PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

FACULTAD DE INGENIERIA

ESCUELA DE SISTEMAS

Disertación de Grado

Previa a la obtención del Título de:

Ingeniera de Sistemas y Computación

TEMA:

"APLICACIÓN DE PATRONES DE DISEÑO PARA LA CONSTRUCCIÓN DE UN SISTEMA DE ADMINISTRACION DE LA INFORMACION DEL PERSONAL QUE LABORA EN EL HOSPITAL GINECO OBSTETRICO ISIDRO AYORA"

María Gabriela Rivera Valarezo
Andrea Gabriela Vega Ortiz

Quito – Ecuador

2013, Abril

DEDICATORIAS

A Dios.

Por permitirme llegar hasta aquí con salud y tener a todos los que amo junto a mí en este punto de mi vida.

A mi madre Laura.
Por haber estado siempre conmigo en todo momento, por sus consejos, sus valores, por haberme enseñado a ser lo que soy, y por sobre todo gracias por su amor.

A mi padre William.
Por su ejemplo de perseverancia y constancia, por ser quien siempre creyó en mí y por supuesto porque me ha mostrado para salir adelante y sobre todo por su gran amor.

A mis familiares.
A mis hermanas Sofía, Paulina y Cristina por ser el ejemplo de hermanas mayores por ser el apoyo que necesite siempre y de las que aprendí en los momentos difíciles; a mi abuelito Arturo Valarezo que me ha enseñado que siempre se debe luchar por lo que se quiere y que debemos dejar huella donde vamos tal como él lo ha hecho; a mis cuñados Gustavo y Byron que siempre me dieron ánimos y consejos en el momento que necesite.

Att.
María Gabriela Rivera Valarezo.

Dedico la presente tesis a mi hijo Mateo que bajo del cielo, para llenar de alegría mi vida por quien cada día tiene sentido, gracias porque eres mi inspiración y fortaleza, una sonrisa tuya ilumina mi mundo y me da las fuerzas necesarias para luchar y conseguir mis metas en busca de un mejor futuro, a él, mi esperanza, mi alegría, mi vida y la culminación de este trabajo y lo que representa

Att.

Andrea Gabriela Vega Ortiz

AGRADECIMIENTOS

A la Pontificia Universidad Católica del Ecuador, porque gracias a esta me pude formar y aprender cada semestre hasta llegar a cumplir esta meta tan añorada. A mis padres William y Laura por apoyarme siempre como lo necesite a mis hermanas, cuñados, sobrinos y abuelitos por siempre darme una palabra de aliento, a mis amigos de toda la vida por darme fuerzas cuando estuve cansada a mis maestros que impartieron su conocimiento para llegar donde estoy y en fin a todos los que participaron directa o indirectamente en la elaboración de esta tesis.

¡Gracias a ustedes!

Att.
María Gabriela Rivera Valarezo.

El presente trabajo de tesis primeramente me gustaría agradecer a Dios por bendecirme para llegar hasta donde he llegado, porque hiciste realidad este sueño tan anhelado, porque has estado conmigo a cada paso que doy, cuidándome y dándome fortaleza para continuar

A mi madre Mónica por siempre haber estado a mi lado apoyándome y acompañándome a lo largo de mi vida, ha velado por mi bienestar y educación siendo mi apoyo en todo momento. Depositando su entera confianza en cada reto que se me presentaba sin dudar ni un solo momento en mi capacidad. Es por ella que soy lo que soy ahora.

A la Pontificia Universidad Católica del Ecuador por darme la oportunidad de estudiar y ser una profesional. También me gustaría agradecer a mis profesores durante toda mi carrera profesional porque todos han aportado con un granito de arena a mi formación

Muchas gracias

Att.

Gabriela Vega Ortiz

TABLA DE CONTENIDO

TABLA DE FIGURAS

CAPITULO I: INTRODUCCION

En este capítulo se dará una introducción sobre lo que motivó al desarrollo de esta tesis, así como una descripción de la propuesta que se dio para resolver el problema expuesto y el detalle de cada capítulo de la disertación.

Esta disertación se realizó para el Hospital Gíneco-Obstetra "Isidro Ayora", tiene como finalidad crear una aplicación que servirá para gestionar la información de los empleados del Hospital.

Debido a la cantidad de información que maneja el Hospital y que la misma es administrada manualmente, consideramos proporcionar una aplicación que ayude llevar una mejor administración de dicha información. De la misma manera la institución se beneficiará automatizando sus procesos, lo que es importante en la actualidad.

La Aplicación denominada STAFFLOG Software y tiene su propio logo de identificación. Los Usuarios podrán acceder a ciertos módulos dependiendo del tipo que sean. STAFFLOG Software ayudará con tareas tales como el registro de nuevos Empleados, registro de Contratos, Vacaciones, Horas Extra, Acciones de Personal y Ausencias. Para el registro de Contratos y Acciones de Personal interactúa con la herramienta de ofimática correspondiente.

STAFFLOG Software está desarrollada en PHP y tiene conexión a una base de datos en MySQL y trabaja con el servidor XAMPP, además para la implementación se hará uso de los Patrones de Diseño Estrategia y Fachada que sirven mucho en la implementación de aplicaciones tipo Web.

Para la realización de esta disertación se realizó un estudio de Patrones de Diseño su significado, su uso, los Patrones más Importantes y conocidos, dando mayor importancia y profundidad a los Patrones Fachada y Estrategia. También se debió hacer una revisión de la metodología RUP y del lenguaje de programación PHP; todo esto está en el Capítulo II que es el Marco Teórico.

Después de hacer el debido estudio de los principales temas a usarse en el desarrollo de la aplicación, plateamos nuestros objetivos y estudiamos la misión y visión del Hospital lo que ayudó en la realización del proyecto esto está planteado en el Capítulo III que es la Presentación, además en esta etapa se recogieron los requerimientos tanto funcionales como no funcionales y la descripción general del sistema.

Para el desarrollo de STAFFLOG Software se realizó la metodología RUP, la cual hace uso de UML para su implementación, los diagramas que se usaron en este proyecto son: Casos de Uso, de Actividad, de Clases y de Secuencia. La arquitectura que utilizamos fue Cliente/Servidor. En la etapa de implementación debido al espacio solo mostramos la parte de crear y modificar usuarios ya que ésta se puede visualizar de mejor forma el uso de los Patrones de los Diseño dentro de la aplicación.

Todo esto se puede visualizar y estudiar en el Capítulo IV llamado Aplicación al caso de Estudio: ADMINISTRACION DE LA INFORMACION DE PERSONAL QUE LABORA EN EL HOSPITAL GINECO OBTETRICO ISIDRO AYORA.

Finalmente en el Capitulo V se presentan las conclusiones y comentarios que llegamos después de la realización de la presente disertación.

CAPITULO II: ELEMENTOS TEORICOS

En el presente capitulo presentaremos el concepto de patrones de diseño, su uso, finalidad, su mejor forma de deducción y sus elementos, los beneficios más significativos que obtenemos de éstos, como ayudan dentro de las aplicaciones web.

También la clasificación principal de los Patrones de Diseño y de cada clasificación los principales y más conocidos, así como, en que tecnologías podemos trabajar con ellos.

Se hará también una exposición de los Patrones de Diseño Estrategia y Fachada los mismos que serán usados para la elaboración del proyecto.

También veremos una definición de la metodología RUP sus dimensiones, características, disciplinas y elementos la misma que va a ser usada en la realización de la Tesis.

Por último se hará una introducción a PHP, ya que es el lenguaje de programación en el que se realizará el presente proyecto.

2.1 Patrones de Diseño:

2.1.1 Que son Patrones de Diseño

Después de un trabajo que realizaron Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, conocidos como "la pandilla de los cuatro", dieron a conocer Patrones de Diseño como concepto en su libro llamado "*Patrones de diseño: Elementos de software orientado a objetos reutilizables*" publicado en 1995. De esta manera se popularizó la utilización de estos en la programación orientada a objetos y se ha llegado a nombrar a dicho libro como "La Biblia" de los Patrones de Diseño.

Los patrones de diseño son simples soluciones para los comunes problemas que se presentan al hacer un diseño orientado a objetos, como son en base a experiencias, éstos nos ayudan guiándonos con consejos de cómo aplicarlos y permitiendo reutilizarlos.

Los Patrones nos ayudan también a reconocer los errores que cometemos al ir realizando una aplicación; por lo que serán de ayuda para evitarlos y darles solución a los que ya cometimos.

Aunque los patrones no sean fáciles de entender en un principio, al estudiarlos y conseguir entenderlos, son fáciles de aplicar y seguir. La finalidad de un patrón no es crear nuevas formas de diseñar sino reusar las ya creadas para que el Ingeniero de Software pueda aplicar todos sus conocimientos y experiencias así como las de sus colegas.

Antes de usar un patrón es importante primero conocerlo bien analizarlo y deducirlo. Se debe escoger el patrón que sirva de ayuda en el problema que se tenga, para lo cual será necesario conocer los diferentes tipos de patrones y sus utilidades.

Elementos de un Patrón de Diseño:

Un patrón está formado por los siguientes componentes:

- ✓ **Nombre:** El cual describirá en una o dos palabras lo que solucionará el Diseño.
- ✓ **Problema:** Una explicación breve del problema que se va a resolver.
- ✓ **Solución:** Indica todo lo que compone el Diseño, sus relaciones y funciones.
- ✓ **Consecuencias:** Aquí están las conclusiones, ventajas y desventajas del Diseño.

