PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS

"IMPLEMENTACIÓN DE UNA SOLUCIÓN INFORMÁTICA DE MANEJO DE ESTADÍSTICAS, PARA EL LABORATORIO DE MICROBIOLOGÍA DEL HOSPITAL DEL IESS CARLOS ANDRADE MARÍN"

SHALOM DAVID MARTÍNEZ GRAJEDA

QUITO, 2015

Contenido

1. INTRODUCCION 4
1.1. Introducción 4
1.2. Datos de la organización o institución 4
1.3. Justificación 4
1.4. Antecedentes 5
1.5. Objetivos 5
1.5.1. Objetivo general 5
1.5.2. Objetivos específicos 5
1.6. Alcance 6
2. MARCO TEÓRICO 6
2.1. Metodología de desarrollo de software. 6
2.1.1. Introducción 6
2.1.2. Revisión de Metodologías. 6
2.2. Metodología de programación ágil Extreme Programming (XP). 40
2.2.1. Fases 42
2.2.2. Entregables. 46
2.3. Selección de Metodología para la disertación. 46
2.4. Arquitectura 47
2.4.1. Lenguaje del lado del servidor PHP. 49
2.4.2. Lenguaje del lado del cliente JavaScript. 50
2.4.3. Sistema de gestión de bases de datos. 50
2.4.4. Servidor Web Apache. 51
3. CASO DE ESTUDIO: Implementación de una solución informática de manejo de estadísticas, para el Laboratorio de Microbiología del Hospital del IESS Carlos Andrade Marín. 53
3.1. Planeación del proyecto: 53
3.1.1. Historias de usuario: 53
3.1.2. Plan de entrega 66
3.1.3. División de Iteraciones. 66
3.1.4. Velocidad del proyecto. 66
3.2. Diseño del proyecto 67
3.2.1. Diseños simples 67

3.2.2. Tarjetas CRC. 69
3.2.3. Diagrama de actividades 72
3.2.4. Diagrama conceptual 73
3.2.5. Diagrama entidad relación: 74
3.2.6. Prototipo 76
3.2.7. Glosario de términos. 77
3.2.8. Riesgos. 77
3.2.9. Refactorizar 78
3.3. Codificación del proyecto. 78
3.4. Pruebas del proyecto. 78
3.5. Diseño del proyecto 84
3.5.1. Diseños simples 84
3.5.2. Tarjetas CRC. 85
3.5.3. Diagrama conceptual 87
3.5.4. Diagrama entidad relación 88
3.6. Codificación del proyecto. 91
3.7. Pruebas del proyecto. 91
4. CONCLUSIONES Y RECOMENDACIONES 94
4.1. Conclusiones 94
4.2. Recomendaciones 95
5. BIBLIOGRAFIA 96

1. INTRODUCCION

1.1. Introducción

El presente proyecto de disertación de grado tiene como objetivo desarrollar una solución informática para el laboratorio clínico del Hospital Carlos Andrade Marín que permita manejar adecuadamente, tener un control y seguimiento de la información que actualmente se maneja, en cuanto al proceso de detección y seguimiento de la tuberculosis.

1.2. Datos de la organización o institución

El Instituto Ecuatoriano de Seguridad Social (IESS) es una entidad, cuya organización y funcionamiento se fundamenta en los principios de solidaridad, obligatoriedad, universalidad, equidad, eficiencia, subsidiariedad y suficiencia. Se encarga de aplicar el Sistema del Seguro General Obligatorio que forma parte del sistema nacional de Seguridad Social

El Hospital del IESS "Carlos Andrade Marín", es la unidad de mayor complejidad de la red de servicios de salud del IESS.

Está regida por políticas y normas dictaminadas por el Consejo Directivo sobre la base de los principios del IESS, para brindar atención médica integral, ética, actualizada y especializada, mediante la utilización de tecnología de punta y capacitación continua en beneficio de la calidad de atención y la satisfacción de las necesidades de sus afiliados y usuarios.

La visión del Hospital Carlos Andrade Marín es ser un hospital líder en la atención de salud a nivel nacional e internacional.

Con autonomía económica y administrativa, enmarcado en los principios legales del Instituto Ecuatoriano de Seguridad Social, IESS, con un modelo de organización dentro del Sistema de Salud, para prestar asistencia integral y especializada.

Lograr excelencia en sus servicios para satisfacer y superar necesidades y expectativas del cliente con el uso racionalizado de los recursos disponibles.

El Hospital Carlos Andrade Marín se encuentra ubicado en las calles Portoviejo y Ayacucho de la ciudad de Quito, Ecuador.

1.3. Justificación

El laboratorio clínico del Hospital Carlos Andrade Marín es el encargado, a través de exámenes médicos de colaborar con el diagnóstico y pronóstico terapéutico de los afiliados al IESS. El servicio de laboratorio clínico atiende a 800 pacientes diarios en el hospital Carlos Andrade Marín y realiza análisis: Hematológicos, Bioquímicos, Inmunológicos, Hormonales, Microbiológicos, Genéticos, y Moleculares.

Los resultados de los exámenes y análisis se envían de manera directa a las computadoras de los médicos, dependiendo de los pedidos y solicitudes consideradas de manera urgente. Actualmente, existe la necesidad de un sistema informático para la obtención de estadísticas, en el laboratorio de Microbiología que permita mantener la información oportuna de sensibilidad y resistencia antimicrobiana de las muestras de los pacientes del hospital. Es por eso que se solicitó a la Pontificia Universidad Católica un sistema estadístico como tema de disertación de grado, el 30 de Abril del 2013 por el Dr. Fernando Romero Jefe de Servicio del laboratorio clínico, al día de hoy todavía existe la posibilidad de desarrollar este proyecto.

1.4. Antecedentes

El hospital Carlos Andrade Marín cuenta con un sistema que está desarrollado en AS/400 en el cual se administra toda la información del hospital. El laboratorio clínico tiene acceso a este sistema, además cuenta con una aplicación privada (Datalab) desarrollada por CLTech Colombia la cual fue proporcionada por ROCHE, empresa que se dedica a la importación y distribución de productos farmacéuticos, para el manejo de la información de los resultados de las muestras de los pacientes del hospital.

Actualmente, el laboratorio clínico utiliza Datalab, que entre sus principales funciones están el ingreso de órdenes de pacientes, realizar trazabilidad de la muestra, configuración de datos demográficos tanto para la historia como para la orden, configuración de grupos de usuarios, configuración del módulo de microbiología, diseño de informes personalizados, módulo de cajas, módulo de citas, 250 niveles de facturación. Estas funciones no logran satisfacer todas las necesidades del laboratorio clínico, debido a que se requiere una mayor flexibilidad en los reportes que se obtienen, más agilidad en el manejo del sistema y obtención de estadísticas. Al ser un sistema médico relacionada con la vida y salud humana es muy importante su adecuado funcionamiento.

1.5. Objetivos

1.5.1. Objetivo general

- Implementar una solución informática de manejo de estadísticas, para el Laboratorio de Microbiología del Hospital del IESS Carlos Andrade Marín.

1.5.2. Objetivos específicos

1. Analizar la situación actual del manejo de información relacionada al laboratorio clínico del hospital del IESS.
2. Analizar los requerimientos del usuario relacionado a los resultados estadísticos necesarios, de los exámenes del laboratorio clínico del hospital del IESS.
3. Seleccionar una metodología de diseño de software adecuada a la aplicación.
4. Diseñar la solución informática de acuerdo a los requerimientos.
5. Definir y seleccionar las herramientas de desarrollo de software.

6. Desarrollar la aplicación para realizar las consultas y cálculos estadísticos necesarios, alienados a los requerimientos funcionales del usuario.
7. Generar manuales técnico y de usuario
8. Describir conclusiones y recomendaciones.

1.6. Alcance

El presente proyecto de disertación de grado culminará con la entrega de la aplicación informática para el manejo de estadísticas, de los resultados del laboratorio clínico del hospital del IESS.

2. MARCO TEÓRICO

2.1. Metodología de desarrollo de software.

2.1.1. Introducción

En la actualidad el desarrollo de sistemas de información es de gran importancia, y se ha vuelto cada vez más complejo a través de los años. Es evidente que la programación y la gestión de proyectos son muy difíciles e incluso imposibles sin la utilización de una metodología de desarrollo de software.

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en la creación de sistemas de información. Podemos pensar en un marco de trabajo que puede ser extendido y adaptado para crear procesos más específicos de ingeniería de software. Existen una gran variedad de metodologías, creadas e introducidas por investigadores y empresas tanto a nivel comercial como académico las cuales nos permiten, estructurar, planificar y controlar el proceso de desarrollo de software, cada una con fortalezas y debilidades específicas, es por eso que no toda metodología es aplicable para cada proyecto, es importante seleccionar una metodología adecuada dependiendo del proyecto que se va a desarrollar.

2.1.2. Revisión de Metodologías.

- Metodologías Estructuradas:

Las primeras metodologías de desarrollo de software surgieron después de la crisis de software en 1970, junto con la programación estructurada, los lenguajes más famosos utilizados eran: Cobol, Fortran, C, Pascal y Modula 2. Los ingenieros de sistemas intentaron reducir las dificultades del desarrollo de software utilizando metodologías estructuradas. Como resultado se logró reducir el número de proyectos de software incompletos, el costo y tiempo de desarrollo. Las metodologías estructuradas son una primera aproximación al problema ya que pasan de una visión general del problema hasta llegar a un nivel de abstracción más sencillo. Están orientadas a procesos, se centran en especificar y descomponer la funcionalidad del sistema, también pueden estar orientados a datos o a ambos.

Metodologías estructuradas orientadas a procesos

- Se centra en la transformación de los datos de entrada para generar la salida esperada.

Metodologías estructuradas orientadas a datos

Estructuras de datos jerárquicas

- Se centran en las entradas y salidas; primero se definen las estructuras de datos y, a partir de éstas, se derivan los componentes procedimentales.

Estructuras de datos no jerárquicas

- Los tipos de datos son el corazón del sistema ya que son más estables que los procesos.

Mixtas

- Se enfocan tanto en el proceso como en los datos tomando desde diversos puntos de vista

Ejemplos de metodologías estructuradas:

Orientadas a procesos:

- De Marco
- Gane & Sarson
- Yourdon/Constantine

Orientadas a datos jerárquicos

- JSP y JSD
- LCP

Orientadas a datos no jerárquicos

- IE

Mixtas

- Merise
- SSADM
- Métrica

Las herramientas que son utilizadas por estas metodologías son diagramas de flujo de datos, especificaciones de procesos, diccionario de datos, diagramas de transición de estados, diagramas entidad relación, que pasamos a describir.

Diagramas de flujo:

Los diagramas de flujo de datos representan los pasos lógicos para realizar una tarea, mediante el uso de símbolos. Cada símbolo tiene su propio significado, en la Figura 1 podemos visualizar los componentes de un diagrama de flujo de acuerdo a la metodología de Yourdon, se utiliza un círculo para representar los procesos, un cuadrado para fuentes o destinos que pueden ser entidades, una flecha para representar una relación entre los componentes y dos líneas paralelas para representar un almacén de datos que puede ser una base de datos, un archivo, etc.

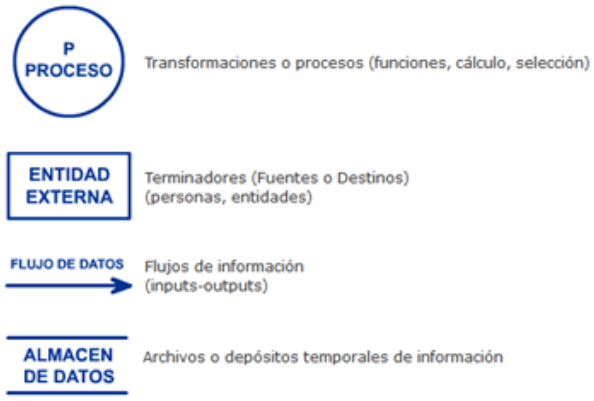


Figura 1: Componentes de un diagrama de flujo de datos metodología Yourdon
Fuente:http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060030/lecciones/Capitulo%204/dfd.htm

Un diagrama de flujo es un conjunto de símbolos que tienen relaciones entre sí, la Figura 2 es un claro ejemplo de un diagrama de flujo de un sistema de pedidos de libros en una librería, desarrollado utilizando la metodología Yourdon.

Figura 2: Diagrama de flujo de datos sistema de pedido de libros metodología Yourdon
Fuente:http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060030/lecciones/Capitulo%204/dfd.html

Especificaciones de procesos:

Las especificaciones de procesos definen que es lo que pasa dentro de un proceso, que es lo que se debe hacer para transformar las entradas del sistema en salidas. La forma más utilizada para representar las especificaciones de procesos es el lenguaje estructurado pero también se utiliza tablas de decisión y árboles de decisión.

Lenguaje Estructurado
Es un subconjunto del lenguaje que lo utilizamos para representar la programación, por ejemplo una expresión algebraica, una frase imperativa, o estructuras de programación como if, else, while, for, etc, pueden ser una frase de lenguaje estructurado.

Tablas de decisión:
Las tablas de decisión representan la toma de decisiones en un proceso, se representa como una matriz donde existen las alternativas y los estados del proceso, la Figura 3 es una representación de la estructura de una tabla de decisión.

Estados de la naturaleza					
Alternativas		e1	e2	…	en
	a1	X11	X12	…	x1n
	a2	X21	X22	…	x2n
	…	…	…	…	…
	an	Xm1	Xm2	…	Xmn

Figura 3: Estructura de una tabla de decisión
Fuente:http://www.slideboom.com/presentations/181016/Especificaci%C3%B3n-de-Procesos

Árboles de decisión:
Un árbol de decisión es una representación de los eventos que pueden surgir a partir de una decisión previa, tienen una representación gráfica como estructura de datos que imita la forma de un árbol. Si queremos encontrar la mejor opción para un conjunto de alternativas de forma visual y probabilística esta es una herramienta adecuada.

Se presenta a continuación la Figura 4 como ejemplo del proceso para conseguir una bodega, existen primero las opciones de construir o comprar una bodega, si se construye se puede conseguir un lote adecuado o no, en cambio si se compra puede ser que se ajuste al presupuesto que tenemos o no.

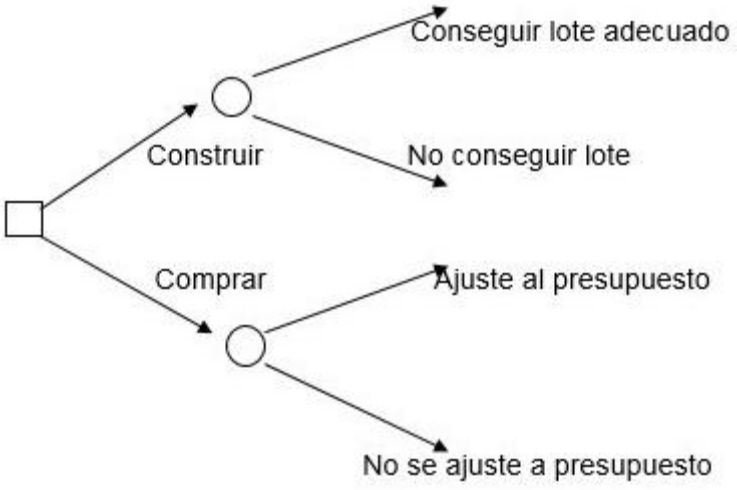


Figura 4: Ejemplo de un árbol de decisión.
Fuente:http://www.estructurayprogramacion.com/matematicas/matematicas-para-la-toma-de-decisiones/arboles-de-decision/

Diccionario de datos:
Un diccionario de datos es un conjunto de metadatos (datos de los datos) que se van a utilizar en el sistema. Incluye la clave única, el nombre del dato, el campo, tipo de dato, tamaño y una descripción.

Aquí se encuentran todos los elementos que forman parte del flujo de datos de todo el sistema, un ejemplo es la Figura 5 que representa un diccionario de datos para la entidad empleado registra cada dato y sus características.

Empleado					
Llave	Nombre	Campo	Tipo	Tamaño	Descripciòn
PK	Còdigo Empleado	cdep	Nùmerico	4	Almacena còdigo del empleado
	Nombres	nbep	Texto	50	Almacena nombre del empleado
	Apellidos	apep	Texto	50	Almacena apellidos del empleado
	Telèfono	tlep	Nùmerico	10	Almacena telèfono del empleado
	Gènero	gnep	Texto	10	Almacena Gènero del empleado
	E_mail	emep	Hipervìnculo	100	Almacena correo electrònico del empleado
	Direcciòn	drep	Memo	100	Almacena direcciòn del empleado
FK	Còdigo barrio	cdbrr	Nùmerico	4	Almacena còdigo del barrio
FK	Còdigo ciudad	cdcdd	Nùmerico	4	Almacena còdigo de la ciudad
FK	Còdigo tipo contrato	cdtc	Nùmerico	4	Almacena còdigo del tipo de contrato
FK	Còdigo localidad	cdl	Nùmerico	4	Almacena còdigo de la localidad

Figura 5: Ejemplo de un diccionario de datos
Fuente:http://evidenciasfase2estefanyperez.files.wordpress.com/2013/04/dic-empleado.png

Diagramas de transición de estados:

Los diagramas de estados representan el comportamiento de un sistema, se representa todos los estados posibles en los que puede estar un objeto específico y sus cambios a lo largo de su ciclo de vida. Los diagramas de estado están compuestos por: estados, eventos y transiciones.

Los estados se pueden dividir en:

Nombre	Descripción	Gráfico
INICIO	Estado inicial en el que se inicia el objeto en su ciclo de vida, ningún estado puede retornar al inicio, se representa por un punto negro.	
FIN	Es el estado final del ciclo de vida de un objeto, ningún evento puede sacar a un objeto de este estado, se representa por un circulo negro rodeado de otro circulo	
ESTADO	Son los diferentes estados en los que puede estar un objeto, salir o retornar.	

Los eventos son los que producen un cambio en el comportamiento del sistema en un momento en su evolución, por ejemplo el método de una clase.

Las transiciones son las líneas de comunicación que une un estado con otro, la representación gráfica es una flecha.

La Figura 6 es un diagrama de estados completo, se puede observar los estados de una orden de compra: inicio, activo, cancelado, procesado, reclamación, fin. Los eventos: crear objeto orden de compra, cancelar orden, procesar orden, verificar producto, reclamar orden, procesar orden, producto finalizado y orden rechazada. Las transiciones están representadas flechas, el objetivo de este diagrama es visualizar los estados en los que se encontrará una orden de compra.

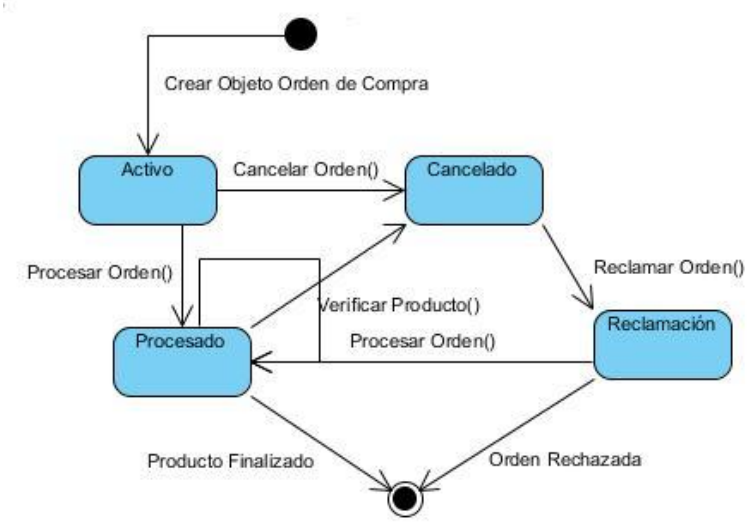


Figura 6: Diagrama de estados orden de compra. Fuente:http://www.elclubdelprogramador.com/2012/04/03/uml-diagrama-de-estados/

Diagramas entidad relación:

Es un modelo de datos que representa una abstracción del mundo real que consiste en un conjunto de objetos básicos llamados entidades y relaciones.
Una entidad es un objeto del mundo real sobre el cual queremos guardar una información. Las entidades poseen atributos, datos que definen al objeto, entre estos datos habrá un dato único al cual se le llama clave principal, y los datos referenciales a otras tablas se las conoce como claves foráneas.

Una relación es una asociación entre entidades, que refleja las interacciones existentes entre entidades pueden ser de tres tipos: relaciones 1-1, relaciones 1-n y relaciones n-n.

La Figura 7 es un diagrama entidad relación que representa un sistema de una cafetería que posee las entidades proveedores, compras, clientes, ventas, inventario con sus respectivos atributos y tiene relaciones entre ellas de 1-n o de n-1 por ejemplo la relación entre proveedores y compras nos indica que un proveedor puede ofrecer varios productos.

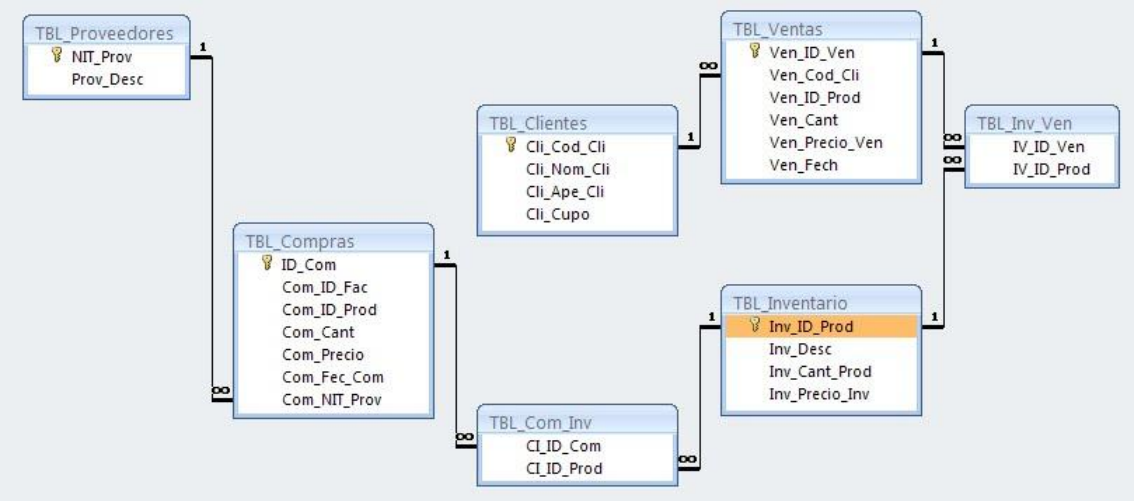


Figura 7: Diagrama entidad relación sistema de una cafetería
Fuente:http://grupodeanalisisydiseno.blogspot.com/2010/09/4-diagrama-entidadrelacion.html

Metodología de Yourdon

El análisis y diseño estructurado surgen en los años 70 de la programación estructurada, con el propósito de entender los requerimientos de software. El análisis estructurado se enfoca en las especificaciones del sistema y sus interfaces utilizando flujos de datos, cada función es representada como un proceso de datos que recibe datos del entorno, transforma los datos, lee o escribe los datos y da una respuesta en forma de datos al entorno. Durante 1980 y a principio de 1990 el análisis estructurado fue creciendo con otras técnicas de especificación de software como técnicas de modelado de información, es así que en 1990, Edward Yourdon público "Análisis Estructurado Moderno", esta es una metodología para el desarrollo de software, que proporciona una guía para el diseño paso a paso de sistemas y programas detallados.

Primero lo que se realiza es un análisis del sistema en el cual se dibuja un diagrama de contexto a nivel superior del problema indicando las fuentes, las salidas y límites del sistema. Después se entrevista a los diferentes usuarios y se realiza una lista de los eventos que el sistema debe responder. La documentación del sistema se completa con las técnicas tradicionales como el modelo de entidades, diagramas de flujo de datos, diagramas de estado.

Esta metodología realiza un gran esfuerzo por aplicar la teoría de varios modelos, se dice que los modelos son los fundamentos de Yourdon, un modelo trata de abstraer las características principales de un sistema para luego representar estas características de una manera útil.

Para el desarrollo de sistemas se utiliza un conjunto definido de herramientas de desarrollo, procedimientos, modelos y técnicas con los cuales se construye modelos de sistemas predominantemente gráficos.

La Figura 8 representa los modelos que se utilizan en la metodología de Yourdon para desarrollar un sistema:

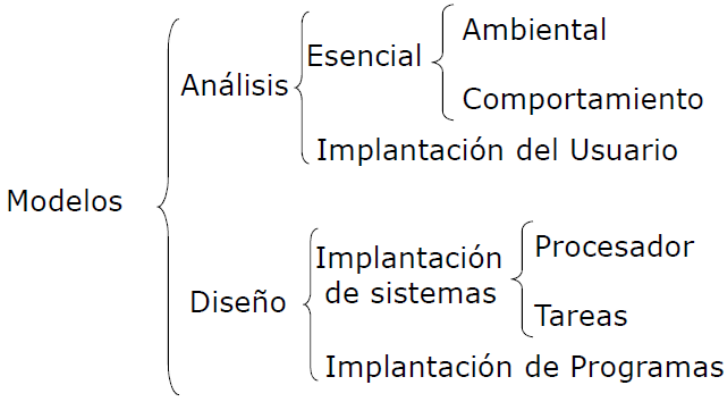


Figura 8: Modelos de la metodología Yourdon.
Fuente: Yourdon Edward, "Análisis estructurado moderno", 1989, ISBN: 0135986249

La metodología de Yourdon se divide en las siguientes etapas:

Análisis:

Para el proceso de análisis se realiza un modelo esencial. El modelo esencial es un modelo de lo que el sistema debe hacer para satisfacer los requerimientos de usuario, sin indicar cómo se implementará ya que esto es parte del diseño. El modelo esencial consiste de dos componentes principales: el modelo ambiental y el modelo de comportamiento.

