UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

PROYECTO DE GRADO

"SISTEMA WEB DE ALMACENES Y COTIZACIONES CASO: EMPRESA FOR SECURITY 4.5"

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA

MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS

AUTOR: JAIME JUNIOR RADA ARANIBAR

TUTOR METODOLÓGICO: Lic. JAVIER HUGO REYES PACHECO

ASESOR: M.Sc. CARLOS MULLISACA CHOQUE

LA PAZ – BOLIVIA

2015

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

DEDICATORIA

A mis padres: Jaime Roberto Rada Ortuño y Yola Aranibar Morales ambos admirables, luchadores y valientes, a la que elevo toda mi gratitud, admiración y respeto, que amo con todo mi ser, quienes me apoyaron en todo momento tanto en dificultades como en éxitos, que con dedicación, espíritu y el calor humano que desprenden de su ser, me dio ánimos, esperanzas y fuerzas para seguir adelante y conseguir mis objetivos.

A mis hermanos Luis, Brayan y Jobani a quienes quiero mucho por apoyarme en esos momentos difíciles y disfrutar muchos momentos de felicidad, por ser parte de mi familia y de mi vida.

Con mucho amor y admiración

Jaime Junior Rada Aranibar

AGRADECIMIENTOS

A Dios porque gracias a él tengo familia, amigos, amigas y que me dio vida, salud y guió mi camino para así poder cumplir una de mis metas más preciadas.

A mi familia que sin su apoyo me hubiese sido muy costoso salir adelante, agradecer a mi padre por brindarme su apoyo, compresión y darme consejos acertados y valiosos en las diferentes etapas de mi vida, a mi mamá por guiarme, corregirme y estrecharme la mano cuando lo necesitaba, a mis hermanos Jaime Luis, Brayan Denis y Jobani Branco, por animarme y sacarme una sonrisa en momentos difíciles; gracias a todos por darme su amor incondicional, los amo.

A quien es como mi segunda madre Eusebia Casas Choque y al amor de mi vida Paola E. Sánchez M.; por darme su apoyo y amor incondicional; las amo.

Al Docente Tutor Metodológico Lic. Javier Hugo Reyes Pacheco, quien con su profesionalismo, experiencia, paciencia me colaboró y me dio consejos en el transcurso de la elaboración del presente proyecto.

Al Docente Asesor M.Sc. Lic. Carlos Mullisaca Choque, quien por su excelentísimo desempeño me colaboró en el seguimiento del proyecto, donde me dio consejos, observaciones y correcciones que fueron fundamentales para la elaboración de este proyecto.

A la Empresa For Security 4.5 por darme la oportunidad, apoyo, colaboración e información necesaria para que se haga posible la implementación del presente proyecto.

A los docentes de la carrera de Informática por los conocimientos transmitidos durante toda la etapa de mi formación universitaria.

RESUMEN

La informática se ha convertido en un factor importante para el desarrollo de las empresas, esto se debe a la gran cantidad de información que estas manejan, lo cual implica la importancia del uso de herramientas automáticas para la generación de informes y reportes en las empresas, las cuales permitirán realizar un control eficiente, y de esta manera lograr que las organizaciones cumplan sus metas y objetivos.

En la empresa For Security 4.5, dedicada a la venta de equipos de seguridad electrónica, existe la necesidad de contar con un sistema de Almacenes y Cotizaciones para apoyar a los procesos de registro, actualización, registros de productos y elaboración de proformas de venta.

Para el análisis, diseño e implementación del presente proyecto se aplicó la metodología Extreme Programming *(XP)*, utilizado para apoyar las actividades del diseño de sitios *Web* e implementación del mismo, además de hacer uso de algunas de las herramientas UML para documentar, ya que XP no utiliza diagramas para la documentación y es uno de los puntos en los que se apoya con herramientas UML.Para el desarrollo del Sistema *Web* se utilizó el lenguaje de programación PHP, y como gestor de base de datos *MySQL* para la función correcta del Sistema *Web*.

En cuanto a la calidad del sistema, la norma ISO 9126, basado en estándares con la funcionalidad y rendimiento total que satisfacen los requerimientos del cliente. Con lo cual se puede concluir que será de mucha utilidad para la empresa.

ABSTRACT

The computer has become an important tool for business development factor, this is due to the wealth of information they handle, which implies the importance of using automated tools for generating reports and reports on companies, which will allow for an efficient control, and thus ensure that organizations meet their goals and objectives.

In the company For Security 4.5, sells electronic security equipment, there is a need for a system of warehouses and Quotes to support registration processes, updating records and making product sales proformas.

For the analysis, design and implementation of this project the Extreme Programming (XP) methodology used to support the activities of website design and implementation thereof, in addition to using some of the tools for documenting UML was applied as XP does not use diagrams for documentation and is one of the points on which it rests with development tools Web UML. System PHP programming language is used, and as manager MySQL database for the correct function of the System Web.

As for the quality of the system, the ISO 9126 standard, standards-based functionality and overall performance that meet customer requirements. Thus it can be concluded that will be very useful for the company.

INDICE

INDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPITULO I

MARCO REFERENCIAL

1.1 INTRODUCCIÓN

Cada día, el avance acelerado de la tecnología adquiere más relevancia en el mundo empresarial, desde que se popularizó el uso de las computadoras y su capacidad de almacenamiento de información, su uso ha repercutido de manera importante en la sociedad en general y en las empresas.

En consecuencia, el manejo de la información que se genera a través de la computadora es distinto al manejo manual, por ser esta de mayor volumen y requerir una mejor organización.

Sin embargo, existen muchas empresas en nuestro medio, que aún no emplean métodos automatizados y llevan los registros de sus operaciones de forma manual, sin respaldo informático corriendo el riesgo de que la información sea errónea y traiga consigo pérdidas económicas.

La implantación de sistemas de información a través de las computadoras es en la actualidad una actividad que forma parte de las necesidades administrativas de micro, pequeñas, medianas y hasta las grandes empresas.

La gestión y estrategias de las empresas dependen de las herramientas e instrumentos tecnológicos de que dispongan. Los sistemas informáticos permiten que dichas empresas posean información útil, rápida, fluida y enfocada a las tácticas y estrategias de las mismas, información que permite un cambio de rumbo oportuno.

En este sentido la información es un arma principal que ayuda a la gerencia, a los productos y a los servicios a entrar en un mundo competitivo.

For Security 4.5, es una empresa que comercializa equipos de seguridad electrónica, como ser cámaras, alarmas, biométricos, controles de acceso, etc., la demanda de estos productos ha incrementado de manera considerable. Actualmente realizan sus registros de almacenes y cotizaciones de forma manual, es por tal razón y para poder entrar dentro de un campo competitivo que proponemos la elaboración de un "Sistema Web de Almacenes y Cotizaciones", que sirva de ayuda para un control efectivo de los productos y apoye en el área de cotizaciones con reportes seguros, oportunos y confiables.

1.2 ANTECEDENTES

La empresa de Seguridad Electrónica For Security 4.5, fue creada el 10 de mayo de 2011, la misma fue saneada, inscrita y establecida en su totalidad el 8 de octubre de 2014.

La principal actividad de la empresa es la comercialización e instalación de los diferentes equipos de seguridad electrónica (sistema de video vigilancia CCTV, alarmas contra robo, biométricos, controles de acceso, etc.), para ello cuenta con varios proveedores para su importación. El departamento de ventas es quien se encarga de los almacenes, donde se tienen almacenados los equipos.

El pedido de los equipos de seguridad electrónica se lo realiza al departamento de ventas (previa verificación de stock), la cotización, la venta o comercialización de estos lo realiza el asesor de ventas. La salida y entrega de los equipos en almacenes lo realiza el encargado de almacenes de forma manual y lenta.

Con el transcurrir del tiempo la demanda de los equipos de seguridad electrónica, fue creciendo debido al aumento de la delincuencia, es por tal motivo que la empresa ofrece a su clientela los distintos sistemas de seguridad.

Se pudo evidenciar que existe bastante información de los productos en cuestión, tanto en precios y características, que requiere ser oportuno, actualizado y controlado al

momento de realizar la cotización al cliente, siendo por consiguiente la necesidad de automatizar todos los procesos involucrados con estas actividades de manera que se pueda contar con una información integra, eficaz y segura.

"Basándose en las funciones se puede decir que el almacén solo alojará lo que se haya pedido o comprado y/o lo que se haya fabricado después de realizada la producción. Además gestionará el tiempo de permanencia del material según las instrucciones del control de calidad permitiendo las salidas de los mismos para su venta"[1]

A continuación se cita proyectos de grado consultados en la Biblioteca de Informática que tratan temas relacionados con el presente proyecto, que describiremos a continuación.