Entre los principales beneficios de los patrones de diseño tenemos:

- Brindan soluciones probadas de diseño, en base a principios no abstractos; además hace el contexto del problema explícito y resumido.
- Mejora de proceso de diseño y ayuda a los diseñadores a aumentar la productividad reduciendo el tiempo empleado en el mismo.
- Reutilización de interfaces compatibles; desarrollando una biblioteca de componentes tanto de diseño de interfaz como de aplicaciones lo cual aumentará la coherencia y facilidad de uso de los diseños. Esto es útil en especial para las empresas.
- Lenguaje común compartido, es bueno cuando se trabaja en equipo ya que los patrones serán entendidos por todos.
- Ayuda para la enseñanza efectiva y una herramienta de referencia, útil para diseñadores novatos como para los que llevan muchos años diseñando y programando.
- Aplicaciones web de beneficio y utilidad; creando un ambiente amigable para los usuarios finales.

En el ambiente web el diseño de interfaces es de difícil ejecución ya que se tienen limitados controles interactivos que sean compatibles en los navegadores todo esto debido a que se tiene poco conocimiento sobre cómo las interacciones del usuario deben ser implementadas. Aquí es donde los patrones de diseño toman su gran importancia en las aplicaciones web.

2.1.2 Principales Patrones

Debido a la variedad de Patrones de Diseño y por su gran número, se los ha clasificado de tal manera que su estudio se simplifique de la siguiente manera:

- ✓ Patrones de Creación
- ✓ Patrones Estructurales
- ✓ Patrones Funcionales o de Comportamiento

- **Patrones de Creación:** En este grupo están los patrones que sirven de guía para construir objetos. Además independizan el qué?, quién?, cómo? y cuándo? en la creación del objeto.
  Entre los más conocidos de esta clase están:

  - ✓ **Patrón de Fábrica Abstracta:** hace abstracto el tipo de familia definido con la que se va a trabajar utilizando objetos de varias familias sin que éstas se mezclen.

  - ✓ **Patrón del Método de Fabricación:** forma una clase constructora común en la que se crearán todos los objetos de un determinado tipo.

  - ✓ **Patrón Constructor:** centraliza el proceso de creación de un objeto, simplificando y agrupando los pasos más complejos en uno solo.

  - ✓ **Patrón Prototipo:** crea objetos nuevos reutilizando los que se encuentren en una instancia ya establecida.

  - ✓ **Patrón de Instancia Única:** permite y garantiza la creación de una única clase para que todos los objetos que la necesiten hagan uso de ésta.

- **Patrones Estructurales:** En este grupo encontramos los patrones que organizan las clases y objetos de características similares y así formar estructuras más grandes.
  En los patrones de esta clase tenemos:

  - ✓ **Patrón Adaptador:** permite que una clase haga uso de una interfaz a la cual no tiene acceso, adaptando la primera especialmente para que sea convertida en otra parecida.

  - ✓ **Patrón Puente:** desacopla una abstracción de su implementación permitiendo modificarla a gusto sin que la primera cambie su forma original.

  - ✓ **Patrón Compuesto:** permite tratar los objetos complejos como si fueran simples, además ayuda a crear objetos complejos a partir de otros más simples y similares entre sí.

  - ✓ **Patrón Decorador:** agrega funcionalidades dinámicamente a un objeto; de esta manera hace que no sea necesario crear subclases de una clase principal.

  - ✓ **Patrón de Fachada:** crea una interfaz de alto nivel única, la cual engloba varias interfaces del sistema facilitando el acceso a cada una de ellas.

  - ✓ **Patrón de Peso Mosca:** es utilizado cuando se necesita de varios objetos ya que éste reduce la información repetida de los mismos, facilitando y simplificando su uso**.**

  - ✓ **Patrón Apoderado:** proporciona un intermediario para controlar el acceso a un objeto.

- **Patrones Funcionales o de Comportamiento:** En el último grupo encontramos los patrones que sirven para organizar, gestionar y combinar el comportamiento de diferentes objetos y del programa en ejecución.
  Los más conocidos son:

- ✓ **Patrón de Cadena de Responsabilidad:** proporciona orden a los mensajes que llegan a los objetos y de esta manera da la oportunidad de que dichos mensajes no se acoplen en un solo receptor. Se lo suele utilizar en conjunto con el Patrón Compuesto.

- ✓ **Patrón de Comando:** encapsula una operación en un objeto sin necesidad de conocer el contenido de éste. Este Patrón es de gran ayuda ya que se permite gestionar y deshacer operaciones con mayor facilidad.

- ✓ **Patrón Intérprete:** define expresiones específicas para manejar las funciones del lenguaje de programación que escojamos para realizar nuestro proyecto.

- ✓ **Patrón Iterador:** define una interfaz la cual tiene los métodos necesarios para acceder secuencialmente a los objetos independientemente de cómo estén implementados.

- ✓ **Patrón Mediador:** define un objeto que ayude en la comunicación de objetos de diferentes clases pero que tienen similitudes entre sí.

- ✓ **Patrón Recuerdo:** guarda el estado de un objeto para utilizarlo cuando se lo vuelva a utilizar desde el punto en el que quedó anteriormente, esto sin romper la encapsulación.

- ✓ **Patrón Observador:** define relación de uno a muchos entre varios objetos, de esta manera al cambiar de estado un objeto se actualizaran todos aquellos que dependan de él.

- ✓ **Patrón de Estado:** modifica el comportamiento de un objeto cada vez que cambie su estado interno.

- ✓ **Patrón de Estrategia:** define una familia de algoritmos para utilizar cualquiera de ellos en el momento que se necesiten.

- ✓ **Patrón del Método Plantilla:** define una operación como esqueleto de todo el algoritmo, delegando subclases de ésta, lo cual permite que se puedan modificar pasos de las subclases sin alterar la estructura.

- ✓ **Patrón Visitante:** permite definir una nueva operación sobre una estructura de clases sin que se modifiquen las clases sobre las que se opera.

2.1.3 Tecnologías que aceptan Patrones

Existen patrones para algunas de las tecnologías de programación más conocidas y usadas por los Ingenieros de Software. Entre las principales tecnologías que aceptan Patrones están PHP, C#, ASP.NET y Java.

2.1.4 Patrón de Diseño Estrategia

En un programa siempre habrá escenarios comunes por lo que las clases sólo se diferencian en su forma de comportarse. En este caso para resumir el código es una buena idea encerrar los algoritmos en uno solo y de esta manera poder seleccionar diferentes algoritmos al tiempo que se los necesite; esto es lo que hace el Patrón de Diseño Estrategia.

El Patrón de Estrategia es un patrón de diseño de comportamiento que nos permite decidir qué curso de acción debería tener un programa. Este encapsula dos algoritmos diferentes dentro de dos clases y decide, en tiempo de ejecución, que estrategia debe seguir.

Para lograrlo se extraen las clases más complejas y se las agrupa para así al momento de necesitarlas se llamará a una sola que decide cual se usará en determinado momento; captura la abstracción en una interfaz y esconde los detalles de implementación de las clases derivadas.

Este patrón es comúnmente utilizado cuando se necesitan distintas variantes del mismo algoritmo, cuando una clase define muchos comportamientos o cuando un algoritmo usa datos que los clientes no tienen por qué conocer.

Ventajas de Usar el Patrón Estrategia:

- Son fáciles comprender
- Permite de forma simple añadir nuevos comportamientos y eliminar o modificar los existentes.
- Se puede reducir a una única clase cuando existan varios objetos cuyo comportamiento sea similar o en algunos casos el mismo.
- Se reduce el uso de subclases.
- Encapsula algoritmos y los hace intercambiables entre sí permitiendo la independencia de los objetos que los usen.

Implementación de Patrón Estrategia:

Las posibles estrategias se ejecutan dentro de un objeto de contexto el mismo que se encarga de recuperar la estrategia apropiada para el cliente.

Cada una de las estrategias implementa una interfaz que define la firma del método de la estrategia.

Para entender mejor este patrón vamos a citar el ejemplo de utilidad que le daremos en este proyecto: se debe desarrollar una clase la cual que puede crear o actualizar el registro de un nuevo usuario o persona, para las dos acciones se necesitan los mismos campos; pero, dependiendo de la situación, se deben usar diferentes funciones.

Para lograrlo se deberían usar las condiciones if'else varias veces, al necesitar usar dicha clase en lugares diferentes se haría demasiado extenso por lo que resultaría más conveniente reducir y puntualizar el contexto.

Otra utilidad de este patrón es cuando se crea una aplicación Web la cual debe estar presentada en varios idiomas o estilos, para solucionarlo primero se debe crear una interfaz común para todos los algoritmos y luego hacer varias implementaciones de la misma, definiendo una clase general y las extensiones con cada subclase. De esta manera se ahorra tiempo y se simplifica la programación.

Estructura

Este patrón tiene 3 participantes:

Contexto: Define una interfaz que permite que la estrategia tenga acceso a sus datos.

Estrategia: Declara una interfaz común para todos los algoritmos que necesiten usar la estrategia. Esta interfaz será usada por el contexto para invocar a la estrategia concreta.

Estrategia Concreta: Implementa lo que va a ser utilizado por la estrategia.

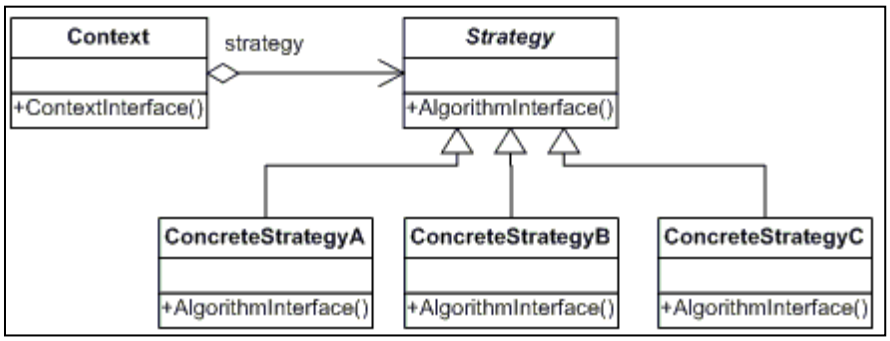

Figura No: II-1Estructura del Patrón de Diseño Estrategia [1]

2.1.5 Patrón de Diseño Fachada

Por lo general en un programa los clientes no deben enterarse de las variables internas que tiene el programa, más aún cuando se está trabajando en ambiente web.