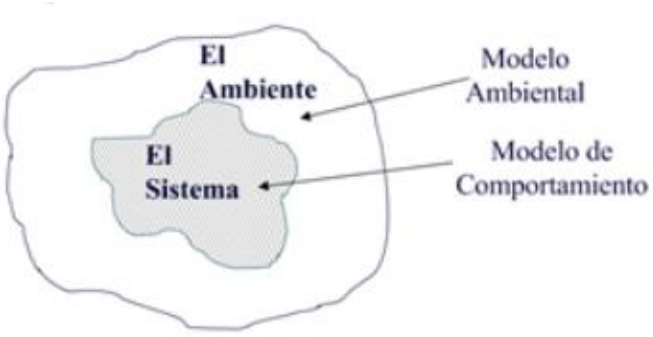


Figura 9: Estructura de un modelo esencial metodología Yourdon.
Fuente: Yourdon Edward, "Análisis estructurado moderno", 1989, ISBN: 0135986249

En la Figura 9 podemos observar la estructura de un modelo esencial, esta posee los modelos: ambiental, de comportamiento del sistema, y se encuentra dentro de un ambiente o entorno.

El modelo ambiental representa la frontera entre el sistema y el resto del mundo. La idea del modelo ambiental es especificar claramente qué es parte del sistema y qué no. El sistema que estamos desarrollando va a ser siempre parte de un sistema más grande. Aquí definimos las interfaces que permiten la interacción del sistema con el ambiente. Para modelar las interfaces entre el sistema y el ambiente exterior, es preciso identificar cuáles son los eventos que ocurren en el ambiente.

Un modelo ambiental consiste de tres elementos: una descripción breve del propósito del sistema, una lista de los eventos del sistema y un diagrama de contexto. La

declaración de propósito es un párrafo el cual nos indica cual es el propósito del sistema que vamos a desarrollar. El diagrama de contexto es un diagrama de flujo de datos que tiene un círculo central que representa el sistema, también posee a personas, y otros sistemas que tienen una relación con el sistema.

La Figura 10 es un ejemplo para un diagrama de contexto de un sistema de pedido de libros, podemos observar que la información de clientes, libros pedidos se guarda en la base de datos editorial, posee una relación con el módulo de administración donde se reportan las ventas, también una relación con el módulo de clientes al cual se envía la información de facturas y lista de envíos y se obtiene los pedidos, pedidos cancelados. Tiene una relación también con el módulo impresores donde se envían ordenes de reimpresión y se obtienen libros a bodega además se envían facturas al módulo de contabilidad que se guardan en la base de datos.

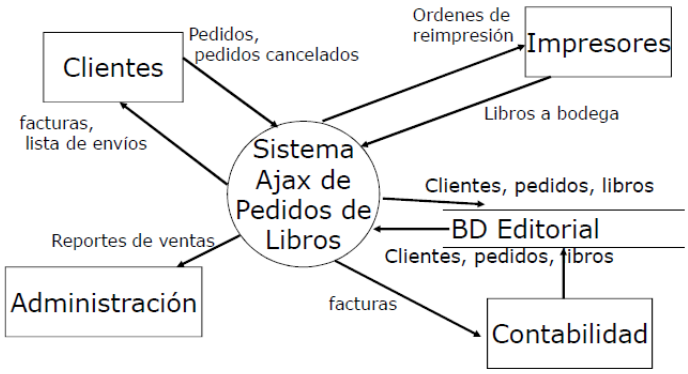


Figura 10: Ejemplo de diagrama de contexto metodología Yourdon.
Fuente: Yourdon Edward, "Análisis estructurado moderno", 1989, ISBN: 0135986249

La lista de acontecimientos indica los eventos que pueden ocurrir fuera del sistema y como responde el sistema ante estos eventos por ejemplo para un sistema de pedido de libros un cliente hace un pedido, cancela ese pedido, la administración pide un reporte de venta, o llega un pedido de reimpresión de un libro a la bodega estos serían los eventos del sistema.

Al definir los límites del sistema el usuario tendrá una idea del límite que tiene el sistema pero existe un área de la cual el usuario no está seguro, o no había tenido en cuenta y que está abierta para la negociación.

La siguiente Figura es una representación gráfica de un modelo esencial para un sistemas de cuentas por cobrar el cual posee una zona que no es conocida esta es la facturación y el control de inventarios del sistema.

Figura 11: Ejemplo de un diagrama esencial metodología Yourdon.
Fuente: Yourdon Edward, "Análisis estructurado moderno", 1989, ISBN: 0135986249

El modelo de comportamiento describe el comportamiento que el sistema debe tener para interactuar con el entorno. Consiste en la creación de diagramas de flujo de datos, un diagrama entidad relación, diagrama de transición de estados, un diccionario de datos y especificaciones de procesos. Se puede obtener mayor información en las páginas anteriores.

En el modelo de implementación de usuario se deben definir las interfaces del sistema con el medio ambiente que lo rodea, determinar cuáles son las metas que debe cumplir el sistema, identificar las actividades de soporte para el sistema, especificar cuáles son las restricciones operacionales.

Las restricciones operacionales son las decisiones que tomemos en cuanto a hardware, que características va a tener el servidor, que sistema operativo vamos a utilizar, puede ser windows, linux, entre otros y sus distribuciones, en que lenguaje de programación vamos a desarrollar la aplicación, etc.

Diseño:

El diseño del sistema involucra el desarrollo de una serie de modelos. El modelo más importante es el modelo de implantación de sistemas que se divide en un modelo de procesador y un modelo de tareas. En esta etapa el modelo esencial es utilizado como base para la creación de los modelos de procesador, modelo de tareas en la Figura 12 se puede visualizar la relación que existe entre estos modelos. El modelo esencial es incorporado al modelo de nivel de procesador este modelo se lo divide a nivel de tareas por cada procesador y cada tarea posee sus módulos del sistema.

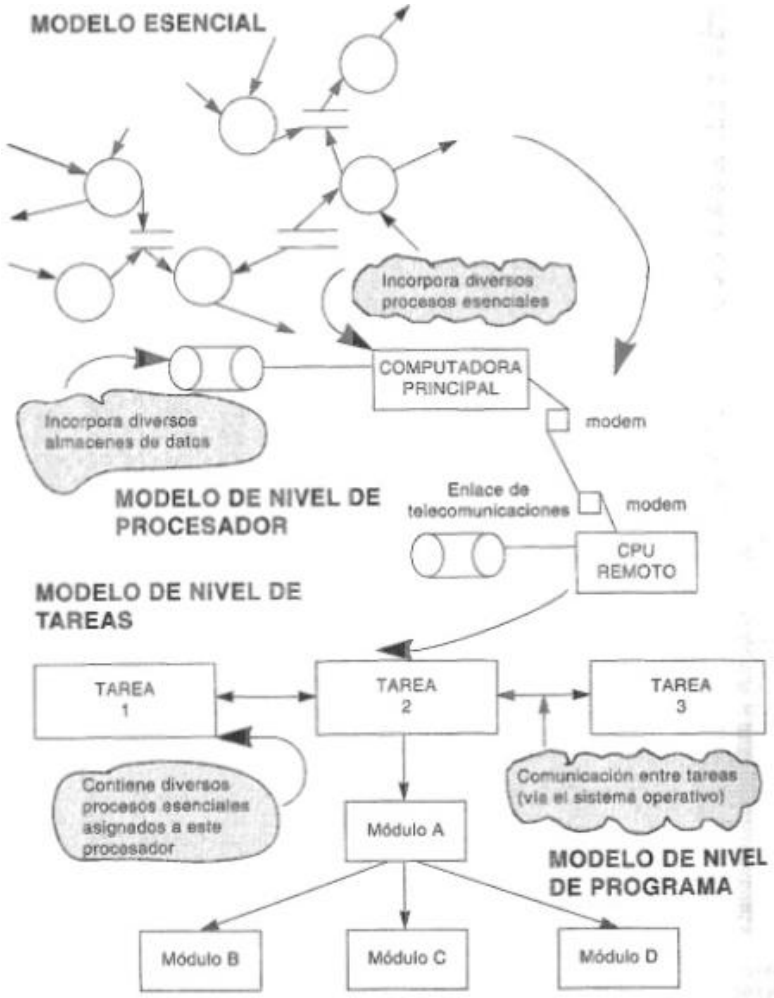


Figura 12: Modelos de análisis y diseño.
Fuente: Yourdon Edward, "Análisis estructurado moderno", 1989, ISBN: 0135986249

El modelo de procesador:

La primera tarea de la fase de diseño es decidir cómo se va a asignar el modelo esencial al hardware y software del sistema, en el nivel de modelo del procesador se trata de decidir cómo asignar el modelo esencial a los distintos procesadores y cómo deben comunicarse entre sí, entre las opciones que se pueden elegir están: asignar el modelo esencial a un solo procesador, esto se conoce como la solución de computadora principal. También se puede asignar a varios procesadores distintos, esto se conoce como la solución distribuida. Se puede escoger una combinación entre una solución distribuida y una solución de computadora principal para minimizar costos, maximizar confiabilidad, etc.

Los almacenes de datos también tienen que ser asignados a los componentes de hardware, se debe decidir si un almacén se realizará como una base de datos en un procesador. Si existe más de un procesador se debe especificar la comunicación entre procesadores, algunas de las opciones son: una conexión directa entre procesadores, por medio de un cable, canal o red local. Un enlace de telecomunicaciones entre procesadores, si los procesadores están separados físicamente por cientos de metros.

O un enlace indirecto entre procesadores en la cual se envían los datos de un procesador a otro por un medio físico.

El modelo de tareas:

Después de tener claro el modelo de procesador, el diseñador debe asignar procesos y almacenes a áreas individuales de cada procesador. Se divide el procesador en áreas separadas, donde cada una se administra con un sistema operativo central la Figura 13 es una representación gráfica de la división del procesador en varias tareas. El sistema es el encargado de asignar las porciones de memoria para cada tarea.

Figura 13: Organización de tareas dentro de un procesador.
Fuente: Yourdon Edward, "Análisis estructurado moderno", 1989, ISBN: 0135986249

El modelo de implantación de programas:

En el modelo de implantación de programas el diseñador deberá asignar módulos a cada tarea. Como podemos ver en la Figura 13 para cada tarea se define que módulos van a existir en ese procesador, con una organización jerárquica establecida dentro de una tarea.

Programación y pruebas:

La programación comienza después que se haya terminado el diseño del sistema. La fase de programación o implantación de un proyecto involucra la escritura de instrucciones en un lenguaje de programación. Las pruebas nos aseguran que el sistema obtenga los resultados esperados de acuerdo a las entradas dadas. Existen varias estrategias de prueba las dos más comunes son las pruebas ascendentes y descendentes. El enfoque ascendente empieza por probar módulos individuales pequeños separadamente, esto se conoce también como prueba de unidades, prueba de módulos o prueba de programas. Luego, se prueban unidades formadas por una combinación de módulos más grandes, finalmente se prueban todos los componentes del sistema, y se realizan pruebas de aceptación, las cuales permiten a los usuarios realizar sus propias pruebas del sistema. El enfoque de prueba descendente supone que se han desarrollado los módulos de alto nivel del sistema, pero todavía no existen los módulos de bajo nivel, dado que no se han implementado todavía muchas de las funciones detalladas, el objetivo es comenzar a verificar las interfaces entre los subsistemas principales.

Metodología SSADM:

SSADM es un acrónimo para método de análisis y diseño de sistemas estructurados, esta es una metodología estructurada de desarrollo de software, creada en la agencia

central de informática y telecomunicaciones del Reino Unido. Fue introducida como la metodología estándar para el desarrollo de proyectos del gobierno del Reino Unido en 1981, actualmente se puede usar por cualquier persona sin la necesidad de ningún permiso. SSADM es un conjunto de guías y estándares para el análisis y diseño de sistemas de información. Los estándares son los que definen la estructura del desarrollo del proyecto en forma de tareas específicas, las guías, técnicas y herramientas nos indican como desarrollar estas tareas para obtener un producto final de acuerdo a lo que necesita el cliente.

Los objetivos de SSADM son: asegurar que el proyecto pueda desarrollarse completamente a pesar de que ocurra algún inconveniente, desarrollar sistemas de calidad, mejorar la forma en que los proyectos son administrados, permitir la participación de personal con o sin experiencia, mejorar la comunicación entre los participantes. Las técnicas más importantes que se utilizan son: modelo de datos lógicos, modelo de flujo de datos, modelo entidad evento. El modelo de datos lógicos se refiere a la identificación y documentación de la información del proceso del negocio, en este modelo se representan las entidades con sus atributos y relaciones del sistema. Un modelo de flujo de datos nos permite identificar, visualizar y documentar los datos que se mueven alrededor de un sistema de información. Este modelo examina los procesos, almacenes de datos, entidades externas y flujos de datos. El modelo entidad evento identifica, y documenta los eventos que afectan a cada entidad y la secuencia en que se producen estos eventos y se diseña para cada caso su proceso. Los roles principales de la metodología SSADM son: director del proyecto, analista, arquitecto, desarrollador, tester, encargado de la liberación.

El director del proyecto es el encargado del seguimiento a los avances del proyecto, la estimación de tiempo, costo, y debe analizar si existen riesgos para el desarrollo del proyecto para buscar soluciones que permitan el desarrollo del software. El analista es quien se encarga del análisis del sistema, determina los actores, casos de uso del sistema, los diagramas de secuencia, los diseños de las interfaces. El arquitecto se centra en la especificación del sistema, de identificar los objetos y de clases del sistema, de desarrollar los diagramas de objetos, de clases, de secuencia y del diseño de la base de datos. El desarrollador codifica y realiza pruebas del sistema, también analiza e implementa el diseño de base de datos. El tester realiza pruebas funcionales del sistema y mide la calidad de los procesos utilizados. El administrador de entregas realiza la liberación del software y sus actualizaciones.

Algunos de las ventajas de SSADM son:

- Al utilizar claras especificaciones de lo que se tiene que realizar y revisar se logra obtener una buena planificación y control del proyecto, y del tiempo en que se terminara el software.
- El modelamiento de las actividades del negocio, los prototipos, la participación de los usuarios finales, permiten mejorar la probabilidad de éxito en grandes y pequeños proyectos.
- El modelamiento del sistema permite concentrar los esfuerzos en desarrollar un producto que el negocio necesita.
- La documentación del sistema representa los objetivos del negocio, el enlace entre el sistema y el negocio, una especificación precisa de diseño, desarrollo y mantenimiento de las aplicaciones.
- La calidad puede ser mejorada mediante la detección de errores temprano en el ciclo de vida, se utilizan técnicas que promueven la exactitud y controles adecuados que buscan la integridad. Al definir la calidad requerida de los

documentos de diseño, y las pruebas se promueve una mejor gestión de la calidad.
- Cada aplicación es diferente, SSADM permite adaptarse a diferentes proyectos mediante la reutilización de recursos, habilidades, y técnicas en otros proyectos.

Algunas de las desventajas de SSADM:

- SSADM se enfoca en el análisis de un sistema y su documentación. Esto puede ocasionar el riesgo de un sobre análisis del proyecto que puede tener como resultado el consumo de tiempo y dinero.
- Dependiendo del tamaño del proyecto los diagramas de flujo de datos, pueden no ser tan claros, por qué se debe incluir toda la información del sistema.
- Se tiene que completar una fase antes de comenzar otra.

Fases de la metodología SSADM.

SSADM posee una secuencia de pasos que se realizan en orden una después de la otra para el desarrollo de software, existen 5 pasos que utiliza la metodología SSADM: estudio de viabilidad, análisis de requerimientos, especificación de requerimientos, especificación del sistema lógico y diseño. En la Figura 14 podemos visualizar esta secuencia de pasos y a la vez sus divisiones o niveles.

Figura 14: Diagrama de etapas del desarrollo de software metodología SSADM
Fuente: https://prezi.com/tia6mqu5m6su/ssadmmetodologia-de-analisis-y-diseno-estructurado/

Fase 0 Estudio de viabilidad:
Nivel 0 Viabilidad:

En esta fase se determina si el proyecto es viable o no, cuales son los objetivos y las implicaciones del proyecto. Dependiendo del tamaño del proyecto esto puede ser necesario o no, por ejemplo si el proyecto es de pequeña escala esto puede no ser necesario, pero en proyectos más grandes se puede realizar un estudio de viabilidad de una forma informal, ya sea porque no hay tiempo para un estudio formal o por qué se debe realizar el proyecto obligatoriamente.
En un estudio de viabilidad existen cuatro áreas principales de análisis estas son:

Técnica
El desarrollo de software necesita recursos para poder hacer realidad el proyecto. Es por eso que es necesario un estudio de viabilidad técnica, para saber si contamos con un equipo de desarrollo de software adecuado, la tecnología necesaria, se debe identificar tanto el software como el hardware que vamos a utilizar. Si existe disponibilidad del usuario, etc.

Financiera:
Para desarrollar un proyecto es preciso conocer si existen los recursos suficientes para lograr alcanzar los objetivos propuestos. La viabilidad financiera es determinar si existe el suficiente dinero para financiar los gastos, inversiones del proyecto.
El estudio de viabilidad financiera debe demostrar que con las diferentes fuentes de financiamiento que tiene acceso el proyecto, es posible financiar todas las etapas del mismo.

Organizacional
Es conveniente realizar un análisis de viabilidad de la organización, de sus prácticas y procesos para saber si el sistema será compatible, de qué forma podemos sacar el mayor provecho del sistema para que sea útil para la organización.

Ética
La parte ética es un asunto un poco más personal, tenemos que analizar si el impacto que tendrá el sistema es socialmente aceptable, si el desarrollo del proyecto es moralmente correcto.

El producto final de esta etapa es un documento de estudio de viabilidad formal del proyecto.

Fase 1 Análisis de requerimientos:
Nivel 1 Investigación del entorno actual:
Esta es una de las etapas más importantes del SSADM. Esta es una investigación de las actividades que se llevan a cabo por la empresa o por la organización para la cual estamos desarrollando el sistema, esto nos permite entender de una mejor manera cuales son las necesidades del usuario. Si podemos llegar a una comprensión completa del sistema esto nos permitirá tener una base sólida para realizar el análisis restante y el diseño. Si existe un sistema anterior este nos proporcionará los requisitos básicos para el nuevo sistema. La participación de los usuarios es una ventaja, ellos pueden aprender del desarrollo del sistema las técnicas y modelos que utilizamos, por otro lado podemos considerar sus deseos como necesidades o requerimientos de usuario. Podemos definir los límites del sistema, identificar problemas relacionados y establecer los roles del sistema.
Los productos que se obtienen de esta etapa son: catálogo de usuarios, aquí se describe a todos los usuarios del sistema y cómo interactúan con él. Catálogo de requerimientos, detallando todos los requerimientos del sistema. Una descripción de

los servicios. Un diagrama de la estructura lógica de datos, un diagrama de contexto, diagrama de flujo de datos del sistema, y un diccionario de datos .

Nivel 2: Opciones del sistema de negocio.
Lo siguiente de la investigación del entorno actual es decidir el diseño general del sistema, se debe desarrollar un conjunto de opciones de sistemas de negocio en base a la investigación que realizamos del entorno actual. Se tiene que analizar las diferentes formas en que puede ser el nuevo sistema de producción, si existe un sistema anterior se debe elegir entre no tirar el viejo sistema por completo o construir uno totalmente nuevo. Se puede realizar una lluvia de ideas para poder visualizar mejor las ideas del nuevo sistema. El objetivo final es elegir la mejor solución de todas las opciones por el analista y los usuarios.

Fase 3 Especificación de requerimientos:
Nivel 3 Definición de requerimientos:
El analista tiene que desarrollar una especificación completa de lo que el nuevo sistema debe hacer. No deben existir errores, ambigüedad e inconsistencia. La especificación de requerimientos no nos dice cómo el sistema se implementará si no nos dice que va a hacer el sistema. El analista debe construir los diagramas de flujo de datos, los diagramas entidad relación, que se utilizan para producir definiciones de las funciones que los usuarios necesitan. El producto de esta etapa son los diagramas de flujo de datos, entidad relación, las definiciones de cada función.

Fase 4 Especificación del sistema lógico
Nivel 4 Opciones técnicas del sistema
Esta etapa es la primera hacia una implementación física del sistema. Existen varias opciones para la aplicación del sistema, se deben generar dos o tres opciones para presentar al usuario que elige la opción final. Se debe considerar las siguientes opciones: la arquitectura de hardware, el software que se va a utilizar, el costo que va a tener la implementación, el personal que se necesita, la distribución, las interfaces del sistema. Estos aspectos deben ajustarse a las restricciones del negocio, como dinero disponible, estándares de hardware y software.

Nivel 5 Diseño Lógico:
Esta etapa se concentra en los requisitos para la interfaz de usuario. El diseño lógico especifica los métodos de interacción en forma de menús y estructuras de mando.
Las necesidades de los usuarios son áreas de actividad, son las principales interfaces con las que los usuarios interactúan con el sistema. Se analiza los eventos del sistema, las consultas de datos, la actualización de datos de los diagramas de la fase 3 para poder determinar una forma de implementar que sea consistente y segura.

Fase 5 Diseño
Nivel 6 Diseño físico:
Esta es la etapa final en que todas las especificaciones lógicas del sistema se convierten en descripciones del sistema en términos de hardware y software, esta es una etapa muy técnica, la estructura lógica se convierte en una arquitectura física en términos de estructuras de base de datos. El objetivo es obtener un diseño físico completo que nos permita implementar el sistema.

- Metodologías Orientadas a Objetos:

Las metodologías orientadas a objetos guían el desarrollo de software mediante la visualización, modelamiento, e implementación de un sistema como una colección de objetos que interactúan entre sí. Se retoman muchas ideas de las metodologías estructuradas, pero con el apoyo de lenguaje orientado a objetos. Durante 1980 estas metodologías ganaron una gran popularidad, debido a que la programación orientada a objetos era el estilo de programación dominante durante esta época. La primera generación de metodologías surge entre 1986 y 1992, estas eran metodologías híbridas que tenían características entre estructuradas y orientadas a objetos. La primera completa orientada a objetos fue creada por Booch en 1986. La segunda generación apareció entre 1992 hasta 1996, durante este período existían 70 metodologías diferentes que competían entre sí por entrar en la industria de desarrollo de software, este periodo se denominó "La guerra de las metodologías". Solo escoger una metodología ya era un esfuerzo, por lo cual gracias a los esfuerzos de la comunidad de ingeniería de software se logró integrar todas estas metodologías en lo que hoy se conoce como UML (Lenguaje de Modelado Unificado), el cual se utiliza para visualizar, especificar construir y documentar un sistema. La tercera generación, se refiere a la integración de ciertas metodologías como resultado se han obtenido otras metodologías que resultan difíciles de manejar. Entre ellas RUP, OPEN, Catalyst, la frustración por la utilización de estas metodologías es la causa del movimiento ágil.

Metodología FUSION

Es una metodología orientada a objetos de desarrollo de software, desarrollada en los laboratorios Hewlett Packard en Bristol, Inglaterra. Fusion integra las mejores características de las primeras metodologías orientadas a objetos, proporciona una guía a los desarrolladores de software, para realizar la implementación de un sistema en un lenguaje de programación, a través del análisis y el diseño. El proceso de desarrollo se divide en fases, los desarrolladores son quienes planifican que se va a hacer durante cada fase y el orden de las actividades, los criterios que indican al desarrollador cuando pasar a la siguiente fase los proporciona la metodología, Fusion utiliza una notación integral, simple, y bien definida para todos sus modelos, y herramientas de gestión. Las salidas de las diferentes fases están identificadas y existen controles para verificar si se logró alcanzar los objetivos. Cada fase tiene sus propias técnicas y se enfoca en un aspecto diferente del proceso de desarrollo de software. Es una metodología adaptable, ya que se puede utilizar una versión ligera en proyectos que no sea necesario utilizar la versión completa, o partes del proceso pueden ser utilizados dentro de otros procesos de desarrollo para fortalecer sus debilidades.

Algunas de las ventajas de Fusion son:

- Utiliza una base de modelos estructurales y de comportamiento para crear un modelo del sistema.
- Las fases y tareas del proceso de desarrollo están bien definidas y se sigue una secuencia de pasos.
- Se puede saber si se cumplió con los requerimientos por medio de pruebas en escenarios de uso de un sistema.

- Los modelos son detallados, claros, útiles para poder entender cómo funciona el sistema en su entorno y sus funcionalidades.

- Se utilizan modelos detallados de objetos, clases y se da una mayor atención a los objetos en especial a su visibilidad y referencia.

Algunas de las desventajas de Fusion son:

- Se necesita un documento de requerimientos para comenzar con el proceso de análisis.
- Los modelos estructurales que se identificaron en la fase de análisis no se los continua en la fase de diseño.
- Se realiza un gran numero de diagramas y otros entregables.

Fases del proceso de desarrollo de software:

La metodología Fusión posee las siguientes fases: análisis, diseño e implementación. En la Figura 15 podemos ver cómo están distribuidas las fases, en la fase de análisis lo primero que se realiza es un documento de requerimientos para luego poder continuar, se modelan los objetos, los cuales nos permitirán crear un modelo de interfaz con los escenarios y el modelo de operación, a continuación se realiza las descripciones de clase y la visualización de herencia en la fase de diseño, también se realizan los gráficos de interacción entre objetos que tienen como objetivo ayudarnos para poder crear gráficos de visibilidad, junto con el diccionario de datos todas estas herramientas se utilizan en conjunto para poder crear el producto final.