- Proyecto de grado "Sistema Integrado de Ventas y Almacenes Carbon film Bolivia Ltda.", autor Jenny Roxana Herréra Lunario, desarrollado el año 2006. Se desarrolla un sistema que mejorará la eficiencia en la atención al cliente, teniendo un control efectivo de los productos ofertados.
- Proyecto de grado "Sistema de Información de manejo y Control de Almacenes", autor Nelly Nieves Quiste Chamba, desarrollado el año 2004.Se desarrolla un sistema que se encarga del control de insumos que entran y salen de los almacenes del Honorable Consejo Municipal de La Paz.
- Proyecto de grado "Sistema de Control de Inventarios para Almacenes Crespal S.A.", desarrollado el año 2006.Se trata de un sistema que mejorara el control de entradas y salidas de medicamentos desde y hasta almacenes.

[1] Roux, 2009: Plan estratégico para los almacenes de la empresa siderúrgica del Orinoco Alfredo Maneiro.

1.3 PLANTEAMIENTO DEL PROBLEMA

1.3.1 Análisis del Problema

En un estudio preliminar realizado de los diferentes procesos y tareas que ejecuta diariamente en almacenes de la empresa For Security 4.5, se pudo identificar los siguientes problemas:

- El registro y búsqueda de los equipos de seguridad electrónica es de forma manual, se emplea demasiado tiempo en su seguimiento.
- El registro de compra de los equipos de seguridad electrónica es de forma manual y no tiene clasificación.
- La actualización periódica del registro de los equipos de seguridad electrónica es manual y deficiente.
- Se emplea mucho tiempo en la consulta de catálogos para conocer las características técnicas y funciones de los diferentes equipos de seguridad electrónica.
- Demoras en la ubicación de proveedores.
- El control de los niveles de stock es deficiente.
- La cotización determinada de los equipos de seguridad electrónica es de forma manual, lo cual afecta el nivel de atención al cliente externo e interno.
- La gerencia no cuenta con la información oportuna para una buena toma de decisiones.

1.3.2 Definición del Problema

El actual proceso de registro y control en almacenes de la empresa For Security 4.5, no dispone de un sistema de información para la automatización del mismo, lo cual al momento genera pérdida de tiempo.

A partir de todo lo descrito anteriormente, se plantea el problema de la siguiente manera:

¿Será posible implementar un sistema web de almacenes y cotizaciones para For Security 4.5, el cual pueda contar con informes actualizados y precisos?

1.4 OBJETIVOS

1.4.1 Objetivo General

Desarrollar e implementar un sistema web de almacenes y cotizaciones para la empresa For Security 4.5, que contribuya a ofrecer un servicio óptimo y eficiente a los clientes y obtener información oportuna y precisa para una buena toma de decisiones.

1.4.2 Objetivos Específicos

- Llevar el control automatizado de los equipos de seguridad electrónica de la empresa.
- Controlar las adquisiciones de los equipos de seguridad electrónica.
- Realizar un catálogo web que contenga información de todas las características técnicas de los equipos de seguridad electrónica.
- Realizar un registro de proveedores para su fácil ubicación.
- Realizar un módulo de cotizaciones para una eficaz atención al cliente.
- Proveer informes de stock de los diferentes equipos de seguridad electrónica.
- Proveer información rápida a gerencia para que la toma de decisiones se las realice oportunamente.

1.5 JUSTIFICACIONES

1.5.1 Justificación Social

El proyecto se justifica socialmente por que proveerá de información rápida, actualizada y oportuna a los departamentos de marketing, contabilidad, ventas y particularmente a almacenes, ayudando a los criterios de decisión a gerencia, ofreciendo un

mejor entorno de trabajo al personal administrativo y operativo, mejorando así el servicio de atención al cliente a un menor costo de tiempo.

1.5.2 Justificación Tecnológica

El desarrollo e implementación del sistema, técnicamente hará uso de la informática, el cual ayudará a llevar un buen control de los equipos de seguridad electrónica en almacenes, se optimizará el servicio de atención al cliente con un sistema de cotizaciones.

1.5.3 Justificación Económica

Se justifica económicamente el proyecto porque el sistema permitirá incrementar los beneficios de la empresa, con la mejora en el control de las ventas.

Dicho sistema podrá determinar la cantidad de equipos de seguridad electrónica que se debe pedir y cuanto pedir. Esto permitirá que no se tenga equipos de seguridad electrónica en demasía y tampoco inexistencia de equipos de seguridad electrónica en demanda, de lo contrario ocasionaría pérdidas económicas considerables.

1.6 ALCANCE, LÍMITES Y APORTES

1.6.1 Alcances

El presente proyecto tendrá un alcance desde el análisis, diseño e implementación del sistema propuesto al departamento de ventas en particular a almacenes de la empresa For Security 4.5, que permitirá llevar un control e información precisa de los equipos de seguridad electrónica. El sistema contara con las siguientes funciones:

- Base de datos actualizada de los equipos de seguridad electrónica.

- Control y registro de ingreso y salida de los equipos de seguridad electrónica de almacenes.
- Verificación y actualización de precios de los equipos de seguridad electrónica.
- Generación de reportes e informes referentes de manera oportuna y confiable reduciendo así el tiempo en la realización de una cotización.
- Emisión de reportes de stock de los diferentes equipos de seguridad electrónica.
- El catalogo web contendrá información técnica básica de los equipos de seguridad electrónica básica.

1.6.2 Límites

El proyecto de grado contemplará como límites los siguientes puntos citados a continuación:

- El presente proyecto no abarcará órdenes de compra y venta, kardex, facturación.

1.6.3 Aportes

Los aportes del presente proyecto de grado, será la de automatizar los procesos rutinarios de la empresa, minimizando y optimizando los tiempos de ejecución generando información que coadyuve a la fácil y correcta toma de decisiones por parte de la gerencia.

El subsistema de inventario facilitará la tarea del encargado de almacén, permitiendo realizar un control adecuado de los equipos de seguridad electrónica, la forma de almacenaje, información actualizada del stock.

El sistema de cotizaciones facilitará la tarea del asesor de ventas, permitiendo así realizar una atención eficaz al cliente, con precios actualizados de los equipos de seguridad electrónica.

1.7 METODOLOGÍA

Los métodos preliminares utilizados para determinar los requerimientos de los usuarios y los procesos a automatizar son:

- Entrevista con personal administrativo, operativo y gerencia.
- Análisis de la situación actual de la empresa.
- Método basado en XP (Extreme Programming).
- Calidad de sistema ISO-9126.
- Uso de herramientas y técnicas:

Hardware

- ❖ Microprocesador Core2duo
- ❖ Capacidad de Memoria RAM 1 GB o superior.
- ❖ Capacidad en disco duro 80 GB o superior.
- ❖ Unidad de CD-ROM 52x
- ❖ Monitor 15 pulgadas.
- ❖ Teclado multimedia, preferentemente español.
- ❖ Mouse.

Sistema

- ❖ Sistema Operativo Windows 7
- ❖ Lenguaje de Programación Php, MySQL

CAPITULO II

MARCO TEÓRICO

2.1 INTRODUCCIÓN

El presente proyecto tiene como objetivo principal, la elaboración de un sistema que se caracteriza por ser adecuado y especializado para uso específico del manejo de Almacenes y Cotizaciones, que sirva de ayuda para un control efectivo de los productos y apoye en el área de cotizaciones con reportes seguros, oportunos y confiables, ya que se asentara en la sistematización de los métodos y procesos manuales que existen en la misma, convirtiéndolos en más óptimos y efectivos, para ello fundamentaremos nuestras actividades recurriendo a las áreas de sistema de información, análisis y diseño de sistemas para garantizar que el desarrollo del proyecto mejore todas las necesidades y requerimientos.

2.2 SISTEMA DE INFORMACIÓN

Un sistema de información puede definirse técnicamente como "un conjunto de componentes interrelacionados que permiten capturar, procesar, almacenar y distribuir la información para apoyar a la toma de decisiones y el control en una organización". [Pressman, 05]

De acuerdo al concepto anterior este sistema de información permitirá como cualquier otro sistema de una organización procesar reportes, almacenar productos, producir información, cotizaciones y otras salidas.

Este sistema de información formado por otros subsistemas que incluyen hardware y Sistema; medios de almacenamiento de datos para archivos y bases datos.

2.3 METODOLOGÍAS AGILES

En el mundo del desarrollo de software (o desarrollo de proyectos en general) podemos decir que existen métodos de desarrollo "pesados" y métodos "ágiles".

Los pesados son los tradicionales, los que todos alguna vez realizamos (o seguimos realizando), no siguen ningún patrón determinado más que el de buenas prácticas de programación, pero se centran en otras cosas. Principalmente centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto.

Las metodologías ágiles nacen como respuesta a los problemas que puedan ocasionar las metodologías tradicionales y se basa en dos aspectos fundamentales, retrasar las decisiones y la planificación adaptativa. Basan su fundamento en la adaptabilidad de los procesos de desarrollo. Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan.

Otra cosa a destacar: la mayoría de las metodologías ágiles están hechas para adaptarse al cambio de requisitos "sobre la marcha" (eso que tanto odiamos os programadores) mientras que en las metodologías tradicionales o pesadas ocurre todo lo contrario [Chin, 04].

