equipo necesita extender o personalizar PostgreSQL de alguna manera, pueden hacerlo con un mínimo esfuerzo, sin costos adicionales. Esto es complementado por la comunidad de profesionales y entusiastas de PostgreSQL alrededor del mundo que también extienden PostgreSQL todos los días.
- ***Multiplataforma:*** PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y ahora en versión nativa para Windows.
- ***Diseñado para ambientes de alto volumen:*** PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mejor respuesta en ambientes de grandes volúmenes.
- ***Herramientas gráficas de diseño y administración de BD:*** Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (Tora, Data Architect).
- Permite desarrollar o migrar aplicaciones desde Access, Visual Basic, Foxpro, Visual Foxpro, C/C++ Visual C/C++, Delphi, etc., para que utilicen a PostgreSQL como servidor de BD. **[10]**

1.5.4 Fundamentación de la metodología a utilizar

¿Por qué elegir XP?

Actualmente XP es la metodología ágil más documentada (hay una colección de libros “XP Series” de Addison Wesley) y extendido. Existe una gran comunidad de desarrolladores XP.

Otra de las ventajas de XP es que no es necesario adoptarlo en forma completa, sino que pueden utilizarse varias de sus prácticas en forma independiente. Esto hace que el costo de su implementación sea mucho más accesible que el de otras metodologías. Un estudio a la bibliografía (Welicki, 2003), muestra las ventajas que tiene XP y que exponemos en los puntos siguientes:

- Puede ser implementado en forma parcial (elegir sólo algunas de las prácticas)
- Puede ser implementado en forma gradual.
- Puede adaptarse a las necesidades de cualquier equipo de desarrollo. De hecho, Kent Beck recomienda a los equipos que lo adapten a sus necesidades (BECK, 1999).
- Exige que se establezca una comunicación más fluida con el cliente y que este tenga mayor participación en el proceso de desarrollo. La consecuencia de esto es que el cliente se involucre más en el desarrollo del producto.
- Actualmente es la metodología ágil más extendida y documentada.
- Se realizan pruebas constantemente del sistema. **[6]**

1.5.5 Fundamentación de la arquitectura a utilizar

Arquitectura 3 Capas. Es un estilo de programación, su objetivo primordial es la separación de la capa de presentación, capa de negocio y la capa de datos.

Ventajas:

- El estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual, permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- El estilo admite muy naturalmente optimizaciones y refinamientos.
- Proporciona una amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa a medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales pueden construirse extensiones o presentaciones específicas. **[1]**

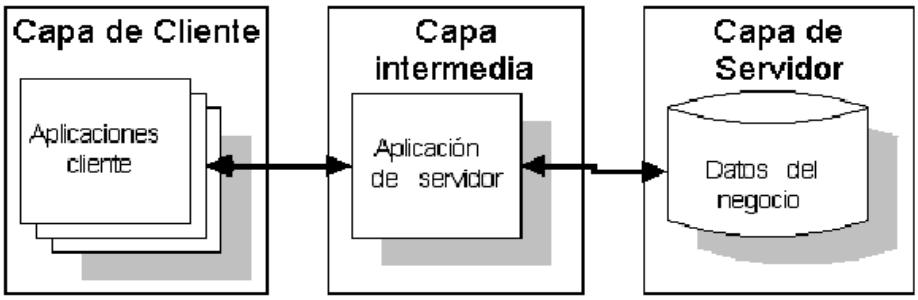

Fig. 1.5 Arquitectura Cliente/Servidor en tres capas.

1.6 Conclusiones

En este capitulo se dio continuidad a la aplicación que se pretende crear especificando el entorno en el cual se creará el sistema. Se describieron de una manera simple, las herramientas con que se desarrollará, abordándose conceptos, ventajas y desventajas entre otras características, pudiéndose así conformar la base para la realización del presente trabajo de diploma.

Capítulo 2 Planeación y Diseño

2.1 Introducción

El reciente capitulo tiene como objetivo fundamentar en la planeación del sistema a desarrollar, relatándose los parámetros substanciales que constituyen la base fundamental de la aplicación. Se presentan las historias de usuarios siendo estas las necesidades especificadas por el cliente, además de realizarse una estimación del esfuerzo para las mismas y de establecerse un plan de iteraciones sobre el sistema, necesarios para su terminación.

2.2 Funcionalidad principal

El proceso de planificar guardias en el instituto, es realizado a mano, el cual en la elaboración del mismo se utilizan recursos (humanos, materiales) que se podrían emplear en otras funciones. Este proceso podría llegar a ser tedioso y por consiguiente en su implementación se cometerían errores, de ser así se realizaría una mala planificación y no cumpliría su objetivo.

Según lo antes expuesto el software tiene como funcionalidad principal la de *planificar las guardias de los estudiantes, por departamento*.

Planificar las guardias de los estudiantes, por departamento

Consiste en planear y asignar de forma adecuada a los estudiantes que pertenecen a un determinado departamento, los turnos de guardias que deben de realizar como forma de proteger los recursos que se encuentran en el instituto.

Personal que trabajará con el sistema

Personal que trabajara con el sistema	Justificación
Administrador	Es el encargado de gestionar la información de los usuarios que accederán a la aplicación.
Planificador	Es el encargado de realizar la planificación correspondiente a su departamento.
Visitante	Podrá observar varias informaciones que la aplicación será capaz de brindar.

Tabla 2.1 Personal que trabajará con el sistema

2.3 Lista de reserva

- Autenticar usuarios.
- Configurar la base de datos del sistema.
- Configurar el acceso a datos del sistema.
- Cambiar contraseña de la configuración del sistema.
- Cambiar contraseña.
- Adicionar usuarios.
- Modificar usuarios.
- Eliminar usuarios.
- Mostrar usuarios.
- Realizar la planificación del departamento.
- Realizar cambios en la planificación de hora.
- Realizar cambios en la planificación de fecha.
- Justificar estudiantes incapacitados.
- Mostrar planificación generada por departamentos.
- Mostrar planificación generada por estudiantes
- Mostrar estudiantes que no realizarán guardia.

- Generar resumen de la planificación.
- Exportar resumen de la planificación.

2.4 Historias de usuario

Las historias de usuario tienen el mismo propósito que los casos de uso. Las escriben los propios clientes, tal y como ven ellos las necesidades del sistema. Las historias de usuario son similares al empleo de escenarios, con la excepción de que no se limitan a la descripción de la interfaz de usuario.

También conducirán el proceso de creación de los test de aceptación (empleados para verificar que las historias de usuario han sido implementadas correctamente). Existen diferencias entre estas y la tradicional especificación de requisitos. La principal diferencia es el nivel de detalle. Las historias de usuario solamente proporcionarán los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de dicha historia de usuario.

<table>
<tr><td colspan="2">Historia de usuario</td></tr>
<tr><td>Número:
(No. Historia de usuario)</td><td>Usuario: (Usuario entrevistado para obtener la función requerida a automatizar)</td></tr>
<tr><td colspan="2">Nombre: (nombre de la historia de usuario que sirve para identificarla mejor entre los desarrolladores y el cliente)</td></tr>
<tr><td>Prioridad en el negocio:
(Importancia de la historia para el cliente)
(Alta / Media / Baja)</td><td>Riesgo en desarrollo:(Dificultad para el programador)
(Alta / Media / Baja)</td></tr>
<tr><td>Puntos estimados:
(Estimación: de 1 a 3 puntos)</td><td>Iteración asignada:
(Iteración a la que corresponde)</td></tr>
<tr><td colspan="2">Programador responsable: Nombre de encargado de programación.</td></tr>
<tr><td colspan="2">Descripción: (Se especifican las operaciones por parte del usuario y las respuestas del sistema)</td></tr>
</table>

Observaciones: (Algunas observaciones de interés, como glosario, información sobre usuario etc.)

Tabla 2.2 Planilla de historia de usuario

HU No.1: Autenticar usuarios.

Historia de usuario	
Número: 1	**Usuarios:** Administradores y planificadores.
Nombre: Validar modo.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Reidelbis Calderón Suárez	
Descripción: El usuario debe de ingresar sus datos para poder acceder al sistema (nombre de usuario y contraseña). El sistema confirma los datos y validará el modo en que se utilizará el software.	
Observaciones: El usuario debe de existir en la base de datos.	