El Patrón Fachada hace que los subsistemas sean fáciles de usar creando una interfaz de alto nivel unificada que engloba a las interfaces del subsistema; de esta manera se reduce al mínimo la comunicación y dependencia entre los subsistemas y de la misma manera con el cliente.

Ventajas de Usar el Patrón Fachada:

- Al usar este patrón se da más portabilidad y seguridad al programa o aplicación que se esté realizando.
- Disminuye el acoplamiento entre el sistema y el cliente.
- Hace los sistemas reusables.
- Reduce dependencias dentro del software.

Implementación de Patrón Fachada:

El patrón indica que al crear una fachada esta hará que el acoplamiento entre los subsistemas y el cliente sea bajo por lo que la fachada será una clase abstracta con subclases concretas.

Se crea una aplicación donde el cliente ingresa sus datos; la aplicación usará los datos como validación y almacenamiento de los mismos de manera que el cliente no tiene que saber cómo, ni en qué momento lo hizo. Al cliente solo le interesan los resultados; más aún si se trata de una aplicación web donde todos los parámetros se los puede visualizar de forma directa en el paso de variables. Esto es lo que hace Fachada crea una interfaz que será visible al cliente y la que llamará a un subsistema escondiendo así los datos que no tiene que visualizar el cliente.

Estructura

Este patrón tiene 2 participantes:

Fachada: Conoce la distribución de las subclases dentro del subsistema y quienes son responsables de las peticiones.

Subsistema: Implementa la funcionalidad del subsistema, es decir realiza el trabajo pedido de la fachada.

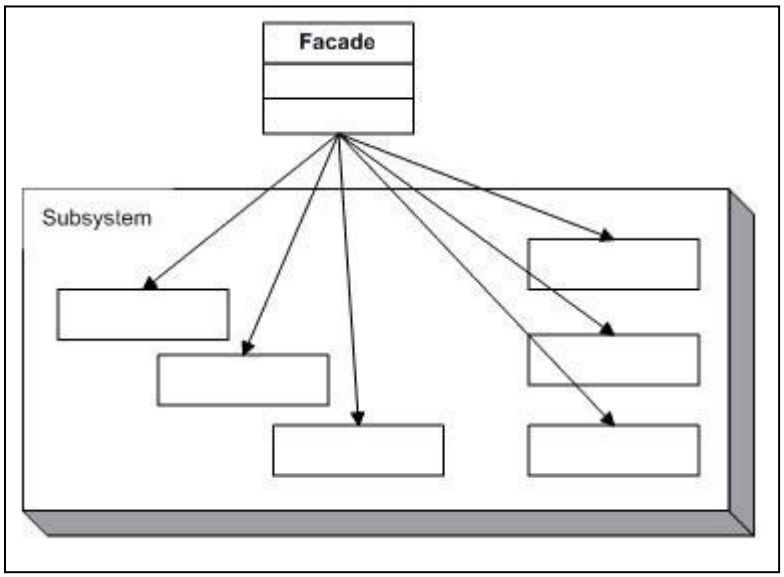

Figura No: II-2Estructura del Patrón de Diseño Fachada [14]

2.2 Metodología RUP

RUP es un proceso de Ingeniería de Software orientado a objetos que se lo podría definir también como un conjunto de metodologías adaptables a las necesidades de una organización.

2.2.1 Definición – RUP

El Rational Unified Process o Proceso Unificado Racional - RUP es un proceso de Ingeniería de Software que suministra un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo; es un método que está orientado a objetos, se lo podría definir también como un conjunto de metodologías adaptables a las necesidades de una organización.

2.2.2 Dimensiones de la metodología RUP

La metodología consta de dos dimensiones:

1. El eje horizontal representa tiempo y demuestra los aspectos del ciclo de vida del proceso.

2. El eje vertical representa las disciplinas, que agrupan actividades definidas lógicamente por la naturaleza.

En la primera dimensión se encuentra representado el aspecto dinámico del proceso y se expresa en términos de fases, de iteraciones, y la finalización de las fases.

La segunda dimensión representa el aspecto estático del proceso: cómo se describe en términos de componentes de proceso, las disciplinas, las actividades, los flujos de trabajo, los artefactos, y los roles.

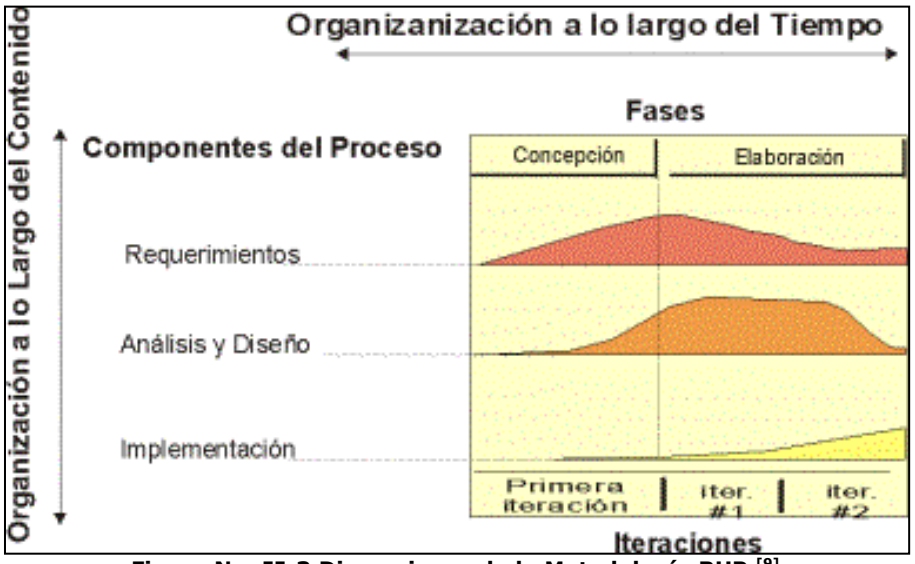

Figura No: II-3 Dimensiones de la Metodología RUP [9]

Claramente como muestra la Figura N° 1 se distingue que en el eje vertical se encuentran especificadas cada una de las disciplinas de los proyectos, sus etapas, mientras que en el eje horizontal se detallan las fases; es decir, el ciclo de vida del proceso en sí como es la fase inicial, la elaboración, construcción y transición.

2.2.3 Fases del proyecto de acuerdo a RUP

De acuerdo a cada una de las fases que se presentan en el eje horizontal de las dimensiones descritas anteriormente tenemos las fases como son:

a) **Inicial**: En esta fase se definen aspectos de entrada importantes en el proyecto como el alcance, el ámbito, los objetivos, la funcionalidad y la capacidad que tendrá el producto a diseñarse.

b) **Elaboración:** Esta fase requiere de temas profundos como es la definición, el análisis y el diseño en sí, con lo cual se podrá definir la estructura básica y los recursos disponibles del proyecto.

c) **Construcción:** Como su nombre lo indica se construye o se desarrolla el producto, se analizan procesos ya establecidos en las fases anteriores como el análisis y diseño; aquí la arquitectura del proyecto es fundamental ya que es en esta fase donde se realizan la programación y las pruebas. Esta es la fase de la implementación propia del proceso.

d) **Transición:** Esta es la fase final del proyecto donde requerimientos como implementación, gestión y control de cambios, la gestión del proyecto y la comunicación alcanzan sus máximos de interacción. El refinamiento de los manuales para el usuario, instalación y técnicos se obtienen en esta etapa.

2.2.4 Gestión de Proyectos

Son las disciplinas que se encuentran en el eje vertical de las dimensiones del RUP, las cuales son:

a) **Requerimientos:** Tiene como objetivo fundamental determinar lo que el sistema debe hacer (Especificar Requisitos), puntualizar los límites del sistema, y una interfaz de usuario, realizar una estimación del costo y tiempo de desarrollo. Utiliza el Modelo de Casos de Uso (CU[1]) para modelar el Sistema que comprenden los CU, Actores y Relaciones, además utiliza los diagramas de Estados de cada CU y las especificaciones adicionales.

[1] CU: Caso de Uso

b) **Diseño:** Especifica la construcción del sistema y tiene como objetivos trasladar requerimientos en especificaciones de implementación, al decir análisis se refiere a transformar CU en clases, y al decir diseño se representa a refinar el análisis para poder implementar los diagramas de clases de análisis de cada CU, los diagramas de colaboración de cada CU, el de clases de diseño de cada CU, el de secuencia de diseño de CU, el de estados de las clases, el modelo de despliegue de la arquitectura.

c) **Implementación:** Aquí el objetivo primordial es implementar las clases de diseño como componentes, asignar los dispositivos a los nodos, probar los componentes en forma individual, integrar los componentes en un sistema ejecutable (enfoque incremental). Utiliza el Modelo de Implementación, conjuntamente con los Diagramas de Componentes para comprender cómo se organizan los Componentes y dependen unos de otros.

d) **Verificación:** Tiene como objetivos verificar la integración de los componentes (prueba de integración), cotejar que todos los requisitos han sido implementados (pruebas del sistema), asegurar que los defectos descubiertos se han resuelto antes de la distribución.

e) **Implantación:** Lo primordial es establecer e implantar los artefactos producidos y sus componentes para poder distribuirlos.

f) **Gestión de Configuración y Control de Cambios:** Es fundamental para controlar el número de artefactos[2] producidos por la cantidad de personal que trabajan en un proyecto conjuntamente.