Se tiene a continuación un resumen de métodos ágiles de desarrollo:

RAD: El desarrollo rápido de aplicaciones (RAD) es una metodología de desarrollo de software, que implica el desarrollo iterativo y la construcción de prototipos. El desarrollo rápido de aplicaciones es un término originalmente utilizado para describir un proceso de desarrollo de software.

RUP: El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

XP (Extreme Programming): La programación extrema es un enfoque de la ingeniería de software formulado por Kent Beck, autor del primer libro sobre la materia, Extreme Programming Explained: Embrace Change (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que estos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

SCRUM: Más que una metodología de desarrollo de software, es una forma de auto-gestión de los equipos de programadores. Un grupo de programadores deciden cómo hacer sus tareas y cuánto van a tardar en ello. Scrum ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro. Scrum permite además seguir de forma clara el avance de las tareas a realizar, de forma que los "jefes" puedan ver día a día como progresa el trabajo.

DSDM: Es probablemente la metodología ágil más completa, sin embargo SCRUM y XP son más fácil de implementar y complementarias porque ellas forman diferentes aspectos de los proyectos de desarrollo y están fundadas sobre los mismos principios de desarrollo ágil.

CRYSTAL CLEAR: No es una metodología en si misma sino una familia de metodologías con un "código genético" común. La idea es poder armar distintas metodologías para distintos tipos de proyectos. Cada proyecto y organización usará este código genético para generar su propia metodología. El nombre Crystal deriva de la

caracterización de los proyectos según dos dimensiones, tamaño y complejidad (como en los minerales, color y dureza).

T.D.D.: Desarrollo guiada por pruebas, o Test-driven development (TDD) es una práctica de programación que involucra otras dos prácticas: Escribir las pruebas primero (Test First Development) y Refactorización (Refactoring). Para escribir las pruebas generalmente se utilizan las pruebas unitarias (unit test en inglés). En primer lugar se escribe una prueba y se verifica que las pruebas fallen, luego se implementa el código que haga que la prueba pase satisfactoriamente y seguidamente se refactoriza el código escrito.

2.4 METODOLOGÍA DE DESARROLLO EXTREME PROGRAMMING (XP)

"La Programación Extrema (XP) es posiblemente el método ágil más conocido y ampliamente utilizado. El nombre fue acuñado por Kent Beck el padre de XP" [Sommerville, 05].

La Programación Extrema una metodología ágil o ligera basada en una serie de valores, reglas y prácticas que cuenta con las fases: Exploración, Planificación, Iteración y Producción persigue el objetivo de aumentar la productividad a la hora de desarrollar programas.

De manera que este proyecto de grado se fundamenta, con esta metodología centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de sistema, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, proporcionando un buen clima de trabajo basándose en realimentación continua entre el cliente y el equipo de desarrollo en niveles extremos teniendo comunicación fluida entre los mismos, dando simplicidad en las soluciones y coraje para enfrentar los cambios.

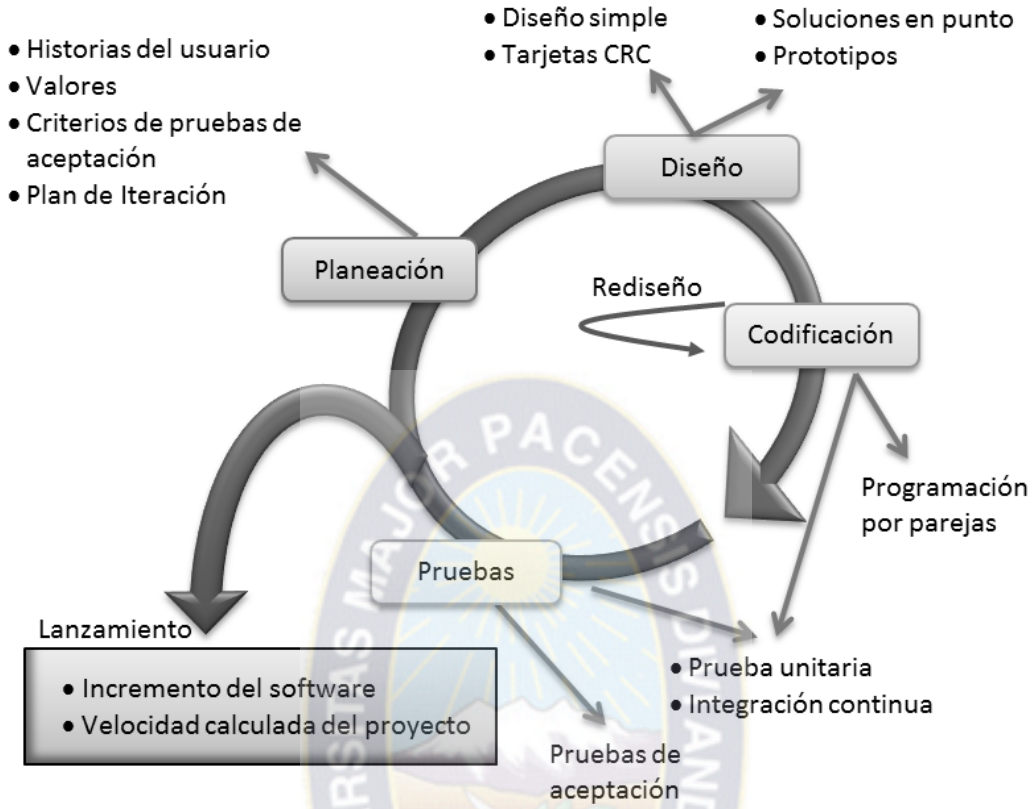


Figura 2.1 Procesos de la metodología XP

Fuente: [Pressman, 98]

2.4.1 Ciclo de vida de Programación Extrema

Entre las fases de esta metodología se menciona los siguientes:

FASE I: EXPLORACIÓN

"Las historias de usuario son similares al empleo de escenarios, con la excepción de que no se limitan a la descripción de la interfaz de usuario. También conducirán el proceso de creación de los test de aceptación (empleados para verificar que las historias de usuario han sido implementadas correctamente)". [Fernández, 02]. En esta fase, los clientes plantean a grandes rasgos las historias de usuarios que son los intereses para la primera

entrega del producto. Al mismo tiempo se logra la familiarizarse con las herramientas, tecnologías y prácticas que se utiliza en el proyecto. La fase de exploración toma de pocas semanas a pocos meses.

FASE II: PLANIFICACIÓN

"La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario asociadas a éstas las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación.". [Joskowicz, 02].

El cliente establece la prioridad de cada historia de usuario y correspondientemente los programadores, se realizan estimaciones del esfuerzo necesario de cada una de ellas. Se determina tanto con el cliente como con equipo de desarrollo el contenido de la primera entrega y un cronograma, una entrega debería obtenerse en no más de tres meses. Esta fase dura pocos días.
Ámbito: ¿Qué es lo que el software debe de resolver para que este genere valor?
Prioridad: ¿Qué debe ser hecho en primer lugar?
Composición de versiones: ¿Cuánto es necesario hacer para saber si el negocio va mejor con software que sin él? En cuanto el software aporte algo al negocio debemos de tener lista las primeras versiones.
Fechas de versiones: ¿Cuáles son las fechas en la presencia del software o parte del mismo pudiese marcar la diferencia?
El personal del negocio no puede tomar en vacío estas decisiones y el personal técnico tomará las decisiones técnicas que proporcionan la materia prima para las decisiones del negocio.
Estimaciones: ¿Cuánto tiempo lleva implementar una característica?
Consecuencias: Informar sobre las consecuencias de la toma de decisiones por parte del negocio. Por ejemplo cambiar las bases de datos a Oracle.
Procesos: ¿Cómo se organiza el trabajo y el equipo?
Programación detallada: Dentro de una versión ¿Qué problemas se resolverán primero?
Pequeñas versiones: Cada versión debe de ser tan pequeña como fuera posible, conteniendo los requisitos de negocios más importantes, las versiones tienen que tener sentido como un todo, me explico no puedes implementar media característica y lanzar la versión.

Tabla 2.1 Determinación de Requerimientos

Fuente [Calero, 03]

FASE III: ITERACIONES

"Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios." [Joskowicz, 02]

En esta fase se incluye varias interacciones sobre el sistema antes de ser entregado. El plan de entrega será compuesto por iteraciones que como máximo duran tres semanas. En la primera iteración se intenta establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias de usuario que contengan los elementos necesarios que den lugar a la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide que historias se implementaran en cada iteración. Al final de la última iteración el sistema estará listo para entrar en producción.

FASE IV: PRODUCCIÓN

"Si bien al final de cada iteración se entregan módulos funcionales y sin errores, puede ser deseable por parte del cliente no poner el sistema en producción hasta tanto no se tenga la funcionalidad completa." [Joskowicz, 02]

Esta fase requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se toman decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

2.4.2 Actividades

Se hará la mención de algunas de las actividades de la metodología XP que son utilizados en el desarrollo de este sistema las cuales son:

Historias de usuario.- Este sistema es desarrollado para el cliente, por lo tanto, ellos son quienes son los que deciden que tareas realizará la aplicación. Este planteamiento se desarrolla a lo largo del proyecto, el cliente es quien decide que hacer. Como primer paso, se debe tener las ideas claras de lo que será el proyecto.