Tabla 2.3 HU No.1: Autenticar usuarios

Para consultar el resto de las HU ir: Anexo1

2.5 Planificación de entregas

Para la planificación de entregas se establece la prioridad de cada historia de usuario así como una estimación del esfuerzo necesario de cada una de ellas con el fin de determinar un cronograma de entregas. **[6]**

Las estimaciones asociadas a la implementación de las historias se establecen empleando como medida el punto de estimación. Un punto de estimación equivale a una semana ideal de programación, donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de

interrupción, este punto de estimación que se utiliza para representar la semana ideal, es de 5 días. Las historias generalmente tienen un valor de 1 a 3 puntos. Además, se mantiene un registro de la velocidad de desarrollo, establecida por puntos de iteración, basado fundamentalmente en la suma de puntos de estimación correspondientes a las HU, que fueron terminadas en la última iteración.

2.5.1 Estimación de esfuerzo por historia de usuario

Para una buena realización del sistema a desarrollar, se realizó una estimación para cada una de las HU identificadas. A continuación se muestran los resultados:

Historia de usuario	Numero	Puntos de estimación (Semanas)
Autenticar usuarios	1	1
Configurar bases de datos	2	1
Cambiar contraseña	3	1
Gestionar Usuarios	4	1
Realizar la planificación del departamento	5	2
Realizar cambios en la planificación	6	1
Justificar estudiantes incapacitados	7	1
Mostrar planificación generada	8	1
Mostrar estudiantes que no realizarán guardia	9	1
Generar el resumen de la planificación.	10	1
Exportar el resumen de la planificación.	11	1

Tabla 2.14 Estimación de esfuerzos por historias de usuario

2.5.2 Planificación de iteraciones

Después de exponer las HU y la estimación del esfuerzo propuesto para el cumplimiento de los mismos, se procederá a la realización de la planificación de la etapa de implementación del sistema. Este plan establece las iteraciones que serán necesarias realizar sobre el sistema, para su terminación.

El plan de iteraciones puede contener indicaciones sobre cuáles HU se incluirán en un reléase, lo cual debería ser consistente con el contenido de una o dos iteraciones. En relación con lo antes tratado se decide realizar el sistema en 3 iteraciones, las cuales se explican de forma detalla a continuación:

Primera iteración:

En esta iteración se le dará cumplimiento a las HU que se consideraron de mayor importancia como tal en el acceso a la aplicación. Ella incluirá con todas las funcionalidades descritas en las historias de usuarios 1 ,2 ,3 y 4, las cuales hacen alusión a Autenticar usuarios, Configurar la base de datos del sistema, Cambiar contraseña y Gestionar usuarios.

Segunda iteración:

Esta iteración tiene como objetivo desarrollar las historias de usuario que complementarán la funcionalidad primordial de la aplicación. Estas son la 5, 6 y 7 que hacen referencia a las funcionalidades de Realizar la planificación, Realizar cambios en la planificación y Justificar estudiantes incapacitados.

Tercera iteración:

En esta última iteración se pretende desarrollar las historias de usuario que tienen como objetivo dar a conocer como quedo establecida la planificación. Funcionalidades como Mostrar planificación generada, Mostrar estudiantes que no realizarán guardia, Generar el resumen de la planificación y Exportar el resumen de la planificación complementarán el mismo, las cuales corresponden a las HU 8, 9,10 y 11.

2.5.3 Plan de duración de las iteraciones

Parte del ciclo de vida de los proyectos guiados con la metodología de desarrollo de software XP es crear el plan de duración de cada una de las iteraciones. En el mismo se expondrán la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de las mismas.

Iteración	Historias de usuario	Duración total
Iteración 1	Autenticar usuarios	4 semanas La entrega se realizará en la primera semana de marzo
	Configurar bases de datos	
	Cambiar contraseña	
	Gestionar Usuarios	
Iteración 2	Realizar la planificación del departamento	4 semanas La entrega se realizará en la primera semana de abril
	Realizar cambios en la planificación	
	Justificar estudiantes incapacitados	
Iteración 3	Mostrar planificación generada	4 semanas La entrega se realizará en la primera semana de mayo
	Mostrar estudiantes que no realizarán guardia	
	Generar el resumen de la planificación.	
	Exportar el resumen de la planificación.	

Tabla 2.15 Plan de duración de las iteraciones

2.6 Tarjetas CRC

Las tarjetas CRC son una metodología para el diseño de software orientado por objetos creada por Kent Beck y Ward Cunningham. Como una extensión informal a UML, la técnica de las tarjetas CRC se puede usar para guiar el sistema a través de análisis guiados por la responsabilidad. Las clases se examinan, se filtran y se refinan basándose en sus responsabilidades con respecto al sistema, y las clases con las que necesitan colaborar para completar sus responsabilidades. **[6]**

Modelo de las tarjetas CRC:

Nombre de la clase	
Tipo de la clase	
Responsabilidades	**Colaboraciones**

Tabla 2.16 Modelo de las tarjetas CRC.

Para el proyecto que se analiza, las tarjetas CRC que se propusieron fueron las siguientes:

Tarjeta CRC No 1: Autenticar usuarios

Nombre de la clase: Autenticar_usuarios	
Tipo de la clase: Lógica del negocio	
Responsabilidades	**Colaboraciones**
Autenticar usuarios	Autenticar_usuarios

Tabla 2.17 Tarjeta CRC No 1: Autenticar usuarios.

Para consultar el resto de las tarjetas CRC ir: Anexo 2

2.7 Conclusiones

En este capítulo se le dio continuidad al desarrollo de la aplicación abordándose en si la fase de planeación y diseño. Se escribieron los relatos de usuario ("user stories") siendo estas tal y como ven los clientes las necesidades del sistema, se realizó la planificación de entrega y se llevó a cabo la planificación de iteraciones de cada HU, llegándose así a la culminación de esta fase y dando paso al inicio de otra.

Capítulo 3 Desarrollo y Pruebas

3.1 Introducción

Al iniciarse este capitulo se le da continuación a la metodología que se está utilizando para la realización de este proyecto, dándose inicio a la fase de desarrollo y prueba. En la misma se presenta el modelo de datos correspondiente y se realiza el desarrollo de las iteraciones a partir del desglose de las historias de usuario en tareas. Se describen además cada una de las pruebas funcionales realizadas durante el desarrollo de la aplicación, detallándose los posibles modos de utilización de la misma en cada una de estas historias de usuario, así como los posibles mensajes de error, información o aceptación que se emite cuando se utilizan las funcionalidades.

3.2 Modelo de datos

Es el modelo físico de la Base de Datos, nos sirve para entender con más facilidad como está diseñada nuestra base de datos y las relaciones de las mismas, así como el tipo de relación que tienen estas, se obtuvieron un total de 3 tablas normalizadas hasta tercera forma normal.

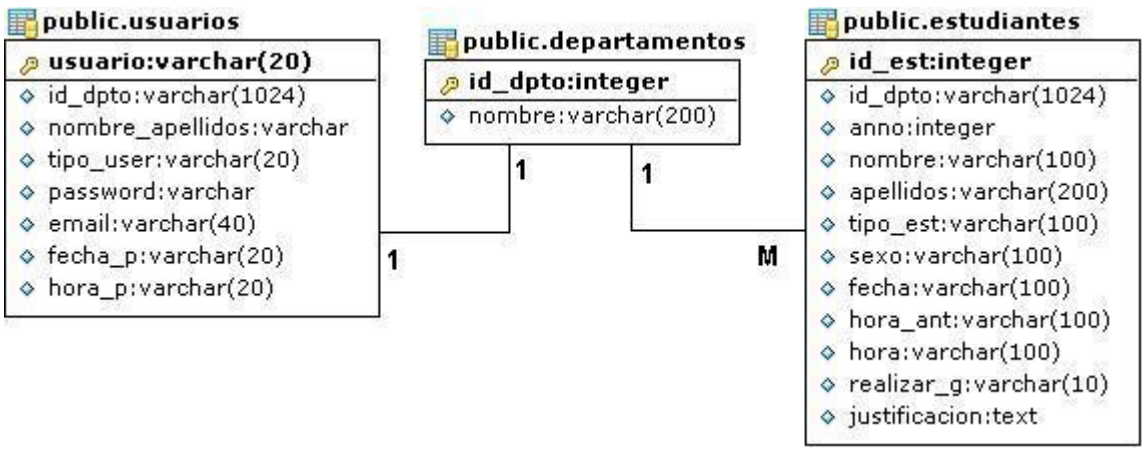

Fig. 3.1 Modelo de datos

3.3 Desarrollo de las iteraciones

En la fase anterior de la metodología se describieron las historias de usuarios las cuales corresponden a cada iteración a desarrollar, teniéndose en cuenta las prioridades y restricciones de tiempo, previstas por el cliente.