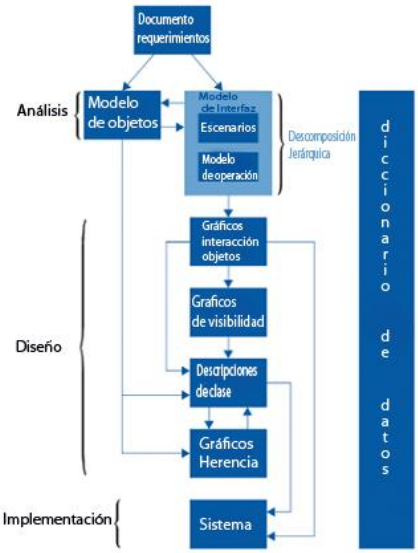


Figura 15: Diagrama del proceso de desarrollo de software fusión.
Fuente: www.hpl.hp.com/hpjournal/96aug/aug96a3a.pdf

Análisis:

En esta etapa se define lo que va a hacer el sistema, cuál va a ser su comportamiento, desde el punto de vista de un usuario, se obtiene los requerimientos, para desarrollar el documento de requerimientos el cual es el documento estándar que nos sirve para comenzar con la fase de análisis. A continuación se realizan varios modelos del sistema que describen las clases de los objetos que existen en el sistema y sus relaciones, las operaciones que puede realizar el sistema y su orden. Estos son dos: un modelo de objetos del sistema y un modelo de la interfaz del sistema. Para ello se desarrolla primero un modelo de objetos del sistema y su entorno, después se identifican los límites del sistema, se crea el modelo de interfaz indicando las operaciones del sistema, finalmente se revisan los modelos de análisis utilizando listas de verificación detalladas. Un modelo de objetos es el proceso de realizar un análisis para obtener la definición de objetos, clases, relaciones y atributos. Para alcanzar la perfección se requiere entrevistas con el cliente y una buena observación.

El ejemplo de la Figura 16 es un modelo de objetos de un sistema de ventas en el cual se presentan los objetos relacionados con el proceso de ventas: empresa, sucursal, producto, cliente, persona, línea de venta y las relaciones que existen.

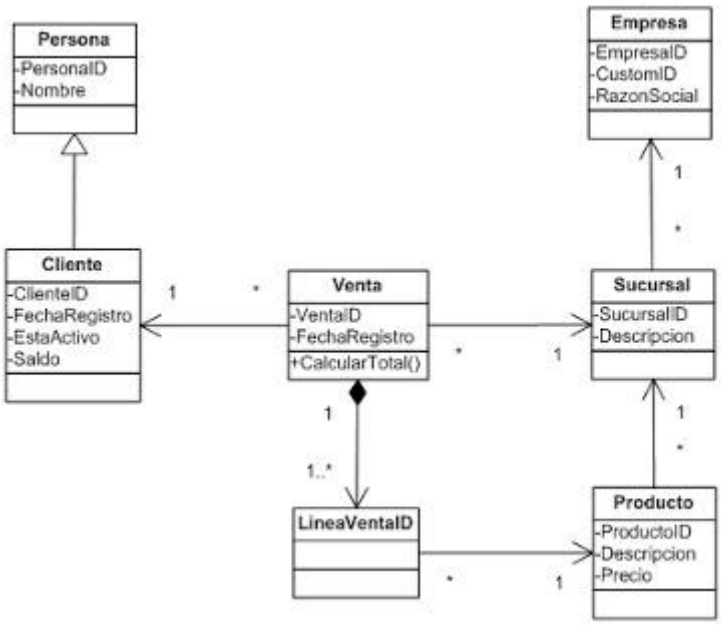


Figura 16: Modelo de objetos
Fuente: http://4.bp.blogspot.com/_MqqufEpbpRw/S9ML-pCEyoI/AAAAAAAAACQ/BtsCnQT8Vss/s400/MOO.jpg

Las relaciones que tiene el sistema con agentes externos como usuarios, otros sistemas son modeladas como diagramas de secuencia, para tener un mejor entendimiento del problema. Al integrar todos los diagramas de secuencia se forman las especificaciones del diagrama de interfaz del sistema, el modelo de objetos del sistema debe tener identificados los límites según los agentes y las operaciones del sistema.

En un diagrama de secuencia se modela la interacción que tiene un actor con el sistema la Figura 17 representa un diagrama de secuencia para el actor visitante en el momento en que quiere registrarse en un sistema, la primera acción que realiza el actor es dar click a un botón registro este lleva a una página registro donde puede ingresar sus datos y enviar un formulario en una capa llamada registro, luego se envía los datos del usuario a la capa de dominio del problema la cual llama a su vez a la base de datos, si se acepta la conexión se inserta el registro en la base de datos y se devuelve un mensaje de confirmación.

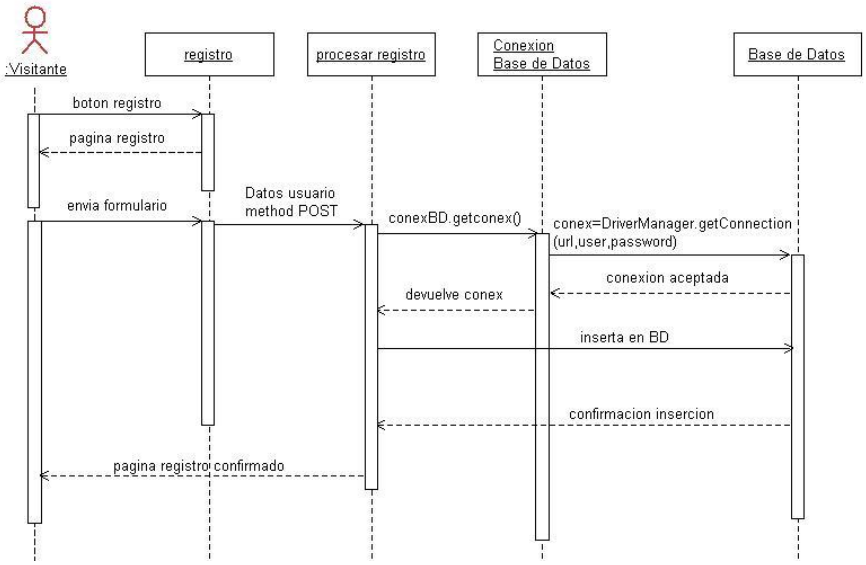


Figura 17: Diagrama de secuencia.
Fuente: http://asojobs.wordpress.com/diseno-inicial/

Diseño:

En esta fase se define como se va implementar las operaciones que realizará el sistema, el enfoque es identificar como se va a obtener lo que se había definido durante el análisis. Durante este proceso se relaciona las operaciones y las clases. Se debe identificar como los objetos interactúan entre sí, y las relaciones apropiadas de herencia entre las clases. Cada modelo debe mostrar como las operaciones del sistema van a ser implementadas, se define las clases con sus respectivos atributos y operaciones para cada objeto.

El diseño consiste de cuatro partes, un gráfico de la interacción que tiene los objetos para la implementación de las operaciones del sistema. Un gráfico que describe la comunicación que tienen los objetos. La descripción detallada de cada clase y un gráfico de la relación de herencia entre las clases. Cada operación del sistema debe ser representada como un gráfico de interacción que describe como se implementarán a través de interacciones de objetos y el intercambio de mensajes.

En la siguiente Figura podemos entender el significado de un diagrama de relación de herencia entre clases, la superclase figura posee los atributos color y los métodos dibujar, borrar, y cambiar color. Las clases línea y círculo tienen una relación de herencia con la clase figura, y por esta razón comparten sus atributos y métodos pero

también tienen métodos y atributos que son propios de cada objeto como centro y radio.

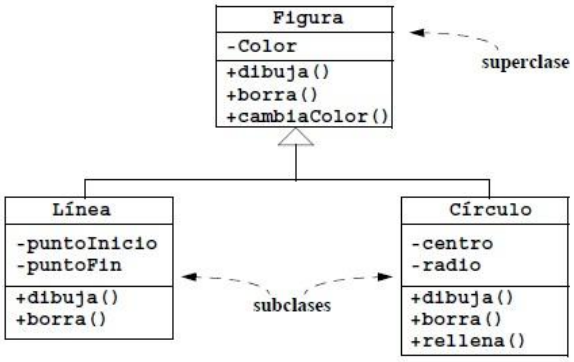


Figura 18: Diagrama de relación de herencia.
Fuente: http://www.redeszone.net/app/uploads/curso_java_volumenXVI_3.jpg

Implementación:

En la implementación se codifica el sistema, el diseño es convertido en código en un determinado lenguaje de programación. Fusion guía este proceso mediante herencia, referencias y los atributos de cada clase que deben ser implementados, las interacciones entre objetos son codificados como métodos que pertenecen a una clase. Las secuencias de operaciones son reconocidas como estados. También se utiliza un diccionario de datos, donde se definen y describen los términos utilizados en el desarrollo de un sistema.

La siguiente Figura representa algunos lenguajes de programación que se utilizan en la implementación de un sistema entre ellos php, visual basic, c++, perl, javascript, python, ruby y java.

Figura 19: Gráfico de lenguajes de programación.
Fuente: http://blog.uchceu.es/informatica/indice-tiboe-ranking-de-lenguajes-de-programacion-mas-utilizados/

Proceso Unificado de Rational (RUP)

El Proceso Unificado de Rational es un proceso de desarrollo de software, que proporciona un enfoque disciplinado para la asignación de tareas y responsabilidades dentro de una organización de desarrollo. El objetivo es asegurar un producto de alta calidad que satisface las necesidades de los usuarios, dentro de una planificación y presupuestos definidos.

RUP es un proceso desarrollado por la empresa Rational Software, actualmente propiedad de IBM, junto con UML constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. RUP mejora el trabajo de equipo con acceso a conocimiento, plantillas, herramientas para el desarrollo del producto, todo el equipo utiliza un lenguaje común que facilita el proceso de desarrollo de software. Es una guía para la utilización eficaz de UML, que nos permite comunicar claramente requerimientos, diseños y arquitecturas. Está apoyado por herramientas que automatizan los procesos, son utilizadas para crear modelos del proceso de ingeniería de software, modelos de visualización, programación, pruebas. Es un conjunto de las mejores prácticas de software que puede ser utilizado para una gran variedad de proyectos y organizaciones. Entre ellas:

El desarrollo de software de forma iterativa, permite definir el problema, diseñar una solución, construir el software y realizar pruebas en varias iteraciones, actualmente los sistemas informáticos son sofisticados y no es posible desarrollar de forma secuencial, es necesario un enfoque incremental para entender el problema a través de un refinamiento sucesivo, y obtener un resultado después de múltiples iteraciones que sea útil para el cliente.

La administración de requerimientos, describe cómo producir, organizar y documentar funcionalidades, restricciones, decisiones, y fácilmente capturar y comunicar los requerimientos del negocio necesarios. Los casos de uso y escenarios han demostrado ser una excelente manera de capturar requisitos, funcionalidades y garantizar que éstos dirijan el diseño, implementación y pruebas de software.

El proceso se centra en el desarrollo temprano y una línea base robusta, antes de comprometer recursos para el desarrollo a gran escala. Se describe cómo diseñar una arquitectura resistente, flexible, con capacidad de cambio, comprensible, y promueve la reutilización de software efectivo.

El modelamiento visual permite capturar la estructura y comportamiento de arquitecturas y componentes. Las abstracciones visuales ayudan a comunicar diferentes aspectos del software, ver cómo los elementos del software encajan entre sí, mantener coherencia entre el diseño y la aplicación, reducir ambigüedades. El UML es la base del éxito del modelado visual.

El aseguramiento de la calidad es una parte importante en el desarrollo de software, un rendimiento pobre de las aplicaciones y una escasa fiabilidad son factores comunes que impiden la aceptación de una aplicación de software. Por lo tanto la calidad debe ser revisada con respecto a los requisitos basados en fiabilidad, funcionalidad, y rendimiento del sistema.

La capacidad de gestión de cambio determina que cada cambio sea aceptable y su seguimiento. El proceso describe cómo controlar, rastrear y monitorear los cambios

para permitir el éxito del desarrollo iterativo. La Figura 20 es el logo de la metodología RUP.

Figura 20: Logo Rational Unified Process.
Fuente: http://sce.uhcl.edu/helm/rationalunifiedprocess/process/about.htm

El Ciclo de Vida del Proceso Unificado

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo constituye una versión del producto para el cliente. La Figura 21 representa el ciclo de vida del proceso unificado en la parte superior se encuentran las fases: inicio, elaboración, construcción y transición, en la parte inferior los ciclos y en la parte izquierda las disciplinas.

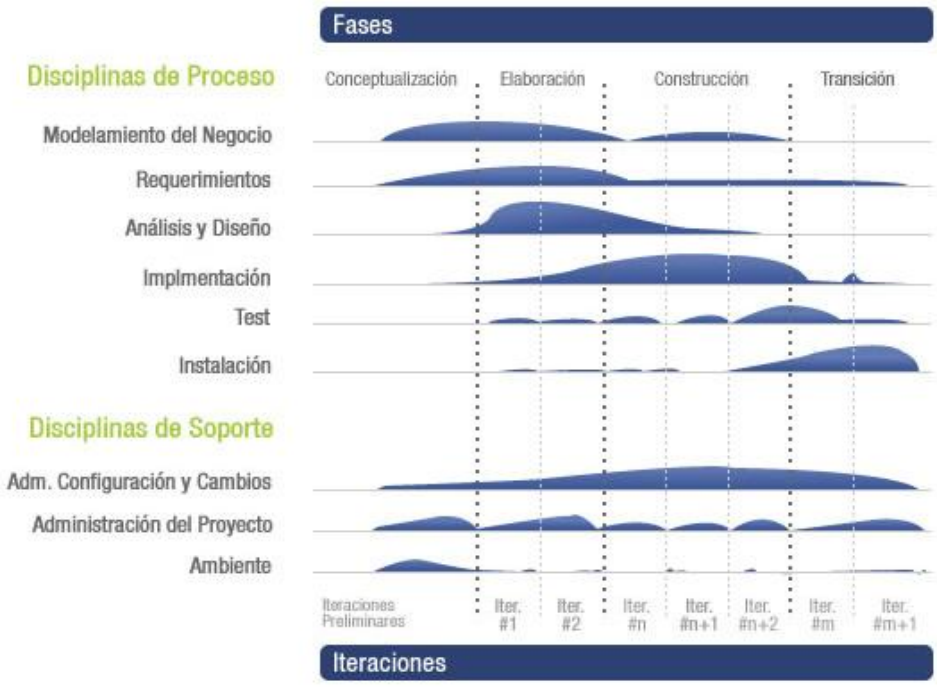


Figura 21: Arquitectura Rational Unified Process.
Fuente:
http://4.bp.blogspot.com/_3AniaubzzJs/S69_M_N38qI/AAAAAAAAAAM/s7103en7XTI/s1600/RUP.jpg

Inicio:

Esta es la primera fase, también conocida como fase de lanzamiento en esta etapa se determina el los requerimientos y el alcance del proyecto. Se identifican los actores principales que van a interactuar con el sistema, los casos de uso más esenciales. Se determina que recursos van a ser asignados al proyecto. Se realiza una propuesta de la arquitectura del sistema, su objetivo es demostrar que se puede crear una arquitectura estable en la siguiente fase. Se identifican riesgos críticos, que puedan afectar el desarrollo del sistema y buscamos si hay una forma de mitigarlos. Podemos construir un prototipo que nos permitirá demostrar a usuarios y clientes que el sistema propuesto es capaz de solventar todas sus necesidades.

Al final de esta etapa los miembros del equipo de trabajo van a saber si es factible desarrollar el proyecto, la intención es minimizar los gastos en tiempo de planificación, esfuerzo y fondos. La fase de inicio no es un estudio completo del sistema propuesto, si no que se busca los elementos necesarios para fundamentar el desarrollo del producto.

En la Figura 21 podemos visualizar de mejor manera que las disciplinas principales que se realizan durante esta fase son el modelamiento del negocio, un análisis de requerimientos, se comienza a realizar la administración del proyecto, y se toma en cuenta el ámbito del proyecto.

Los objetivos de esta fase son:

- Determinar la factibilidad del proyecto, identificar las necesidades del cliente, los límites del proyecto y llegar a un acuerdo entre los miembros del proyecto.
- Encontrar los principales casos de uso del sistema, es decir los escenarios básicos del sistema.
- Estimar el costo, tiempo, riesgos, fuentes de incertidumbre del proyecto.
- Preparar un ambiente de apoyo, que incluye hardware, software, procesos y recursos.

Criterios de evaluación

- Existe un acuerdo entre los interesados del proyecto sobre el alcance del proyecto y las estimaciones realizadas.
- Hay un entendimiento en los requerimientos de software, y los principales Casos de Uso.
- Las estimaciones de tiempo, costo y riesgo son creíbles.
- Comprensión total de cualquier prototipo de la arquitectura desarrollado.
- Los gastos hasta el momento se asemejan a los planeados.

Elaboración:

En la fase de elaboración se seleccionan los casos de uso más importantes, se modela el sistema para luego desarrollarlo, se crea una arquitectura inicial en base a la cual debe evolucionar en las siguientes iteraciones para convertirse en el sistema final, para esto se utilizan varios diagramas entre ellos: diagramas de clases, modelo entidad – relación, diagramas de secuencia, diagramas de estados, diagramas de colaboración, modelo de dominio, diagramas de componentes.

El resultado de la fase de elaboración es una arquitectura estable que va a guiar al sistema, también se planifica la construcción del software con gran precisión para esto es necesario:

Crear una línea base para la arquitectura que cubrirá las funcionalidades del sistema y las características importantes para las personas involucradas. Identificar riesgos que podrían perturbar el desarrollo del software. Especificar los requerimientos tanto funcionales y no funcionales, planificar la siguiente fase del proyecto. Preparar una propuesta de planificación costos, personal necesario.

En esta fase se da más importancia a las disciplinas de modelamiento del negocio, requerimientos, análisis y diseño como podemos observar en la Figura 21.

Los objetivos de esta fase son:

- Definir, validar y consolidar la arquitectura de software.
- Completar la visión del producto.
- Crear un plan viable para la fase de construcción. Este plan puede evolucionar en las siguientes iteraciones.
- Demostrar que la arquitectura propuesta se podrá implementar con un costo y tiempo determinados.

Los criterios de evaluación de esta fase son los siguientes:

- La visión del producto es estable.
- La arquitectura es estable.
- Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos.
- El plan para la fase de construcción es detallado y preciso. Las estimaciones son creíbles.
- Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales.
- Los gastos hasta ahora son aceptables.

Construcción:

El objetivo de la fase de construcción viene indicado pos su tarea fundamental la capacidad de construir un producto listo para ser distribuido como versión beta y ser sometido a pruebas. En esta fase utilizamos todo el trabajo anteriormente realizado, requerimientos del usuario, análisis, diseño para implementarlos en el producto final de tal forma que se logre cumplir las expectativas del cliente. La construcción de un sistema informático se realiza mediante el proceso de escribir código fuente en un lenguaje de programación, la Figura 22 representa un entorno de programación en lenguaje de programación java.

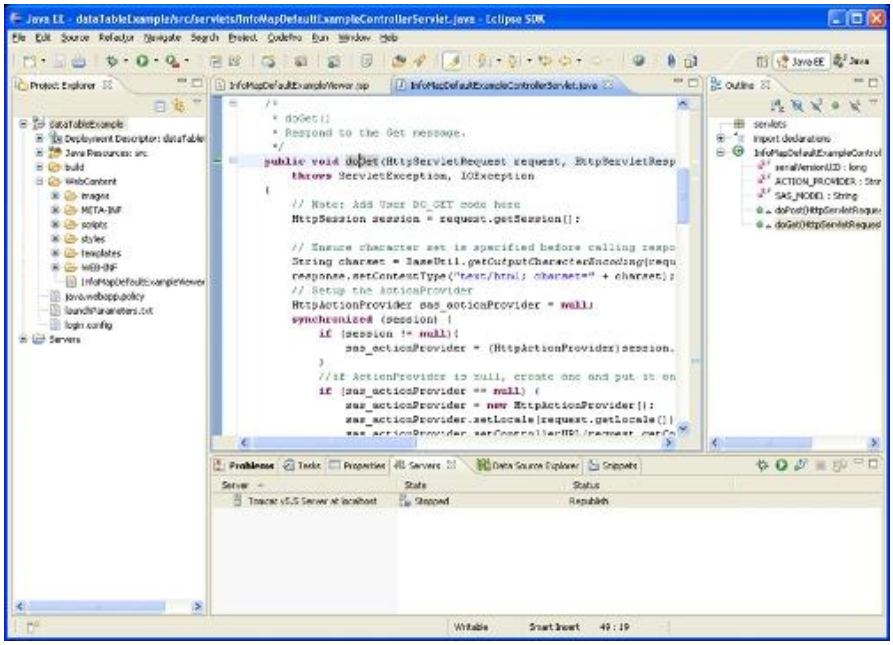


Figura 22: Ejemplo de programación en java.
Fuente: https://www.seoclerk.com/Programming/157776/i-will-create-simple-software-using-JAVA

Esta fase es la fase más larga, pero se divide todo el trabajo en varias iteraciones para equilibrar los recursos asignados. Es importante que el análisis y diseño de los requerimientos, la arquitectura de software esté bien formada por que si no, nos e va a poder cumplir con el costo y tiempo estimado.

Los objetivos concretos incluyen:

- Minimizar los costos de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- Conseguir una calidad adecuada tan rápido como sea posible.
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) del sistema.

Los criterios de evaluación de esta fase son los siguientes:

- El producto es estable y maduro como para ser entregado a la comunidad de usuario.
- Es posible comenzar a realizar pruebas con el producto.

Transición

El objetivo de esta fase es poner el producto en manos de los usuarios finales, para ésto, se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar a los usuarios en el manejo del producto y en general tareas relacionadas al ajuste, configuración, instalación y soporte. A menudo primero se entrega una versión beta del sistema. La organización distribuye un producto de software capaz de un funcionamiento inicial a una muestra representativa de la comunidad de usuarios. El funcionamiento del producto en el entorno de los usuarios es frecuentemente una prueba del estado de desarrollo del producto.

La fase de transición termina con la entrega del producto final. Sin embargo, antes de que el equipo del proyecto abandone el proyecto, los líderes del equipo llevan a cabo un estudio del sistema con los siguientes objetivos: encontrar, discutir, evaluar y registrar las "lecciones aprendidas" para referencias futuras. Registrar asuntos útiles para la entrega de una versión siguiente.

Las siguientes Figuras representan un producto de software terminado, y sus manuales para la entrega al cliente final.

Figura 23: Ejemplo de un producto de software terminado.
Fuente: http://www.interempresas.net/Envase/FeriaVirtual/Producto-Software-de-gestion-de-almacenes-Macsa-Integra-Almacenes-108119.html

Los principales objetivos de esta fase son:

- Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario.

Los criterios de evaluación de esta fase son los siguientes:

- El usuario se encuentra satisfecho.
- Son aceptables los gastos actuales versus los gastos planificados.

Entregables:

Inicio:

- Documento de visión general de los requerimientos del proyecto, características clave y restricciones principales.
- Modelo inicial de casos de uso.
- Glosario inicial.
- Lista de riesgos y plan de contingencia.
- Plan del proyecto incluyendo fases e iteraciones.
- Modelo de negocio.
- Prototipo exploratorio.

Elaboración:

- Modelo de casos de uso completo al menos hasta el 80%.

- Requisitos adicionales que capturan los requisitos no funcionales y cualquier requisito no asociado con un caso de uso específico.
- Descripción de la arquitectura software.
- Un prototipo ejecutable de la arquitectura.
- Lista de riesgos y caso de negocio revisados.
- Plan de desarrollo para el proyecto.
- Un caso de desarrollo actualizado que especifica el proceso a seguir.
- Un manual de usuario preliminar (opcional).

Construcción:

- Modelos completos (casos de uso, análisis, diseño, despliegue e implementación)
- Arquitectura íntegra (mantenida y mínimamente actualizada)
- Riesgos presentados mitigados
- Plan del proyecto para la fase de transición.
- Manual inicial de usuario (con suficiente detalle)
- Prototipo operacional – beta
- Caso del negocio actualizado

Transición:

- Prototipo operacional
- Documentos legales
- Caso del negocio completo
- Línea de base del producto completa y corregida que incluye todos los modelos del sistema
- Descripción de la arquitectura completa y corregida
- Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión.

Metodologías Ágiles:

El desarrollo de software ha evolucionado, desde los primeros años en que la mayoría de requerimientos eran estables y el desarrollo seguía los pasos sin mayores cambios, hasta la actualidad donde los proyectos son más dinámicos, críticos, y han aparecieron nuevas dificultades. Entre ellas está la falta de comprensión de los requerimientos del sistema, por la mala comunicación entre el cliente y el desarrollador, la carencia de una visión clara de lo que necesitan, y cambios continuos debido a la evolución del negocio. La poca participación del cliente en el desarrollo del proyecto, poco presupuesto, plazos ajustados y grandes necesidades. La Figura 24 es un comic en el cual podemos observar un claro ejemplo de las dificultades en el desarrollo de software de un columpio en un árbol.