Las historias de usuario son utilizadas como herramienta para dar a conocer los requerimientos del sistema al equipo de desarrollo. Son pequeños textos en los que el cliente describe sus actividades que realiza el sistema; la redacción de los mismos se realizó bajo la terminología del cliente, no del desarrollador, de forma que sea clara y sencilla, sin profundizar en detalles. Las historias de usuario solo muestran la silueta de una tarea a realizarse por esta razón hubo un representante que estuvo disponible en todo momento para solucionar las dudas.

Iteraciones.- En la metodología XP, la creación del sistema se divide en etapas para facilitar su realización. Por lo general, los proyectos constan de más de tres etapas, las cuales toman el nombre de iteración, de allí el concepto de metodología iterativa. La duración de una iteración es de una a tres semanas. Para cada iteración se define un módulo o conjunto de historias que se van a implementar, al final de la iteración se obtiene como resultado la entrega de los módulos de almacén y cotizaciones, el cual supera las pruebas de aceptación que establece el cliente para verificar el cumplimiento de los requisitos.

Reuniones.- El planeamiento es esencial para cualquier tipo de metodología, es por ello que XP requiere de una revisión continua del pan de trabajo. A pesar de ser una metodología que evita la documentación exagerada, en muy estricta en la organización del

trabajo. Por tal motivo las reuniones fueron constantes para obtener un producto satisfactorio teniendo la disponibilidad de los usuarios.

Entregas pequeñas.- La duración de una iteración varía entre una semana y tres, al final de la cual se hizo la entrega de los avances del producto, los cuales son completamente funcionales. Estas entregas se caracterizaron por ser frecuentes.

2.5 PROYECTO DE SISTEMA

"Los proyectos de sistema se realizan de manera planificada y controlada por una razón principal: es la única forma conocida de gestionar la complejidad". (Pressman, 02, pág. 643)

a) Gestión de Proyectos

La Gestión de Proyectos se puede describir como un proceso de planteamiento, ejecución y control de un proyecto, desde su comienzo hasta su conclusión, con el propósito de alcanzar un objetivo final en un plazo de tiempo determinado, con un coste y nivel de calidad determinados, a través de la movilización de recursos técnicos, financieros y humanos. Incorporando variadas áreas del conocimiento, su objetivo final es el de obtener el mejor resultado posible del trinomio coste plazo- calidad. (Knoow.Net, 08)

b) Técnicas y Herramientas de Gestión de Proyectos

Existen una gran variedad de técnicas utilizadas en el planteamiento y control de proyecto, entre muchas tenemos a las más usadas: los Gráficos de Gantt (incluyendo cronogramas, gráficos de carga, entre otros), el PERT [2] y el CPM[3]. (Knoow.Net, 08)

[2]Programe Evaluation and ReviewTechnique
[3]Critical Path Method

c) Ingeniería del Sistema

Para definir este punto no basta con analizar y estudiar un concepto en general, debido a que existen varios autores que definen la Ingeniería del Sistema de distintas maneras, entre los más importantes podemos citar:

Según la definición de IEEE[4], citado por Lewis, sistema es una suma total de los programas de computadora, procedimientos, reglas, documentación asociada, y los datos que pertenecen a un sistema de cómputo y un artículo de sistema es diseñado para un usuario o un conjunto de usuarios (Lewis, 94).

IEEE, también define a la Ingeniería de Sistema como: "la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del sistema."(Hans, 02).

Existen muchas definiciones dadas por organismos internacionales, profesionales de prestigio tales como la IEEE, entre ellas las más comprensible que se describe a continuación: "la Ingeniería del sistema es la aplicación del enfoque sistemático, disciplinado y cuantificado al desarrollo, operación y mantenimiento del mismo; es decir, la aplicación de ingeniería al sistema" (Pressman, 02).

Ciclo de Vida de la Ingeniería del Sistema

La Ingeniería de Sistema requiere llevar a cabo numerosas tareas agrupadas en etapas, al conjunto de estas etapas se le denomina ciclo de vida. Las etapas comunes a casi todos los modelos de ciclo de vida son las siguientes:

[4] IEEE: *Institute for Electrical and Electronics Engineers*.

Análisis de requisitos: Extraer los requisitos de un artículo de sistema es la primera etapa para crearlo. Mientras que los clientes piensan que ellos saben lo que el sistema tiene que hacer, se requiere habilidad y experiencia para reconocer requisitos incompletos, ambiguos o contradictorios.

Especificación: La especificación de requisitos describe el comportamiento esperado en el sistema una vez desarrollado. Gran parte del éxito de un proyecto de sistema radicará en la identificación de las necesidades del negocio (definidas por la alta dirección), así como la interacción con los usuarios funcionales para la recolección, clasificación, identificación, priorización y especificación de los requisitos del sistema.

Arquitectura: La integración de infraestructura, desarrollo de aplicaciones, bases de datos y herramientas gerenciales, requieren de capacidad y liderazgo para poder ser conceptualizados y proyectados a futuro, solucionando los problemas de hoy. Un diseño arquitectónico describe en general el cómo se construirá una aplicación de sistema.

Programación: Reducir un diseño a código puede ser la parte más obvia del trabajo de ingeniería de sistema, pero no necesariamente es la que demanda mayor trabajo y ni la más complicada. La complejidad y la duración de esta etapa está íntimamente relacionada al o a los lenguajes de programación utilizados, así como al diseño previamente realizado.

Prueba: Consiste en comprobar que el sistema realice correctamente las tareas indicadas en la especificación del problema. Una técnica de prueba es probar por separado cada módulo del sistema, y luego probarlo de forma integral, para así llegar al objetivo.

Documentación: Todo lo concerniente a la documentación del propio desarrollo del sistema y de la gestión del proyecto, pasando por modelaciones; todo con el propósito de eventuales correcciones, usabilidad, mantenimiento futuro y ampliaciones al sistema.

Mantenimiento: Mantener y mejorar el sistema para corregir errores descubiertos e incorporar nuevos requisitos. Esto puede llevar más tiempo incluso que el desarrollo del sistema inicial.

Fases de la Ingeniería del Sistema

La ingeniería del sistema es una técnica multicapas formada por herramientas, métodos, procesos, y un enfoque de calidad. En la ingeniería del sistema se definen tres fases genéricas independientemente del área de aplicación, tamaño o complejidad del proyecto.

Las cuales se detallan a continuación:

• Fase de Definición

La fase de definición se centra en que información será procesada, que función, comportamiento y rendimiento se desea del sistema, que interfaces serán establecidas, que restricciones del diseño existen y que criterios de validación se necesitan para definir un sistema correcto. El método aplicado durante esta fase varía dependiendo del paradigma de ingeniería del sistema que se elija, de alguna manera tendrá lugar tres tareas principales como son: La ingeniería de sistemas o de información, planificación del proyecto del sistema y el análisis de los requerimientos.

• Fase de Desarrollo

La fase de Desarrollo intenta definir como se diseña las estructuras de datos, la implementación los detalles procedimentales, las interfaces, la traducción del diseño en un lenguaje de programación y como se realiza las pruebas. Los métodos aplicados mediante la fase de desarrollo varían, pero siempre se darán lugar a las tres tareas específicas como son: El diseño de sistema, generación de código y prueba de sistema.

• Fase de Mantenimiento

Para (Jacobson, 00) la fase de mantenimiento e centra en el cambio que va asociado a la corrección de errores, las adaptaciones requeridas a medida que evoluciona el entorno del sistema, y a los cambios a las mejoras producidas por los requisitos cambiantes de los usuarios. Esta fase vuelve a aplicar las fases de definición y desarrollo, pero en el contexto del sistema ya existente. Durante la etapa se encuentran cuatro tipos de cambios como son la corrección, adaptación, mejora y la prevención.

Tomando en cuenta las tres fases de la ingeniería del sistema se define como "un conjunto de etapas parcialmente ordenadas con la intención de lograr un objetivo", para la obtención de un artículo de sistema de calidad. El proceso de desarrollo de la misma es aquel en el que las necesidades del usuario son traducidas en requerimientos de programa de computador, estos requerimientos transformados en diseño, y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo. Concretamente define quien está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo.

2.5.1 Métodos y Herramientas de la Ingeniería del Sistema

La ingeniería del sistema está compuesta por las capas de procesos, métodos y herramientas orientadas hacia la calidad.

Métodos de la Ingeniería de Sistema

"Un método de Ingeniería de Sistema es un enfoque estructurado para el desarrollo de sistema cuyo propósito es facilitar la producción de sistema de alta calidad de una forma costeable. Métodos como Análisis Estructurado y JSD fueron los primeros desarrollados en los años 70. Estos métodos intentaron identificar los componentes funcionales básicos de un sistema de tal forma que los orientados a funciones aún se utilizan ampliamente.