3.3.1 Tareas por historias de usuario

A continuación se detallan las tareas correspondientes por cada historia de usuario.

Historia de usuario	**Tareas**
Autenticar usuarios.	Autenticar usuarios.
Configurar bases de datos.	Configurar la base de datos del sistema. Configurar el acceso a datos del sistema. Cambiar contraseña de la configuración del sistema.
Cambiar contraseña.	Cambiar contraseña.
Gestionar usuarios.	Insertar usuarios. Modificar usuarios Eliminar usuarios. Mostrar listado de usuarios.
Realizar la planificación del departamento.	Realizar la planificación.
Realizar cambios en la planificación.	Realizar cambios en la planificación de hora. Realizar cambios en la planificación de fecha.
Justificar estudiantes incapacitados.	Justificar estudiantes incapacitados
Mostrar planificación generada.	Mostrar planificación por departamentos.

	Mostrar planificación por estudiantes.
Mostrar estudiantes que no realizarán guardia.	Mostrar estudiantes que no realizarán guardia.
Generar el resumen de la planificación.	Generar el resumen de la planificación.
Exportar el resumen de la planificación.	Exportar el resumen de la planificación.

Tabla 3.1 Tareas por historias de usuarios

3.3.2 Iteración No.1

En esta iteración se le dio cumplimiento a la implementación de las Historias de usuarios que se consideraron de mayor importancia como tal para lograr el acceso a la aplicación.

Historias de usuario abordadas en la primera iteración.

Historia de usuario	Tiempo de estimación (semanas)	
	Estimación inicial	Real
Autenticar usuarios.	1	0.4
Configurar bases de datos.	1	1
Cambiar contraseña.	1	0.2
Gestionar Usuarios.	1	1

Tabla 3.2 Historias de usuario abordadas en la primera iteración

Después de mostradas las historias de usuarios correspondientes a la primera iteración se detallarán las tarjetas de tarea propias de ellas.

Tarjeta de tarea No.1: Autenticar usuarios

	Tarea ingeniería
Número tarea: 1	**Número historia:** 1
Nombre tarea: Autenticar usuarios.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 1/2/2011	**Fecha fin:** 2/2/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea posibilita el acceso de los usuarios al sistema mediante un nombre de usuario y una contraseña.	

Tabla 3.3 Tarjeta de tarea No.1 Autenticar usuarios

Para ver las otras Tarjetas de tareas de la primera Iteración ir al: Anexo 3

3.3.3 Iteración No.2

Esta iteración tuvo como objetivo desarrollar las historias de usuario que complementaron la funcionalidad primordial de la aplicación

Historias de usuario abordadas en la segunda iteración.

Historia de usuario	Tiempo de estimación (semanas)	
	Estimación inicial	Real
Realizar la planificación del dpto.	2	2.4
Realizar cambios en la planificación	1	0.8
Justificar estudiantes incapacitados	1	0.2

Tabla 3.12 Historias de usuario abordadas en la segunda iteración.

Para ver las Tarjetas de tareas de la segunda Iteración ir al: Anexo 3

3.3.4 Iteración No.3

En esta última iteración se desarrollaron las historias de usuario que tuvieron como objetivo dar a conocer como quedo establecida la planificación, entre otras funcionalidades de índole utilitarias.

Historias de usuario abordadas en la tercera iteración.

Historia de usuario	Tiempo de estimación (semanas)	
	Estimación inicial	Real
Mostrar planificación generada	1	0.4
Mostrar estudiantes que no realizarán guardia.	1	0.2
Generar el resumen de la planificación.	1	0.4
Exportar el resumen de la planificación.	1	0.6

Tabla 3.17 Historias de usuario abordadas en la tercera iteración.

Para ver las Tarjetas de tareas de la tercera Iteración ir al: Anexo 3

3.4 Principales interfaces de la aplicación

A continuación se presentan las principales interfaces de la aplicación llevadas a cabo en la fase de Implementación de la metodología expuesta en el capítulo inicial.

- Interfaz principal del sistema.
- Interfaz de la planificación de la guardia.
- Interfaz de la gestión de usuarios.
- Interfaz de la realización de cambios, de fecha.
- Interfaz del resumen y exportación de la planificación.

3.5 Pruebas

Las pruebas en la metodología XP juegan un papel fundamental pues permiten verificar si el software funciona correctamente, las mismas deben desarrollarse a medida que se va creando la aplicación, esperar a realizar el desarrollo del software para luego realizar la prueba se constituye en una práctica de alto riesgo. Esto se debe a que las fallas encontradas al realizar el desarrollo del software pueden ser muchas y difíciles de remover, ya que se está probado todo el sistema y es difícil saber dónde se encuentra el defecto.

3.5.1 Desarrollo dirigido por pruebas

El principio de pruebas: se tiene que establecer un período de pruebas de aceptación del programa (llamado también período de caja negra) donde se definirán las entradas al sistema y los resultados esperados de estas entradas. Es muy recomendable automatizar estas pruebas para poder hacer varias simulaciones del sistema en funcionamiento. Para hacer estas simulaciones automatizadas, se pueden utilizar Ambientes de Prueba (Unit testing frameworks). Un buen ejemplo de un ambiente de prueba es el JUnit para Java (www.junit.org/index.htm). Otros ambientes de pruebas para otros lenguajes como C, C++, JavaScript, XML y servicios Web. **[6]**

3.5.2 Pruebas de aceptación

Las pruebas de aceptación se crean a partir de los relatos de usuario. El cliente define los escenarios de prueba para verificar si el relato de usuario ha sido correctamente implementado. Un relato de usuario puede tener una o varias pruebas de aceptación. El cliente es responsable de verificar el pasaje de las pruebas de aceptación y priorizar la corrección de las pruebas fallidas. **[6]**

Las pruebas de aceptación son pruebas tipo caja negra a nivel del sistema: cada prueba de aceptación corresponde a un resultado producido por el

sistema, deben ser automáticas, correrse frecuentemente, publicarse sus resultados y programarse su corrección para la próxima iteración.

Para realizar las pruebas de aceptación el cliente utiliza la siguiente planilla.

Planilla de prueba de aceptación

Prueba de aceptación
HU: Nombre de la Historia de usuario que va a comprobar su funcionamiento.
Nombre: Nombre del caso de prueba.
Descripción: Descripción del propósito de la prueba.
Condiciones de ejecución: Precondiciones para que la prueba se realice.
Entrada/Pasos ejecución: Pasos para probar la funcionalidad.
Resultado: Resultado que se desea de la prueba.
Evaluación de la prueba: (Aceptada o denegada) o (Satisfactoria o no satisfactoria)

Tabla 3.23 Plantilla de prueba de aceptación.

Pruebas de aceptación para la HU: Autenticar usuarios.

Prueba de aceptación
HU: Autenticar usuarios.
Nombre: Prueba para comprobar la Autenticación de usuarios. (Nombre de usuario y contraseña).
Descripción: Validación de entrada de los datos de los usuarios.
Condiciones de ejecución: El usuario debe introducir su nombre de usuario y contraseña.
Entrada/Pasos ejecución: El usuario escribe su nombre de usuario y contraseña y luego da clic en el botón “Iniciar”.
Resultado: • Si el usuario tiene acceso para entrar a la aplicación e inserta sus datos correctamente accederá sin problemas al sistema. Se emitirá un mensaje de error en caso de que: • Se inserte los datos de un usuario no válido para el sistema o estén incorrectos ya sea su nombre de usuario o su contraseña.

• Se dé clic en el botón “Iniciar” sin insertar nada en los campos de texto. • Si no se logra realizar una conexión a la base de datos especificada.
Evaluación de la prueba: Aceptada.

Tabla 3.24 Prueba para comprobar la Autenticación de usuarios.