[2] Artefacto: Pieza de Información

Los controles sobre los cambios son de mucha ayuda ya que evitan confusiones costosas como la compostura de algo que ya se había arreglado (defectos de diseño y sus componentes), y aseguran que los resultados de los artefactos no entren en conflicto con algunos problemas como:

- Actualización simultánea: Es la actualización de algo elaborado con anterioridad, sin saber que alguien más lo está actualizando.

- Notificación limitada: Al realizar alguna modificación, no se deja información sobre lo que se hizo, por lo tanto no se sabe quién, cómo, y cuándo se hizo.

- Versiones múltiples: No saber con exactitud, cual es la última versión, y al final no se tiene un orden sobre que modificaciones se han realizado a las diversas versiones.

g) **Gestión del proyecto:** *En la gestión de proyecto su objetivo es equilibrar los objetivos competitivos, administrar el riesgo, y superar restricciones para entregar un producto que satisface las necesidades de los ambos clientes con éxito (los que pagan el dinero) y los usuarios. Con la Gestión del Proyecto se logra una mejoría en el manejo de una entrega exitoso de software[E].*

h) **Gestión de calidad:** En la disciplina de gestión de calidad es fundamental para que lo producido se mantenga en constante mantenimiento, con niveles y estándares de cero deficiencias y problemas de su arquitectura y que pueda seguir el producto sin defectos ante más requerimientos o especificaciones.

2.2.5 Características de la Metodología RUP

Los autores de RUP (Jacaboson, I., Booch, G., Rumbaugh J.) destacan que el proceso de software propuesto por RUP posee tres características fundamentales:

- Está dirigido por los Casos de Uso
- Está centrado en la arquitectura, y
- Es iterativo e incremental.

Proceso dirigido por Casos de Uso

"Los Casos de Uso son una técnica de captura de requisitos que fuerza a pensar en términos de importancia para el usuario y no sólo en términos de funciones que sería bueno contemplar." [D]. Según Kruchten, también dice que *se define un Caso de Uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido.*

Los Casos de Uso constituyen una herramienta primordial de trabajo, un elemento integrador y una guía del trabajo, con los cuales se puede desarrollar y crear los modelos de análisis y diseño, luego procedemos a la implementación que los lleva a cabo, y se comprueba que efectivamente el producto implemente adecuadamente cada Caso de Uso y que todos los patrones deben estar adaptados con el modelo de Casos de Uso.

Centrado en la Arquitectura

Un proceso que se encuentre centrado en la arquitectura nos muestra la forma del sistema y si debe diseñarse de forma que pueda evolucionar no únicamente de su desarrollo inicial, sino en futuras generaciones.

La arquitectura de un sistema es la organización o estructura de sus partes principales, situación que permite tener una visión común entre todos los involucrados en el proyecto (desarrolladores y usuarios) y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo.

La arquitectura también involucra aspectos estáticos y dinámicos más significativos del sistema, ésta se relaciona directamente con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden.

Por otra parte la definición de la arquitectura debe tomar en consideración elementos de calidad del sistema, rendimiento, reutilización y capacidad de evolución por lo que debe ser flexible durante todo el proceso de desarrollo.

En el caso de RUP además de utilizar los Casos de Uso para guiar el proceso se presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento. Cada producto tiene tanto una función como una forma. La función corresponde a la funcionalidad reflejada en los Casos de Uso y la forma la proporciona la arquitectura. Existe una interacción entre los Casos de Uso y la arquitectura, los Casos de Uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los Casos de Uso requeridos, actualmente y en el futuro. Esto provoca que tanto arquitectura como Casos de Uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software.

Iterativo e incremental

Jacaboson, I., Booch, G., Rumbaugh J., el equilibrio correcto entre los Casos de Uso y la arquitectura es algo parecido al equilibrio de la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo.

Para esto, la estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto. Una iteración puede realizarse por medio de una cascada; es decir, se pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Cada iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos requisitos o han cambiado los existentes, afectando a las iteraciones siguientes. Durante la planificación de los detalles de la siguiente iteración, el equipo también examina cómo afectarán los riesgos que aún quedan al trabajo en curso. Toda la retroalimentación de la iteración pasada permite reajustar los objetivos para las siguientes iteraciones. Se continúa con esta dinámica hasta que se haya finalizado por completo con la versión actual del producto.

2.2.6 Elementos del RUP

El proceso RUP consta de cuatro elementos para su modelado, los cuales son:

1. Roles
2. Actividades
3. Artefactos
4. Flujos de trabajo

Figura No: II-4 Elementos del RUP [9]

ROLES – ACTORES

Un rol define el comportamiento y las responsabilidades de una persona trabajando en el proyecto. Estos actores se dividen en varias categorías: Analistas, Desarrolladores, Probadores, Encargados, Otros actores. A continuación se presenta una lista de actores de acorde a las categorías mencionadas con anterioridad:

Analistas

- Analista del Proceso del Negocio.
- Diseñador del Negocio.
- Revisor del Modelo del Negocio.
- Revisor de Requerimientos.
- Analista del Sistema.
- Especificador de Casos de Uso.
- Diseñador de Interfaz del Usuario.

Desarrolladores

- Arquitecto.
- Revisor de la Arquitectura.
- Diseñador de Cápsulas.
- Revisor del Código y Revisor del Diseño.
- Diseñador de la Base de Datos.
- Diseñador.
- Implementador y un Integrador.

Probadores Profesionales

- Diseñador de Pruebas.
- Probador.

Encargados

- Encargado de Control del Cambio.
- Encargado de la Configuración.
- Encargado del Despliegue.
- Ingeniero de Procesos.
- Encargado de Proyecto.
- Revisor de Proyecto.

Otros

- Cualquier trabajador.
- Artista Gráfico.

- Stakeholder.[3]
- Administrador del Sistema.
- Escritor técnico.
- Especialista de Herramientas.

ACTIVIDADES

Una actividad representa una unidad de trabajo desempeñada por un determinado rol. El propósito de la actividad es crear artefactos.

ARTEFACTOS

Un artefacto es una pieza de información que se produce, modifica, o usa por un proceso. Es el producto tangible del proyecto que puede ser:

- Un modelo, tal como Modelo de Casos de Uso
- Un documento, tal como el documento de la Arquitectura del Sistema,
- Una Pieza Hardware o un Programa Ejecutable del Sistema.

FLUJOS DE TRABAJO

Un flujo de trabajo es una secuencia de actividades que produce resultados observables. Se representan dos tipos de flujos de trabajo:

[3] Stakeholder: Se refiere a quienes pueden afectar o ser afectados por las actividades de una empresa.

a) **Flujo de Trabajo Principal:** Es una colección de actividades relacionadas, que representa un componente del proceso de desarrollo RUP.

b) **Flujo de Trabajo Detallado:** Representan el desglosamiento de las actividades que se muestran en el flujo de trabajo principal en otro grupo de actividades más pequeñas que interactúan con roles y artefactos.

2.3 PHP

2.3.1 Definición PHP

PHP (Hypertext Pre-processor) es un lenguaje de programación creado en 1995 por Ramsu Lerdford, con una gran librería de funciones y mucha documentación. Este se ejecuta desde un servidor web, generando una pagina html para ser usada por un usuario.

PHP necesita de un servidor web que este en la máquina que se va a trabajar, y así mismo instalar el interprete de PHP par todo esto se pueden usar herramientas como XAMPP.

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 Microsoft Windows, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

2.3.2 Ventajas de PHP

El 13 de julio de 2004, fue lanzado PHP 5, utilizando el motor Zend Engine 2.0 (o Zend Engine 2). Incluye todas las ventajas que provee el nuevo Zend Engine 2 como:

- Mejor soporte para la programación orientada a objetos, que en versiones anteriores era extremadamente rudimentario.
- Mejoras de rendimiento.
- Mejor soporte para SQL con extensión completamente reescrita.
- Mejor soporte a XML (XPath, DOM, etc.).
- Soporte nativo para SQLite.
- Soporte integrado para SOAP.
- Iteradores de datos.
- Manejo de excepciones.
- Mejoras con la implementación con Oracle.

2.3.3 Características

Una de las funcionalidades principales de PHP es crear páginas dinámicas y fáciles de usar para el cliente. Estas pueden trabajar conectadas a una base de datos que pueden ser cualquiera de las conocidas, conexiones en red, y así poder crear lo que verá el cliente al final. Este solo recibe una página HTML resultante de la ejecución de la PHP, la misma que es compatible con la mayoría de navegadores; entre las principales características de PHP están:

- Multiplataforma.
- Manejo de excepciones.

- Posee una biblioteca nativa de funciones.
- Permite técnicas de programación orientada a objetos.
- Amplia documentación en su página oficial y en internet en general.
- Fácil conectividad con MySQL.
- Es software libre.
- Seguridad.

Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar (muchos otros lenguajes tampoco lo hacen), aun estando dirigido a alguna en particular, el programador puede aplicar en su trabajo cualquier técnica de programación y/o desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del Patrón de diseño Modelo Vista Controlador o MVC, que permiten separar el tratamiento y acceso a los Datos, la Lógica de control y la Interfaz de usuario en tres componentes independientes.

2.3.4 Seguridad

PHP ha sido diseñado específicamente para ser un lenguaje seguro de escribir programas y con la correcta selección de las opciones de configuración de tiempo de compilación y ejecución se consigue la exacta combinación de libertad y seguridad que se necesita. Ya que existen diferentes modos de utilizar PHP, existe también una multitud de opciones de configuración que permiten controlar su funcionamiento. Una gran selección de opciones garantiza que se pueda usar PHP para diferentes aplicaciones, pero también significa que existen combinaciones de estas opciones y configuraciones del servidor que producen instalaciones inseguras.