En los años 80 90, estos métodos orientados a funciones fueron complementados por métodos orientados a objetos como los propuestos por Booch y Rumbaugh. Estos diferentes enfoques se han integrado a un solo enfoque unificado, basado en UML. No existe método ideal y métodos diferentes tienen diferentes áreas donde son aplicables. Por ejemplo, los métodos orientados a objetos a menudo son apropiados para sistemas interactivos, pero no para sistemas de tiempo real con requerimientos severos…" (Somerville, 02, pág. 11).

Por otro lado, la planificación de la gestión ágil es informal (algunos modelos llegan a prohibir el uso de diagramas de Gantt) y solo cubren el ciclo de sistema que se está elaborando (generalmente un mes). Las principales referencias de la gestión ágil de proyectos son: Scrum es un modelo ágil no centrado en prácticas de programación como XP, sino en prácticas de gestión. Rational Unified Process es un proceso iterativo para desarrollo de sistema creado por Rational Sistema - IBM. Muchas organizaciones exitosas hoy día, utilizan una combinación de metodologías formales e informales de desarrollo, y una mezcla de herramientas, para conducir proyectos de manera predecible, que aceleren el desarrollo y reduzcan los riesgos innatos en estos…" (Aguilar Ramos, 2005, págs. 23-24)

Requerimientos

"…Los requerimientos describen el comportamiento de un sistema. A medida que el sistema actúa sobre los datos o las instrucciones, los objetos y las entidades se mueven desde un estado de ser a otro: por ejemplo, de lleno a vacío, de ocupado a libre, o de enviando a recibiendo. Esto es, en un estado dado, el sistema satisface un conjunto de condiciones; cuando el sistema actúa puede cambiar su estado global cambiando el estado de un objeto. Los requerimientos expresan los estados del sistema y de los objetos y las transiciones de un estado a otro.

Los requerimientos pueden pensarse de dos maneras: funcional y no funcional, lo que ayuda a describirlos. Un requerimiento funcional describe una interacción entre el sistema y su ambiente. Por ejemplo, para determinar los requerimientos funcionales se decide cuáles estados son aceptables para el sistema. Además, los requerimientos funcionales describen cómo debe comportarse el sistema ante determinado estímulo.

Se pueden utilizar diversas técnicas para determinar los requerimientos funcionales, incluyendo los casos de uso.

Tanto los requerimientos funcionales como los no funcionales son obtenidos del cliente de manera formal y cuidadosa. Esta extracción formal de los requerimientos es necesaria porque los clientes no siempre son buenos para describir con exactitud lo que desean o necesitan, y los desarrolladores no siempre son buenos para comprender los intereses del negocio de los demás.

Los clientes conocen su negocio pero no siempre pueden describir sus problemas de negocios a terceros; las descripciones están llenas de expresiones en jerga y de suposiciones con las cuales los desarrolladores no están familiarizados. Por otra parte, los desarrolladores conocen las soluciones de computación, pero no siempre saben cómo afectarán las soluciones posibles a las actividades de negocios de los clientes…". (Laurence Pleeger, 02, pág. 166).

2.6 ELECCIÓN METODOLÓGICA

2.6.1 Introducción a las Metodologías de Desarrollo Ágil

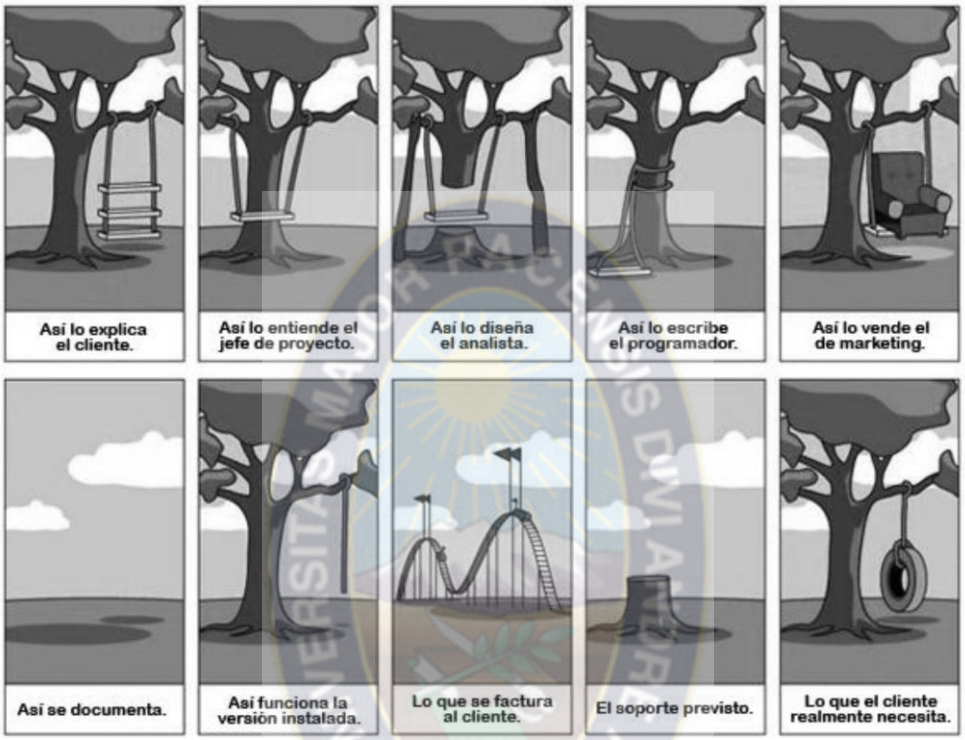


Figura 2.2 La realidad del desarrollo de Sistema[5]

La figura 2.2, muestra con humor, pero de forma precisa, las diversas situaciones que se dan en un gran número de proyectos de desarrollo de sistema. ¿Por qué es difícil desarrollar sistema? Una respuesta puntual a esa pregunta, es que hay perspectivas distintas al interpretar la realidad, lo que realmente requiere el cliente. Rumbaugh (OMT y UML), Jacobson (SDL y UML) y Booch (UML) y los unifica en una sola metodología. (Jacobson, 00, pág. 31).

[5]Figura. La realidad sobre el desarrollo de sistema. Adaptado de "Modelo de Desarrollo Productivo", por Contreras E. Recuperado de http://richzendy.org/docs/DesarrolloProductivo/

Esto hace que el desarrollo de un proyecto de sistema sea riesgoso y difícil de controlar, por eso es necesario tener un plano en que apoyarse. Es decir, que el desarrollador, debe llevar una metodología de por medio, que le ayude a saber cómo identificar las necesidades o problemas existentes, qué criterios usar para llegar a decidir la solución computarizada que se debe aplicar en el proyecto de sistema, entre otros.

Actualmente, existe una gran variedad de propuestas metodológicas que incluyen diversas herramientas en el proceso de desarrollo. Principalmente, estas metodologías se dividen en dos: las Metodologías Ágiles y las Tradicionales o Clásicas.

En este apartado, se hablará solo de las metodologías ágiles pues estas han sido la base fundamental para el éxito de este proyecto.

El desarrollo ágil de Sistema hace uso directo del Manifiesto Ágil de Sistema, mismo que sigue estos principios que Beck (2001) afirma:

- La mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de sistema con valor.
- Es aceptable que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entrega sistema funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores deben trabajar juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El sistema funcionando es la medida principal de progreso.

- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de minimizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

2.6.2 Análisis de las Metodologías ágiles de Desarrollo MSF, AUP, XP

2.6.2.1 Microsoft Solution Framework (MSF)

M.S.F. representa un resumen de las mejores prácticas que han aparecido en cuanto a administración de proyectos. Se la puede ver como una metodología rígida de administración de proyectos, pero es una serie de modelos que si se los aplica en un tipo adecuado de plan puede adaptarse a cualquier proyecto de tecnología de información; sin embargo es demasiado largo, debido a que cada una de sus fases requieren una extensa documentación, por lo que no es recomendable para proyectos que funcionen en base a tiempo o para proyectos cortos, sino para proyectos largos y que necesitan una visión clara del proyecto desde el inicio de éste; además es primordial para poder aplicar ésta metodología que se tengan claros los requerimientos del cliente desde el principio, pues cada retroalimentación con esta metodología representa un alto coste. El modelo de proceso de M.S.F. se compone de 5 fases o etapas iterativas:

- ❖ Visión y Alcances.
- ❖ Planificación.
- ❖ Desarrollo.
- ❖ Estabilización.
- ❖ Implantación.

Al final de cada fase se tienen que lograr alcances específicos –definidos por hitos- y que generan entregables para agregar valor al proyecto.

2.6.2.1.1 Principales ventajas de MSF

- Puede ser utilizada en un proyecto de desarrollo sobre cualquier tecnología.
- No es requerido alcanzar un punto específico en una de las fases para poder avanzar a la siguiente.
- Crea una disciplina de análisis de riesgos que evoluciona a lo largo del proyecto.
- Permite la reutilización de componentes ya desarrollados en ciclos anteriores.