Para ver las otras pruebas de aceptación ir al: Anexo 4.

3.6 Conclusiones

En el capitulo que concluye se desarrolló la fase de Desarrollo y Pruebas en la cual se presentó el modelo de datos del sistema a obtener. Se detalló las tarjetas de tareas correspondientes lográndose una mejor visión del resultado final y del tiempo a utilizar, concluyendo con una serie de pruebas las cuales dieron ciertas respuestas sobre el resultado a esperar a primera instancia de nuestro sistema.

Capítulo 4 Estudio de Factibilidad

4.1 Introducción

El análisis de la factibilidad en un software determina la realización o no del mismo, ejerciendo esta una gran influencia en el desarrollo de un producto. Para analizar esta problemática haremos uso de la Metodología Costo Efectividad (Beneficio) la cual plantea que la conveniencia de la ejecución de un proyecto se determina por la observación conjunta de dos factores:

- El costo, que involucra la implementación de la solución informática, adquisición y puesta en marcha del sistema hardware/software y los costos de operación asociados
- La efectividad, que se entiende como la capacidad del proyecto para satisfacer la necesidad, solucionar el problema o lograr el objetivo para el cual se ideó, es decir, un proyecto será más o menos efectivo con relación al mayor o menor cumplimiento que alcance en la finalidad para la cual fue ideado (costo por unidad de cumplimiento del objetivo). **[16]**

4.2 Efectos económicos

- Efectos directos.
- Efectos indirectos
- Efectos externos
- Intangibles

4.2.1 Efectos directos

✓ **Positivos**

- Se mejora la eficiencia y calidad del proceso de planificar guardias.
- Se gestiona la información necesaria a la que los usuarios finales del sitio podrán acceder.
- Los estudiantes tendrán un sitio al cual acceder para observar cuando le corresponde su turno de guardia, sin necesidad de esperar a que se les informe.

✓ **Negativos**

- Para usar la aplicación es imprescindible el uso de un ordenador, aparejado a los gastos que este trae de consumo de corriente eléctrica y mantenimiento.

4.2.2 Efectos indirectos

- Los efectos económicos observados que pudiera repercutir sobre otros mercados no son perceptibles, aunque este proyecto no está construido con la finalidad de comercializarse.

4.2.3 Efectos externos

- Se contara con una herramienta disponible que permitirá a los departamentos realizar la planificación de los turnos de guardia de sus estudiantes, ahorrándose tiempo y otros recursos.

4.2.4 Intangibles

- En la valoración económica siempre hay elementos perceptibles por una comunidad como perjuicio o beneficio, pero al momento de ponderar en unidades monetarias esto resulta difícil o prácticamente imposible.

A fin de medir con precisión los efectos, deberán considerarse dos situaciones:

- **Situación sin proyecto**

Para llevar a cabo la planificación de la guardia de los estudiantes los responsables deben tener en cuenta los siguientes pasos:

1. Saber que estudiantes están disponibles para la realización de la misma.
2. Conocer el turno de guardia que le correspondió anteriormente a cada estudiante, para una mejor y correcta planificación.
3. Hacerle llegar esta información a los estudiantes para que estos procedan con la realización de la misma.

- **Situación con proyecto**

Para la entrada de los datos al sistema propuesto debemos seguir los siguientes pasos:

1. Entrar los datos correspondientes del usuario que hará uso de la aplicación.
2. Proveer al sistema con los datos de la planificación correspondiente.
3. Mostrará como ha quedado expuesta la planificación.
4. Obtener de forma inmediata un reporte en formato de impresión.

4.3 Beneficios y Costos Intangibles en el proyecto

COSTOS:

- ✓ Resistencia al cambio.

BENEFICIOS:

- ✓ Mayor comodidad para los usuarios.
- ✓ Mejora en la calidad de la información.
- ✓ Menor tiempo empleado en la introducción de los datos.
- ✓ Obtención del resultado en un menor tiempo.
- ✓ Facilidad a la hora de buscar la información.

4.4 Ficha de costo

Para determinar el costo económico del proyecto se utilizará el procedimiento para elaborar una Ficha de Costo de un Producto Informático [Dra. Ana María Gracia Pérez, UCLV]. Para la elaboración de la ficha se consideran los siguientes elementos de costo, desglosados en moneda libremente convertible y moneda nacional.

4.4.1 Costos en Moneda Libremente Convertible

- **Costos Directos.**
  1. Compra de equipos de cómputo: No procede.
  2. Alquiler de equipos de cómputo: No procede.
  3. Compra de licencia de Software: No procede.
  4. Depreciación de equipos: $ 25.00
  5. Materiales directos: No procede.

Total: $ 25.00

- **Costos Indirectos.**
  1. Formación del personal que elabora el proyecto: No procede.
  2. Gastos en llamadas telefónicas: No procede.
  3. Gastos para el mantenimiento del centro: No procede.
  4. Know How: No procede.
  5. Gastos en representación: No procede

Total: $0.00

- **Gastos de distribución y venta.**
  1. Participación en ferias o exposiciones: No procede.
  2. Gastos en transportación: No procede.
  3. Compra de materiales de propagandas: No procede.

Total: $25.00

4.4.2 Costos en Moneda Nacional

- **Costos Directos.**
  1. Salario del personal que laborará en el proyecto: $110.00.
  2. El 12.5% del total de gastos por salarios se dedica a la seguridad social: No procede.
  3. El 9.09% de salario total, por concepto de vacaciones a acumular: No procede.
  4. Gasto por consumo de energía eléctrica: $ 88,80.
  5. Gastos en llamadas telefónicas: No procede.
  6. Gastos administrativos: No procede.

- **Costos Indirectos.**
  1. Know How: $ 198,8

Total: $ 198,8

Como se hizo referencia anteriormente, la técnica seleccionada para evaluar la factibilidad del proyecto es la Metodología Costo-Efectividad. Dentro de esta metodología la técnica de punto de equilibrio aplicable a proyectos donde los beneficios tangibles no son evidentes, el análisis se basa exclusivamente en los costos. Para esta técnica es imprescindible definir una variable discreta que haga variar los costos. Teniendo en cuenta que el costo para este proyecto es despreciable, tomaremos como costo el tiempo en horas empleado para

resolver los reportes de la verías y el posible mantenimiento y la variable sería la complejidad de las pruebas que se realizan durante este proceso.

Valores de la variable (Solución manual).

1. Realizar la planificación de la guardia, 5 variables (80 min.).
2. Realizar cambios en la planificación según las necesidades de algunos estudiantes, 2 variables (20 min.).

Valores de la variable (Solución con el programa).

1. Cargar formulario con los datos necesarios para realizar la planificación, 5 variables (4 min.).
2. Cargar formulario con los datos necesarios para realizar cambios en la planificación, 2 variables (5 min.).

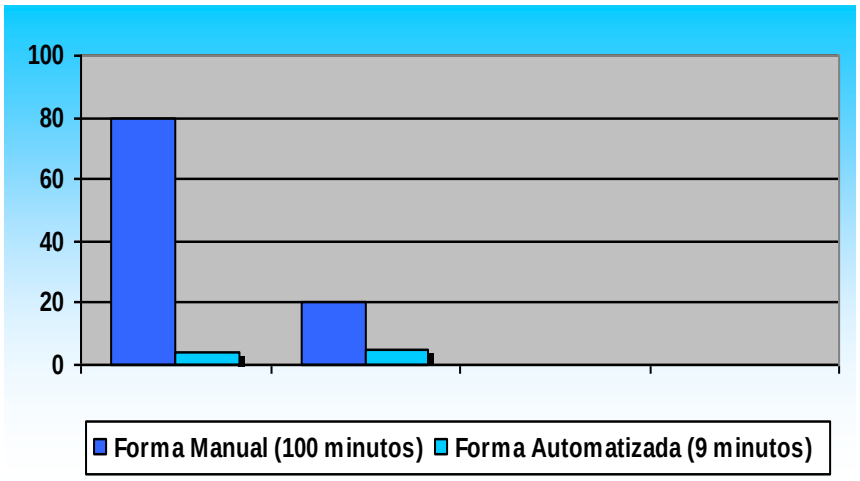

Fig. 4.1 Gráfica de la solución manual y solución automatizada

Al analizar los resultados representados en la gráfica, se ha podido demostrar la factibilidad de la aplicación, el cual se evidencia en el tiempo de darle solución al problema.