CAPITULO III: SITUACION ACTUAL

En este capítulo se hará la presentación general del Hospital Gíneco-Obstetra "Isidro Ayora", los objetivos tanto generales como específicos que tenemos nosotras para con el mismo, así como la misión y visión del Hospital.

También se hará referencia a los requerimientos funcionales y no funcionales que el Hospital necesita en el sistema y los que creemos que serán necesarios para la implementación del mismo.

3.1 Objetivos

3.1.1 General

Elaborar una aplicación la cual servirá de ayuda al Hospital Gíneco-Obstetra "Isidro Ayora" para automatizar el manejo de información del personal que labora en el mismo.

3.1.2 Específicos

- Implementar una aplicación que sea accesible para los responsables de la administración de la información del personal que labora en el Hospital Gíneco-Obstetra "Isidro Ayora".
- Ayudar a la institución en el correcto uso de la información de su personal.
- Usar las metodologías aprendidas para el funcionamiento de la aplicación para de esta manera servir a la comunidad en general.
- Construir pruebas de la aplicación en el hospital Gíneco-Obstetra "Isidro Ayora".
- Utilizar Patrones de Diseño en la construcción de la Aplicación.

3.2 Misión del Hospital Gíneco-Obstetra "Isidro Ayora"

Atender oportuna y permanentemente a los/as usuarios/as e internos/as en las especialidades de ginecología, obstetricia y neonatología, precautelando la salud sexual y reproductivas de mujeres y recién nacidos, mediante actividades de promoción, prevención, atención y rehabilitación de la salud; en un ambiente físico y humano digno y adecuado; con equidad, respeto, integridad y en el marco de los derechos.

Participar en investigaciones, así como en docencia y formación de talentos humanos con calidad técnico-científica acordes a las necesidades de los/as usuarios/as y del país.

3.3 Visión del Hospital Gíneco-Obstetra "Isidro Ayora"

Para el año 2014 somos un hospital de tercer nivel con atención especializada y personalizada en Ginecología, Obstetricia, Neonatología, como parte del sistema de referencia y contra referencia nacional.

Líderes en calidad de la atención, docencia e investigación.

Talento humano comprometido, con competencias sociales y técnicas, que trabajan en equipo para brindar atención integral y trato digno a los/as usuarios/as internos /as y externos/as, con enfoque de derechos

3.4 Antecedentes del Hospital Gíneco-Obstetra "Isidro Ayora"

Al momento, el Hospital Gíneco-Obstétrico "Isidro Ayora" de Quito (HGOIA) es un Hospital de Especialidad, en donde se brinda atención a madres y recién nacidos/as (RN) de todo el país. El hospital cuenta con más o menos 800 personas entre personal administrativo, médicos y enfermeras, las cuales brindan sus servicios y atención en las áreas de administración, obstetricia, ginecología, pediatría y odontología, para madres adultas, adolescentes y sus recién nacidos/as.

3.5 Información de la Gestión del Proyecto

3.5.1 Antecedentes:

Debido a la cantidad de información que maneja la maternidad y a que la misma es administrada manualmente, se ha visto la necesidad de proporcionar una aplicación que ayude llevar una mejor administración de dicha información y así sea más útil y tenga un fácil acceso. De la misma manera la institución se beneficiará automatizando sus procesos, lo que es importante en la actualidad.

3.5.2 Estrategia:

En esta fase se creará las clases básicas del proyecto:

- Técnicos
- Usuarios
- Empleados

Cada clase tiene operaciones de ingreso, modificación y eliminación respectivamente.

Igualmente se crearán las clases que tiene como fin modelar los procesos, que son:

- Manejar Contratos
- Administración de Horas Extras
- Gestión de Acciones Personal
- Administración de Ausencias
- Administración de Vacaciones

Además se realizará la Administración de Contratación que tiene como objetivo hacer reportes del Personal, Consulta de Horas Extras y Consulta de Acciones de Personal que se encuentran en la base de datos.

3.5.3 Requerimientos Funcionales

Por razones de espacio se presenta solo el Requerimiento F2, los demás Requerimientos están expuestos en el Manual Técnico dentro de la carpeta en el CD.

- **F2: Manejo de Contratos**

  El sistema será capaz de manejar los contratos del personal que labora en el Hospital.

  - ✓ **F2.1: Crear Contrato**

    El sistema será capaz de crear un nuevo contrato para un nuevo empleado.

  - ✓ **F2.2: Modificar Contrato**

    El sistema será capaz de modificar los datos de un contrato.

  - ✓ **F2.3: Buscar Contrato**

    El sistema será capaz de buscar y mostrar los datos de un contrato.

  - ✓ **F2.4: Anular Contrato**

    El sistema será capaz de anular el contrato de un empleado eliminándolo al mismo de la base.

3.5.4 Requerimientos No Funcionales

- Solo tendrá un Administrador que podrá hacer lo mismo que los otros usuarios con la única diferencia de poder crear nuevos usuarios.
- Tendrá un nombre y un logo creado por las programadoras
- El entorno del sistema será Web

CAPITULO IV: DISEÑO DE STAFFLOG SOFTWARE

En este capítulo verán ejemplos de los diagramas de Casos de Uso, de Clases, de Actividad y los de Secuencia de acuerdo a las diferentes administraciones ya detalladas en el capítulo anterior que servirán de apoyo para la implementación del sistema.

Finalmente se mostrará cómo está implementado STAFFLOG Software indicando dónde y cómo se usaron los Patrones de Diseño.

4.1 Casos de Uso

Los casos de uso son una representación gráfica del comportamiento de un sistema desde el punto de vista de un usuario, en estos se determinan los requisitos funcionales del sistema, es decir, representan las funcionalidades que el sistema puede ejecutar y así facilitan su interpretación.

4.1.1 Diagrama General

En el diagrama general se puede visualizar las funcionalidades que tendrá el sistema y de qué manera interactuarán los usuarios con cada uno de los módulos de STAFFLOG Software.

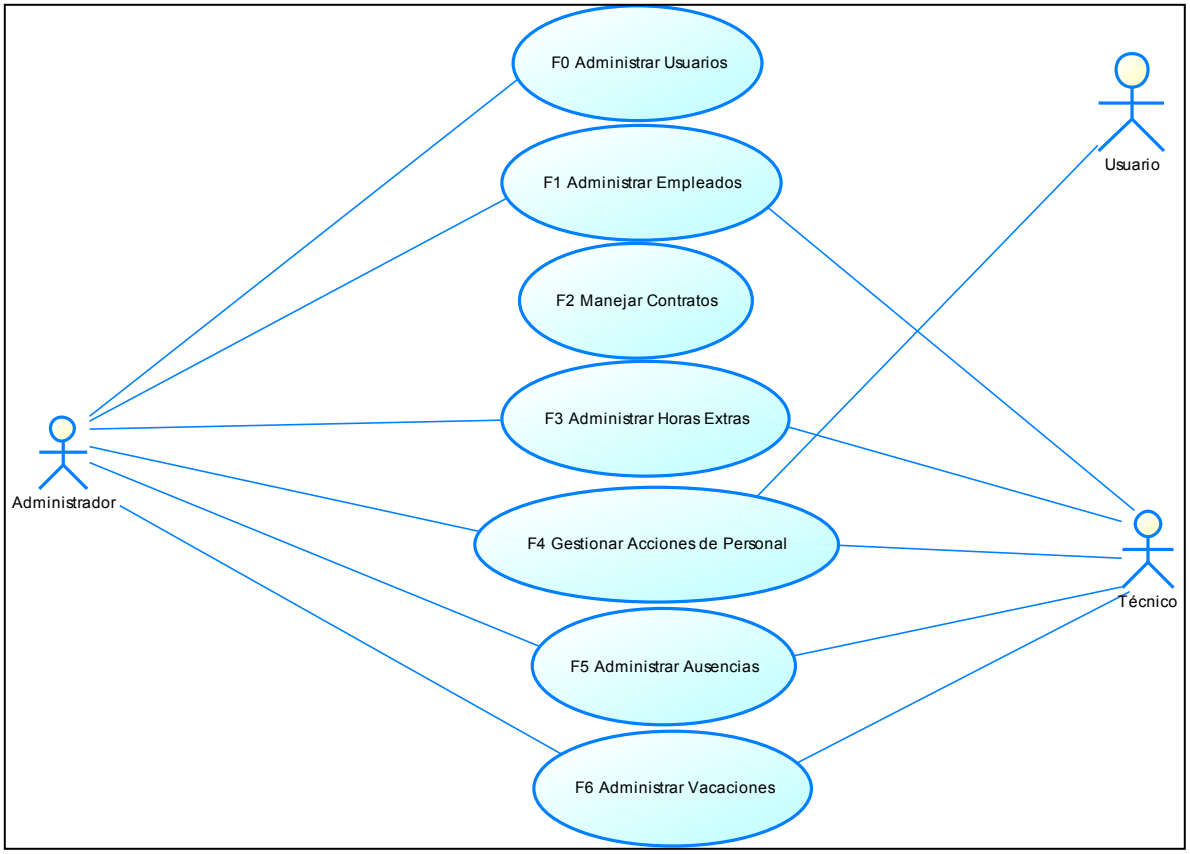

Figura No: IV-1 Diagrama Genera de Casos de Uso [A]

Por Motivos de Espacio solo presentaremos el módulo de Manejo de Contratos, los demás módulos están completos en el Manual Técnico dentro de la Carpeta Anexos en el CD.