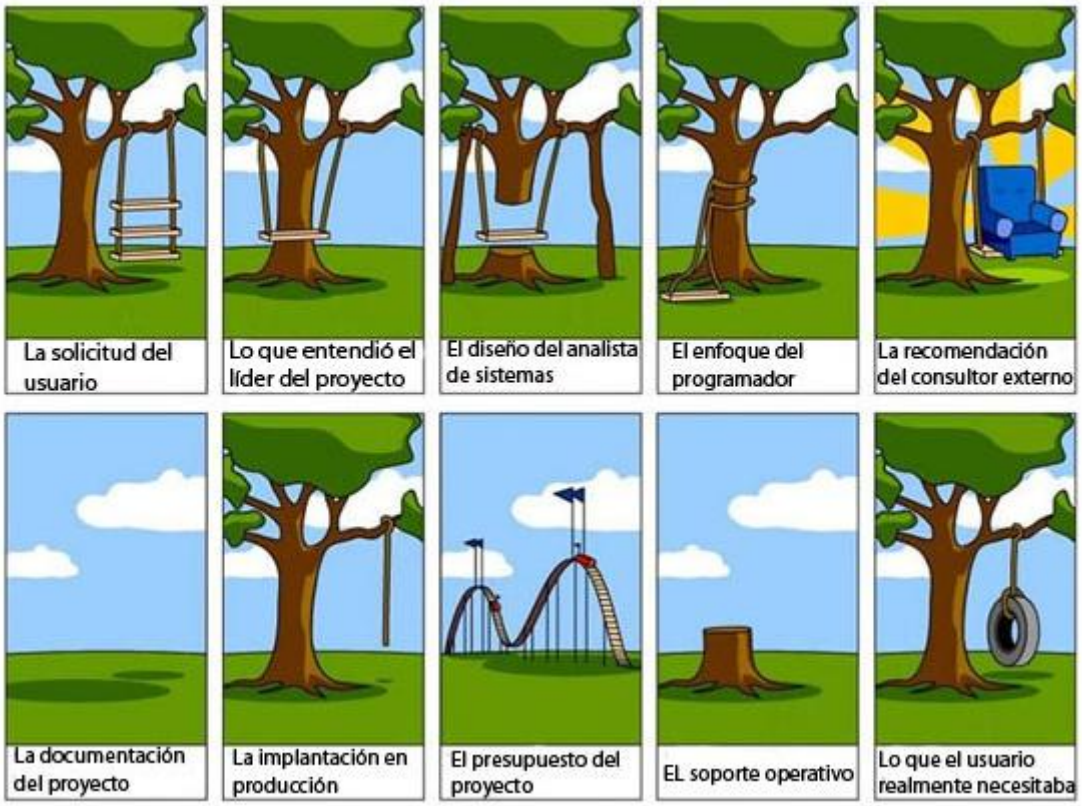


Figura 24: Comic dificultades desarrollo de software
Fuente: http://stevedempsen.blogspot.com/2013/08/agile-software-requirements-comic.html

Las metodologías orientadas a objetos no pueden superar estas dificultades y alcanzar los objetivos, para esto es necesario utilizar nuevas metodologías. Un número de profesionales de TI comenzaron a trabajar individualmente en nuevos enfoques para el desarrollo de software. El resultado de sus investigaciones fueron nuevas metodologías de desarrollo que tenían características similares.

El 17 de febrero del 2001 se reunieron en la conferencia de Utah diecisiete críticos de la propuesta de mejora del desarrollo del software, en la reunión se acuñó el término "Métodos Ágiles" para definir los métodos que estaban surgiendo como alternativa a las metodologías formales (CMMI, SPICE), el resumen de estos métodos son la base del "Manifiesto Ágil", el cual puede ser leído en la Figura 25.

Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

Individuos e interacciones sobre procesos y herramientas
Software funcionando sobre documentación extensiva
Colaboración con el cliente sobre negociación contractual
Respuesta ante el cambio sobre seguir un plan

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.

Figura 25: Manifiesto Ágil.
Fuente: http://agilemanifesto.org/iso/es/

Estas filosofías nos ayudan a superar los problemas antes mencionados, al aceptar los cambios, satisfacer los requerimientos del cliente, desarrollar más rápido y entregar el producto de software final. La Figura 26 representa un cambio del comic de dificultades del desarrollo de software hacia un enfoque orientado a las metodologías ágiles. La diferencia entre los dos gráficos es que en el primer gráfico se observan los problemas en el desarrollo de software y no se puede alcanzar los objetivos, con las metodologías agiles aunque existen problemas, por medio de varias iteraciones se logra alcanzar lo que el cliente necesita.

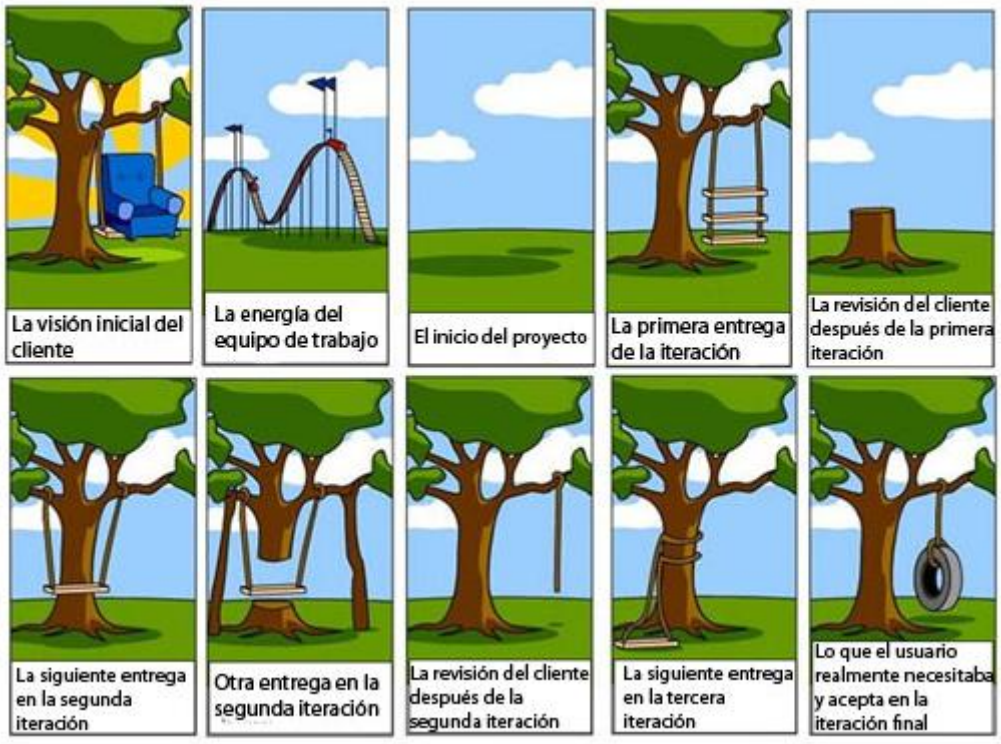


Figura 26: Comic desarrollo de software utilizando metodologías ágiles.
Fuente: http://stevedempsen.blogspot.com/2013/08/agile-software-requirements-comic.html

Metodología SCRUM

Scrum es un marco de trabajo ágil que puede ser utilizado en casi cualquier proyecto, sin embargo es más utilizado en el desarrollo de software.

Es un conjunto de buenas prácticas para trabajar colaborativamente en equipo y obtener el mejor resultado posible de un proyecto, consiste en equipos, roles, eventos, artefactos, reglas, que cumplen con un propósito específico y cada uno es esencial para el éxito de un proyecto.

El proceso de desarrollo de scrum es utilizado para proyectos de requerimientos cambiantes y entornos complejos. Está especialmente indicado para proyectos donde se necesita obtener resultados pronto, en los que las entregas se demoran demasiado, los costos se disparan o la calidad no es aceptable, la moral de los equipos es baja, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

La planificación de un proyecto se realiza con la información real del progreso de un proyecto, cada parte se la divide en varios pedazos y cada uno es construido a partir de pedazos anteriores. Se busca obtener un resultado como consecuencia de la experiencia obtenida en varias iteraciones al haber desarrollado el proyecto por partes.

Algunas de las ventajas de scrum son las siguientes:

- Adopta una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- La calidad del resultado es consecuencia del conocimiento de las personas, en lugar de los procesos empleados.
- El desarrollo se realiza mediante un solapamiento de las diferentes fases, en lugar de un ciclo secuencial.
- Permite minimizar riesgos durante el desarrollo de un proyecto mediante la entrega de varios avances de acuerdo a cada iteración.
- Se realiza un seguimiento diario de los avances logrando que los miembros del equipo estén unidos y el cliente pueda observar los avances del proyecto.

Algunas de las desventajas de scrum son las siguientes:

- Dificultad de aplicación en grandes proyectos, trabaja con equipos pequeños.
- El cliente tiene que estar muy involucrado en el desarrollo de software, tiene que participar de forma activa, continua y revisar los avances.
- Exige una gran responsabilidad al equipo, el equipo tiene que tomar sus propias decisiones, si los miembros del proyecto no están convencidos el proyecto nunca finalizará o fallará.
- Se puede optar por tomar el camino más corto cuando no se logra alcanzar los objetivos a tiempo.
- Esta metodología necesita que los miembros del equipo sean experimentados, si no, no se logrará terminar el producto a tiempo.

El proceso de desarrollo de software utilizando la metodología SCRUM es iterativo, La Figura 27 representa el proceso de desarrollo de un producto, comienza con los requerimientos del sistema y termina cuando la retroalimentación indique que el producto está terminado. La gestión del backlog identifica los requerimientos que se van a entregar durante esta iteración. Luego se planifica el sprint, entre las actividades están: seleccionar que trabajo se realizará, preparar el sprint backlog que detalla el tiempo que llevará realizar el trabajo. Se ejecuta el sprint y se realiza la entrega del resultado del trabajo de la iteración. Lo cual nos proporciona la retroalimentación para la siguiente iteración.

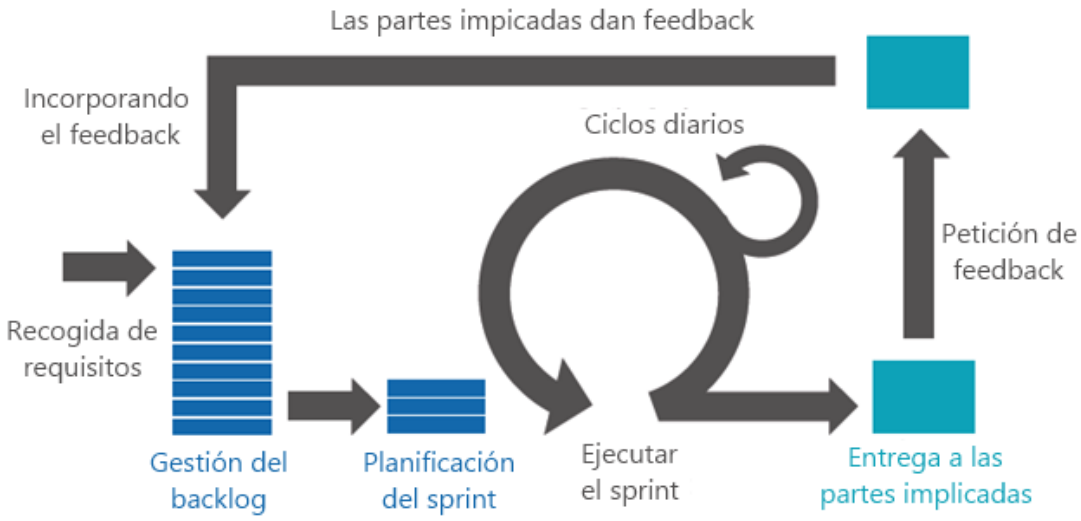


Figura 27: Proceso de desarrollo de un producto utilizando la metodología Scrum.
Fuente: http://programandonet.com/web/scrum-con-tfs/

Reglas de scrum:

El equipo de scrum:

- Propietario del producto: Es el encargado de comunicar la visión del producto al equipo de desarrollo. Debe representar los intereses del cliente en cuanto a requerimientos y dar prioridades. Este es el rol con más autoridad en el proyecto y el que tiene más responsabilidades.

- Scrum Master: Es un facilitador entre el propietario del producto y el equipo de desarrollo. No maneja el equipo sino que es el encargado de eliminar cualquier impedimento que obstaculiza al equipo para alcanzar los objetivos.

- Equipo de desarrollo: Es responsable de completar el trabajo. Equipos ideales consisten en siete miembros. Para el desarrollo de software un equipo típico consiste en ingenieros de software, arquitectos, programadores, analistas, personas encargadas de realizar pruebas, diseñadores. En cada iteración se debe resolver que parte del proyecto se va a completar, esto brinda a los miembros libertad pero a su vez responsabilidad.

Figura 28: Roles de la metodología Scrum.
Fuente: http://es.wikipedia.org/wiki/Scrum#mediaviewer/File:Ficha_scrum.png

Eventos de scrum:

El Sprint es un bloque de tiempo de un mes o menos durante el cual se crea un incremento de producto terminado, utilizable. Cada Sprint comienza después de la finalización de un sprint previo. Cada sprint consiste de la reunión de planificación del sprint, los scrums diarios, el trabajo de desarrollo, la revisión del sprint y la retrospectiva del sprint. Cada uno se puede considerar un proyecto que posee su propio objetivo, una definición de lo que hay que hacer, su propio diseño y un plan flexible.

La planificación del proyecto se realiza en la reunión de planificación por el equipo scrum completo. Tiene un máximo de duración de ocho horas, el scrum master es el encargado de que se lleve a cabo, y que se mantenga en el bloque de tiempo. La reunión de planificación determina cual va a ser el incremento resultante del sprint, el dueño del producto determina cuales son los elementos de la lista de producto, y los miembros del equipo de desarrollo evalúan que es lo que se va a lograr. El objetivo del sprint es lo que se debe implementar de la lista de producto. El equipo de desarrollo determina como se va a construir cada funcionalidad para obtener un incremento del producto terminado. Los elementos de la lista de producción más el plan de cómo hacerlos recibe el nombre de lista de pendientes del sprint.

El scrum diario es una reunión de 15 minutos para que el equipo de desarrollo sincronice sus actividades y cree un plan para las siguientes 24 horas. Así se mejora la comunicación, la toma de decisiones, el nivel de conocimiento del equipo de desarrollo, se eliminan impedimentos relativos al desarrollo, y no hace falta mantener otras reuniones. El objetivo es evaluar el trabajo realizado y hacer una proyección de lo que puede completarse.

Figura 29: Eventos de la metodología Scrum.
Fuente: http://es.wikipedia.org/wiki/Scrum#mediaviewer/File:Ficha_scrum.png

Al final del sprint se lleva a cabo una revisión del sprint, para inspeccionar el incremento y adaptar la lista si es necesario, se trata de una reunión informal que tiene como objetivo facilitar la retroalimentación de información y fomentar la colaboración. El resultado es una lista de producto revisada, que define los elementos de la lista de producto posibles para el siguiente sprint.

La retrospectiva de sprint es una oportunidad de inspeccionarse a sí mismo y crear un plan de mejoras, se la realiza después de la revisión del sprint. Es una reunión de máximo tres horas que tiene el propósito de inspeccionar cómo fue el último sprint en cuanto a personas, relaciones, procesos, herramientas, identificar y ordenar los elementos más importantes y las posibles mejoras, crear un plan para implementar las mejoras.

Artefactos de scrum:

Una lista de producto es una lista ordenada de todos los requisitos del producto, es la fuente para cualquier cambio que se requiera realizar. El contenido, disponibilidad y orden son responsabilidades del dueño del producto. La lista de producto es dinámica, cambia de acuerdo los requerimientos, es por eso que nunca está completa. Al principio refleja las necesidades entendidas, pero estas evolucionan a medida que el producto y su entorno cambian. Las funcionalidades, requisitos, mejoras, cambios, correcciones del software son parte de la lista de procesos, cada elemento que posee una lista posee una descripción, orden, estimación y valor. A menudo varios equipos scrum trabajan juntos en el desarrollo de un mismo producto, para describir el trabajo que se va a realizar se utiliza una lista de producto, el dueño del producto y el equipo de desarrollo colaboran en los detalles, estimaciones y orden de los elementos a esto se le conoce como refinamiento. El orden de los elementos de la lista determina si son más claros y detallados con un orden de mayor a menor, Los elementos que se desarrollaran en el siguiente sprint son más precisos, han sido refinados, se los conoce como elementos "preparados" o "accionables."

La lista de pendientes del sprint es el conjunto de elementos de la lista de producto seleccionados para el sprint, un plan para entregar el incremento de producto y conseguir el objetivo del sprint. Es una predicción del equipo de desarrollo sobre las funcionalidades que formaran parte del próximo incremento. Si se requiere más trabajo se añade a la lista de pendientes, y si un elemento es considerado innecesario es eliminado. La lista dependientes del sprint pertenece únicamente al equipo de desarrollo.

El incremento es la suma de todos los elementos de la lista de producto completos durante un sprint y el valor de los incrementos de sprints anteriores. Al final de un sprint el nuevo incremento debe estar en condiciones de ser utilizado sin importar si el dueño del producto desea liberarlo o no.

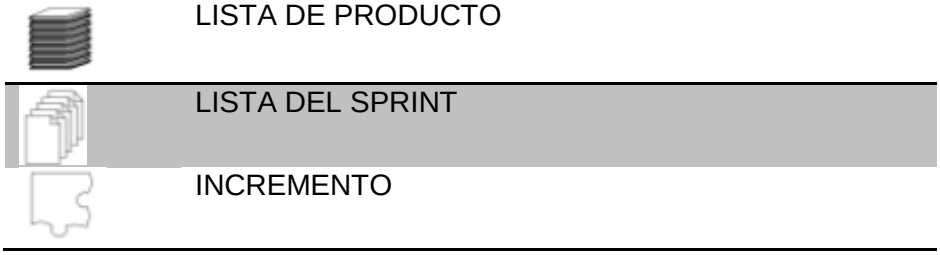


Figura 30: Artefactos de la metodología Scrum.
Fuente: http://es.wikipedia.org/wiki/Scrum#mediaviewer/File:Ficha_scrum.png

2.2. Metodología de programación ágil Extreme Programming (XP).

Extreme Programming (XP) es una metodología ligera, eficiente, de bajo riesgo, flexible, predecible, científica y divertida para desarrollar software.

XP fue concebido y desarrollado para abordar las necesidades específicas de desarrollo conducido por equipos pequeños, en requerimientos cambiantes e imprecisos. Por medio de esta metodología podemos reducir los riesgos del proyecto, mejorar la capacidad de respuesta ante cambios del negocio y mejorar la productividad de un sistema.

La metodología XP está basada en valores que nos permiten mejorar el software, estos son: simplicidad, comunicación, respeto y coraje.

La simplicidad se ve reflejada en el desarrollo del software al definir el alcance del proyecto, así se realiza solo lo necesario y no se emplea tiempo en otras características del sistema que no se haya llegado a un acuerdo, al dividir el proyecto en iteraciones vamos a alcanzar el objetivo paso a paso, resolviendo los problemas que aparezcan.

Es importante que haya una buena comunicación durante el desarrollo del proyecto, un entendimiento entre la parte del cliente y los desarrolladores ayuda en la elaboración del software, todos somos parte del equipo y vamos a trabajar juntos para obtener la mejor solución del problema.

Todos nos merecemos respeto, cada individuo colabora con el crecimiento del proyecto, incluso si es con solo entusiasmo, cada persona tiene una forma distinta de pensar, los desarrolladores respetan la experiencia del cliente y viceversa.

Siempre necesitamos tener el coraje de decir la verdad en cuanto a estimaciones y el progreso del proyecto, no tememos a nada y si existen cambios nos vamos a adaptar a ellos

El propósito de XP es entregar un sistema que cumpla con lo que el cliente necesita, en el tiempo requerido, además una de las razones del éxito de esta metodología es aceptar los cambios en cualquier momento durante el desarrollo.

Algunas de las ventajas de Extreme Programming son las siguientes:

- Algunos proyectos tienen requerimientos poco claros o dinámicos en estos casos es conveniente utilizar XP.
- Existe una gran probabilidad que el sistema cumpla con los requerimientos de usuario.

- Adaptabilidad a cambios en del negocio con menores costos.
- Se libera una versión de software verificada durante cada iteración la cual puede revisar el cliente, esto disminuye el riesgo del proyecto.
- Al realizar constantes pruebas, los defectos del sistema se logran identificar en las primeras etapas.

Algunas de las desventajas de Extreme Programming son las siguientes:

- El cliente tiene que estar involucrado todo el tiempo.
- Es necesaria una completa cooperación entre el cliente y los desarrolladores.
- XP funciona mejor con equipos medianos entre 2 y 12 personas.
- La falta de documentación adecuada en grandes proyectos puede ser un problema especialmente cuando se cambian los miembros del proyecto.
- Una gran cantidad de refactorización puede ser una pérdida de tiempo.
- XP posee un enfoque orientado a la codificación y no al diseño.

2.2.1. Fases

La metodología XP posee las fases de planeación, diseño, codificación, pruebas, y para cada una de las fases existen tareas que debemos realizar. La siguiente Figura representa cada una de las fases de la metodología XP y sus tareas.

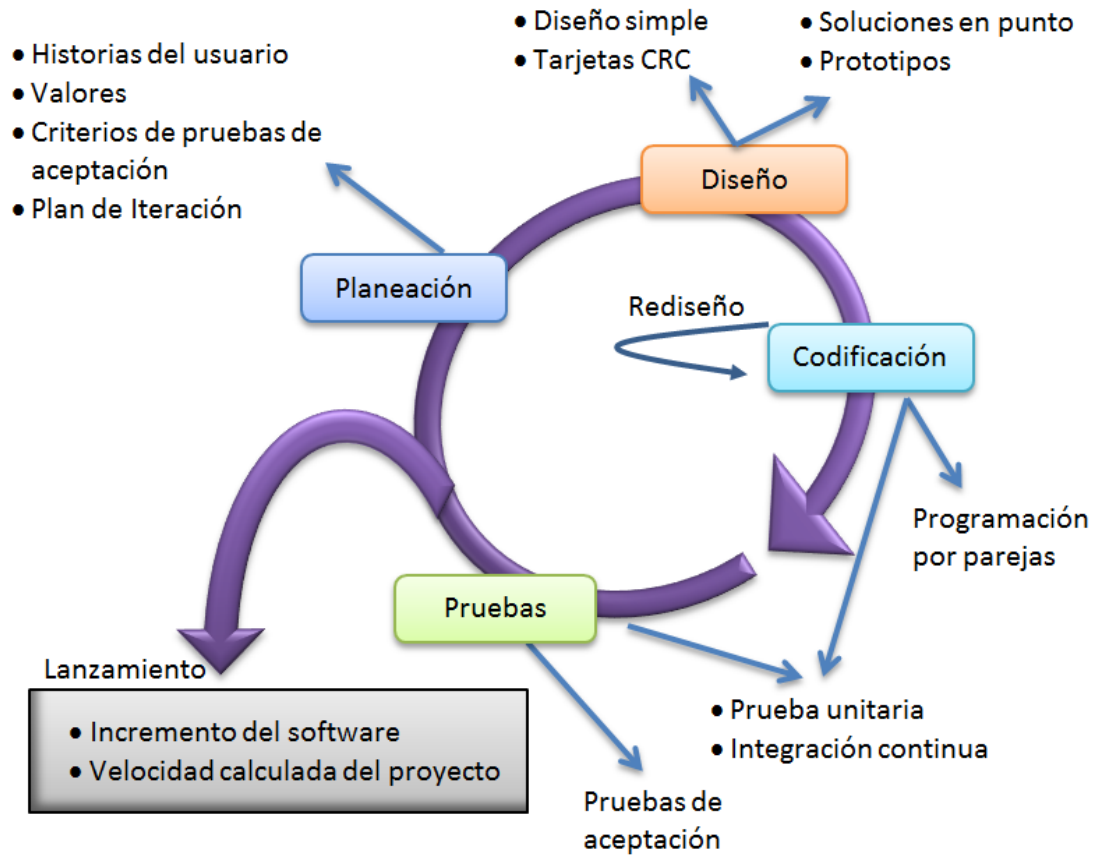


Figura 31: Fases de desarrollo de software metodología XP.

Fuente: http://www.codejobs.biz/es/blog/2013/06/05/programacion-extrema-xp#sthash.DQ2RYslj.dpbs

Planeación:

Es la primera actividad de la metodología XP, esta actividad nos permite obtener los requerimientos del cliente para entender el contexto del negocio, así podremos definir las características y funcionalidades del software que se va a elaborar.

La actividad de planificación comienza escuchando, lo cual nos lleva a la creación de historias de usuario. Estas son, una forma simple de conseguir lo que necesitan los clientes, para que realice el sistema por ellos. Se escriben aproximadamente 3 frases por el cliente, utilizando sus propios términos.

La diferencia de las historias de usuario con la especificación de requerimientos tradicionales es el nivel de detalle. Es importante obtener solo suficientes detalles para estimar cuanto tiempo se necesita para implementar una historia, esta se estima en número de semanas, si son mayores a tres semanas se deben dividir en historias más pequeñas, y si son menores a una se pueden combinar varias. En la etapa de implementación los desarrolladores deben recibir una descripción más detallada de los requerimientos.

Se debe buscar llegar a un acuerdo entre el cliente y el desarrollador, para eso las historias de usuario deben cumplir con las siguientes características: tienen que proporcionar valor para el cliente, cada una debe ser independiente de otra, el tamaño de cada una debe permitir que se realice el desarrollo de unas pocas en cada iteración, y por último cada historia debe ser comprobable.

Las historias de usuario permiten realizar una estimación de tiempo, con la cual se elabora el plan de entrega.

Antes de elaborar el plan de entrega es importante tener una reunión, en cada una de las iteraciones del proyecto, para su respectiva elaboración con el cliente. El plan de entrega determina cuales historias de usuario se van a implementar, en que ciclo y su respectiva fecha. El objetivo es estimar en semanas cuanto tiempo tomará implementar cada historia de usuario, e identificar que historias se desarrollarán primero y cuales después. La responsabilidad del cliente es definir las historias de usuario, decidir el valor que tiene cada historia y cuales se van a realizar en una primera iteración, en cambio la del desarrollador de software es estimar cuanto tiempo se necesita para elaborar cada historia, prevenir al cliente de riesgos técnicos significativos y medir el progreso del proyecto.