4.5 Conclusiones

En este capítulo se realizó el estudio de factibilidad mediante La Metodología Costo Efectividad (Beneficio), analizándose los efectos económicos, los beneficios y costos intangibles, además se calculó el costo de ejecución del proyecto mediante la ficha de costo arrojando como resultado $ 25,00 CUC. $198.8 MN demostrándose la factibilidad del proyecto.

Conclusiones Generales

Al finalizar el trabajo antes expuesto hemos llegado a las siguientes conclusiones:

1. Se desarrolló una aplicación capaz de planificar las guardias de los estudiantes.
2. Se llevó a cabo un estudio de las principales metodologías, lenguajes y herramientas que se consideraron factibles para el desarrollo del sistema, escogiéndose las más viables.
3. Se realizaron todas las fases correspondientes a la metodología implementada en el proceso, identificándose en su desarrollo, entre otros datos, los requisitos funcionales, las historias de usuarios, el modelo de datos, las tarjetas CRC y las tareas del ingeniero, que hicieron posible la culminación del software.
4. Se realizaron las pruebas de aceptación a cierta cantidad de requisitos funcionales arrojándose como resultado la aceptación por parte del usuario.
5. Se efectuó un estudio detallado de la factibilidad del producto final, el cual arrojó los resultados esperados.

Recomendaciones

Con vistas al desarrollo futuro de este proyecto se recomienda:

1. Se propone la utilización del software.
2. Perfeccionar e incorporar nuevas funcionalidades que ayuden a ganar en eficiencia la solución propuesta.
3. Realizar un estudio más profundo de este sistema en vista a perfeccionarlo en versiones futuras.

Fuentes bibliográficas

[1] Arquitectura [Consultado 2010-5-11]. Disponible en: http://www.mailxmail.com/curso/informatica/generadores/capitulo2.htm.

[2] php_manual_es [Consultado 2010-10-13]. Disponible en: http://www.php.net/manual/es/.

[3] Gestión de Turnos TURNEX [Consultado 2011-02-23]. Disponible en: http://www.solex.cl/SOLUTIONS/OperationsManagement/LaborShiftsManagement.aspx.

[4] Metodología Tradicional [Consultado 2011-03-03]. Disponible en: http://www.bi-la.com/profiles/blogs/sobre-las-metodologias.

[5] Metodología XP vs. Metodología RUP [Consultado 2011-03-03]. Disponible en: http://metodologiaxpvsmetodologiarup.blogspot.com.

[6] XP [Consultado 2011-03-03]. Disponible en: http://www.extremeprogramming.org.

[7] PHP [Consultado 2011-03-03]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/p/h/p/PHP_2fec.html.

[8] HTML [Consultado 2011-03-06]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/c/r/c/CRC_tarjetas_f4e3.html.

[9] Lenguaje de programación [Consultado 2011-03-06]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/l/e/n/Lenguaje_de_programaci%C3%B3n.html.

[10] PostgreSQL [Consultado 2011-03-06]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/p/o/s/PostgreSQL_399b.html.

[11] Servidor HTTP Apache [Consultado 2011-03-06]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/s/e/r/Servidor_HTTP_Apache_4593.html.

[12] JavaScript [Consultado 2011-03-06]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/j/a/v/JavaScript_6861.html.

[13] Sitio Web [Consultado 2011-03-06]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/s/i/t/Sitio_web.html.

[14] Pagina Web [Consultado 2011-03-06]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/p/%C3%A1/g/P%C3%A1gina_web.html.

[15] Sistemas Gestores de Bases de Datos [Consultado 2011-03-06]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/s/g/b/SGBD_3a1b.html.

[16] Santana Díaz, Osmany. “Sistema de gestión y control de las practicas laborales”, Trabajo de Diploma, ISMM, Moa, 2010.

[17] Tarjetas CRC [Consultado 2011-03-17]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/c/r/c/CRC_tarjetas_f4e3.html.

[18] MySQL [Consultado 2011-03-28]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/m/y/s/MySQL_62a0.html.

[19] SQL [Consultado 2011-03-28]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/s/q/l/SQL_9778.html.

[20] Tecnohospital [Consultado 2011-04-01] Disponible en: http://www.tecnohospital.es/.

[21] Planificación [Consultado 2011-05-11]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/p/l/a/Planificaci%C3%B3n.html.

[22] Informática [Consultado 2011-05-11]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/i/n/f/Inform%C3%A1tica.html.

[23] Servidor Web [Consultado 2011-05-11]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/s/e/r/Servidor_web.html.

[24] Aplicación Web [Consultado 2011-05-11]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/a/p/l/Aplicaci%C3%B3n_web.html.

[25] Programación Extrema [Consultado 2011-05-12]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/p/r/o/Programaci%C3%B3n_Extrema3b63.html.

[26] Tomcat [Consultado 2011-05-12]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/t/o/m/Tomcat.html.

[27] IIS [Consultado 2011-05-12]. Disponible en: http://wikipedia.uo.edu.cu/es/articles/i/n/t/Internet_Information_Services_fb37.html

Glosario de términos

PHP: PHP Hypertext Pre-processor Lenguaje de programación para el desarrollo de Webs dinámicas, con sintaxis parecida a la de C. Originalmente se conocía como Personal Home Page tools, herramientas para páginas personales (en Internet).

HTML: Hyper Text Markup Language, o simplemente HTML, es un lenguaje de etiquetas de programación muy sencillo que se utiliza para crear los textos y las páginas Web. Si se hace la traducción de su nombre del inglés al castellano, sería “Lenguaje de Marca de Hipertextos”, ya que es justamente un lenguaje que se basa en las marcas para crear los hipertextos.

Software Libre: es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente, aunque conserve su carácter de libre, puede ser vendido comercialmente.

Ciclo de vida: Es un proceso por el cual los analistas de sistemas, los ingenieros de software, los programadores y los usuarios finales elaboran sistemas de información y aplicaciones informáticas.

Cliente: Persona, organización o grupo de personas que encargan la construcción de un producto software.

Herramientas: Son los ambientes de apoyo necesario para automatizar las prácticas de Ingeniería de Software.

Metodología ágil: Nuevo enfoque metodológico orientado a la gente y los resultados.

Metodología de desarrollo: Es una versión amplia y detallada de un ciclo de vida COMPLETO de desarrollo de sistemas que incluye: Reglas,

procedimientos, métodos, herramientas, funciones individuales y en grupo por cada tarea, productos resultantes, normas de Calidad.

Metodologías tradicionales: Metodologías basadas en procesos.

Proceso: secuencia de actividades que tienen un marcado inicio y fin.

Pruebas de aceptación: son las pruebas realizadas por el cliente para validar el software.

Anexos

Anexo # 1 Historias de usuarios.

HU No.2: Configurar bases de datos.

Historia de usuario	
Número: 2	**Usuario:** Administrador.
Nombre: Configurar bases de datos.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Reidelbis Calderón Suárez	
Descripción: El usuario podrá reconfigurar la conexión a la base de datos. El sistema guardara los datos.	
Observaciones: Deben de existir las base de datos, no siendo así las tablas en la base de datos del sistema, pero si en la de acceso a datos.	

Tabla 2.4 HU No.2: Configurar bases de datos.

HU No.3: Cambiar contraseña.

Historia de usuario	
Número: 3	**Usuarios:** Administradores y planificadores.
Nombre: Cambiar contraseña.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Reidelbis Calderón Suárez	
Descripción: El usuario será capaz de cambiar su contraseña con que accede a la aplicación, además de que el sistema validará la misma a la hora de entrar al sistema.	
Observaciones: El usuario debe de existir en la base de datos.	

Tabla 2.5 HU No.3 Cambiar contraseña.

HU No.4: Gestionar Usuarios.

Historia de usuario	
Número: 4	**Usuario:** Administrador.
Nombre: Gestionar Usuarios.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Reidelbis Calderón Suárez	
Descripción: El administrador podrá insertar un nuevo usuario, cuando un departamento del instituto le solicite el acceso a la aplicación, así como modificar o eliminar cualquiera de estos u observar todos los usuarios que acceden a la aplicación. El sistema guardará los datos.	
Observaciones: Solo un administrador podrá realizar estas operaciones.	