2.6.2.1.2 Principales desventajas de MSF

- Existe dependencia al administrador de proyectos, ya que es el responsable directo de la interacción con el cliente y el grupo de trabajo; para tomar decisiones en caso de cambios o el tratamiento de nuevos requerimientos.
- Está basada en tecnología Microsoft, por lo que trata de obligar a utilizar sus propias herramientas. Generando dependencias de tecnologías propietarias.
- Demasiada documentación en sus fases.
- Los precios de licencias, capacitación y soporte de Microsoft son costosos.

2.6.2.2 Extreme Programming

La Programación Extrema está basada en la comunicación, la simplicidad, y la reutilización continua de código. Es considerada la más destacada de los procesos ágiles de desarrollo de sistema. Se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad.

Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos.

Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. Las fases que se siguen en la metodología XP son las siguientes:

- ❖ Planificación
- ❖ Diseño
- ❖ Desarrollo
- ❖ Pruebas
- ❖ Finalización del Proyecto

Características fundamentales de la metodología:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas.
- Programación por parejas.
- Frecuente interacción del equipo de desarrollo con el cliente o usuario.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para
- aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento.
- Simplicidad en el código: La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

2.6.2.2.1 Principales ventajas de XP

- El manejo, coste del cambio no depende de la fase o etapa de desarrollo.
- Comunicación constante entre usuarios y desarrolladores.
- Retroalimentación frecuente entre el equipo de desarrollo, y el cliente.

- Simplicidad en el análisis de la solución y codificación de los módulos del sistema.

2.6.2.2.2 Principales desventajas de XP

- Debido a la falta de documentación no se puede tener un control completo de las iteraciones que se han ido realizando durante todo el proceso de desarrollo.
- El uso de una metodología Ágil, como X.P. sirve más para proyectos sencillos, de alcance corto; pues hace énfasis en el empleo del menor tiempo posible y no generar mucha documentación. Sin embargo, esto representa una desventaja ya que el documentar, puede servir para el control y seguimiento de lo que realiza el equipo de desarrollo.

2.6.2.3 Agil Unified Process (AUP)

La metodología AUP es una versión simplificada del Proceso Unificado de Rational (RUP); que describe de manera más simple la forma de desarrollar aplicaciones de sistema usando técnicas ágiles. Las disciplinas de AUP son:

- ❖ Modelado
- ❖ Implementación
- ❖ Prueba
- ❖ Despliegue
- ❖ Administración de la configuración
- ❖ Administración o gerencia del Proyecto
- ❖ Entorno

AUP es ágil, porque está basada en los siguientes principios:

El personal sabe lo que está haciendo. La gente no va a leer de forma detallada el proceso de documentación, pero algunos quieren una orientación de alto nivel y/o formación de vez en cuando.

Simplicidad.- Todo se describe concisamente.

Agilidad.- El ajuste a los valores y principios de la Alianza Ágil.

Centrarse en actividades de alto valor.- La atención se centra en las actividades que se ve que son esenciales para el de desarrollo.

2.6.2.3.1 Principales ventajas de AUP

- Metodología robusta con muchos artefactos y disciplinas por elegir.
- Prioridades establecidas sobre la base de mayor riesgo.
- La documentación ayuda a comunicarse en entornos distribuidos.

2.6.2.3.2 Principales desventajas de AUP

- La documentación es mucho más formal que en la mayoría de las metodologías ágiles.
- Mínima atención a la dinámica del equipo de trabajo.

2.7 COMPARACIÓN ENTRE LAS METODOLOGÍAS CANDIDATAS

A continuación se realiza un análisis comparativo entre algunas metodologías ágiles, de manera que se justifique por qué se optó por la utilización de XP para este proyecto.

2.7.1 Diferencias entre Metodología Tradicionales y Ágiles

Metodologías Tradicionales	Metodologías Ágiles
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.	Basadas en heurísticas provenientes de prácticas de producción de código.
Cierta resistencia a los cambios.	Especialmente preparados para cambios durante el proyecto.
Impuestas externamente.	Impuestas internamente por el equipo.
Proceso mucho más controlado, con numerosas políticas o normas.	Proceso menos controlado, con pocos principios.
Más artefactos.	Pocos Artefactos.
Más roles.	Pocos roles.
Grupos grandes y posiblemente distribuidos.	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.
La arquitectura del sistema es esencial y se expresa mediante modelos.	Menos énfasis en la arquitectura del sistema.
Existe un contrato prefijado.	No existe contrato tradicional o al menos es bastante flexible.

Tabla 2.2 Comparación entre las Metodologías Ágiles y las Tradicionales

CRITERIOS	METODOLOGÍAS		
	Microsoft Solutions Framework (MSF)	**Programación Extrema (XP)**	**Proceso Unificado Ágil (AUP)**
Pocos roles y flexibles.	2	3	1
Iterativa.	1	3	1
Entorno amplio de proyectos de sistema.	2	3	2
Permite desarrollar sistema sobre cualquier tecnología.	2	2	3
Nivel de conocimiento del grupo de trabajo.	0	3	1
Soporte de Orientación a Objetos.	3	3	3
Afinidad con el sistema.	2	3	2
No presenta Resistencia a los cambios.	1	3	1
Proceso Controlado.	2	2	2
Comunicación con el cliente.	1	3	1
TOTAL:	16	28	17

Tabla 2.3 Matriz de comparación entre las Metodologías Candidatas

Valores:

Nivel Alto = 3 **Nivel Bajo = 1**

Nivel Medio = 2 **Nivel Nulo = 0**

Realizado el análisis comparativo entre las tres metodologías de desarrollo, se ha seleccionado Extreme Programming como determinante para orientar el proyecto XP, será bastante útil durante el desarrollo de éste proyecto, dado que lo que se quiere es evitar un diseño rígido del sistema, que cuando el cliente en la etapa final solicite algún cambio no sea difícil de realizar, y que, en el caso de que se lo hiciera, no alteraría muchos aspectos con respecto a lo contemplado desde un principio. Durante el desarrollo de este sistema, la

realidad va ser otra; pues se tiene que ofrecer al cliente la posibilidad de conseguir un sistema acorde a sus requerimientos (que muchas veces no los tienen claros), por lo que, se irá mostrando el proyecto a tiempo, para ir cambiándolo y poder retroceder a una fase para rediseñarlo a su gusto. Es decir, existirá una retroalimentación concreta y frecuente entre el equipo de desarrollo, el cliente y los usuarios finales. Esta metodología se acopla perfectamente al proyecto, pues: el equipo de desarrollo es pequeño, se necesita reducir tiempos de desarrollo en base a simplificar el diseño, y la disminución del número de procesos involucrados.

2.8 DESCRIPCIÓN DE LA METODOLOGÍA SELECCIONADA

2.8.1 Metodología XP (Extreme Programming)

2.8.1.1 Introducción a la metodología XP

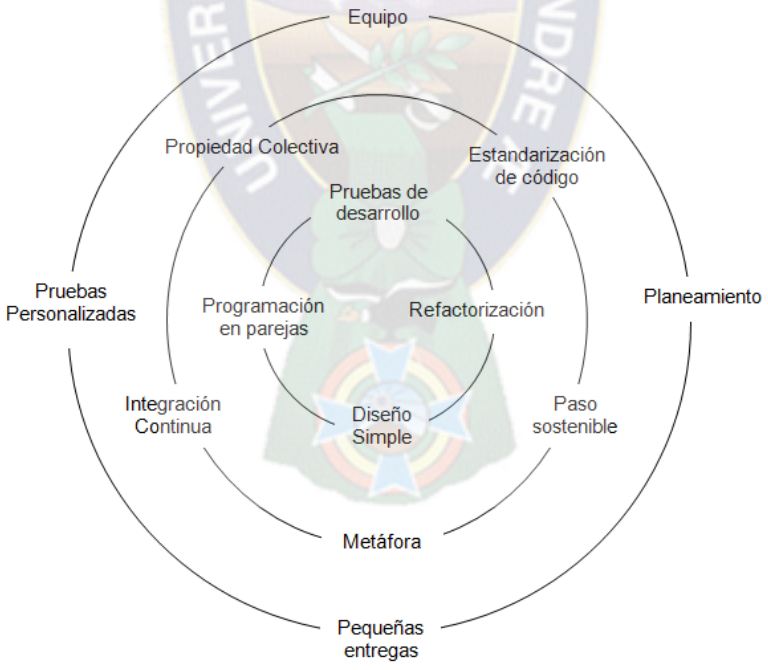


Figura 2.3 Prácticas XP

"Todo en el sistema cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar."[6]Kent Beck (1999)

Extreme Programming o Programación Extrema (XP) es una metodología ligera para el desarrollo de sistema, enunciada por Kent Beck, Ward Cunningham, y Ron Jeffries. Esta metodología persigue dos objetivos:

El objetivo principal es satisfacer los requerimientos del cliente (Implica responder de forma rápida a los cambios en las necesidades del cliente, incluso cuando éstos cambios se produzcan al final del ciclo de vida del sistema).

El segundo objetivo es potenciar al máximo el trabajo en equipo (tanto Jefes de Proyecto, como desarrolladores y clientes, forman parte del equipo encargado de la implementación del sistema).