El siguiente paso es dividir el proyecto en iteraciones, el desarrollo iterativo nos permite tener una mayor agilidad, control en el desarrollo de software. Se debe llevar un seguimiento de cada iteración, la cual dura alrededor de 3 semanas. Al comienzo se selecciona las historias de usuario que van a ser implementadas y las que no pasaron las pruebas de la iteración anterior, cada historia se divide en tareas de 1 a 3 días de duración, que van a ser realizadas.

Diseño:

La simplicidad es la clave. XP recomienda realizar siempre lo más simple que pueda funcionar, si algo es muy complejo se debe reemplazarlo con algo más simple. Un diseño simple nos permitirá conseguir facilidad a la hora de implementar, y un mayor entendimiento del sistema.

Por medio de las siguientes características, podemos determinar si el diseño es simple. Es navegable, si se puede encontrar en el diseño cualquier cosa cuando se necesite, utilizar polimorfismo, herencia nos facilita el desarrollo de un sistema navegable. Es fácil de verificar su correcto funcionamiento por medio de pruebas unitarias, de aceptación. Es comprensible, si se puede entender cómo funciona el sistema. Es explicable si es posible explicar el funcionamiento del sistema.

En esta fase se diseñan las Tarjetas de Clases Responsabilidades y Colaboración (CRC) las cuales nos permiten pensar con un enfoque orientado a objetos. Cada tarjeta representa un objeto, la clase de cada objeto puede ser escrita en la parte de arriba de la tarjeta, las responsabilidades en la parte izquierda inferior, y las clases de colaboración en la parte derecha.

Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semánticas, sus responsabilidades son sus atributos y métodos. Las clases de colaboración son las clases con las que interactúa para llevar a cabo sus actividades. Es conveniente indicar al cliente las clases que se han identificado, para llegar a un acuerdo sobre su validez.

Una vez que tenemos las historias de usuario es conveniente realizar un prototipo, para reducir riesgos técnicos, básicamente es un programa simple, en el cual vamos a tener una estimación de cómo se va a implementar una historia de usuario, esto es útil si no tenemos ninguna idea, y nunca hemos desarrollado una historia similar.

Codificación:

En el plan de iteraciones, se obtienen las historias de usuario del cliente, estas se dividen en tareas que son asignadas y van a ser desarrolladas.

Uno de los requerimientos de XP es que el cliente siempre esté disponible, para cada una de las fases, la comunicación con el cliente es muy importante y es preferible que sea cara a cara. Los detalles de cada historia de usuario se quedaron aparte, es por eso que es necesario hablar con el cliente para obtener los requerimientos de una forma más detallada para cada tarea de programación.

En XP la programación se realiza por dos personas sentadas juntas en una máquina, a estó se le conoce como Pair Programming. Primero se trabaja en una tarea a la vez, después al finalizar las tareas se integran al software liberado, a continuación se corren todas las pruebas unitarias del sistema para verificar que no haya ningún problema y todo esté funcionando al 100%.

Al programar XP recomienda que el código exprese las intenciones que tenemos, así podemos lograr un código claro y rápido que ayudará la próxima vez que una persona revise el código, de esta forma se puede reducir la documentación. Si se puede entender el significado de un método, ya no es necesario escribir comentarios o documentar las características de ese método. Una buena práctica es utilizar estándares de programación, estos nos ayudan a mantener orden, por ejemplo si utilizamos estándares en los nombres durante la programación se vuelve más fácil y rápido encontrar lo que buscamos.

Utilizamos diseños simples durante la programación, cuando hacemos una tarea se busca realizar lo más simple posible, es decir solo lo que sea necesario para que el código pueda funcionar.

Después de haber realizado las pruebas y comprobado la simplicidad del código estamos listos para la liberación.

Al realizar cambios la idea es tener las pruebas unitarias verificadas al 100%, realizar liberación de código a menudo de esta forma habrá pocos conflictos.

Pruebas:

Antes de escribir código se definen las pruebas unitarias, esto es útil porque no sabemos las funcionalidades del sistema, ni cómo va a evolucionar.

El objetivo al escribir el código del sistema es realizar solo lo suficiente para que se pueda validar las pruebas, las pruebas son una guía donde apoyarnos. En el momento que se requiera cambiar el código, las pruebas son las que brindan confianza, por su inmediata retroalimentación, nos ayuda a saber si los cambios no generan ningún problema.

Las Pruebas de Aceptación son creadas a partir de las historias de usuario. En cada iteración las historias seleccionadas serán traducidas a pruebas de aceptación. El cliente definirá los escenarios en que se evaluarán las historias de usuario que han sido implementadas.

Las pruebas de aceptación son pruebas de caja negra, cada una representa un resultado esperado del sistema. El cliente es responsable de verificar el resultado esperado de cada prueba, decidir cuales están bien y cuales fallaron e identificar una prioridad de estas últimas. En cada iteración se crean nuevas pruebas de aceptación, esto representa el progreso del equipo.

Las Pruebas Unitarias bien redactadas son una de las mejores documentaciones posibles, nos indican claramente la intención que tiene el código que estamos escribiendo. Cada parte que se codifique debe poseer una prueba unitaria, esto nos permite saber exactamente si está funcionando o no cada parte del código, no se puede liberar código sin su respectiva prueba. Si encontramos algún defecto se crean nuevas pruebas que tienen que ser verificadas.

Una vez que se han superado las pruebas unitarias, las pruebas de aceptación y todo esté funcionando al 100% podemos liberar el código.
La Figura 32 nos permite visualizar de mejor manera el ciclo de trabajo de la metodología XP. Comenzamos con las historias de usuario que nos permiten tener más claro los requerimientos de usuario y los escenarios de prueba del sistema. Un primer esbozo de la arquitectura nos da una primera impresión de cómo va a ser el sistema. Con esta información podemos realizar una planificación de las entregas, esto define el trabajo que se debe realizar durante cada iteración y la velocidad del proyecto, que es el número de historias de usuario para cada iteración. De las iteraciones se obtiene la última versión del sistema, si se encuentran errores estos se quedan para una próxima iteración. Finalmente se realizan las pruebas de aceptación de la última versión del sistema si el cliente aprueba esta versión se realiza su entrega.

Figura 32: Secuencia de desarrollo de software utilizando la metodología XP.
Fuente: http://www.extremeprogramming.org/map/project.html

Para cada iteración las historias de usuario, la velocidad del proyecto que representa el número de historias de usuario que se pueden implementar en una iteración, y pruebas de aceptación fallidas, nos permiten planificar las iteraciones que van a ser desarrolladas, las tareas incompletas se las deja para una nueva iteración, después de realizar el desarrollo se obtiene nuevas funcionalidades o se corrige errores anteriores esta es la última versión d la cual se pueden obtener nuevas necesidades o historias de usuario. Como podemos observar en la Figura 33.

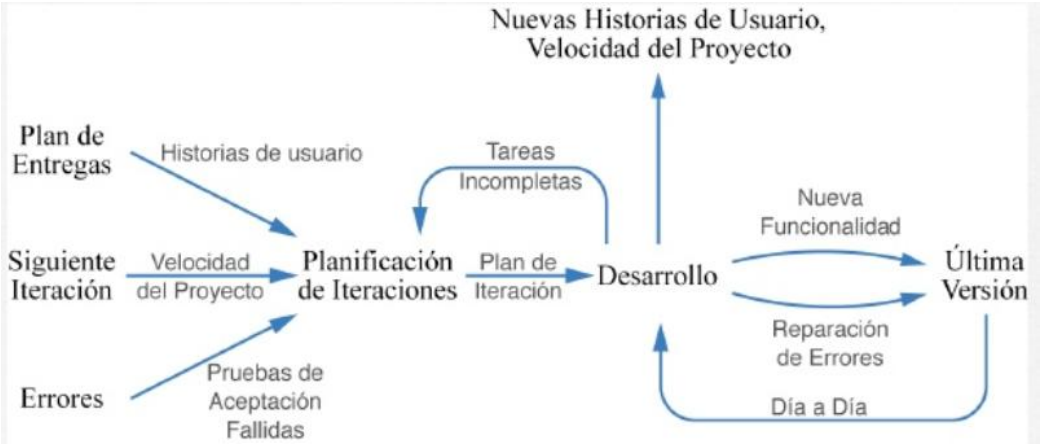


Figura 33: Secuencia de desarrollo de software previa a la codificación, para cada iteración.
Fuente: http://es.slideshare.net/joaquin_win/extreme-programming-456979

2.2.2. Entregables.

Planeación:

- Acta de Reuniones
- Historias de Usuario
- Plan de entrega
- Plan de Iteraciones

Diseño:

- Diseños simples.
- Glosario de términos
- Tarjetas CRC
- Prototipo

Codificación:

- Estándares de Codificación
- Pruebas Unitarias

Pruebas:

- Pruebas Unitarias
- Pruebas Aceptación

2.3. Selección de Metodología para la disertación.

Una vez que se han presentado las principales características de las diferentes metodologías se resume en el siguiente cuadro sus propiedades fundamentales.

METODOLOGÍA DE DESARROLLO DE SOFTWARE	**TIPO DE DESARROLLO**	**TIEMPO DE DESARROLLO**	**TIPO DE REQUERIMIENTOS**	**ORIENTADO A OBJETOS**	**PARTICIPACIÓN DEL USUARIO**	**TAMAÑO DEL PROYECTO**	**INTEGRANTES DEL PROYECTO**	**APRENDIZAJE**
METODOLOGÍAS ESTRUCTURADAS								
METODOLOGÍA DE YOURDON	SECUENCIAL	PROYECTOS DE LARGA DURACIÓN	RESISTENCIA A CAMBIOS	NO	MEDIO	GRANDES	GRUPOS GRANDES	LENTO
METODOLOGÍA SSADM	SECUENCIAL	PROYECTOS DE LARGA DURACIÓN	RESISTENCIA A CAMBIOS	NO	MEDIO	GRANDES	GRUPOS GRANDES	LENTO
METODOLOGÍAS ORIENTADAS A OBJETOS								
METODOLOGÍA FUSION	SECUENCIAL	PROYECTOS DE DURACIÓN MEDIA, LARGA	RESISTENCIA A CAMBIOS	SI	MEDIO	MEDIANOS, GRANDES	GRUPOS MEDIANOS	LENTO
METODOLOGÍA RUP	ITERATIVO E INCREMENTAL	PROYECTOS DE DURACIÓN MEDIA, LARGA	REQUERIMIENTOS CAMBIANTES	SI	MEDIO	MEDIANOS, GRANDES	GRUPOS MEDIANOS	LENTO
METODOLOGÍAS ÁGILES								
METODOLOGÍA SCRUM	ITERATIVO E INCREMENTAL	PROYECTOS DE CORTA DURACIÓN	REQUERIMIENTOS CAMBIANTES	SI	ALTO	PEQUEÑOS,MEDIANOS,GRANDES	GRUPOS PEQUEÑOS	RÁPIDO
METODOLOGÍA XP	ITERATIVO E INCREMENTAL	PROYECTOS DE CORTA DURACIÓN	REQUERIMIENTOS CAMBIANTES	SI	ALTO	PEQUEÑOS, MEDIANOS, GRANDES	GRUPOS PEQUEÑOS	RÁPIDO

Figura 34: Cuadro de comparación de metodologías de desarrollo de software.

Del cuadro anterior se destacan las metodologías ágiles debido a que se va a desarrollar un proyecto de tamaño mediano, que cuenta con un límite de tiempo y un grupo de desarrollo pequeño. Entre las metodologías ágiles que se estudió SCRUM y XP se eligió la metodología XP, aunque las dos metodologías comparten ciertas características su forma de trabajo es distinta. En SCRUM se trabajan en iteraciones que tienen una duración de entre dos semanas y un mes, en XP se trabaja en iteraciones de una o dos semanas. Además XP se adapta de mejor manera a cambios, en SCRUM es necesario que se termine primero el acuerdo al que se llega de una parte del software para luego poder realizar cambios. Finalmente en XP se da prioridad a lo que los usuarios necesitan que se desarrolle en cambio en SCRUM se realiza primero lo que los desarrolladores deciden.

2.4. Arquitectura

La arquitectura de software nos indica la estructura, funcionamiento e interacción entre las partes del software. Su evolución ha sido conforme al avance de la tecnología. Es así que desde el surgimiento de la computación a mediados de la década de los 50's hasta inicios de los 90, la arquitectura predominante era la arquitectura basada en host la cual era un ordenador conectado a una red, que podía ofrecer recursos de información, servicios y aplicaciones a los usuarios de la red.

La siguiente Figura es una representación gráfica de una arquitectura basada en host. La computadora central ofrece recursos, servicios y aplicaciones a las otras computadoras que están conectadas en red.

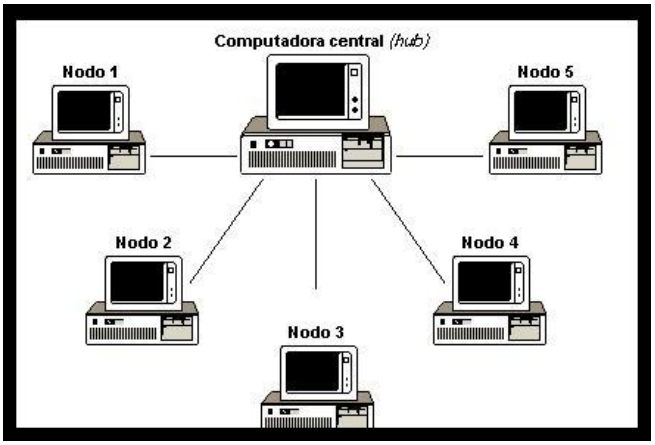


Figura 35: Arquitectura basada en Host.
Fuente: http://laprofeenpracticas.blogspot.com/2013/01/informatica.html

En 1960 empieza a desarrollarse el internet y se vuelve más popular el modelo arquitectónico cliente servidor. Este es un modelo de sistema en el que se organiza un conjunto de servicios y servidores que están asociados, más unos clientes que acceden a estos servicios por medio de una red.

Ejemplos de servidores son: servidores de impresoras que ofrecen servicios de impresión, servidores de compilación que brindan el servicio de compilar un lenguaje de programación. Los clientes pueden conocer los nombres de los servidores disponibles y sus servicios, sin embargo los servidores no necesitan conocer la identidad de los clientes. El procedimiento es el siguiente: un cliente realiza una petición al servidor a traves de un protocolo de petición como por ejemplo el protocolo http y espera hasta recibir una respuesta, el servidor recibe la solicitud y entrega una respuesta. La siguiente Figura representa una arquitectura cliente servidor a través de internet. Existen una computadora portátil, un Smartphone y una computadora de escritorio que realizan peticiones a un servidor.

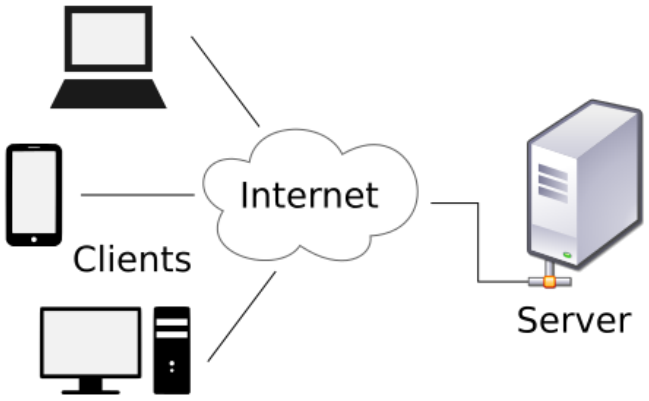


Figura 36: Arquitectura cliente servidor.
Fuente: http://es.wikipedia.org/wiki/Cliente-servidor#mediaviewer/File:Client-server-model.svg

En 1970 surge el modelo de capas el cual organiza el sistema en capas. Cada una de las cuales proporciona un conjunto de servicios. Un ejemplo es el modelo de referencia OSI de protocolos de red. La aproximación por capas soporta el desarrollo incremental de sistemas. A medida que se desarrolla una capa algunos servicios pueden estar disponibles para los usuarios. Esta arquitectura soporta bien los cambios

pero tiene la desventaja de que la estructuración de los sistemas puede resultar difícil. La siguiente Figura representa una arquitectura por capas, la primera capa es la capa de presentación esta capa es la que el usuario puede ver, es la que presenta y captura la información del usuario, también es conocida como interfaz gráfica. La siguiente capa es la de negocio es donde residen los programas que se ejecutan recibiendo las peticiones del usuario y enviando la respuesta. La capa de negocio se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados, también se comunica con la capa de datos para solicitar los datos de la base de datos. La capa de datos es donde residen los datos del sistema.

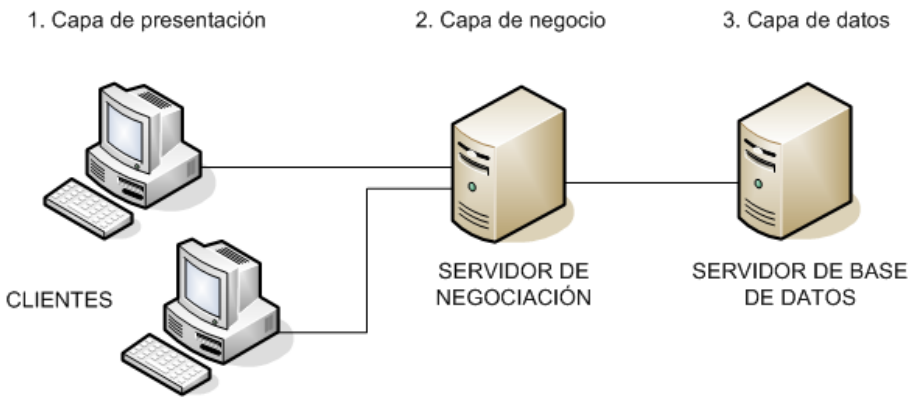


Figura 37: Arquitectura por capas.
Fuente: http://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas#mediaviewer/File:Tres_capas.PNG

2.4.1. Lenguaje del lado del servidor PHP.

Es un lenguaje de programación que nace en el año 1994. Creado por Rasmus Lerdorf en Perl con base de un conjunto simple de ficheros binarios escritos en C para mostrar su hoja de vida y guardar ciertos datos importantes como el tráfico que recibía su página web. A este conjunto de scripts lo llamo "Personal Home Page Tools". Desde la versión PHP 3 se cambió el nombre a PHP: Hypertext Preprocesor, se dio soporte a la programación orientada a objetos y una sintaxis de lenguaje mucho más potente, las siguientes versiones se fue mejorando PHP y el manejo de objetos.

Actualmente PHP se encuentra en la versión 5.6.3, es uno de los lenguajes de programación más utilizados en sitios web, entre ellos Facebook, Yahoo!, Wikipedia, WordPress. La principal razón es que se caracteriza por ser rápido, flexible, de código abierto y funciona en cualquier plataforma. La siguiente Figura es el logo de PHP.

Figura 38: Logo de PHP.
Fuente: http://php.net/

Se utilizará PHP como lenguaje del lado del servidor para realizar consultas a la base de datos, crear registros, modificarlos. También para crear los objetos y clases del sistema.

2.4.2. Lenguaje del lado del cliente JavaScript.

Es el lenguaje de programación de la web, la mayoría de sitios web y navegadores de computadoras, tablets, Smart phones incluyen intérpretes de javascript, esta es la característica más importante por la cual utilizar javascript. Es una de las tres tecnologías que todo desarrollador web debe saber entre ellas HTML para definir el contenido de una página web, CSS para especificar la presentación de una página web y javascript para determinar el comportamiento de una página web.

Fue creado por Netscape durante los primeros días de la web, es una marca registrada por Oracle, es un lenguaje orientado a objetos, ligero, no es útil como lenguaje independiente, comparte su sintaxis con java pero es completamente diferente. Posee un conjunto central de objetos como arreglos fecha y objetos matemáticos, además de operadores, estructuras de control y sentencias.

Javascript del lado del cliente permite tener un control del navegador y su modelo de objeto documentado, como por ejemplo podemos ubicar elementos de un formulario HTML y responder a los eventos del usuario.

2.4.3. Sistema de gestión de bases de datos.

MySQL Es un sistema de gestión de base de datos muy conocido y ampliamente utilizado por su simplicidad, rendimiento y distribución por internet bajo licencia GPL. MySQL surge en el año de 1990, creada por la empresa sueca MySQL AB como una interfaz SQL compatible a Mini SQL que no tenía ni flexibilidad ni rapidez. Inicialmente MySQL carecía de algunos elementos esenciales de las bases de datos relacionales, pero debido a su facilidad de uso llamo la atención de desarrolladores de páginas web con contenido dinámico, de tal forma se fue incorporando poco a poco todos estos elementos faltantes por desarrolladores internos y desarrolladores de software libre.

Entre sus principales características tenemos que su principal objetivo es la rapidez y la robustez, tiene una muy buena velocidad de respuesta, posee un amplio conjunto de tipos de datos, su administración se basa en usuarios y privilegios, es altamente confiable en cuanto a estabilidad, puede trabajar en distintos sistemas operativos, el servidor soporta mensajes de error en cualquier lengua. No necesita un computador con una gran cantidad de recursos. La siguiente Figura es el logo de MySQL.

Figura 39: Logo MySql.
Fuente: http://www.mysql.com/

2.4.4. Servidor Web Apache.

Apache es uno de los servidores web más utilizados desde abril de 1996, desarrollado por la Apache Software Foundation. La siguiente Figura es el logo de Apache.

Figura 40: Logo de Apache
Fuente: www.apache.org

Algunas de sus ventajas es que es de código abierto, utiliza la licencia Apache creada por la Apache Software Foundation que permite al usuario la libertad de usarlo para cualquier propósito, distribuirlo, modificarlo, y distribuir versiones modificadas de este software, es multiplataforma ya que se puede utilizar con Windows, Linux, Unix, Solaris, Mac OSX, NetWare, entre otros. Es fácil conseguir ayuda o soporte porque es un servidor web popular. También posee varios módulos que permite tener mayores funcionalidades en un servidor web. Actualmente se encuentra en la versión 2.4.10 y es posible descargarla desde la página web http://httpd.apache.org/. La siguiente Figura es la página web de apache.

Figura 41: Página web de Apache.
Fuente:www.apache.org

El servidor web apache es el que se encargará de recibir las peticiones de los usuarios que deseen acceder al sistema y gestionar su entrega o denegación de acuerdo a políticas de seguridad establecidas.

Finalmente se presenta a continuación una tabla de resumen de las herramientas utilizadas.

HERRAMIENTA	SELECCIÓN	RAZÓN
Metodología de desarrollo de software	EXTREMME PROGRAMMING	Se eligió XP por ser una metodología ágil, que reduce la cantidad de documentación y el tiempo de desarrollo.
Lenguaje de programación	PHP/JAVASCRIPT	PHP es un lenguaje potente, popular y gratuito. JAVASCRIPT se utiliza para mejorar la interfaz de usuario.
Sistema de gestión de base de datos	MYSQL	No se requiere pagar por utilizar la herramienta, tiene facilidad de configuración e instalación, soporta varios sistemas operativos.
Servidor HTTP	APACHE	Apache es un popular componente en una infraestructura donde se utiliza php y mysql.

Figura 42: Cuadro resumen de herramientas.

3. CASO DE ESTUDIO: Implementación de una solución informática de manejo de estadísticas, para el Laboratorio de Microbiología del Hospital del IESS Carlos Andrade Marín.

3.1. Planeación del proyecto:

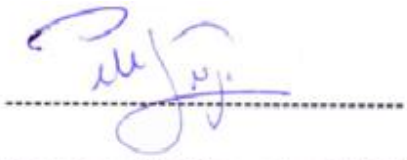
La metodología XP nos indica que la planeación es la primera actividad que se debe realizar, nos sirve para entender los requerimientos del cliente al desarrollar un producto de software, en esta primera etapa se documentará las necesidades para la implementación de una solución informática de manejo de estadísticas, para el laboratorio de microbiología del Hospital del IESS Carlos Andrade Marín. Como se observó anteriormente en el capítulo 2.2, la forma para comenzar un proyecto es con las historias de usuario. Es por eso que a continuación se presentan las que se utilizarán para el desarrollo del caso de estudio.

3.1.1. Historias de usuario:

Estas son las funcionalidades que estarán presentes en el sistema, y se las implementará de acuerdo a una división de iteraciones.

1. Administración de usuario.

Historia de Usuario	
Número: 1	**Nombre:** Administración de usuario
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:**1
Prioridad en Negocio: Alta (Alta/Media/Baja)	**Puntos Estimados:** 0.3
Riesgo en Desarrollo: Media (Alta/Media/Baja)	**Puntos Reales:** 0.4
Descripción: El sistema tendrá la opción para administrar usuarios, al ingresar en esta opción se desplegará un listado con la información de cada usuario, su nombre, apellido, contraseña y perfil. Se podrá ingresar la información de un usuario nuevo, modificar o eliminar su información. Además el sistema permitirá el acceso a usuarios que se hayan autentificado con su respectivo usuario y contraseña.	
Observaciones:	

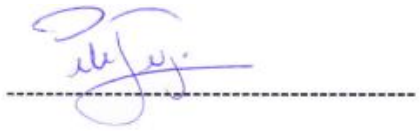


Responsable Proyecto HCAM

Responsable Desarrollo Software

2. Administración de paciente.

Historia de Usuario	
Número: 2	**Nombre:** Administración de paciente
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:**1
Prioridad en Negocio: Alta (Alta/Media/Baja)	**Puntos Estimados:** 0.3
Riesgo en Desarrollo: Media (Alta/Media/Baja)	**Puntos Reales:** 0.4
Descripción: El sistema tendrá la opción administrar pacientes, se encontrará dentro de un módulo de pre análisis, al ingresar en esta opción se desplegará un listado con la información de cada paciente, se podrá ingresar un nuevo paciente, modificar o eliminar.	
Observaciones: Es importante tomar como una base, la información de los pacientes que se utiliza en el programa de control de la tuberculosis del ministerio de salud pública.	