Tabla 2.6 HU No.4 Gestionar Usuarios.

HU No.5: Realizar la planificación del departamento.

Historia de usuario	
Número: 5	**Usuario:** Planificador.
Nombre: Realizar la planificación del departamento.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Reidelbis Calderón Suárez	
Descripción: El usuario tendrá a su disposición las variables principales para poder realizar una correcta planificación de su departamento. El sistema guardara los datos.	
Observaciones: Solo un usuario con permiso para planificar podrá realizar esta operación.	

Tabla 2.7 HU No.5 Realizar la planificación del departamento.

HU No.6: Realizar cambios en la planificación.

Historia de usuario	
Número: 6	**Usuario:** Planificador.
Nombre: Realizar cambios en la planificación.	
Prioridad en el negocio: Media	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Reidelbis Calderón Suárez	
Descripción: El usuario después de realizar la planificación podrá efectuar cambios en la misma, ya sea por solicitud de los estudiantes o porque el lo requiera. El sistema guardara los datos.	
Observaciones: Solo un usuario con permiso para planificar podrá realizar esta operación.	

Tabla 2.8 HU No.6 Realizar cambios en la planificación.

HU No.7 Justificar estudiantes incapacitados.

Historia de usuario	
Número: 7	**Usuario:** Planificador.
Nombre: Justificar estudiantes incapacitados.	
Prioridad en el negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Reidelbis Calderón Suárez	
Descripción: El usuario antes de realizar la planificación de la guardia podrá seleccionar a los estudiantes que por un motivo o por otro no puedan realizar la misma pudiendo justificarlos. El sistema guardara los datos.	
Observaciones: La justificación es opcional.	

Tabla 2.9 HU No.7 Justificar estudiantes incapacitados.

HU No.8: Mostrar planificación generada.

Historia de usuario	
Número: 8	**Usuarios:** Administradores, planificadores y visitantes.
Nombre: Mostrar planificación generada.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Reidelbis Calderón Suárez	
Descripción: El usuario podrá observar todas las planificaciones que se han realizado, por departamento o estudiante.	
Observaciones: El sistema procesará la petición definida por el usuario y le devolverá la solicitud realizada.	

Tabla 2.10 HU No.8 Mostrar planificación generada.

HU No.9: Mostrar estudiantes que no realizarán guardia.

Historia de usuario	
Número: 9	**Usuarios:** Administradores, planificadores y visitantes.
Nombre: Mostrar estudiantes que no realizarán guardia.	
Prioridad en el negocio: Media	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Reidelbis Calderón Suárez	
Descripción: El usuario podrá observar por departamentos, todos los estudiantes que por una razón u otra no realizarán guardia.	
Observaciones: El sistema procesará la petición definida por el usuario y le devolverá la solicitud realizada.	

Tabla 2.11 HU No.9 Mostrar estudiantes que no realizarán guardia.

HU No.10: Generar el resumen de la planificación.

Historia de usuario	
Número: 10	**Usuarios:** Administradores, planificadores y visitantes.
Nombre: Generar el resumen de la planificación.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Reidelbis Calderón Suárez	
Descripción: El usuario podrá observar, en general, la planificación que existe del departamento seleccionado.	
Observaciones: El sistema procesará la petición definida por el usuario y le devolverá la solicitud realizada.	

Tabla 2.12 HU No.10 Generar el resumen de la planificación.

HU No.11: Exportar el resumen de la planificación.

Historia de usuario	
Número: 11	**Usuarios:** Administradores, planificadores y visitantes.
Nombre: Exportar el resumen de la planificación.	
Prioridad en el negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Reidelbis Calderón Suárez	
Descripción: El usuario podrá exportar la planificación del departamento que seleccionó para generar el resumen.	
Observaciones: El sistema procesará la petición definida por el usuario y le devolverá la solicitud realizada.	

Tabla 2.13 HU No.11 Exportar el resumen de la planificación.

Anexo # 2 Tarjetas CRC.

Tarjeta CRC No 2: Configurar bases de datos.

Nombre de la clase: C_base_datos	
Tipo de la clase: Lógica del negocio	
Responsabilidades	**Colaboraciones**
Configurar la base de datos del sistema	C_bd_sistema
Configurar el acceso a datos del sistema	C_acceso_datos
Cambiar contraseña de la configuración del sistema	C_cont_sistema

Tabla 2.18 Tarjeta CRC No 2: Configurar la base de datos del sistema.

Tarjeta CRC No 3: Cambiar contraseña

Nombre de la clase: Cambiar_contraseña	
Tipo de la clase: Lógica del negocio	
Responsabilidades	**Colaboraciones**
Cambiar contraseña	Cambiar_contraseña

Tabla 2.19 Tarjeta CRC No 3: Cambiar contraseña.

Tarjeta CRC No 4: Gestionar usuarios

Nombre de la clase: Gestionar_usuarios	
Tipo de la clase: Lógica del negocio	
Responsabilidades	**Colaboraciones**
Insertar usuarios	Insertar_usuarios
Modificar usuarios	Modificar_usuarios
Eliminar usuarios	Eliminar_usuarios
Mostrar usuarios	Mostrar_usuarios

Tabla 2.20 Tarjeta CRC No 4: Gestionar Usuarios.

Tarjeta CRC No 5: Realizar la planificación del departamento

Nombre de la clase: Realizar_planificación	
Tipo de la clase: Lógica del negocio	
Responsabilidades	**Colaboraciones**
Realizar la planificación del departamento	Realizar_planificación

Tabla 2.21 Tarjeta CRC No 5: Realizar la planificación del departamento.

Tarjeta CRC No 6: Realizar cambios en la planificación

Nombre de la clase: R_cambios_planificación	
Tipo de la clase: Lógica del negocio	
Responsabilidades	**Colaboraciones**
Realizar cambios en la planificación de fecha	R_cambios_fecha
Realizar cambios en la planificación de hora	R_cambios_hora

Tabla 2.22 Tarjeta CRC No 6: Realizar cambios en la planificación.

Tarjeta CRC No 7: Justificar estudiantes incapacitados

Nombre de la clase: Justificar_estudiantes	
Tipo de la clase: Utilitaria	
Responsabilidades	**Colaboraciones**
Justificar estudiantes incapacitados	Justificar_estudiantes

Tabla 2.23 Tarjeta CRC No 7: Justificar estudiantes incapacitados.

Tarjeta CRC No 8: Mostrar planificación generada

Nombre de la clase: M_planificación_generada	
Tipo de la clase: Utilitaria	
Responsabilidades	**Colaboraciones**
Mostrar planificación por departamentos	M_planificación_dpto
Mostrar planificación por estudiantes	M_planificación_est

Tabla 2.24 Tarjeta CRC No 8: Mostrar planificación generada.

Tarjeta CRC No 9: Mostrar estudiantes que no realizarán guardia

Nombre de la clase: M_estudiantes_no_guardia	
Tipo de la clase: Utilitaria	
Responsabilidades	**Colaboraciones**
Mostrar estudiantes que no realizarán guardia	M_estudiantes_no_guardia

Tabla 2.25 Tarjeta CRC No 9: Mostrar estudiantes que no realizarán guardia.

Tarjeta CRC No 10: Generar el resumen de la planificación.

Nombre de la clase: Generar_resumen	
Tipo de la clase: Utilitaria	
Responsabilidades	**Colaboraciones**
Generar el resumen de la planificación.	Generar_resumen

Tabla 2.26 Tarjeta CRC No 10: Generar el resumen de la planificación.

Tarjeta CRC No 11: Exportar el resumen de la planificación.

Nombre de la clase: Exportar_resumen	
Tipo de la clase: Utilitaria	
Responsabilidades	**Colaboraciones**
Exportar el resumen de la planificación.	Exportar_resumen

Tabla 2.27 Tarjeta CRC No 11: Exportar el resumen de la planificación.

Anexo # 3 Tarjetas de Ingeniería.

Tarjeta de tarea No.2: Configurar la base de datos del sistema.

	Tarea ingeniería
Número tarea: 2	**Número historia:** 2
Nombre tarea: Configurar la base de datos del sistema.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 3/2/2011	**Fecha fin:** 4/2/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita la instalación de la aplicación.	

Tabla 3.4 Tarjeta de tarea No.2 Configurar la base de datos del sistema.