El término "extremo" se debe a la idea planteada por Kent Beck: "Llevar las buenas prácticas de Ingeniería de Sistema al extremo". Por ejemplo:

Tabla 2.4 Lo "extremo" de XP

Buenas Prácticas	Prácticas Extremas
Sistema de pruebas estructurado	Desarrollo guiado por pruebas durante todo el desarrollo del proyecto.
Revisiones de código	Programación en parejas
Sistema funcionando totalmente	Entregas incrementales e integración continua
Tener alineado al cliente	Cliente en sitio.

[6]Figura. Procedimiento o conjunto de prácticas de Extreme Programming. Adaptado de "*Metodologías acordes con Agile Manifiesto*", por Balzar E. A. Recuperado de http://gmodulo.sourceforge.net/docs/html/manual/ch02s04.html

Si revisar el código es bueno, entonces en XP se revisará el código de manera exhaustiva, valiéndose de la programación en pareja.

Si realizar pruebas es bueno, entonces en XP se realizarán pruebas continuamente, a través de las pruebas unitarias que realiza el desarrollador, hay que añadir a este aspecto, las pruebas que realiza el cliente (pruebas de aceptación).

Si diseñar la solución del proyecto es bueno, entonces XP lo hace parte del trabajo diario del equipo de desarrollo, valiéndose de la Refactorización.

Si desarrollar realizando iteraciones cortas a lo largo del proyecto es bueno, entonces XP plantea la realización de iteraciones que en un corto periodo de tiempo.

Esta metodología está formada por cuatro partes fundamentales: valores, principios, actividades y prácticas; que se ejecutan durante todo el ciclo de desarrollo del proyecto.

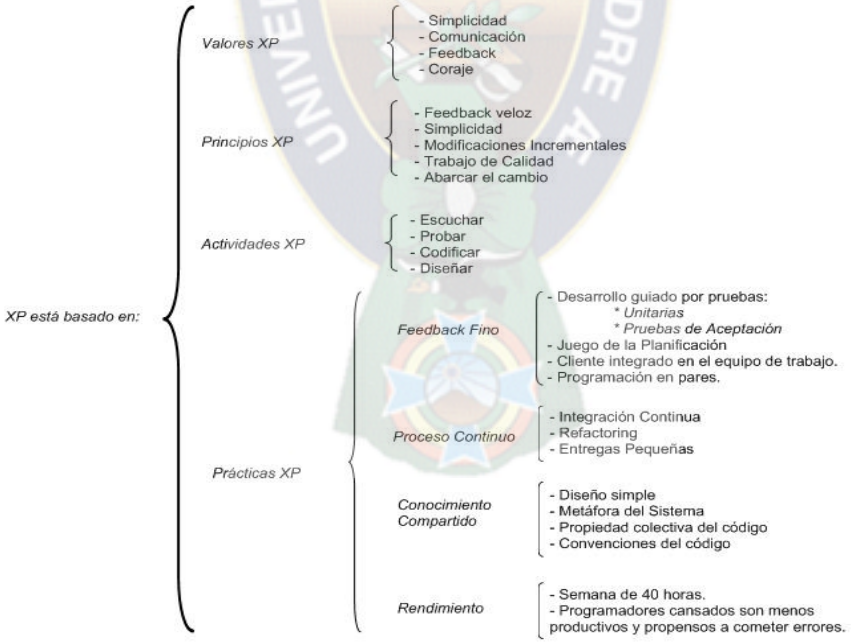


Figura 2.4 Partes fundamentales en las que se basa XP

2.8.1.2 Valores de XP

Tabla 2.5 Valores de XP

Valores	Descripción
Simplicidad	Siempre se intenta simplificar el diseño con lo cual se agilita el desarrollo y facilita el mantenimiento. Para mantener la simplicidad es necesario refactorizar el código a medida que este va creciendo, también se aplica el valor de simplicidad al momento de documentar, haciéndolo en su justa medida.
Comunicación	Constituye un valor fundamental de esta metodología. Es necesario establecer una comunicación con el cliente ya que también formará parte del equipo de desarrollo, además con la ayuda de la programación en pareja se logra mayor interacción entre los desarrolladores del proyecto.
Retroalimentación	Se ve aplicada en distintas dimensiones o etapas del desarrollo de sistema: Retroalimentación del Sistema: Al realizar pruebas unitarias o de integración, el desarrollador realiza una retroalimentación directa con el resto del equipo, indicando el estado que tiene el sistema tras haber realizado algún cambio. Retroalimentación del Cliente: Las pruebas de aceptación son escritas por el cliente y por los testers. Con esta práctica, ambos miembros del equipo reciben una retroalimentación concreta. Retroalimentación del equipo de desarrollo: Si el cliente viene al equipo de desarrollo con nuevos requerimientos, el equipo estima el tiempo que tardará en implementarlos
Coraje	El desarrollador debe ser capaz de trabajar rápidamente y rediseñar si es necesario, teniendo como soporte las pruebas de unidad y de aceptación. El desarrollador tiene la libertad para reconstruir código, modificarlo o desechar código obsoleto sin tomar en cuenta cuánto esfuerzo le costó realizarlo; si con ello se asegura que futuros cambios se implementarán fácilmente.
Respeto	El respeto tiene como base los 4 valores ya mencionados. Los miembros del equipo de desarrollo se respetan en el sentido de que no deben realizar cambios al sistema que haga fallar las pruebas de unidad existentes, o retraiga el trabajo de sus compañeros.

2.8.1.1 Actividades de XP

Tabla 2.6 Actividades de XP

Actividad	Descripción
Escuchar	Escuchar a los clientes y principalmente cuales son los problemas de su negocio, se debe mantener siempre una escucha activa, además de explicar lo que es fácil y difícil de obtener, y la realimentación entre ambos permite a todos a entender los problemas.
Probar	Todas las características del sistema deben ser demostradas mediante pruebas. Las pruebas brindan la oportunidad de saber si lo que se implementa es lo que en un principio se entendió con las historias de usuario.
Codificar	Sin código fuente no hay programa. Por tanto se necesita codificar y crear nuestras ideas a través del código, ya que al basarse XP en la reutilización esta carecería de sentido si no hay código que reutilizar.
Diseñar	El diseño crea una estructura que organiza la lógica del sistema, un buen diseño permite que el sistema crezca con cambios en un solo lugar. Los diseños deben ser sencillos, si alguna parte del sistema es de desarrollo complejo, dividirla en varias. Si hay fallos en el diseño o malos diseños, estos deben de ser corregidos cuanto antes.

2.8.1.2 Principios de XP

Esta metodología tiene cinco principios fundamentales, que toman como base los valores antes descritos.

Tabla 2.7 Principios de XP

Principio	Descripción
Feedback veloz	Aplicación de pequeños ciclos de retroalimentación.
Asunción de la Simplicidad	Ver al problema como una pieza fácil de resolver.
Modificaciones incrementales	Las modificaciones incrementales resuelven problemas realizando cambios pequeños. Se aplica a las fases de: planificación, desarrollo y diseño del proyecto.
Trabajo de Calidad	X.P eleva la calidad del código generado, por medio de pruebas unitarias y de integración antes de realizar una entrega.
Abarcar el cambio	Los Tener distintas posibles viables, mientras se resuelven problemas que son críticos para la entrega.

2.9 ARQUITECTURA TECNOLOGICA

2.9.1 Cliente - Servidor

La tecnología cliente servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesamiento, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales. Desde el punto de vista funcional, cliente servidor es una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aun en entornos multiplataforma.

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor, y este envía uno o varios mensajes con la respuesta. En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de servidor para algunas tareas y el rol de cliente para otras.

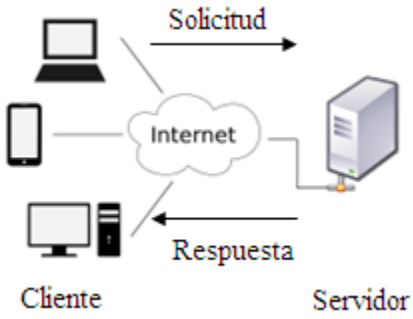


Figura 2.5Modelo Cliente Servidor

Fuente: [Delgado,08]

Tanto clientes como servidores son entidades independientes que operan conjuntamente a través de una red para realizar una tarea. Los sistemas cliente servidor deberían cumplir las siguientes características:

- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basada en el concepto de servicio que se establece entre cliente y servidor.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando a su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a peticiones de servicios al servidor. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.
- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio. Las plataformas de software y hardware entre clientes y servidores son independientes. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de su plataforma. [Delgado, 08].

2.9.2 Componentes de un Modelo Cliente/Servidor

Cliente/Servidor en un modelo basado en la idea del servicio, en que el cliente es un proceso consumidor de servicios y el servidor es un proceso proveedor de servicios. Esta relación es la petición de servicio y la realización del mismo. De este concepto se desprenden los tres elementos fundamentales sobre los cuales se desarrollan e implantan los sistemas cliente/servidor, son: el proceso cliente que es quien inicia el dialogo, el proceso servidor que pasivamente espera a que lleguen peticiones de servicio y el middleware que corresponde a la interfaz que provee la conectividad entre el cliente y el servicio para poder intercambiar mensajes.