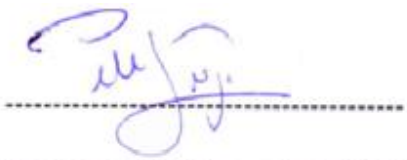


Responsable Proyecto HCAM

Responsable Desarrollo Software

3. Administración de muestra.

Historia de Usuario	
Número: 3	**Nombre:** Administración de muestra
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:**1
Prioridad en Negocio: Alta (Alta/Media/Baja)	**Puntos Estimados:** 0.7
Riesgo en Desarrollo: Alta (Alta/Media/Baja)	**Puntos Reales:** 0.9
Descripción: El sistema tendrá la opción de administrar las muestras de cada paciente que se van a examinar, está opción se encontrará dentro de un módulo de pre análisis, al ingresar en esta opción se desplegará un listado con las muestras que corresponden a cada paciente. Se podrá ingresar la información de una nueva muestra, modificarla o eliminarla.	
Observaciones: Se requiere que se realicen los exámenes de varias muestras siguiendo un orden específico que depende del tipo de muestra.	

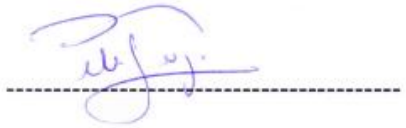


Responsable Proyecto HCAM

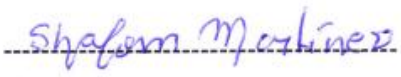
Responsable Desarrollo Software

4. Administración de análisis de muestra.

Historia de Usuario	
Número: 4	**Nombre:** Administración de análisis de muestra
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:**1
Prioridad en Negocio: Alta (Alta/Media/Baja)	**Puntos Estimados:**0.7
Riesgo en Desarrollo: Alta (Alta/Media/Baja)	**Puntos Reales:** 0.8
Descripción: El sistema tendrá la opción de administrar los análisis de muestra, la cual se encontrará dentro de un módulo de análisis, esta opción permitirá ingresar el resultado de los análisis realizados a las muestras de los pacientes, modificarlos y eliminarlos.	
Observaciones: Existen varios tipos de exámenes que se realizan bajo ciertas condiciones.	



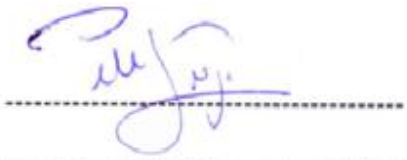
Responsable Proyecto HCAM



Responsable Desarrollo Software

5. Administración de institución

Historia de Usuario	
Número: 5	**Nombre:** Administración de institución
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:**1
Prioridad en Negocio: Media (Alta/Media/Baja)	**Puntos Estimados:**0.2
Riesgo en Desarrollo: Baja (Alta/Media/Baja)	**Puntos Reales:** 0.3
Descripción: El sistema tendrá la opción de administrar las instituciones públicas y privadas que atienden casos de tuberculosis. Se podrá ingresar la información de una institución, modificarla o eliminarla.	
Observaciones:	

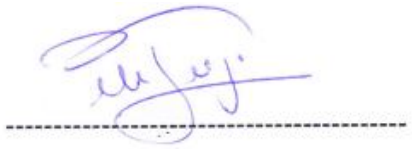


Responsable Proyecto HCAM

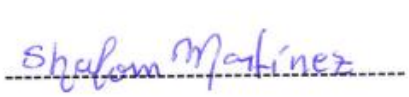
Responsable Desarrollo Software

6. Administración de establecimiento de salud

Historia de Usuario	
Número: 6	**Nombre:** Administración de establecimiento de salud
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:**1
Prioridad en Negocio: Media (Alta/Media/Baja)	**Puntos Estimados:**0.2
Riesgo en Desarrollo: Baja (Alta/Media/Baja)	**Puntos Reales:** 0.3
Descripción: El sistema tendrá la opción de administrar los establecimientos de salud que pertenecen a las diferentes instituciones. Se debe permitir ingresar un nuevo establecimiento de salud, modificar o eliminar.	
Observaciones:	



Responsable Proyecto HCAM



Responsable Desarrollo Software

7. Administración de programación

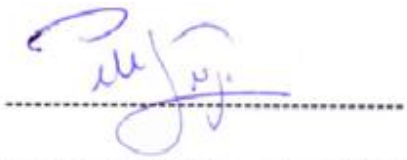
Historia de Usuario	
Número: 7	**Nombre:** Administración de programación
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:**2
Prioridad en Negocio: Baja (Alta/Media/Baja)	**Puntos Estimados:** 0.2
Riesgo en Desarrollo: Baja (Alta/Media/Baja)	**Puntos Reales:**
Descripción: El sistema tendrá la opción de realizar la programación de las actividades de control de la tuberculosis. Se podrá ingresar la información de una nueva programación, modificarla o eliminarla.	
Observaciones: En reuniones con el cliente se ha acordado que por ser una historia de baja prioridad, su implementación queda postergada para el próximo reléase de la aplicación	

Responsable Proyecto HCAM

Responsable Desarrollo Software

8. Administración de contacto

Historia de Usuario	
Número: 8	**Nombre:** Administración de contacto
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:** 1
Prioridad en Negocio: Media (Alta/Media/Baja)	**Puntos Estimados:** 0.2
Riesgo en Desarrollo: Baja (Alta/Media/Baja)	**Puntos Reales:** 0.2
Descripción: El sistema tendrá la opción de administrar la información de los contactos que posee un paciente. Se podrá ingresar la información de un contacto nuevo, modificarla o eliminarla.	
Observaciones:	

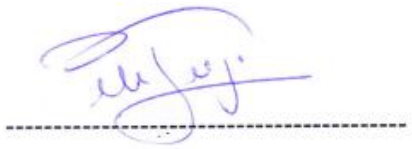


Responsable Proyecto HCAM

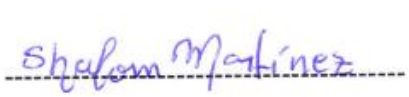
Responsable Desarrollo Software

9. Administración de laboratorio

Historia de Usuario	
Número: 9	**Nombre:** Administración de laboratorio
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:**1
Prioridad en Negocio: Media (Alta/Media/Baja)	**Puntos Estimados:** 0.3
Riesgo en Desarrollo: Baja (Alta/Media/Baja)	**Puntos Reales:** 0.3
Descripción: El sistema tendrá la opción de administrar la información de los laboratorios que analizarán las muestras de un paciente, se podrá ingresar la información de un laboratorio, modificarla o eliminarla.	
Observaciones:	



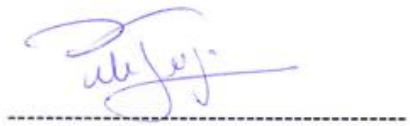
Responsable Proyecto HCAM



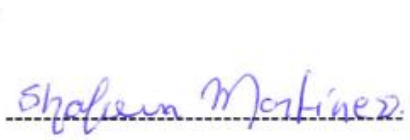
Responsable Desarrollo Software

10. Administración de diagnóstico

Historia de Usuario	
Número: 10	**Nombre:** Administración de diagnóstico
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:**1
Prioridad en Negocio: Alta (Alta/Media/Baja)	**Puntos Estimados:**0.7
Riesgo en Desarrollo: Media (Alta/Media/Baja)	**Puntos Reales:** 0.4
Descripción: El sistema tendrá la opción de ingresar el diagnóstico de un paciente, y que seguimiento se le va a realizar. También modificar un diagnóstico o eliminarlo.	
Observaciones:	



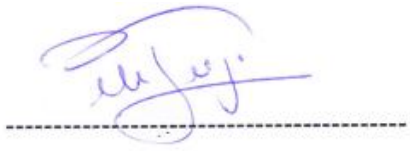
Responsable Proyecto HCAM



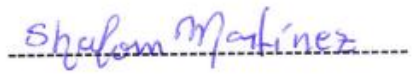
Responsable Desarrollo Software

11. Administración de seguimiento

Historia de Usuario	
Número: 11	**Nombre:** Administración de seguimiento
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:**1
Prioridad en Negocio: Alta (Alta/Media/Baja)	**Puntos Estimados:** 0.7
Riesgo en Desarrollo: Media (Alta/Media/Baja)	**Puntos Reales:** 0.3
Descripción: El sistema tendrá la opción de administrar el seguimiento de los pacientes con tuberculosis dependiendo del resultado y el tipo de paciente se le asignará el esquema 1 o el esquema 2. Para el esquema 1 se realizará un seguimiento de 6 meses, para el esquema 2 un seguimiento de 8 meses. Cada mes se debe realizar una baciloscopia y en el esquema 1 en el mes 2 un cultivo si la baciloscopia es positiva, en el esquema 2 en el mes 3. Finalmente se debe ingresar el resultado del seguimiento.	
Observaciones:	



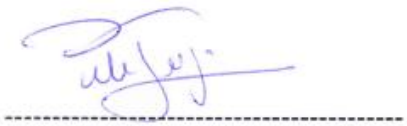
Responsable Proyecto HCAM



Responsable Desarrollo Software

12. Administración de control de calidad

Historia de Usuario	
Número: 12	**Nombre:** Administración de control de calidad
Usuario: Cliente	
Modificación de Historia Número:	**Iteración Asignada:** 2
Prioridad en Negocio: Alta (Alta/Media/Baja)	**Puntos Estimados:** 0.5
Riesgo en Desarrollo: Media (Alta/Media/Baja)	**Puntos Reales:**
Descripción: El sistema tendrá la opción de ingresar un control de calidad de un análisis de muestra, y modificarla dependiendo de su perfil.	
Observaciones:	



Responsable Proyecto HCAM

Responsable Desarrollo Software

3.1.2. Plan de entrega

El plan de entrega nos permite visualizar la prioridad que tiene cada historia de usuario, los puntos estimados que representan el número de semanas para desarrollar cada historia y en que iteración se las va a realizar, a continuación se presenta una tabla con el plan de entrega que se utilizará para el caso de estudio.

HISTORIA DE USUARIO	PRIORIDAD	PUNTOS ESTIMADOS	ITERACIÓN
Administración de usuario	1	0.3	1
Administración de paciente	1	0.3	1
Administración de muestra	1	0.7	1
Administración de institución	1	0.2	1
Administración de establecimiento de salud	1	0.2	1
Administración de programación	3	0.2	1
Administración de contacto	2	0.2	1
Administración de laboratorio	1	0.3	1
Administración de análisis de muestra	1	0.7	2
Administración de diagnóstico	1	0.7	2
Administración de seguimiento	1	0.7	2
Administración de control de calidad	2	0.5	2

Tabla 1: Plan de entrega del proyecto

3.1.3. División de Iteraciones.

Extreme programming nos indica que el proyecto debe ser dividido en varias iteraciones que duran entre 1 y 3 semanas. Para este caso de estudio se lo dividió en dos iteraciones una de 3 semanas y otra de 2 semanas. Es decir que se realizarán dos entregas de partes funcionales del sistema.

3.1.4. Velocidad del proyecto.

La velocidad del proyecto son el número de historias que se van a realizar por cada iteración a continuación se presenta una tabla con cada iteración y su número respectivo de historias de usuario.

ITERACIÓN	NÚMERO DE HISTORIAS DE USUARIO
1	8
2	4

Tabla 2: Velocidad del proyecto

3.2. Diseño del proyecto

3.2.1. Diseños simples.

El diseño del sistema debe ser simple, el objetivo del diseño es entender las funcionalidades del sistema que vamos construir. Para poder visualizar, especificar, documentar, construir este sistema se va a utilizar UML por ser el lenguaje de modelado más conocido y utilizado en la actualidad.

- Diagramas de casos de uso:
  Los siguientes diagramas representan al sistema de la forma en que lo perciben los usuarios externos llamados actores, cada funcionalidad del sistema es representada como un caso de uso.

Diagrama general:
En este diagrama se va a modelar las funcionalidades que tendrá el sistema de forma general, sin incurrir en sus detalles para visualizar las relaciones entre las funcionalidades y actores.

El sistema constará con los siguientes actores:

- Director
- Administrador
- Médico
- Supervisor
- Laboratorista

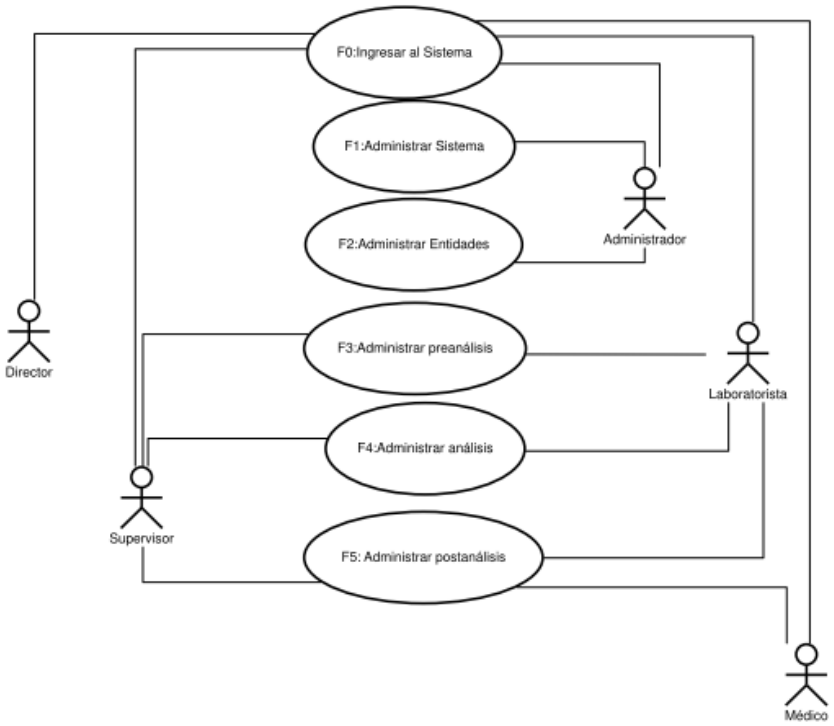


Figura 43: Diagrama general

Primera Iteración

A continuación se va a presentar diagramas de casos de uso por cada funcionalidad. En la primera iteración se diseñaron las siguientes funcionalidades:

F1: Administrar sistema

Definición: A través de esta operación el sistema permitirá el ingreso de los actores al sistema.

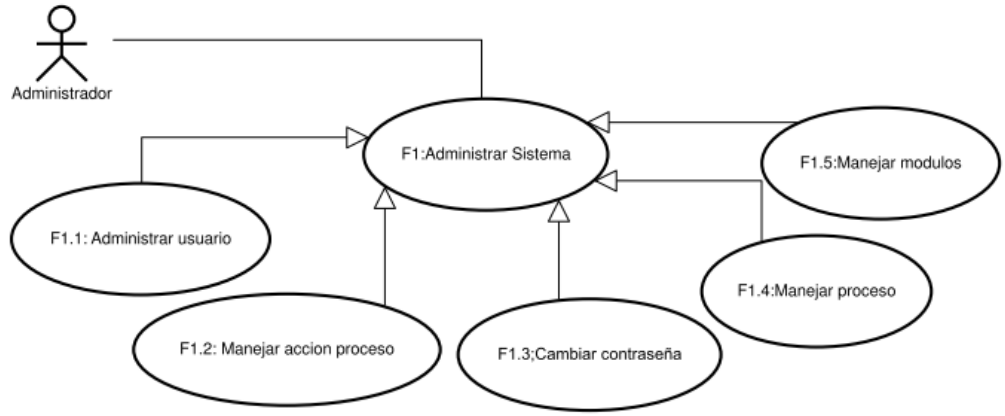


Figura 44: Administrar sistema

F2: Administrar entidades

Definición: A través de esta operación el sistema permitirá el manejo de información de cada institución y establecimiento de salud.

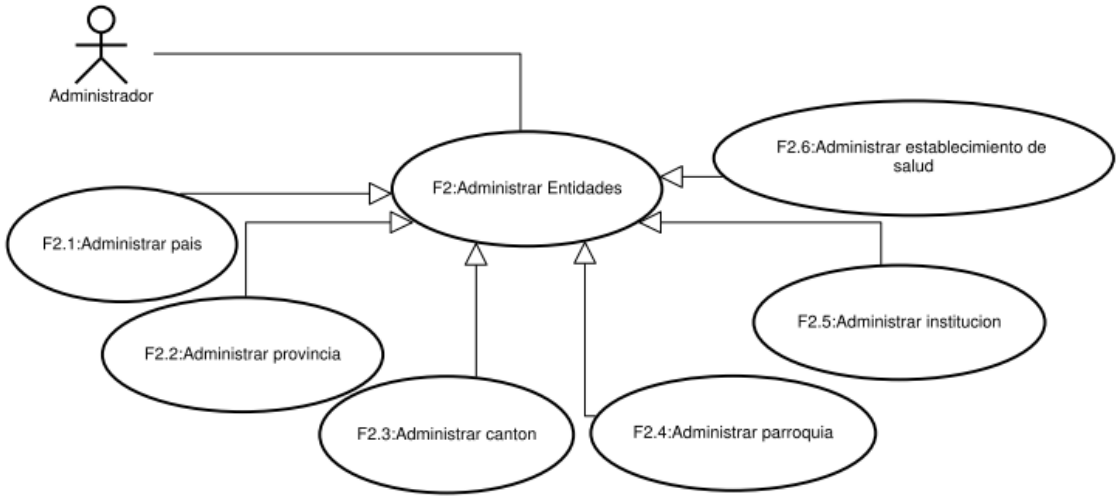


Figura 45: Administrar entidades

F3: Administrar pre análisis

Definición: A través de esta operación el sistema permitirá realizar un pre análisis de la información de un paciente.

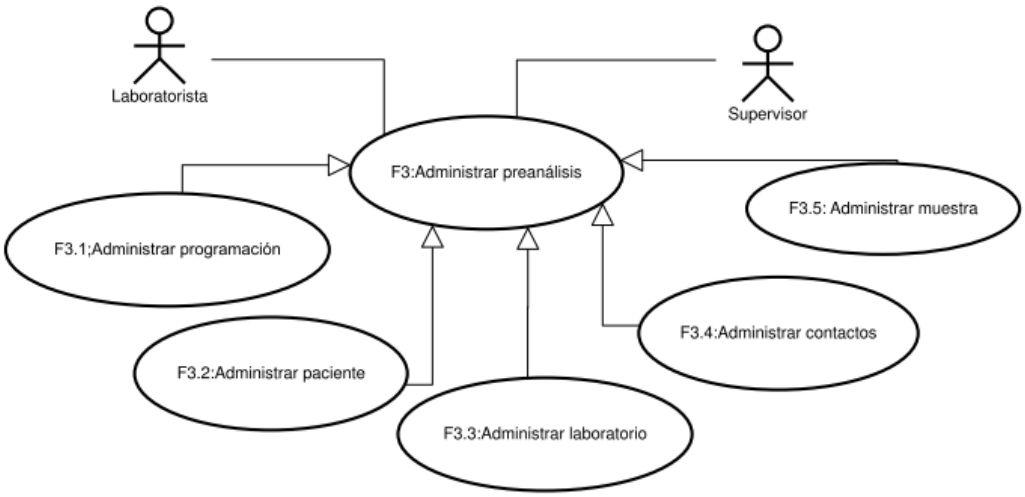


Figura 46: Administrar pre análisis

3.2.2. Tarjetas CRC.

Las tarjetas de Clases Responsabilidades y Colaboración son una herramienta, como vimos en el capítulo 2.2, que nos permite representar los objetos, clases y relaciones que van a existir en el sistema. A continuación se presenta las tarjetas CRC de la primera iteración del sistema.

1. Usuario

NOMBRE: USUARIO	
RESPONSABILIDADES:	**COLABORADORES:**
Ingresar usuario	TODAS LAS OTRAS CLASES
Modificar usuario	
Eliminar usuario	
Permitir el acceso de un usuario al sistema	
Determinar el usuario que ingresa información al sistema.	

2. Paciente

NOMBRE: PACIENTE	
RESPONSABILIDADES:	**COLABORADORES:**
Ingresar paciente	MUESTRA
Modificar paciente	DIAGNOSTICO
Eliminar paciente	CONTACTO
	ENFERMEDAD

3. Muestra

NOMBRE: MUESTRA	
RESPONSABILIDADES:	**COLABORADORES:**
Ingresar la información de una muestra	PACIENTE
Modificar muestra	LABORATORIO
Eliminar muestra	ANÁLISIS MUESTRA

4. Institución

NOMBRE: INSTITUCIÓN	
RESPONSABILIDADES:	**COLABORADORES:**
Ingresar Institución	ESTABLECIMIENTO DE SALUD
Modificar de una Institución	
Eliminar Institución	

5. Establecimiento de salud

NOMBRE: ESTABLECIMIENTO DE SALUD	
RESPONSABILIDADES:	**COLABORADORES:**
Ingresar establecimiento de salud	INSTITUCIÓN
Modificar establecimiento de salud	PARROQUIA
Eliminar establecimiento de salud	ENFERMEDAD

6. Contacto

NOMBRE: CONTACTO	
RESPONSABILIDADES:	**COLABORADORES:**
Ingresar contacto	PACIENTE
Modificar contacto	
Eliminar contacto	

7. Laboratorio

NOMBRE: LABORATORIO	
RESPONSABILIDADES:	**COLABORADORES:**
Ingresar laboratorio	MUESTRA
Modificar laboratorio	
Eliminar laboratorio	

3.2.3. Diagrama de actividades

Un diagrama de actividades representa una actividad: cada paso de un flujo de trabajo o la ejecución de una operación. A continuación se representará los procesos reales más importantes del caso de estudio.

La Figura 46 representa el proceso para obtener el diagnóstico de un paciente, que permite determinar si el paciente tiene tuberculosis, los actores que se encuentran relacionados son: el paciente a quien se va a realizar los exámenes, el laboratorista quien va a analizar las muestras obtenidas de un paciente, y el médico que va a dar su criterio en cuanto al resultado.

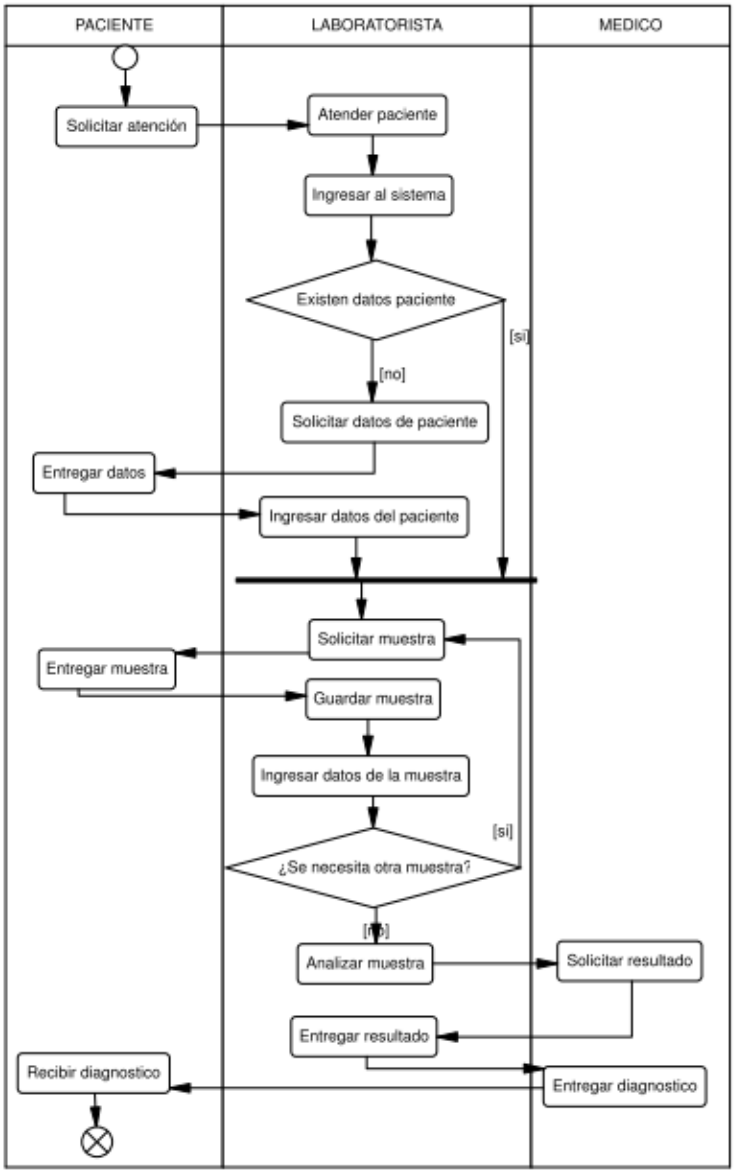


Figura 47: Diagrama actividades diagnóstico.

3.2.4. Diagrama conceptual

Este diagrama se lo realiza antes del diagrama entidad relación, es una representación del sistema donde se puede observar las clases que intervienen en el dominio del problema y sus relaciones, no se incluyen atributos debido a que es necesario tener una visión general del problema para obtener una solución adecuada.

La Figura 47 corresponde al diagrama conceptual que se va a utilizar para el desarrollo de software.