4.1.2 F2: Manejo de Contratos

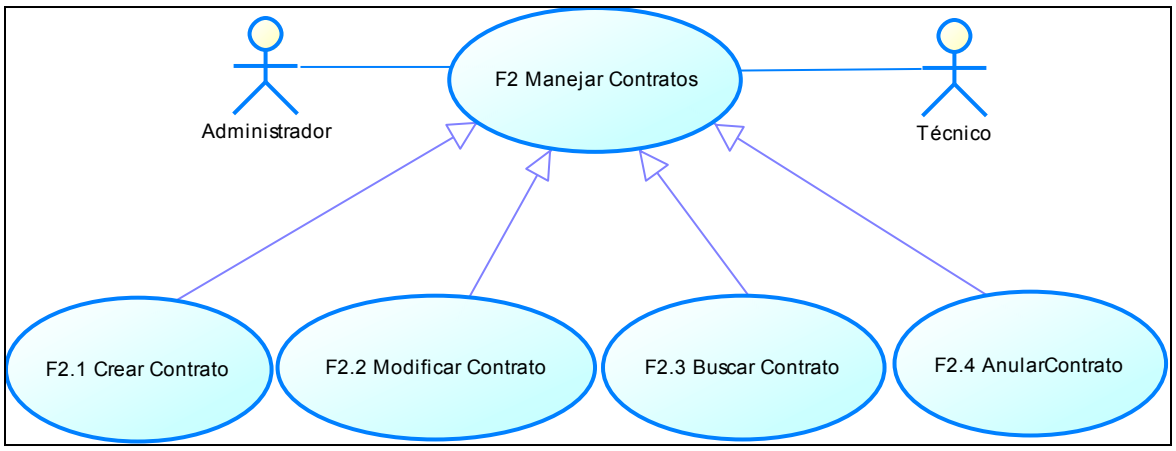

Figura No: IV-2 Diagrama de Casos de Uso F2 [A]

- **F2.1: Crear Contrato**

  Actor: Administrador, Técnico

  Descripción: El Actor tendrá la capacidad de crear un Contrato utilizando los datos de un empleado dentro de la base de datos.

  Flujo Principal:

  1. *El Actor* selecciona Generar Contratos del Menú de Creación.
  2. *El Sistema* presenta la ventana para Ingresar un código.
  3. *El Actor ingresa el código del Empleado*.
  4. *El Actor* presiona Buscar.
  5. *El Sistema* verifica el Empleado.(E1)

6. *El Sistema* abre una ventana mostrando los campos que se requieren para ingresar un contrato nuevo.
7. *El Actor* llena los campos del nuevo contrato.
8. *El Actor* presiona Insertar.
9. *El Sistema* genera el contrato en Microsoft Word.(E2)
10. *El Sistema* almacena la información. (E1)

Flujo Alterno:

6. El Sistema ejecuta Caso de Uso F2.2, F2.3 ó F2.4.

Excepciones:

E1. Error en la Base de Datos.
E2. No tienen instalado Microsoft Word.

- **F2.2: Modificar Contrato**

Actor: Administrador, Técnico

Descripción: El Actor tendrá la capacidad de modificar un Contrato.

Flujo Principal:

1. *El Actor* selecciona Contratos del Menú de Edición.
2. *El Sistema* presenta la ventana para Ingresar un código.
3. *El Actor ingresa el código del Empleado*.
4. *El Actor* presiona Buscar.
5. *El Sistema* verifica el Empleado.(E1)
6. *El Sistema* presenta los campos del contrato.
7. *El Actor* llena los campos del contrato.
8. *El Actor* presiona Actualizar.
9. *El Sistema* genera el contrato en Microsoft Word.(E2)

10. *El Sistema* almacena la información. (E1)

Flujo Alterno:

6. El Sistema ejecuta Caso de Uso F2.1, F2.3 ó F2.4.

Excepciones:

E1. Error en la Base de Datos
E2. No tienen instalado Microsoft Word

- **F2.3: Buscar Contrato**

Actor: Administrador, Técnico

Descripción: El Actor tendrá la capacidad de buscar un Contrato utilizando el código de un empleado dentro de la base de datos.

Flujo Principal:

1. *El Actor* selecciona Contratos del Menú de Consultas.
2. *El Sistema* presenta la ventana para Ingresar un código.
3. *El Actor ingresa el código del Empleado*.
4. *El Actor* presiona Buscar.
5. *El Sistema* verifica el Empleado.(E1)
6. *El Sistema* presenta los campos del contrato.

Flujo Alterno:

6. El Sistema ejecuta Caso de Uso F2.1, F2.2 ó F2.4.

Excepciones:

E1. Error en conexión a Base de Datos

- **F2.4: Anular Contrato**

Actor: Administrador, Tecnico

Descripción: El Administrador tendrá la capacidad de anular un Contrato utilizando el código de un empleado dentro de la base de datos.

Flujo Principal:

1. *El Actor* selecciona Anular Contratos del Menú de Eliminar.
2. *El Sistema* presenta la ventana para Ingresar un código.
3. *El Actor ingresa el código del Empleado*.
4. *El Actor* presiona Buscar.
5. *El Sistema* verifica el Empleado.(E1)
6. *El Sistema* presenta los campos del contrato.
7. *El Actor* presiona Eliminar.
8. *El Sistema* elimina la información.

Flujo Alterno:

6 .El Sistema ejecuta Caso de Uso F2.1, F2.2 ó F2.3

Excepciones:

E1. Error en la BDD

4.1.3 Diagrama de Actividades de Manejo de Contratos

El diagrama de actividades representa gráficamente los flujos de trabajo paso a paso del Manejo de Contratos, empezando por quien solicita la generación del nuevo contrato y cómo termina el proceso.

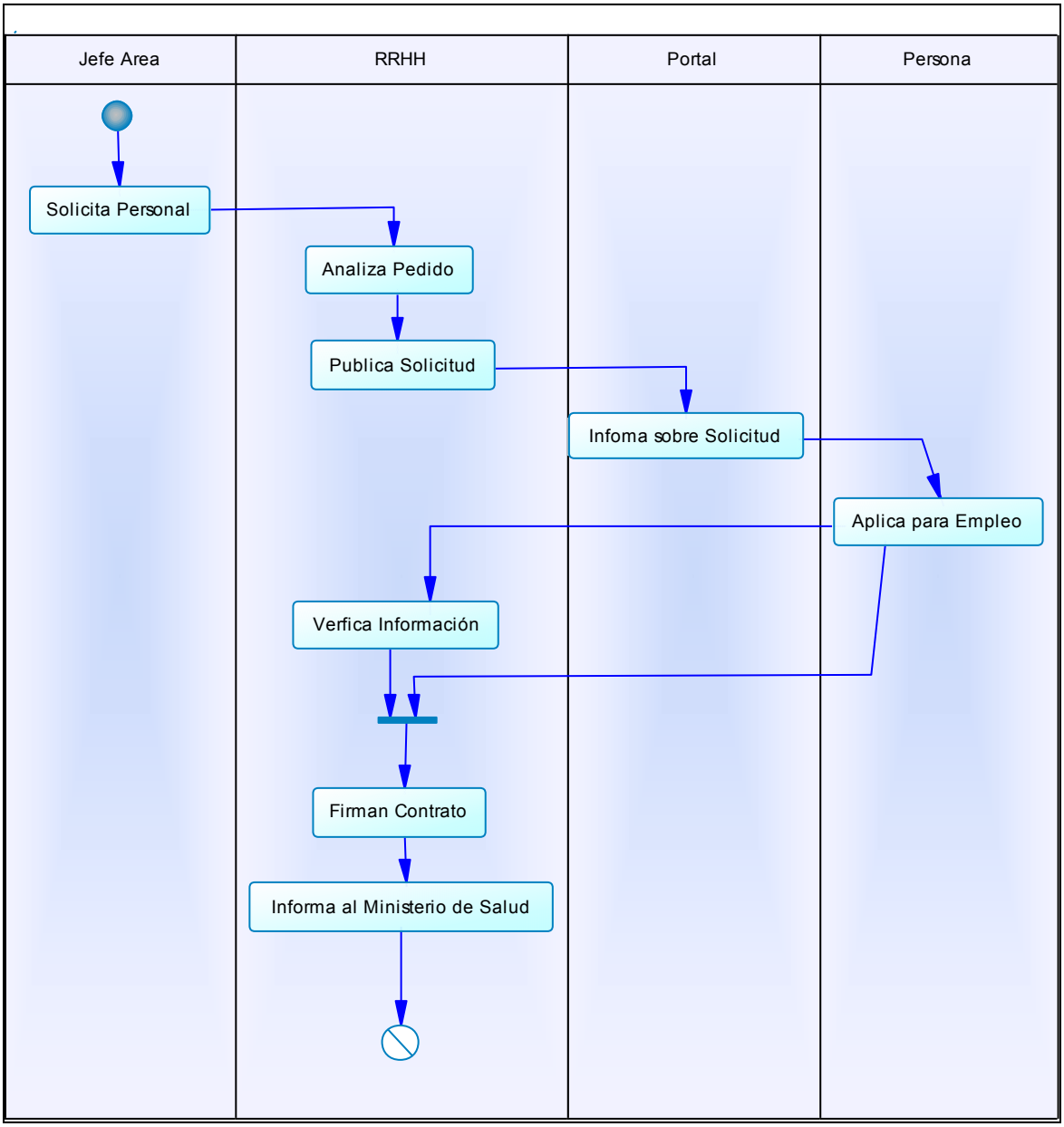

Figura No: IV-3 Diagrama Actividades del Manejo de Contratos [A]

4.2 Diagrama de Clases General

El diagrama de clases describe la estructura del sistema mostrando sus clases, atributos y las relaciones entre ellos.