Tarjeta de tarea No.3: Configurar el acceso a datos del sistema.

	Tarea ingeniería
Número tarea: 3	**Número historia:** 2
Nombre tarea: Configurar el acceso a datos del sistema.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 7/2/2011	**Fecha fin:** 8/2/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita el acceso a los datos de los estudiantes si ocurre un cambio repentino de servidor de la base de datos.	

Tabla 3.5 Tarjeta de tarea No.3 Configurar el acceso a datos del sistema.

Tarjeta de tarea No.4: Cambiar contraseña de la configuración del sistema.

	Tarea ingeniería
Número tarea: 4	**Número historia:** 2
Nombre tarea: Cambiar contraseña de la configuración del sistema.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 9/2/2011	**Fecha fin:** 9/2/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita que el usuario que configura el acceso a las bases de datos pueda cambiar su contraseña.	

Tabla 3.6 Tarjeta de tarea No.4 Cambiar contraseña de la configuración del sistema..

Tarjeta de tarea No.5: Cambiar contraseña.

	Tarea ingeniería
Número tarea: 5	**Número historia:** 3
Nombre tarea: Cambiar contraseña.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10/2/2011	**Fecha fin:** 10/2/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita que los usuarios puedan cambiar su contraseña.	

Tabla 3.7 Tarjeta de tarea No.5 Cambiar contraseña.

Tarjeta de tarea No.6: Insertar usuarios.

	Tarea ingeniería
Número tarea: 6	**Número historia:** 4
Nombre tarea: Insertar usuarios.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 11/2/2011	**Fecha fin:** 11/2/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita insertar los usuarios que interactuarán con el software.	

Tabla 3.8 Tarjeta de tarea No.6 Insertar usuarios.

Tarjeta de tarea No.7: Modificar usuarios.

	Tarea ingeniería
Número tarea: 7	**Número historia:** 4
Nombre tarea: Modificar usuarios.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 14/2/2011	**Fecha fin:** 15/2/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita modificar algunos datos de los usuarios que acceden a la aplicación.	

Tabla 3.9 Tarjeta de tarea No.7 Modificar usuarios.

Tarjeta de tarea No.8: Eliminar usuarios.

	Tarea ingeniería
Número tarea: 8	**Número historia:** 4
Nombre tarea: Eliminar usuarios.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 16/2/2011	**Fecha fin:** 16/2/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita eliminar los usuarios que ya no van a interactuar con el software.	

Tabla 3.10 Tarjeta de tarea No.8 Eliminar usuarios.

Tarjeta de tarea No.9: Mostrar usuarios.

	Tarea ingeniería
Número tarea: 9	**Número historia:** 4
Nombre tarea: Mostrar usuarios.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17/2/2011	**Fecha fin:** 17/2/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita ver los usuarios que acceden a la aplicación.	

Tabla 3.11 Tarjeta de tarea No.9 Mostrar usuarios.

Tarjeta de tarea No.10: Realizar la planificación.

	Tarea ingeniería
Número tarea: 10	**Número historia:** 5
Nombre tarea: Realizar la planificación.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 2
Fecha inicio: 18/2/2011	**Fecha fin:** 7/3/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita la planificación de la guardia.	

Tabla 3.13 Tarjeta de tarea No.10 Realizar la planificación.

Tarjeta de tarea No.11: Realizar cambios en la planificación de hora.

	Tarea ingeniería
Número tarea: 11	**Número historia:** 6
Nombre tarea: Realizar cambios en la planificación de hora.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 8/3/2011	**Fecha fin:** 9/3/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita realizar cambios (de hora) a la planificación de la guardia.	

Tabla 3.14 Tarjeta de tarea No.11 Realizar cambios en la planificación de hora.

Tarjeta de tarea No.12: Realizar cambios en la planificación de fecha.

	Tarea ingeniería
Número tarea: 12	**Número historia:** 6
Nombre tarea: Realizar cambios en la planificación de fecha.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 10/3/2011	**Fecha fin:** 11/3/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita realizar cambios (de fecha) a la planificación de la guardia.	

Tabla 3.15 Tarjeta de tarea No.12 Realizar cambios en la planificación de fecha.

Tarjeta de tarea No.13: Justificar estudiantes incapacitados.

	Tarea ingeniería
Número tarea: 13	**Número historia:** 7
Nombre tarea: Justificar estudiantes incapacitados.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 14/3/2011	**Fecha fin:** 14/3/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita que los usuarios que van a ver la información vean el porque algunos estudiantes no van a realizar guardias.	

Tabla 3.16 Tarjeta de tarea No.13 Justificar estudiantes incapacitados.

Tarjeta de tarea No.14: Mostrar planificación por departamentos.

	Tarea ingeniería
Número tarea: 14	**Número historia:** 8
Nombre tarea: Mostrar planificación generada por departamentos.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 15/3/2011	**Fecha fin:** 15/3/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea muestra la planificación de las guardias (por departamentos) realizadas por sus respectivos usuarios.	

Tabla 3.18 Tarjeta de tarea No.14 Mostrar planificación por departamentos.

Tarjeta de tarea No.15: Mostrar planificación por estudiantes.

	Tarea ingeniería
Número tarea: 15	**Número historia:** 8
Nombre tarea: Mostrar planificación generada por estudiantes.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 16/3/2011	**Fecha fin:** 16/3/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea muestra la planificación de las guardias (por estudiantes) realizadas por sus respectivos usuarios.	

Tabla 3.19 Tarjeta de tarea No.15 Mostrar planificación por estudiantes.

\

Tarjeta de tarea No.16: Mostrar estudiantes que no realizarán guardia.

	Tarea ingeniería
Número tarea: 16	**Número historia:** 9
Nombre tarea: Mostrar estudiantes que no realizarán guardia.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17/3/2011	**Fecha fin:** 17/3/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea muestra los estudiantes que no van a realizar guardias con sus respectivas justificaciones (si tienen).	

Tabla 3.20 Tarjeta de tarea No.16 Mostrar estudiantes que no realizarán guardia.

Tarjeta de tarea No.17: Generar el resumen de la planificación.

	Tarea ingeniería
Número tarea: 17	**Número historia:** 10
Nombre tarea: Generar el resumen de la planificación.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 18/3/2011	**Fecha fin:** 21/3/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea posibilita crear un resumen de la planificación que se le realizó a cierto departamento, incluyendo a los estudiantes que no realizarán guardias.	

Tabla 3.21 Tarjeta de tarea No.17 Generar el resumen de la planificación.

Tarjeta de tarea No.18: Exportar el resumen de la planificación.

	Tarea ingeniería
Número tarea: 18	**Número historia:** 11
Nombre tarea: Exportar el resumen de la planificación.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 1
Fecha inicio: 22/3/2011	**Fecha fin:** 24/3/2011
Programador responsable: Reidelbis Calderón Suárez.	
Descripción: Esta tarea facilita la exportación (a un formato de impresión) del resumen.	

Tabla 3.22 Tarjeta de tarea No.18 Exportar el resumen de la planificación.

Anexo # 4 Pruebas de Aceptación.

Pruebas de aceptación para la HU: Configurar bases de datos.

Prueba de aceptación
HU: Configurar bases de datos.
Nombre: Prueba para comprobar la configuración de las bases de datos.
Descripción: El usuario debe ser administrador de la configuración del sistema, el mismo podrá reconfigurar el acceso a la base de datos donde el sistema guarda los datos y a la que accede para obtener datos relacionados con los estudiantes del instituto o podrá cambiar su contraseña.
Condiciones de ejecución: El usuario debe introducir su nombre de usuario y contraseña de configuración.
Entrada/Pasos ejecución: El usuario escribe el nombre de usuario y la contraseña de configuración del sistema aparte de los otros datos que se piden y oprime el botón “Configurar” en el caso de configuración o “Cambiar” si va a cambiar su contraseña.
Resultado: ➢ Al configurar el sistema correctamente: • Se emitirá un mensaje de confirmación. • Si no existe la tabla usuarios en la base de datos se emitirá un mensaje de información. ➢ Al configurar el acceso a datos correctamente: • Se emitirá un mensaje de confirmación. ➢ Al cambiar la contraseña: • Se emitirá un mensaje de confirmación. Se emitirá un mensaje de error en caso de que: • Si existen campos de textos vacíos. • Si el nombre de usuario o la contraseña son incorrectos. • Si no se logra realizar una conexión a la base de datos especificada.