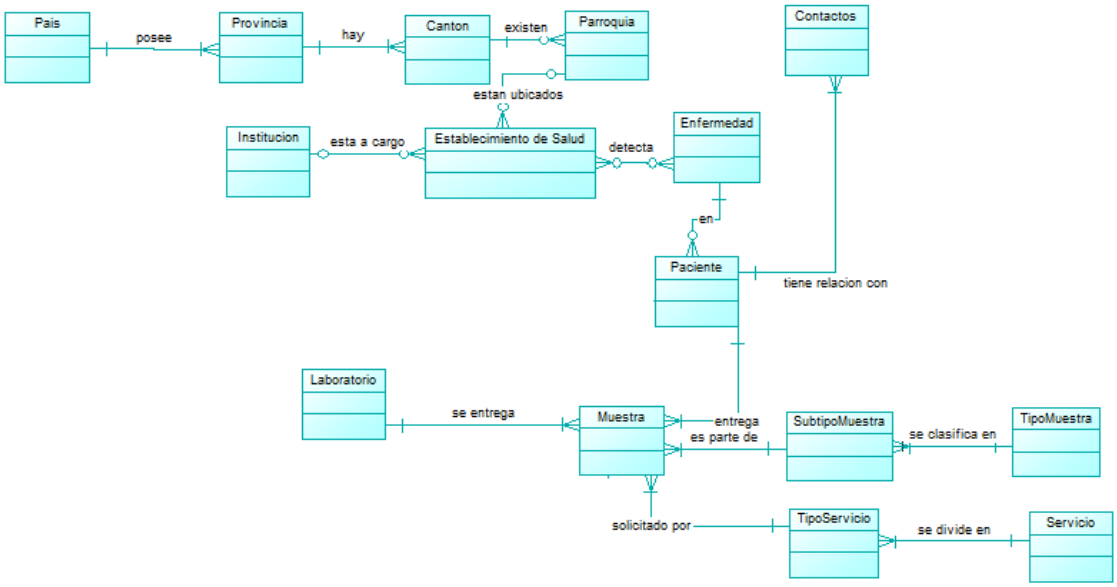


Figura 48: Diagrama conceptual.

3.2.5. Diagrama entidad relación:

Representa el sistema como una percepción del mundo real, que consiste en un conjunto de objetos llamados entidades y sus relaciones. Cada entidad posee sus respectivos atributos que son características o propiedades que identifican a cada objeto, y tienen un valor en un momento determinado. A continuación se presentan 3 Figuras que corresponden a los diagramas de entidad relación con los cuales se va a desarrollar el caso de estudio.

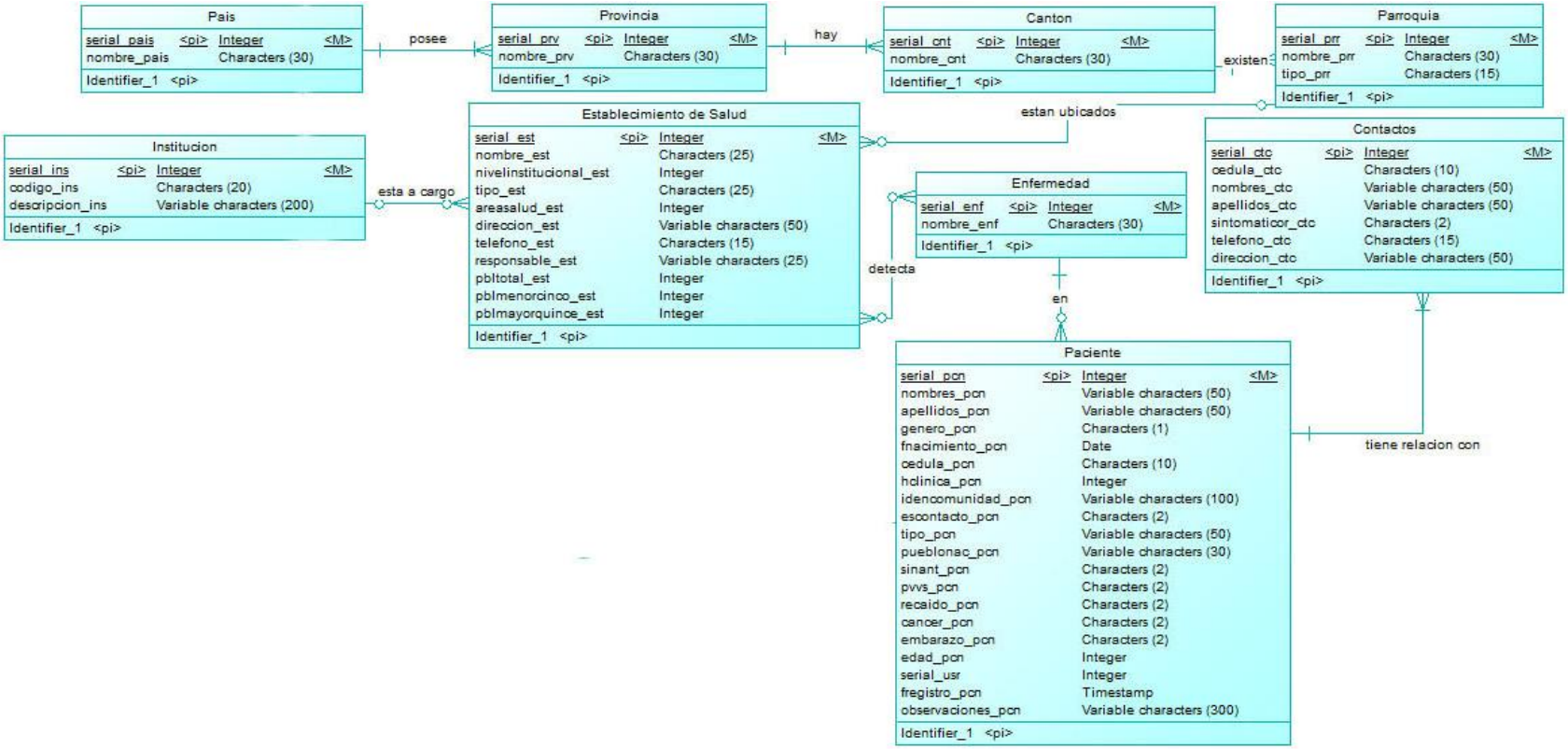


Figura 49: Diagrama entidad relación primera parte

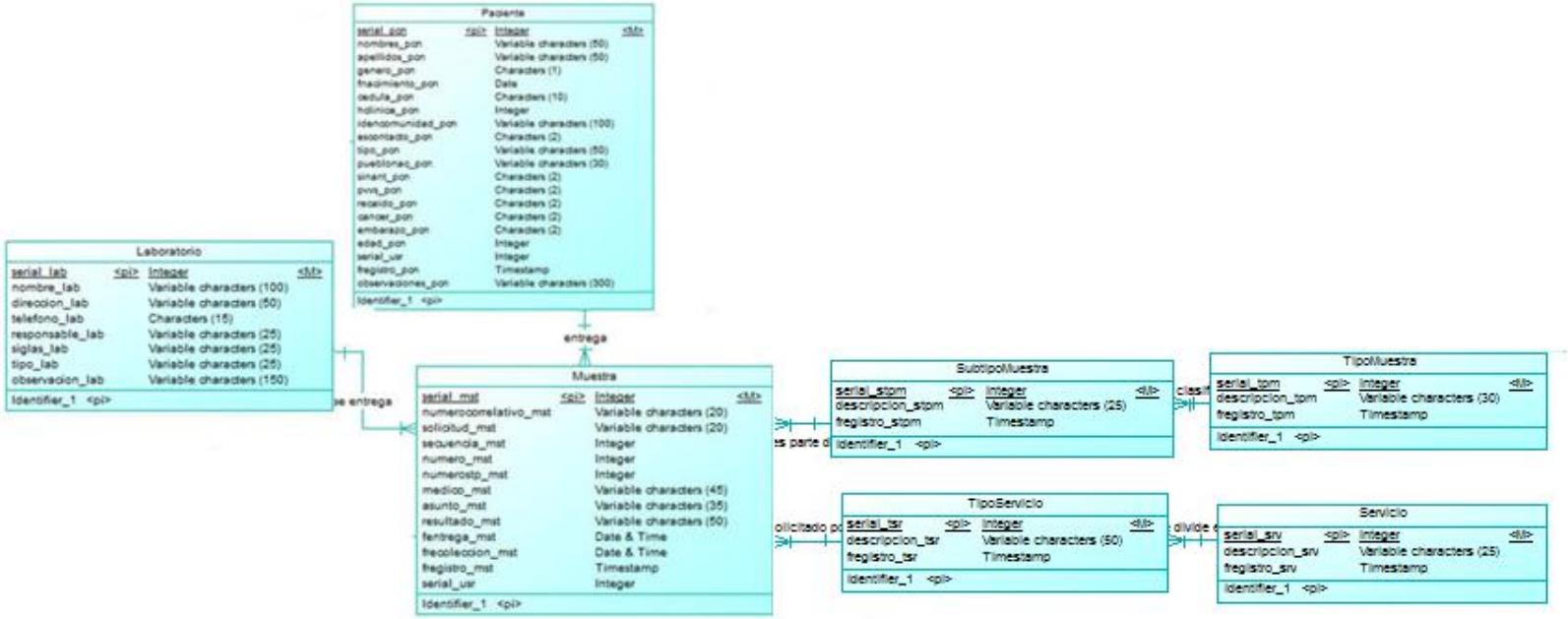


Figura 50: Diagrama entidad relación segunda parte

3.2.6. Prototipo

Un prototipo es una solución para poder identificar las necesidades de los usuarios, durante el desarrollo de un sistema informático. Es un programa muy simple que nos permite explorar soluciones. Para el desarrollo del caso de estudio fue de gran ayuda, porque proporcionó una visualización de cómo sería el sistema, el entendimiento y acuerdo de los requerimientos de usuario, y una retroalimentación general. A continuación se visualizan las Figuras 53 y 54 que representan el prototipo del sistema.

Figura 51: Pantalla inicio.

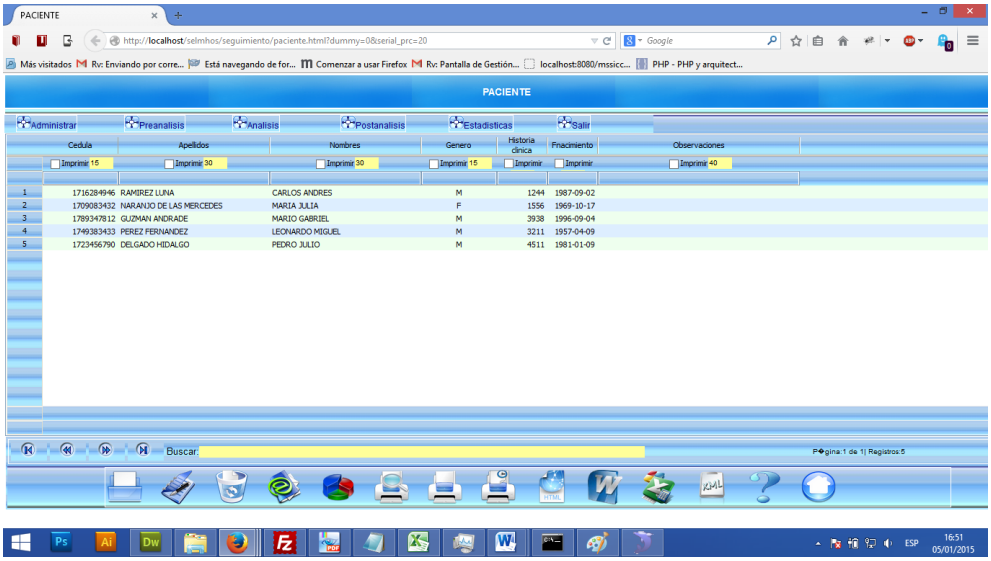


Figura 52: Pantalla administración paciente.

3.2.7. Glosario de términos.

Baciloscopia, es la técnica fundamental para la detección de casos de tuberculosis y para su control, la baciloscopia permite identificar al 70%-80% de casos pulmonares positivos

Contacto, es una persona que ha pasado tiempo con alguien que tiene tuberculosis en etapa infecciosa.

Cultivo, es una herramienta útil para el diagnóstico de tuberculosis en el caso de que existan pacientes con baciloscopia negativa y cuadro clínico sugestivo de TB.

Diagnóstico, se puede determinar que una persona tiene tuberculosis mediante las pruebas de baciloscopia y/o cultivo.

Establecimiento de salud, unidad de salud que pertenece a una institución la cual solicita un examen bacteriológico.

Institución, organismo público o privado que se encarga de atender un número importante de casos de tuberculosis

Muestra, cantidad de una sustancia utilizada para representar y analizar un material determinado.

Prueba de Sensibilidad, es una herramienta útil para detectar algunos casos de resistencia a fármacos en pacientes que presentan cultivos positivos.

Tipificación, es una prueba complementaria para cultivos positivos, por muestras extra pulmonares.

Tuberculosis (TB), es una enfermedad contagiosa causada por la bacteria "bacilo de koch"**,** afecta principalmente a los pulmones pero puede atacar otras partes del cuerpo causando pequeños tumores que destruyen los tejidos. Algunos de sus síntomas son la tos, fatiga, pérdida de peso, dificultad de respirar y la fiebre.

3.2.8. Riesgos.

Los riesgos de un proyecto son eventos o condiciones inciertas, que tienen un efecto positivo o negativo en un proyecto, generalmente se gestionan los riesgos negativos, debido a que suponen una amenaza para el proyecto.

Entre los riesgos del caso de estudio están:

- Cambio de requerimientos
- Diseño inadecuado
- Diferencias con los clientes
- El proyecto carece de promotores.
- Desarrollo de funciones inadecuadas o erróneas
- Problemas personales

3.2.9. Refactorizar

Refactorizar significa mejorar el diseño y la codificación de un sistema. A diferencia de otras metodologías en las cuales el diseño es global y completo, extreme programming nos indica que el diseño con el que vamos a empezar debe ser simple y general, para luego poder mejorarlo a medida que el proyecto avanza.

3.3. Codificación del proyecto.

Para la primera iteración se va a presentar toda la codificación en el CD que se va a adjuntar.

3.4. Pruebas del proyecto.

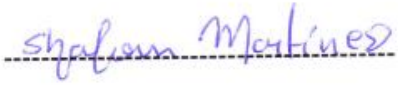
Las pruebas del sistema nos permiten verificar el resultado del sistema, decidir cuáles están bien, cuáles fallaron con el objetivo de obtener un software útil para los usuarios. A continuación se presentan las pruebas realizadas del sistema en la primera iteración. Para mayor información consultar el capítulo 2.2.

1. Administración de usuario:

Funcionalidad	Datos	Resultado esperado	Cumple
Ingresar datos de usuario	Serial, código de usuario, clave de ingreso, nombres, apellidos, estado, email, perfil, fecha de registro	Se insertan los datos en la tabla USUARIO.	SI
Modificar datos de usuario	Código de usuario, clave de ingreso, nombres, apellidos, estado, email, perfil, fecha de registro	Se modifican los datos de la tabla USUARIO de un usuario determinado	SI
Eliminar datos de usuario	Serial, código de usuario, clave de ingreso, nombres, apellidos, estado, email, perfil, fecha de registro	Se eliminan los datos de la tabla USUARIO del usuario seleccionado	SI
Ingresar al sistema	Código de usuario, clave de ingreso.	Se valida el código de usuario y clave con los datos que ya han sido guardados en la tabla USUARIO si son iguales se permite el ingreso al sistema en caso contrario no se permite.	SI

Bitácora de usuario	Código de usuario	Se registra el usuario y la acción que realizó en el sistema.	SI

Responsable Proyecto HCAM

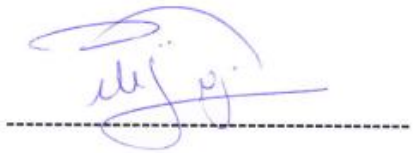


Responsable Desarrollo Software

2. Administración de paciente

Funcionalidad	**Datos**	**Resultado esperado**	**Cumple**
Ingresar datos de paciente	Serial, nombres, apellidos, cédula, género, fecha de nacimiento, historia clínica, es contacto, identificado en la comunidad, tipo de paciente, resistencia, nacionalidad, antecedentes, diagnostico	Se insertan los datos en la tabla PACIENTE	SI
Modificar datos de paciente	Nombres, apellidos, cédula, género, fecha nacimiento, historia clínica, es contacto, identificado en la comunidad, tipo de paciente, resistencia, nacionalidad, antecedentes, diagnostico.	Se modifican los datos de la tabla PACIENTE de un paciente determinado	SI
Eliminar datos de paciente	Nombres, apellidos, cédula, género, fecha nacimiento, historia clínica, es contacto, identificado en la comunidad, tipo de paciente, resistencia, nacionalidad, antecedentes, diagnostico	Se eliminan los datos de la tabla PACIENTE de un paciente determinado	SI

Color según tipo de paciente.	Antecedentes, fecha nacimiento.	Se visualiza un color dependiendo de los antecedentes, o edad del paciente	SI



Responsable Proyecto HCAM

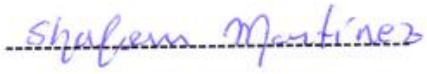
Responsable Desarrollo Software

3. Administración de muestra

Funcionalidad	**Datos**	**Resultado esperado**	**Cumple**
Ingresar datos de muestra	Serial, número correlativo, solicitud, secuencia, número de muestra, número de subtipo de muestra, médico solicitante, seguimiento, resultado, fecha entrega, fecha de recolección fecha de registro	Se insertan los datos en la tabla MUESTRA	SI
Modificar datos de muestra	Número correlativo, solicitud, secuencia, número de muestra, número de subtipo de muestra, médico solicitante, seguimiento, resultado, fecha entrega, fecha de recolección fecha de registro	Se modifican los datos de la tabla MUESTRA de una muestra determinada	SI

Eliminar datos de muestra	Serial, número correlativo, solicitud, secuencia, número de muestra, número de subtipo de muestra, médico solicitante, seguimiento, resultado, fecha entrega, fecha de recolección fecha de registro.	Se eliminan los datos de la tabla MUESTRA de una muestra determinada	SI

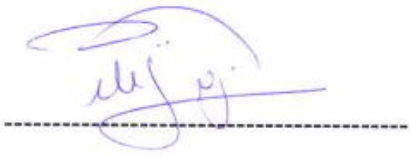
Responsable Proyecto HCAM



Responsable Desarrollo Software

4. Administración de institución

Funcionalidad	**Datos**	**Resultado esperado**	**Cumple**
Ingresar datos de institución	Serial, código, descripción	Se insertan los datos en la tabla INSTITUCION	SI
Modificar datos de institución	Código, descripción	Se modifican los datos de la tabla INSTITUCION de una institución determinada	SI
Eliminar datos de institución	Serial, código, descripción	Se eliminan los datos de la tabla INSTITUCION de una institución determinada	SI



Responsable Proyecto HCAM

Responsable Desarrollo Software

5. Administración de establecimiento de salud

Funcionalidad	Datos	Resultado esperado	Cumple
Ingresar datos de establecimiento de salud	Serial, nombre, nivel institucional, tipo, área de salud, dirección, teléfono, responsable	Se insertan los datos en la tabla ESTABLECIMIENTOSALUD	SI
Modificar datos de establecimiento de salud	Nombre, nivel institucional, tipo, área de salud, dirección, teléfono, responsable	Se modifican los datos de la tabla ESTABLECIMIENTOSALUD de un establecimiento determinado	SI
Eliminar datos de establecimiento de salud	Serial, nombre, nivel institucional, tipo, área de salud, dirección, teléfono, responsable	Se eliminan los datos de la tabla ESTABLECIMIENTO DE SALUD de un establecimiento determinado	SI

Responsable Proyecto HCAM

Responsable Desarrollo Software

6. Administración de contacto

Funcionalidad	Datos	Resultado esperado	Cumple
Ingresar datos de contacto	Serial, cédula, nombres, apellidos, sintomático, teléfono, dirección	Se insertan los datos en la tabla CONTACTO	SI
Modificar datos de contacto	Cédula, nombres, apellidos, sintomático, teléfono, dirección	Se modifican los datos de la tabla CONTACTO de un contacto determinado.	SI
Eliminar datos de contacto	Serial, cédula, nombres, apellidos, sintomático, teléfono, dirección	Se eliminan los datos de la tabla CONTACTO de un contacto determinado	SI

Responsable Proyecto HCAM

Responsable Desarrollo Software

7. Administración de laboratorio

Funcionalidad	Datos	Resultado esperado	Cumple
Ingresar datos de laboratorio	Serial, nombre, dirección, teléfono, responsable, siglas, tipo, observación	Se insertan los datos en la tabla LABORATORIO	SI
Modificar datos de laboratorio	Nombre, dirección, teléfono, responsable, siglas, tipo, observación	Se modifican los datos de la tabla LABORATORIO de un laboratorio determinado.	SI
Eliminar datos de laboratorio	Serial, nombre, dirección, teléfono, responsable, siglas, tipo, observación	Se eliminan los datos de la tabla LABORATORIO de un laboratorio determinado	SI

Responsable Proyecto HCAM

Responsable Desarrollo Software

Segunda iteración

3.5. Diseño del proyecto

3.5.1. Diseños simples

Para la segunda iteración se van a desarrollar las siguientes funcionalidades:

F4: Administrar análisis

Definición: A través de esta operación el sistema permitirá realizar un análisis de las muestras de un paciente.

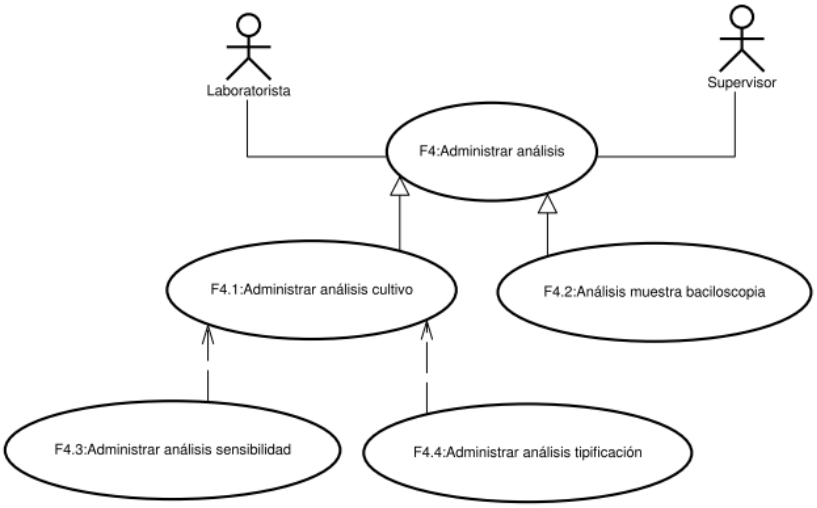


Figura 53: Administrar análisis

F5: Administrar post análisis

Definición: A través de esta operación el sistema permitirá realizar un post análisis de los resultados del paciente.

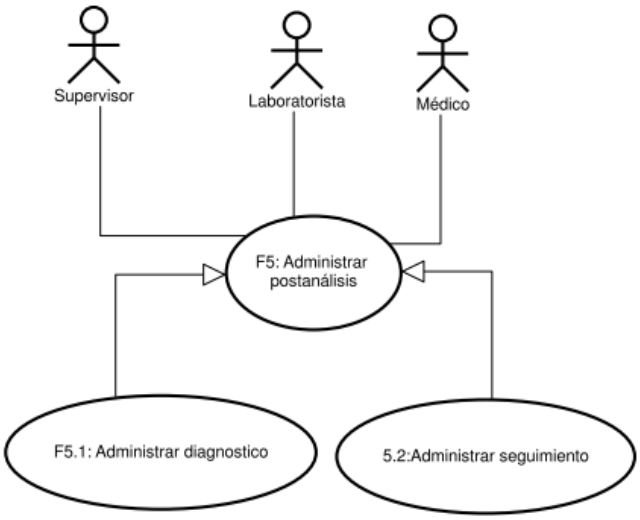


Figura 54: Administrar postanálisis

3.5.2. Tarjetas CRC.

En la segunda iteración se desarrollaron las siguientes tarjetas CRC.

1. Análisis muestra

NOMBRE: ANÁLISIS MUESTRA	
RESPONSABILIDADES:	**COLABORADORES:**
Ingresar examen	MUESTRA
Modificar examen	REVISION
Eliminar examen	

2. Diagnóstico

NOMBRE: DIAGNOSTICO	
RESPONSABILIDADES:	**COLABORADORES:**
Ingresar diagnóstico	PACIENTE
Modificar diagnóstico	SEGUIMIENTO
Eliminar diagnóstico	TRATAMIENTO

3. Seguimiento

NOMBRE: SEGUIMIENTO	
RESPONSABILIDADES:	**COLABORADORES:**
Ingresar Seguimiento	DIAGNOSTICO
Modificar Seguimiento	
Eliminar Seguimiento	

4. Tratamiento

NOMBRE: TRATAMIENTO	
RESPONSABILIDADES:	**COLABORADORES:**
Ingresar tratamiento	DIAGNOSTICO
Modificar tratamiento	
Eliminar tratamiento	

3.5.3. Diagrama conceptual

En la segunda iteración se aumentó en el diagrama conceptual las clases vistas anteriormente en las tarjetas CRC.