Como se puede observar todas las clases están asociadas a la clase Empleado de la Siguiente manera:

El usuario Maneja los Empleados, además el usuario tiene una Categoría, un Empleado tiene un Contrato, un Empleado puede tener varias Horas Extra, varias Acciones de Personal, Varias Ausencias o varios Días de Vacación.

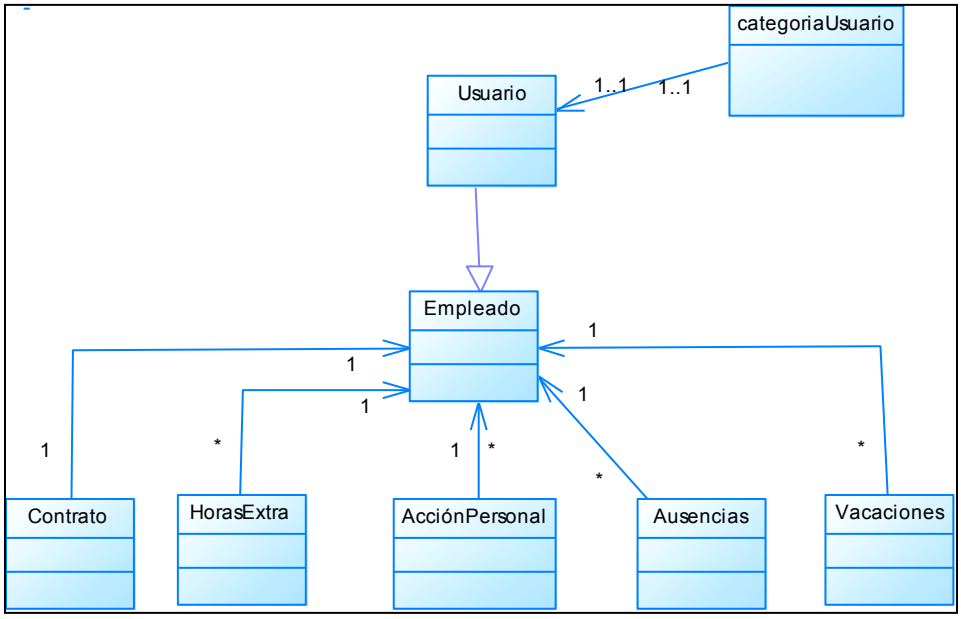

Figura No: IV-4 Diagrama Genera de Clases [A]

4.2.1 Diagrama de Clases con Atributos

En estos se mostrará la Clase Contrato con cada uno de sus atributos y principales métodos. Las demás clases se las podrá visualizar el Manual Técnico dentro de la Carpeta Anexos en el CD.

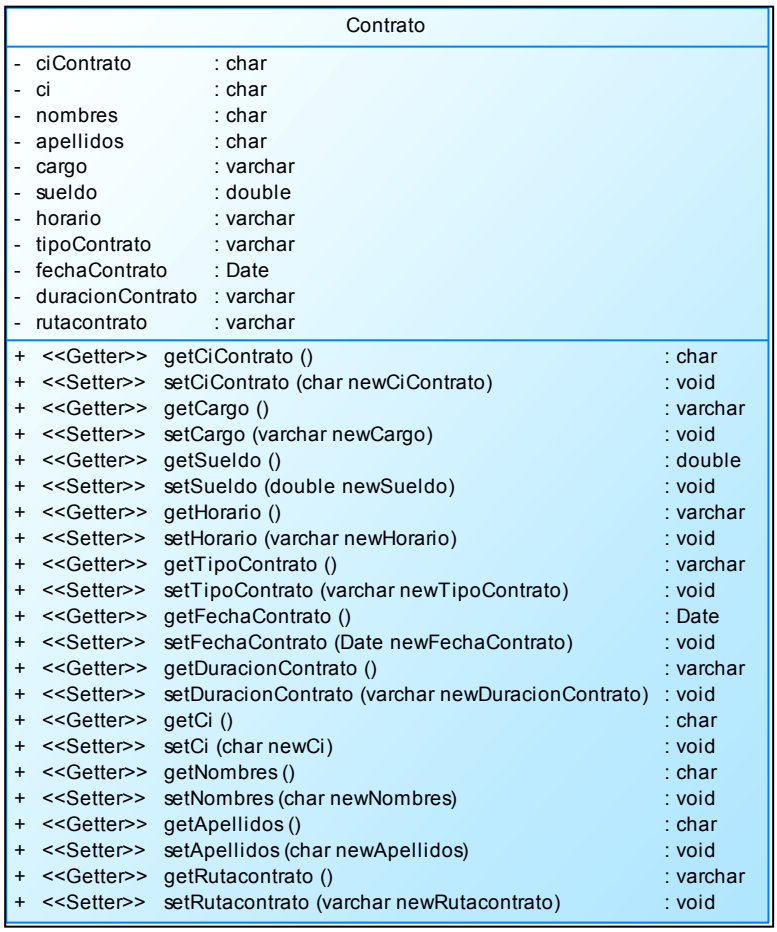

Figura No: IV-5 Diagrama de la Clase Contrato [A]

4.3 Diagrama de Secuencia

Los diagramas de Secuencia muestran la interacción de los objetos durante su ciclo de vida. Especificando de mejor manera los casos de uso para así comprender como interactuarán el Usuario y el Sistema desde que se ingresa a la Aplicación para generar un nuevo Contrato. Los Diagramas de Secuencia del resto de los Casos de Uso están el Manual Técnico dentro de la carpeta Anexos del CD.

4.3.1 Caso de uso: Crear Contrato

Identificador: F2.1

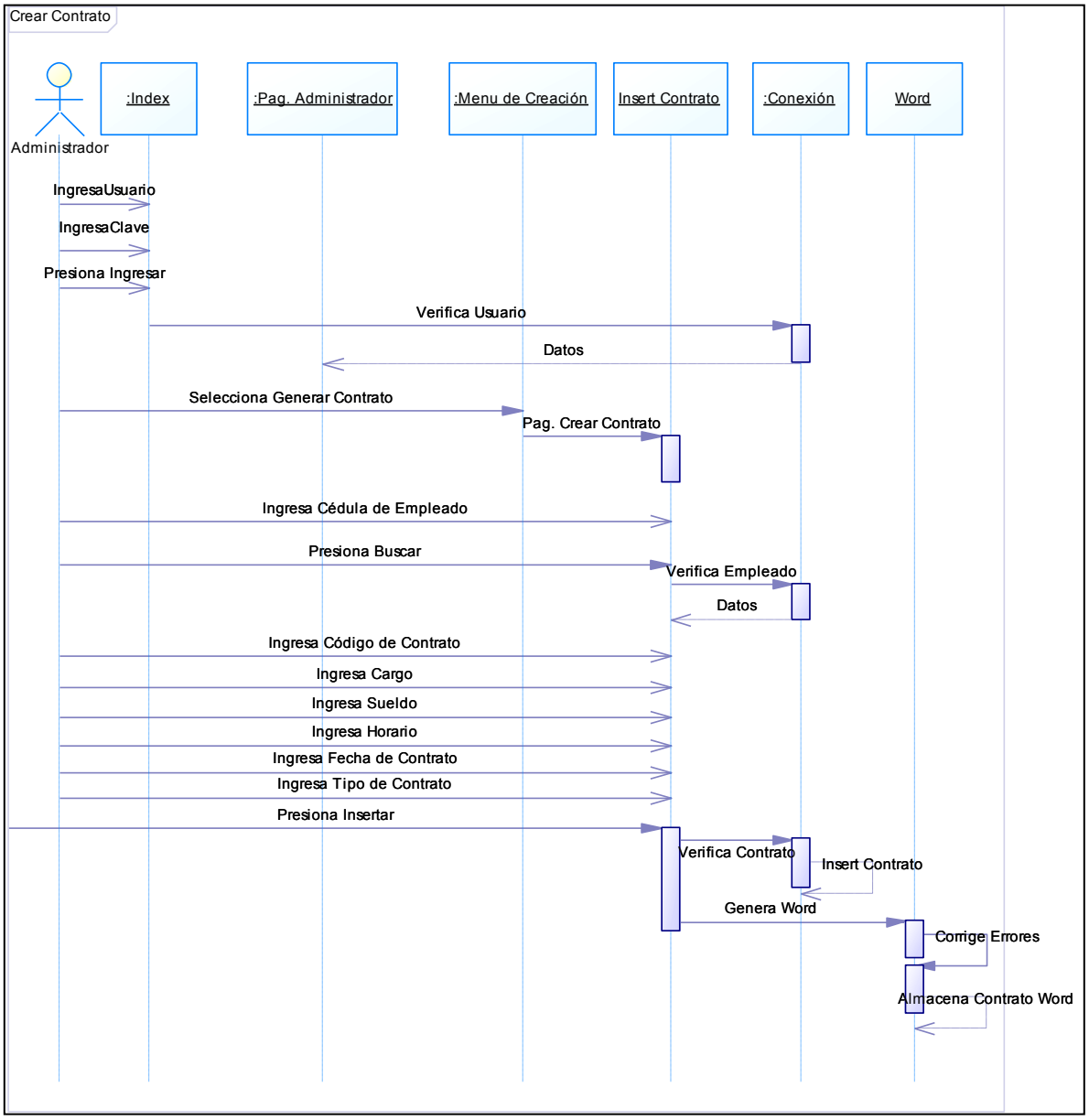

Figura No: IV-6Diagrama de Secuencia de la Generación de Contratos [A]

4.4 Implementación

En esta parte vamos a revisar la implementación del módulo Usuarios.