Evaluación de la prueba: Aceptada.

Tabla 3.25 Prueba para comprobar la configuración de las bases de datos.

Pruebas de aceptación para la HU: Cambiar contraseña.

Prueba de aceptación
HU: Cambiar contraseña
Nombre: Prueba para comprobar el cambio de contraseña.
Descripción: Cambio de contraseña del usuario.
Condiciones de ejecución: El usuario debe introducir su nombre de usuario y contraseña.
Entrada/Pasos ejecución: El usuario escribe su nombre de usuario, contraseña, la nueva contraseña y la confirmación de la nueva contraseña, luego da clic en el botón “Cambiar”.
Resultado: • .Se emitirá un mensaje confirmando que ha cambiado su contraseña. Se emitirá un mensaje de error en caso de que: • Al presionar el botón “Cambiar” sin llenar todos los campos solicitados. • El usuario o la contraseña sean incorrectos. • Al ser diferentes la nueva contraseña y su confirmación.
Evaluación de la prueba: Aceptada.

Tabla 3.26 Prueba para comprobar el cambio de contraseña.

Pruebas de aceptación para la HU: Gestionar usuarios.

Prueba de aceptación
HU: Gestionar usuarios.
Nombre: Prueba para verificar la gestión de usuarios.
Descripción: Validación de la gestión de usuarios.
Condiciones de ejecución: El usuario debe ser administrador del sistema.
Entrada/Pasos ejecución: El administrador para poder insertar un usuario deberá llenar los campos solicitados, y después será capaz de modificar o eliminar al mismo o simplemente observar algunos de sus datos en el listado que se muestra al acceder a la gestión de usuarios.
Resultado: • Si se ha insertado correctamente el usuario se emitirá un mensaje de confirmación, igual que si se elimina o modifica el usuario. Se emitirá un mensaje de error en caso de que: ➢ Al insertar usuarios: • Si se presione el botón “Insertar” sin llenar todos los campos solicitados. • Si ya existe un usuario con ese mismo nombre de usuario. • Si ya existe un usuario que planifique para cierto departamento. • Si son diferentes la nueva contraseña y su confirmación. ➢ Al modificar un usuario: • Si se presione el botón “Modificar” sin llenar todos los campos solicitados. • Si al modificar los datos se ponen nuevos correspondientes a otro usuario. • Si son diferentes la nueva contraseña y su confirmación. ➢ Al eliminar un usuario: • Si existe un solo administrador en la base de datos o si se trata de eliminar el mismo que esta accediendo a la aplicación.
Evaluación de la prueba: Aceptada.

Tabla 3.27 Prueba para verificar la gestión de usuarios.

Pruebas de aceptación para la HU: Realizar la planificación del dpto.

Prueba de aceptación
HU: Realizar la planificación del departamento.
Nombre: Prueba para comprobar la planificación de la guardia.
Descripción: Se realizará la planificación del departamento al cual le corresponde planificar al usuario que accedió al sistema.
Condiciones de ejecución: El usuario debe de ser un planificador de la guardia del sistema.
Entrada/Pasos ejecución: El usuario escogerá el día de inicio y culminación de la guardia, además de los años o estudiantes que por una razón u otra no realizarán la misma pudiendo justificarlos. Después de realizada estas operaciones podrá presionar el botón “Planificar”.
Resultado: • Si se ha realizado correctamente las operaciones pertinentes se dará un mensaje de confirmación. Se emitirá un mensaje de error en caso de que: ➢ Al presionar el botón “Planificar”: • Si se escribió una o ambas fechas incorrectamente. • Si la fecha de inicio es mayor a la de culminación. ➢ Al presionar el botón “Enviar”: • Si no se seleccionó ningún estudiante.
Evaluación de la prueba: Aceptada.

Tabla 3.28 Prueba para comprobar la planificación de la guardia.

Pruebas de aceptación para la HU: Realizar cambios en la planificación.

Prueba de aceptación
HU: Realizar cambios en la planificación.
Nombre: Prueba para comprobar los cambios en la planificación.
Descripción: Se realizarán cambios en la planificación del departamento al cual le corresponde planificar al usuario que accedió al sistema.
Condiciones de ejecución: El usuario debe de ser un planificador de la guardia del sistema.
Entrada/Pasos ejecución: El usuario debe de dirigirse al menú Planificación/Realizar cambios y una vez allí podrá escoger que tipo de cambios querrá realizarle a la planificación, escogiéndose primeramente al estudiante para el cual se mostrarán los estudiantes disponibles.
Resultado: • Se dará un mensaje confirmando lo realizado.
Evaluación de la prueba: Aceptada.

Tabla 3.29 Prueba para comprobar los cambios en la planificación

Pruebas de aceptación para la HU: Mostrar planificación generada.

Prueba de aceptación
HU: Mostrar planificación generada.
Nombre: Prueba para comprobar como se muestra la planificación generada.
Descripción: Se mostrará esta información a cualquiera que acceda a la aplicación.
Condiciones de ejecución: (No existen)
Entrada/Pasos ejecución: Ir al menú que dice “MOSTRAR INFORMACIÓN” y seleccionar la opción “Planificación”, estado allí puede seleccionar “Por departamento” o “Por estudiantes”.
Resultado: • Se mostrará la planificación según la selección realizada. • Se mostrarán mensajes en forma de información en caso de que la selección realizada presente algún problema.
Evaluación de la prueba: Aceptada.

Tabla 3.30 Prueba para comprobar como se muestra la planificación.

Pruebas de aceptación para la HU: Mostrar estudiantes que no realizarán guardia.

Prueba de aceptación
HU: Mostrar estudiantes que no realizarán guardia.
Nombre: Prueba para comprobar como se muestra los estudiantes que no realizarán guardia.
Descripción: Se mostrará esta información a cualquiera que acceda a la aplicación.
Condiciones de ejecución: (No existen)
Entrada/Pasos ejecución: Ir al menú que dice “MOSTRAR INFORMACIÓN” y seleccionar la opción “Estudiantes sin guardia”.
Resultado: • Se mostrarán los estudiantes que no realizarán guardia según el departamento seleccionado. • Se mostrarán mensajes en forma de información en caso de que la selección realizada no tenga ningún estudiante incapacitado.
Evaluación de la prueba: Aceptada.

Tabla 3.31 Prueba para comprobar como se muestra los estudiantes que no realizarán guardia.

Pruebas de aceptación para la HU: Generar el resumen de la planificación.

Prueba de aceptación
HU: Generar resumen de la planificación.
Nombre: Prueba para comprobar como se genera el resumen de la planificación del departamento.
Descripción: Se mostrará esta información a cualquiera que acceda a la aplicación.
Condiciones de ejecución: (No existen)
Entrada/Pasos ejecución: Ir al menú que dice “RESUMEN” y seleccionar la opción “Generar resumen”.
Resultado: • Se mostrará la planificación realizada del departamento seleccionado. • Se mostrarán mensajes en forma de información en caso de que la selección realizada no tenga ninguna planificación.
Evaluación de la prueba: Aceptada.

Tabla 3.32 Prueba para comprobar como se genera el resumen de la planificación.

Pruebas de aceptación para la HU: Exportar el resumen de la planificación.

Prueba de aceptación
HU: Exportar el resumen de la planificación.
Nombre: Prueba para comprobar como se exporta el resumen de la planificación.
Descripción: Cualquiera que acceda a la aplicación podrá exportar esta información.
Condiciones de ejecución: (No existen)
Entrada/Pasos ejecución: Una ve realizado el resumen, el mismo se podrá exportar en un formato de impresión.
Resultado: • Se exportará la planificación realizada del departamento seleccionado. Se emitirá un mensaje de error en caso de que: • No exista información para exportar.
Evaluación de la prueba: Aceptada.

Tabla 3.33 Prueba para comprobar como se exporta el resumen de la planificación.

Anexo # 5 Principales interfaces de la Aplicación.

Interfaz principal del sistema.

Interfaz de la planificación de la guardia.

Interfaz de la gestión de usuarios.

Interfaz de la realización de cambios, de fecha.

Interfaz del resumen y exportación de la planificación.