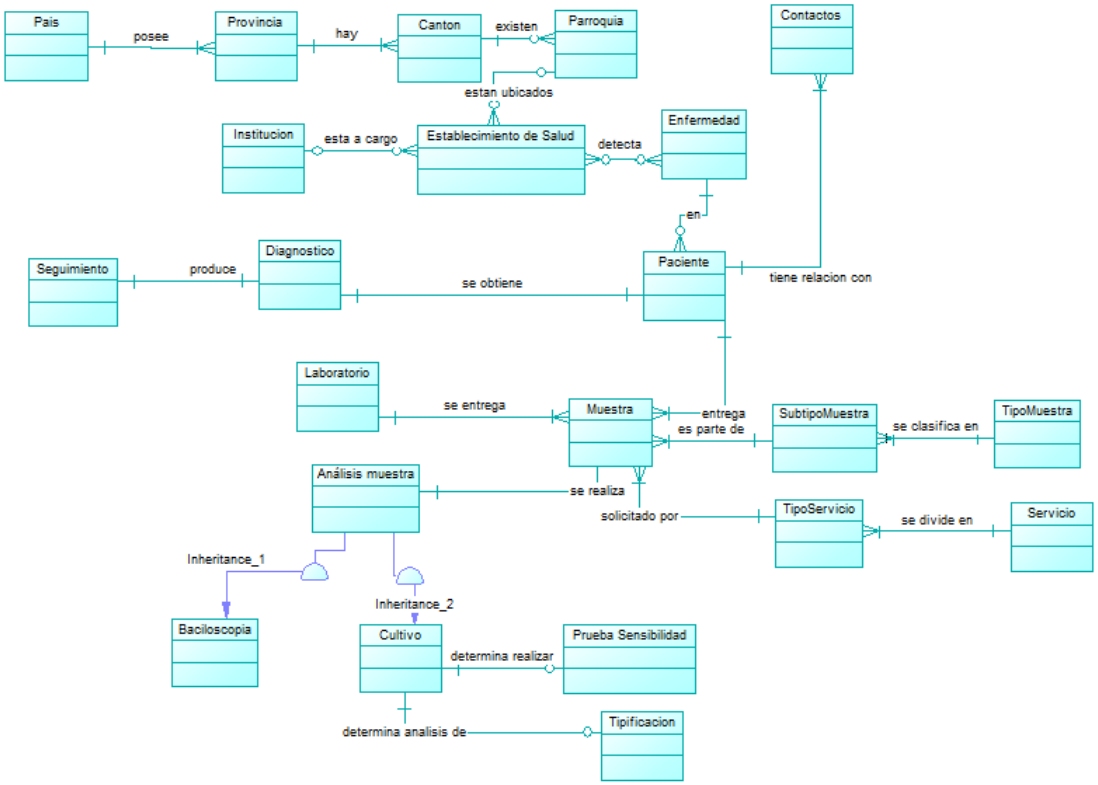


Figura 55: Diagrama conceptual.

3.5.4. Diagrama entidad relación

En la segunda iteración se presenta a continuación 3 Figuras donde se encuentran las clases y atributos de las funcionalidades implementadas.

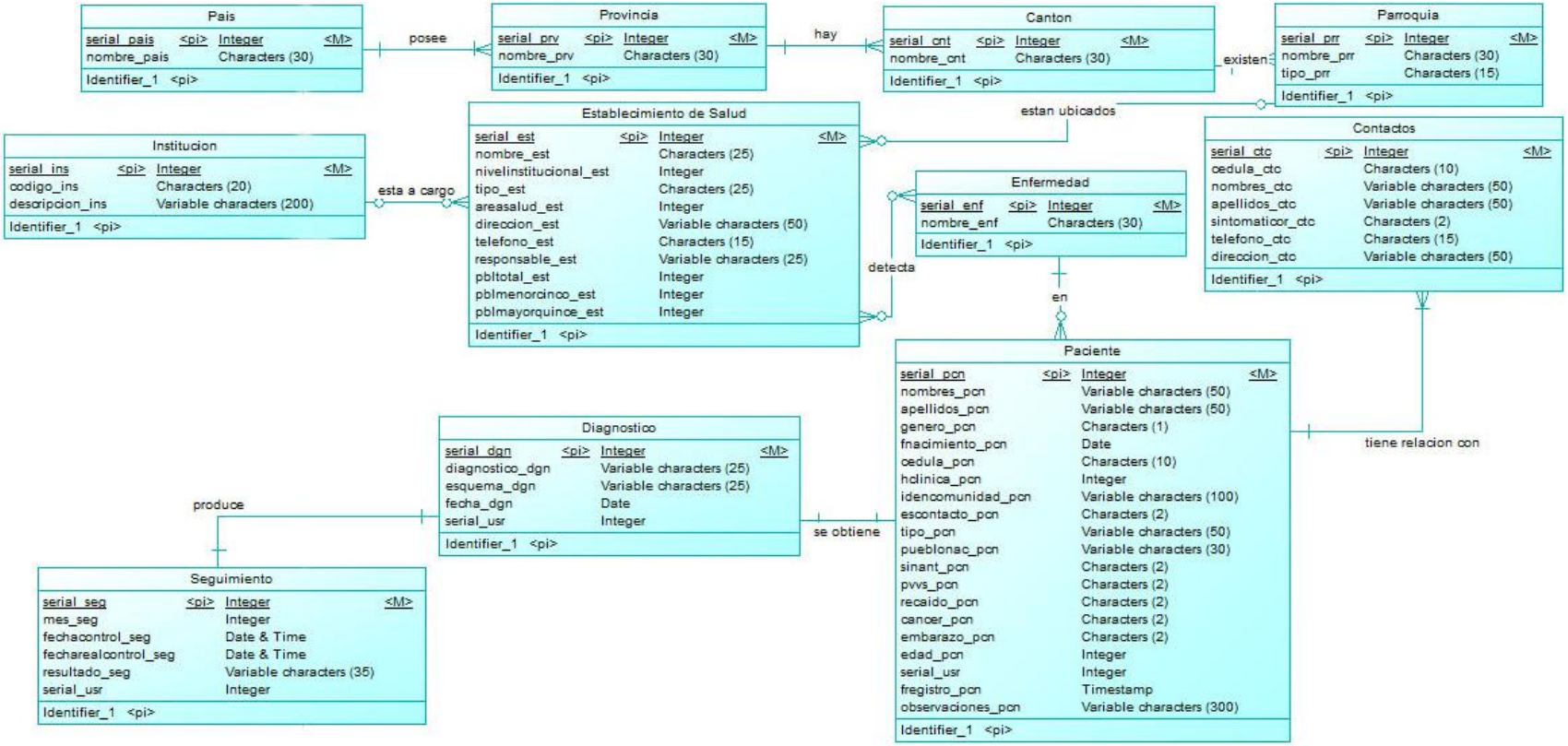


Figura 56: Diagrama entidad relación primera parte

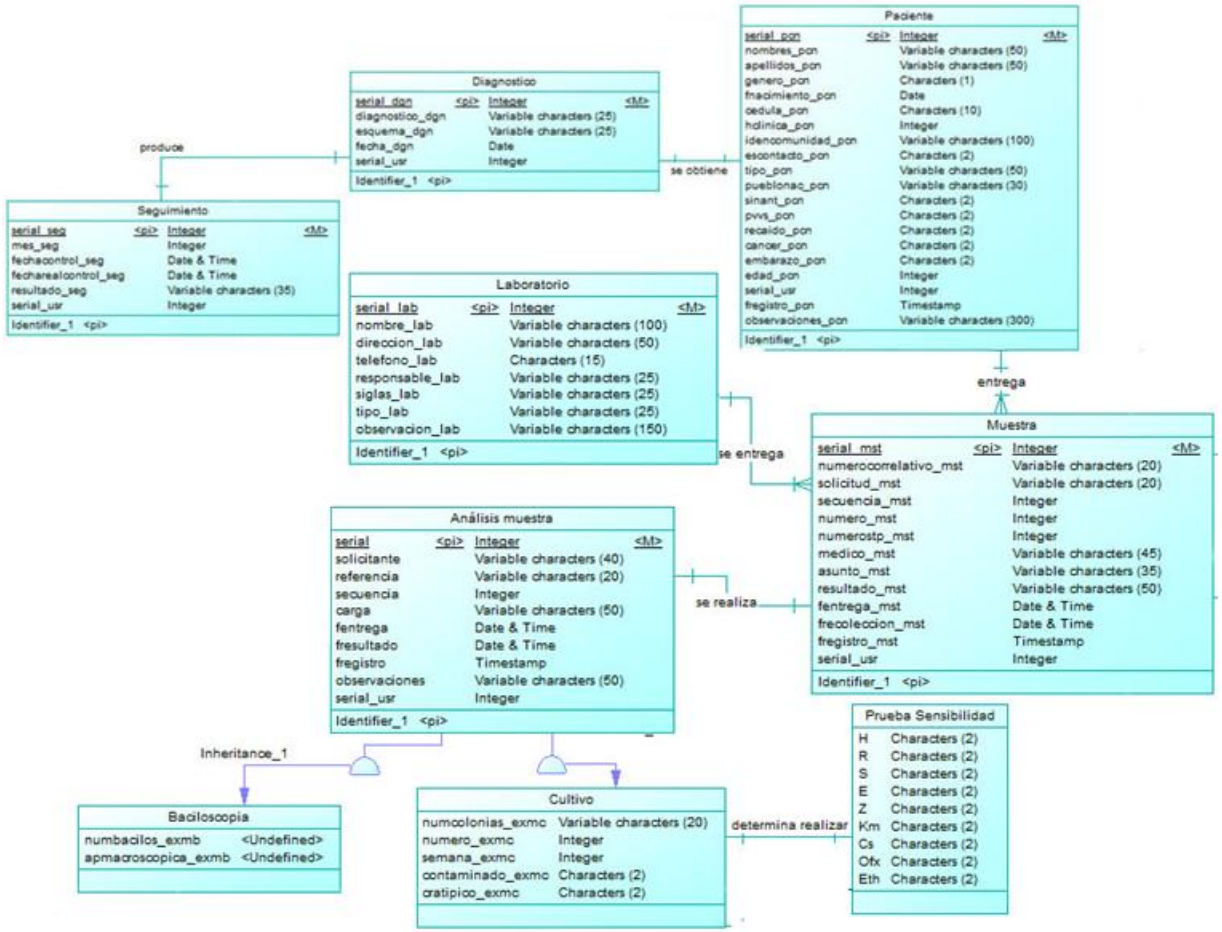


Figura 57: Diagrama entidad relación primera parte

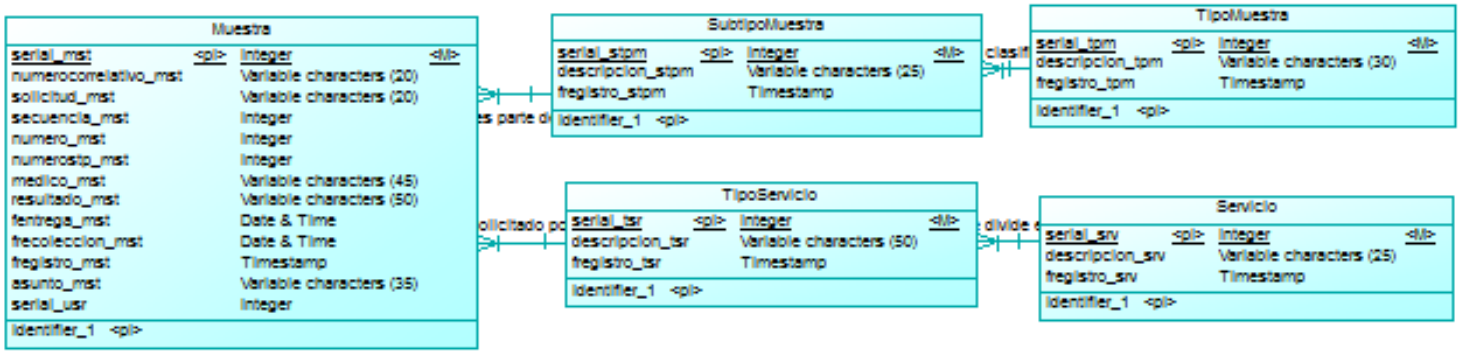


Figura 58: Diagrama entidad relación tercera parte

3.6. Codificación del proyecto.

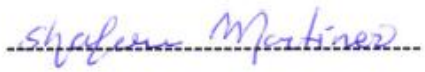
Para la segunda iteración se va a presentar la codificación de administración de baciloscopia, el resto estará en el CD que se va a adjuntar.

3.7. Pruebas del proyecto.

1. Administración de análisis de muestra
   1.1. Administración de baciloscopia

Funcionalidad	Datos	Resultado esperado	Cumple
Ingresar datos de baciloscopia	Serial, referencia, secuencia, numero de bacilos, apariencia macroscópica carga, fecha entrega, fecha resultado, fecha registro, observaciones	Se insertan los datos en la tabla EXAMENBACILOSCOPIA	SI
Modificar datos de baciloscopia	Referencia, secuencia, numero de bacilos, apariencia macroscópica carga, fecha entrega, fecha resultado, fecha registro, observaciones	Se modifican los datos de la tabla EXAMENBACILOSCOPIA de una baciloscopia determinada.	SI
Eliminar datos de baciloscopia	Serial, referencia, secuencia, numero de bacilos, apariencia macroscópica carga, fecha entrega, fecha resultado, fecha registro, observaciones	Se eliminan los datos de la tabla EXAMENBACILOSCOPIA de una baciloscopia determinado	SI

Responsable Proyecto HCAM



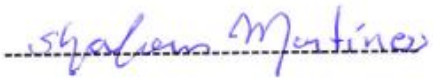
Responsable Desarrollo Software

1.2. Administración de cultivo

Funcionalidad	Datos	Resultado esperado	Cumple
Ingresar datos de cultivo	Serial, referencia, secuencia, numero de colonias, carga, semana, contaminado, crecimiento atípico fecha entrega, fecha resultado, fecha registro,	Se insertan los datos en la tabla EXAMENCULTIVO	SI

	observaciones		
Modificar datos de cultivo	Referencia, secuencia, numero de colonias, carga, semana, contaminado, crecimiento atípico fecha entrega, fecha resultado, fecha registro, observaciones	Se modifican los datos de la tabla EXAMENCULTIVO de un cultivo determinado.	SI
Eliminar datos de cultivo	Serial, referencia, secuencia, numero de colonias, carga, semana, contaminado, crecimiento atípico fecha entrega, fecha resultado, fecha registro, observaciones	Se eliminan los datos de la tabla EXAMENCULTIVO de un cultivo determinado	SI

Responsable Proyecto HCAM



Responsable Desarrollo Software

1.3. Administración de sensibilidad

Funcionalidad	**Datos**	**Resultado esperado**	**Cumple**
Ingresar datos de sensibilidad	Serial, H, R, S, E, Z, Km, Cs, Ofx, Eth, fecha registro	Se insertan los datos en la tabla EXAMENSENSIBILIDAD	SI
Modificar datos de sensibilidad	H, R, S, E, Z, Km, Cs, Ofx, Eth, fecha registro	Se modifican los datos de la tabla EXAMENSENSIBILIDAD de un análisis de sensibilidad determinado.	SI
Eliminar datos de sensibilidad	Serial, H, R, S, E, Z, Km, Cs, Ofx, Eth, fecha registro	Se eliminan los datos de la tabla EXAMENSENSIBILIDAD de un análisis de sensibilidad determinado	SI

Responsable Proyecto HCAM

Responsable Desarrollo Software

1.4. Administración de tipificación

Funcionalidad	Datos	Resultado esperado	Cumple
Ingresar datos de tipificación	Serial, MTB detectada, MTB no detectada, fecha registro	Se insertan los datos en la tabla EXAMENTIPIFICACION	SI
Modificar datos de tipificación	MTB detectada, MTB no detectada fecha registro	Se modifican los datos de la tabla EXAMENTIPIFICACION de un análisis de tipificación determinado.	SI
Eliminar datos de tipificación	Serial, MTB detectada, MTB no detectada, fecha registro	Se eliminan los datos de la tabla EXAMENTIPIFICACION de un análisis de tipificación determinado	SI

Responsable Proyecto HCAM

Responsable Desarrollo Software

1.5. Administración de diagnóstico

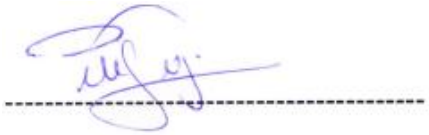
Funcionalidad	Datos	Resultado esperado	Cumple
Ingresar datos de diagnóstico	Serial, diagnóstico, esquema, fecha	Se insertan los datos en la tabla DIAGNOSTICO	SI
Modificar datos de diagnóstico	Diagnóstico, esquema, fecha	Se modifican los datos de la tabla DIAGNOSTICO de un diagnóstico determinado.	SI
Eliminar datos de diagnóstico	Serial, diagnóstico, esquema, fecha	Se eliminan los datos de la tabla DIAGNOSTICO de un diagnóstico determinado	SI

Responsable Proyecto HCAM

Responsable Desarrollo Software

2. Administración de seguimiento

Funcionalidad	Datos	Resultado esperado	Cumple
Ingresar datos de seguimiento	Serial, diagnóstico, esquema, fecha	Se insertan los datos en la tabla SEGUIMIENTO	SI
Modificar datos de seguimiento	Diagnóstico, esquema, fecha	Se modifican los datos de la tabla SEGUIMIENTO de un seguimiento determinado.	SI
Eliminar datos de seguimiento	Serial, diagnóstico, esquema, fecha	Se eliminan los datos de la tabla - SEGUIMIENTO de un seguimiento determinado	SI



Responsable Proyecto HCAM

Responsable Desarrollo Software

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

Este proyecto consistió en el desarrollo de una solución informática para el Hospital del IESS "Carlos Andrade Marín" con los siguientes objetivos: permitir a los usuarios ingresar información de pacientes, muestras, diagnósticos y seguimiento para el control de la tuberculosis. Disponer de datos para tomar decisiones de una manera más técnica, precisa, rápida. Facilitar el acceso a la información generada por el laboratorio para un funcionamiento organizacional del programa para el control de la tuberculosis. Proporcionar una herramienta que facilite elaborar informes y reportes. Contar con datos para verificar información mediante cruces. Ahorrar tiempo mediante la automatización del ingreso de información que actualmente se realiza manualmente.

El software que se ha desarrollado facilitará el control de la tuberculosis, permitirá un mejor control de la información, reduciendo errores, proporcionando una trazabilidad de los resultados y un log (bitácora) de las actividades realizadas por usuario con el fin de mejorar el desempeño de los usuarios, cumplir de una manera adecuada con los procesos de control de tuberculosis, de acuerdo a los estándares del programa de control de tuberculosis del Ministerio de Salud Pública del Ecuador, proporcionar una detección y seguimiento de pacientes con tuberculosis más rápida y efectiva.

El proceso de desarrollo de software ha seguido la metodología XP, acompañada del lenguaje de modelado unificado (UML) que permitió crear un modelo más adecuado del producto. Por una parte XP reduce el tiempo destinado al proceso de diseño, pero carece en su documentación es por eso que se utilizó UML como un soporte para el diseño de software. Para la codificación del proyecto se utilizó las herramientas que se consideraban más adecuadas dadas las características del proyecto, con el objetivo de

lograr cumplir con las necesidades del usuario, y adaptarse al entorno tecnológico en el que se encuentra el HCAM, donde existen ciertas restricciones como el cuidado de la información y la dificultad de conseguir un servidor donde instalar la aplicación, porque las computadores pertenecen a otras instituciones privadas con las que el hospital negocia. Se logró conseguir una computadora con sistema operativo Windows XP, que al principio parecía que estaba dañada, pero había un cable quemado que se reemplazó. Los equipos no podían salir de la institución lo que hizo más difícil la configuración y las pruebas. Se instaló las herramientas para que se ejecute la aplicación, la mejor opción fue instalar XAMPP, que es una distribución de Apache, Mysql y PHP gratis y fácil de conseguir, la razón fue que no se sabía si la información en el equipo era importante y si al formatear al equipo habría problemas con los propietarios del equipo.

El desarrollo para una institución pública, puede ser difícil debido a las restricciones que existen, también puede que el software desarrollado no se utilicé debido a los cambios de personal. Pueden existir problemas de apoyo por la institución hacia las personas encargadas del proyecto. A pesar de estos problemas se espera que el software desarrollado se pueda utilizar, y que sea de ayuda, facilite las actividades diarias de las personas que trabajan en el laboratorio del HCAM, y ayude a disminuir las muertes causadas por la tuberculosis, las cuales pueden ser prevenidas mediante un adecuado seguimiento y tratamiento de los pacientes.

4.2. Recomendaciones

Es recomendable que al utilizar herramientas de tecnología nuevas al principio es difícil saber cómo utilizarlas, es por eso que se necesita una investigación preliminar para tener una mejor idea, después podremos tener una experiencia real que nos permitirá hacer las cosas de una manera más rápida y confiable.

Se puede seguir una metodología de desarrollo de software y utilizar material de apoyo de la forma en que se necesite, también hay que tener el criterio de saber elegir entre hacer lo que dice la metodología y que no. Por ejemplo la metodología XP nos indica que las historias de usuario deben ser escritas por el usuario pero es muy difícil lograrlo debido a que desde el principio un usuario no sabe lo que quiere. Es por eso que un prototipo, diagramas de casos de uso son de gran valor para identificar los requerimientos de usuario.

También es importante para un proyecto utilizar herramientas conocidas con las que se cuente con una experiencia previa, disminuye el tiempo, facilita el desarrollo de software. Es preciso saber elegir las herramientas que se van a utilizar, no siempre se deben utilizar las mismas, dependen del entorno y la situación. Hay proyectos donde se necesita mayor velocidad de los procesos, otros donde se requiere una mayor seguridad y control de la información. No todos los sistemas son iguales.

Se recomienda utilizar el software desarrollado ya que proporciona una gran ayuda al momento de llevar un control de todo el proceso de diagnóstico y seguimiento de la tuberculosis mediante reportes, consultas fáciles y rápidas, mayor control del ingreso, modificación y eliminación de la información. La aplicación es fácil de usar, pero si requiere una capacitación y acompañamiento al usuario para que pueda entender cómo se utiliza cada opción. Para poder utilizar la aplicación es necesario poseer un servidor, con apache, php y mysql instalados.

Finalmente se recomienda a la facultad ayudar a los estudiantes dando a conocer si existen solicitudes para realizar una disertación de grado por parte de instituciones públicas o privadas, porque puede ser difícil saber que hacer, como disertación de grado,

para graduarse. También sería bueno dar una mayor agilidad para apoyar proyectos a la colectividad.

5. BIBLIOGRAFIA

IESS. Quiénes Somos. Internet. http://www.iess.gob.ec/es/web/guest/institucion Acceso: (30/04/2014).

Hospital Carlos Andrade Marín. Nuestra Institución. Internet. http://www.hcam.gob.ec/index.php/el-hcam/quienes-somos.html Acceso: (30/04/2014).

CLTech. DataLab 2005. Internet. http://www.cltech.net/spanish/desarrollo-de-software-para-laboratorio-clinico.html Acceso: (30/04/2014).

Dr. Salazar, Ramiro. Servicio de Laboratorio Clínico, Internet. http://www.hcam.gob.ec/index.php?option=com_content&view=article&id=128:servicio-de-laboratorio-clinico&catid=46:servicios-medicos&Itemid=178
Acceso: (29/04/14).

ExtremeProgramming, Internet http://www.extremeprogramming.org, internet Acceso:(18/08/2014).

Programación Extrema. Internet. http:// http://programacionextrema.tripod.com, internet Acceso: (22/08/2014).

Addison Wesley, "Extreme Programming Applied Playing To Win", 01 de Octubre del 2001 ISBN.0201616408.

Kent Beck, Martin Fowler, "Planning Extreme Programming", 12 de Octubre del 2000 ISBN. 0201710919.

Ron Jeffries, Ann Anderson, Chet Hendrickson "Extreme Programming Installed" ISBN.0201708426.

Yourdon Edward, "Análisis estructurado moderno", 1989, ISBN-10: 0135986249
Metodología Scrum y Metodologías Ágiles, internet http://scrummethodology.com/
Acceso: (16/08/2014).

"¿Qué es SCRUM?", internet. http://www.proyectosagiles.org/que-es-scrum
Acceso:(16/08/2014)

"¿Para qué sirve el SCRUM en la metodología ágil?", internet, http://www.i2btech.com/blog-i2b/tech-deployment/para-que-sirve-el-scrum-en-la-metogologia-agil/ Acceso: (16/08/2014).

"Selección de un enfoque de desarrollo", internet, http://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf, Acceso (24/08/2014).

Ian Sommerville, "Ingeniería de Software", 2007, 7ma edición, ISBN: 8478290745.

"¿Qué es el proceso unificado de rational?", internet, http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jan01/WhatIstheRationalUnifiedProcessJan01.pdf Acceso:(30/08/2014).

"Una introducción a la metodología de diseño y análisis de sistemas estructurados (SSADM)", internet, http://www.ogcio.gov.hk/en/infrastructure/methodology/ssadm/doc/s3a_pub.pdf, Acceso: (02/09/2014).

Masoud Yaghini, "A Framework for Selection of Information Systems Development Methodologies", Febrero de 2009

"Criterios de selección de metodologías de desarrollo de software", Oscar Tinoco Gómez, Pedro Pablo Rosales López, Julio Salas Bacalla, Industrial Data, vol. 13, núm. 2, julio, 2010, pp. 70-74, Universidad Nacional Mayor de San Marcos

"APACHE",internet,http://www.ntchosting.com/encyclopedia/hosting/apache-web-server/ Acceso:(04/12/2014)

"Qué hace un servidor web como Apache",internet,http://www.digitallearning.es/blog/apache-servidor-web-configuracion-apache2-conf/ Acceso:(04/12/2014)

"Historia de PHP", internet, http://php.net/manual/es/history.php.php Acceso:(04/12/2014)

"Concepto de JavaScript", internet, https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Obsolete_Pages/Gu%C3%ADa_JavaScript_1.5/Concepto_de_JavaScript Acceso:(04/12/2014)

Luis Alberto C., Marc Gibert G., Oscar Pérez Mora, Base de datos en MySQL, internet, http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02151.pdf Acceso:(05/12/2014)

"Arquitectura de software", internet, http://es.wikipedia.org/wiki/Arquitectura_de_software Acceso:(05/12/2014)

"Entorno cliente/servidor", internet, http://es.kioskea.net/contents/148-entorno-cliente-servidor Acceso:(06/12/2014)

"Arquitectura de tres niveles", internet, http://www.soluciones-del-valle.net/archivos/ tecnologico/antologias/ene-jun-13/li8/Arquitectura_de_3niveles.pdf Acceso:(06/12/2014)