En todos los módulos de la aplicación se verá reflejada la implementación del Patrón de Diseño Fachada el cual ayudará a que las variables que se pasen de página a página no sean visibles al usuario llamando dentro de una interfaz visible a otra escondida que será la que realice las operaciones. Es decir la interfaz Fachada llamada creusuario.php llamará a la interfaz subsistema insertusu.php la cual hará las operaciones necesarias para crear o modificar usuarios. La llamada al subsistema se puede observar en la siguiente Figura.

```
<?php //aquí se hace la llamada al php que hará el trabajo de insertar o modificar el usuario ¿>
        <iframe frameborder="0" scrolling="auto" height="550" width="900" src="../insertusu.php" align="center"></iframe>
        <p> </p>
    </div>
```

Figura No: IV-7 Aplicación del Patrón Fachada [A]

En crear y modificar Usuario se visualizará el uso del Patrón de Diseño Estrategia al hacer el ingreso o modificación del usuario, para realizar cualquiera de estas dos acciones se presentara la misma interfaz y el usuario tendrá que ingresar los datos y la aplicación decidirá si hay que ingresar un nuevo usuario o simplemente modificarlo.

```
//identifica si el Id de usuario ya esta en la base o no (esto es para el patron estrategia)
mysql_select_db($database_miconexion, $miconexion);
$consulta="select * from usuario where IDUSUARIO =".$_POST['IDUSUARIO'];
$resultado=mysql_query($consulta) or die ("ERROR1!!!");
//si es usuario está en la base actualiza el registro con los nuevos valores
if (mysql_num_rows($resultado)>0)
{
$updateSQL = "UPDATE usuario SET USERUSUARIO = '".$_POST['USERUSUARIO']."', PASSWORDUSUARIO = '".$_POST['PASSUSU']."' WHERE IDUSUARIO =".$_POST['IDUSUARIO'];
mysql_select_db($database_miconexion, $miconexion);
$Result1 = mysql_query($updateSQL, $miconexion) or die(mysql_error());

$updateSQL = "UPDATE categoriausuario SET DETALLECATEGORIA = '".$_POST['cat']."' WHERE IDUSUARIO =".$_POST['IDUSUARIO'];
mysql_select_db($database_miconexion, $miconexion);
$Result1 = mysql_query($updateSQL, $miconexion) or die(mysql_error());

//si el usuario no esta en la base agrega un usuario nuevo
}else {$insertSQL = sprintf("INSERT INTO usuario (IDUSUARIO, USERUSUARIO, PASSWORDUSUARIO) VALUES (%s, %s, %s)",
GetSQLValueString($_POST['IDUSUARIO'], "text"),
GetSQLValueString($_POST['USERUSUARIO'], "text"),
GetSQLValueString($_POST['PASSUSU'], "text"));

mysql_select_db($database_miconexion, $miconexion);
$Result1 = mysql_query($insertSQL, $miconexion) or die("ERROR2!!");

$insertSQL = sprintf("INSERT INTO categoriausuario (IDUSUARIO, DETALLECATEGORIA) VALUES (%s, %s)",
GetSQLValueString($_POST['IDUSUARIO'], "text"),
GetSQLValueString($_POST['cat'], "text"));

mysql_select_db($database_miconexion, $miconexion);
$Result1 = mysql_query($insertSQL, $miconexion) or die(mysql_error());
}
```

Figura No: IV-8 Aplicación del Patrón Estrategia [A]

Por motivos de espacio solo presentaremos una parte del código, el resto se encuentra en el CD, dentro de la carpeta STAFFLOG1.

El certificado de instalación de STAFFLOG Software en el Hospital Gíneco-Obstetra "Isidro Ayora" está en el Anexo 1 del presente Documento.

CAPITULO V: CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Se elaboró la aplicación STAFFLOG donde se usó los Patrones de Diseño Estrategia y Fachada y se concluye que el uso de Patrones debe usarse principalmente con lenguajes orientados a objetos ya que estos manejan Clases y ahí es donde en realidad se ve reflejado el uso de Patrones como ayuda al programador.

- Que a pesar de que PHP no es un lenguaje plenamente orientado a objetos, y los Patrones de Diseño trabajan con el uso de clases se pudo adaptar a su uso en el diseño de STAFFLOG Software aplicando los Patrones Estrategia y Fachada, demostrando que no estos pueden modificarse sin que cambie la esencia de su significado.

- Se realizaron pruebas con diferentes datos otorgados por el Hospital Gíneco-Obstetra "Isidro Ayora" los cuales se acoplaron con la aplicación y proporcionaron los resultados esperados, por lo que se concluye que STAFFLOG Software cumple con el objetivo de automatizar las tareas de manejo de la información de empleados.

- Que STAFFLOG Software es fácil de usar y completamente accesible para los usuarios del Hospital Gíneco-Obstetra "Isidro Ayora", minimizando tiempo de elaboración de procesos y mejorando la calidad de datos.

- Que los patrones de diseño pueden tener variaciones en su contexto ya que no todos los problemas que puedan surgir son iguales, además se puede trabajar en diferentes plataformas sin que el Patrón cambie su esencia.

- Al realizar este proyecto de Tesis podemos notar que muchas veces hemos usado patrones de diseño en nuestros programas y

aplicaciones sin haberlo notado, ya que estos se acoplan de acuerdo a las necesidades del programador.

- Antes de utilizar un Patrón de diseño se debe estudiar a fondo sus diferentes usos e implementaciones, ya que algunas veces puede resultar confuso.

- Que un ingeniero de Sistemas necesita tener una mente capaz de realizar análisis, observar cada detalle, saber administrar y aprovechar los recursos al máximo.

5.2 Recomendaciones

- STAFFLOG Software optimiza y agilita los principales procesos del departamento de Recursos Humanos por lo que se recomienda hacer el debido mantenimiento y actualización de los módulos.

- Se debe tener especial cuidado al manejar archivos desde PHP, ya que al ser aplicación Web los deberán ser descargados al computador cada vez que se los quiera visualizar desde la aplicación y se profundicen los conocimientos de las nuevas técnicas de programación dentro de la carrera.

- No se logró ubicar imágenes en la creación de los archivos WORD, por lo que es recomendable usar scripts y clases ya creadas para hacerlo.

- Para mejorar el uso de la aplicación es recomendable que el Hospital Gíneco-Obstetra "Isidro Ayora", invierta en la implantación de una intranet y hasta una extranet de esta forma no solo STAFFLOG Software sino otras aplicaciones futuras podrán prestar más y mejores beneficios.

BIBLIOGRAFIA

[A] Rivera, G., & Vega, G. (2013). Aplicación de Patrones de Diseño para la Contrucción de un Sistema de Administración de la Información del Personal que Labora en el Hospital Gineco Obtétrico Isidro Ayora. Quito.

[B] Erich Gamma, R. H. (1995). Design Patterns: Elements of Reusable Object-Oriented Software.

[C] Javier Nieto Diego, P. R. (s.f.). El Patrón Fachada. Salamaca.

[D] Kruchten, P. (2000). The Rational Unified Process: An Introduction.

[E] Roger, P. (1998). Ingenieria De Software: Un Enfoque Práctico. España: McGrawhill.

[F] Soto, O. A. (09 de Junio de 2009). Patrones de Diseño.

[G] VORA, P. (2009). Web Application DESIGN PATTERNS. Burlington: Elsevier.

[1] Adry. (s.f.). Adrysmodelos. Obtenido de Introducción a los Patrones de Diseño: http://adrysmodelos.blogspot.com/

[2] Balu. (19 de 07 de 2010). Baluarte.net 2.0. Recuperado el 10 de 05 de 2012, de

TECNOLOGIA, ENTRETENIMIENTO Y CULTIRA: http://www.baluart.net/articulo/introduccion-a-los-patrones-de-diseno-con-php

[3] Caytuiro, N. (21 de 02 de 2011). Slides Share. Recuperado el 20 de 05 de 2012, de http://www.slideshare.net/lapiedradigital/patrones-de-diseo-7006476

[4] Codigo Programación. (s.f.). Recuperado el Enero de 2012, de http://codigoprogramacion.com/curso-php/111-introduccion-a-php-y-que-es-php.html

[5] IBM. (s.f.). developerWorks. Recuperado el 2012, de http://www.ibm.com/developerworks/library/os-php-designptrns/

[6] Ingeniero Software. (s.f.). Recuperado el Enero de 2012, de http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php

[7] Jaramillo, D. E. (13 de 02 de 2009). Slides Share. Recuperado el 20 de 05 de 2012, de http://www.slideshare.net/2008PA2Info3/info-exposicion-temas-de-revision

[8] Kioskea.net. (16 de 10 de 2008). Recuperado el 02 de 06 de 2012, de http://es.kioskea.net/contents/genie-logiciel/design-patterns.php3

[9] Laredo, V. (s.f.). SISTEMAS DE INFORMACIÓN EN LA ADMINISTRACIÓN DE LOS NEGOCIOS. Obtenido de http://www.vlaredo.galeon.com/index2.htm

[10] osvaldonafi. (13 de Enero de 2010). Buenas Tareas. Recuperado el Enero de 2012, de http://www.buenastareas.com/ensayos/Introduccion-a-Rup/91607.html

[11] PHP. (s.f.). PHP. Recuperado el Enero de 2012, de http://www.php.net/manual/es/language.oop5.patterns.php

[12] SCRIBD. (s.f.). Recuperado el Enero de 2012, de http://www.scribd.com/doc/46891651/Tecnologia-Rup

[13] Tratando de Entenderlo. (2010). Recuperado el 2012, de http://tratandodeentenderlo.blogspot.com/2010/11/patrones-de-diseno-strategy.html

[14] V., K. Z. (s.f.). Patrones Estructurales y Algo más . Obtenido de http://patronestructuralesyalgomas.blogspot.com/2009/05/patron-facade.html

ANEXO 1