Para entender de una forma más ordenada y clara los conceptos y elementos involucrados en esta tecnología se puede aplicar una descomposición o arquitectura por niveles. Esta descomposición principalmente consiste en separar los elementos estructurales de esta tecnología en función de aspectos más funcionales de la misma:

- **Nivel de Presentación:** Agrupa a todos los componentes asociados al componente cliente.
- **Nivel de Aplicación:** Agrupa a todos los elementos asociados al componente servidor.
- **Nivel de Comunicación:** Agrupa a todos los elementos que hacen posible la comunicación entre los componentes cliente y servidor.
- **Nivel de Base de Datos:** Agrupa a todas las actividades asociadas al acceso de los datos. [Delgado, 08].

2.9.2.1 Cliente

El cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor, se lo conoce como el término front-end. Este normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario.

Las funciones que lleva a cabo el proceso cliente son:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de base de datos.
- Recibir resultados del servidor.

2.9.2.2 Servidor

Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrativo por él. Al proceso servidor se lo conoce con el término back-end.

Las funciones que lleva a cabo el proceso servidor son:

- Aceptar los requerimientos de base de datos que hacen los clientes.
- Procesar requerimientos de base de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de las aplicaciones y realizar validaciones a nivel de base de datos.

2.10 HERRAMIENTAS

Entre las herramientas que se utiliza son:

- **Base De Datos:** "Base de Datos es un conjunto de información estructurada en registros y almacenada en un soporte electrónico legible desde un ordenador. Cada registro constituye una unidad autónoma de información que puede estar a su vez estructurada en diferentes campos o tipos de datos que se recogen en dicha base de datos" [Matsukawa, 05].

Una Base de Datos es un conjunto auto descriptivo de registros integrados persistente que es utilizado este sistema. Por persistentes queremos decir, de manera intuitiva, que el tipo de datos de la base de datos difieren de otros datos más efímeros como los datos de entrada y salida.

- **Motor de Base de Datos Mysql:** "Es el servicio para almacenamiento, procesamiento y manejo de la seguridad de los datos. Proporciona acceso controlado y procesamiento rápido de transacciones para satisfacer los requerimientos de las aplicaciones empresariales que demandan gran consumo de datos, y soporte para alta disponibilidad de datos" [Matsukawa, 05].
- **Intranet:** "es una red privada corporativa que emplea para su configuración y funcionamiento operativo los protocolos de la tecnología de Internet (IP), ubicada privadamente en un servidor y la que tienen acceso únicamente las personas autorizadas".[Esteban, 09]
- **Servidor Web Apache, Xampp:** Es un servidor local que permite que podamos acceder a nuestro sitio en forma local (en nuestro ordenador) como si estuviésemos accediendo a él a través de internet, es decir, con una dirección. Xampp una distribución de Apache que incluye MySQL, PHP y otras herramientas para el desarrollo de aplicaciones web, como phpMyAdmin.
- **Lenguaje de Programación PHP**: Es un lenguaje de script del lado del servidor, los scripts PHP están incrustados en los documentos HTML y el servidor los interpreta y ejecuta antes de servir las páginas al cliente, el cliente no ve el código PHP sino los resultados que produce.

Es un lenguaje de programación con variables, sentencias condicionales, ciclos (bucles), funciones, no es un lenguaje de marcado como podría ser HTML, XML o WML. Está más cercano a JavaScript o a C, es por tal motivo que se decide realizar con este sistema con este lenguaje.

- **Lenguaje de Modelado Unificado (UML)**

Lenguaje estándar para describir planos de sistema, el cual puede utilizarse para visualizar, construir y documentar los artefactos de los sistemas.

Para ayuda en el análisis del sistema se empleara el siguiente diagrama:

Diagrama de Clases.- Que muestra el conjunto de clases y objetos importantes que forman parte de un sistema, junto con las relaciones existentes entre las clases y objetos.

Clase: Representa un conjunto de entidades que tienen propiedades comunes.

Objeto: Denominado instancia de la clase, representada por u rectángulo con tres divisiones internas.

Atributo: Representa una propiedad de una entidad.

Operación: El conjunto de operaciones que describen el comportamiento de los objetos de una clase.

Asociación: Representa las relaciones entre instancias de clase. Es una línea que une dos o más clase.

Dependencia: Es una relación donde existe entidades independientes y otras dependientes.

Los tipo de asociación entre clases presentes en un diagrama estático son: Binaria, n-aria, composición.

Diagrama de casos de uso.- Un diagrama de casos de uso, muestra las distintas operaciones que se esperan de una aplicación o sistema y como se relaciona con su entorno (usuario u otras aplicaciones).

Los casos de uso representan en el diagrama de un eclipse que denota un requerimiento solucionado por el sistema. Cada caso de uso es una operación completa desarrollada por los actores y por el sistema de dialogo.

2.11 MÉTRICAS DE CALIDAD - CALIDAD DE SISTEMA

ISO 9126 es un estándar internacional para la evaluación de la calidad del sistema. Está reemplazado por el proyecto SQuaRE, ISO 25000:2005, el cual sigue los mismos conceptos. Este estándar está dividido en cuatro partes las cuales dirigen, realidad, métricas externas, métricas internas y calidad en las métricas de uso y expendido. El modelo de calidad establecido en la primera parte del estándar, ISO 9126-1, clasifica la calidad del sistema en un conjunto estructurado de características y sub-características de la siguiente manera:

2.11.1 Confiabilidad (Fiabilidad)

En términos estadísticos se define como la probabilidad de operación libre de fallos del sistema en un entorno determinado y durante el tiempo (Pessman, 2006), este factor es esencial para la evaluación de la calidad del sistema.

El nivel de confiabilidad del sistema está estrictamente relacionado con la cantidad de errores que lanza el sistema durante el tiempo de ejecución y no en tiempo real. La confiabilidad es expresada en una escala del 0 al 1, en el que el sistema será muy confiable si el valor obtenido se acerca a 1 y viceversa, es decir que a menor error, mayor será la confiabilidad ante los ojos del usuario.

2.11.2 Facilidad de Mantenimiento

Se centra más en el cambio que va asociado a la corrección de errores durante la creación del sistema (mantenimiento correctivo), dicho en otras palabras se concentra más a las adaptaciones requeridas a medida que evoluciona el entorno del sistema (mantenimiento

adaptativo), esto como consecuencia de cambios en las políticas de la empresa, a la adaptación de nuevos sistemas operativos, y al descubrimiento de funciones adicionales que van a producir beneficios más allá de sus requisitos funcionales originales (mantenimiento perpetuo) y finalmente el cambio del artículo pensando en mejoras futuras(Mantenimiento preventivo o ingeniería del sistema).

La facilidad de mantenimiento de un sistema es la facilidad con la que se puede localizar y corregir un error o errores dentro del sistema.

2. 11.3 Usabilidad

La usabilidad se define como el esfuerzo necesario para comprender el sistema en general reparar los datos de entrada e interpretar las salidas. Es el intento por medir lo amigable que puede ser un programa con el usuario final.

2.12 SEGURIDAD

Es muy importante conocer la función informática, de forma esencial cuando su manejo está basado en tecnología moderna, para esto se debe conocer que la información esta almacenada y procesada en computadoras, y esta puede ser confidencial para algunas personas o a escala institucional, puede ser mal utilizada o divulgada, puede estar sujeta a robos, sabotaje o fraudes.

Resguardo de la Información: Efectuar copias de seguridad periódicas del sitio, con un esquema adecuado de frecuencia, tipo de resguardo, rotación y reutilización de medios de almacenamiento. Las copias deben ser almacenadas en un lugar físicamente seguro, de manera de prevenir daños físicos, robo o pérdida de las mismas. Probar periódicamente la correcta restauración de las copias de seguridad. En los casos en las cuales se disponga la eliminación de información o bien la reutilización de medios de almacenamiento, efectuar la eliminación de la información de forma segura.

Control de Cambios: Gestionar los cambios de forma de garantizar que los mismos se efectúan de forma autorizada y segura. Documentar e implementar procedimientos para la solicitud, evaluación, diseño, prueba e implementación de los cambios. Implementar un mecanismo de solicitud de cambios formalizado. Documentar e implementar un procedimiento de análisis de los cambios solicitados, incluyendo la evaluación del impacto del cambio sobre la seguridad de la información.

2.13 IMPLEMENTACIÓN

Es poner en marcha el resultado del diseño en términos de componentes del software, para ello se debe definir la organización del código, implementar los códigos fuente y ejecutables y probar dichos componentes para integrarlos al proyecto final.

2.13.1 Pruebas (Diseño de casos de prueba)

Es necesario crear los casos de prueba, donde se especifica la forma de probar el sistema como un todo. Esto quiere decir realizar:

- Pruebas de instalación, instalación en la plataforma del cliente.
- Pruebas de configuración.
- Pruebas negativas, encontrar debilidades del sistema.
- Pruebas de tensión o de estrés, cuando hay recursos insuficientes.
- Prueba de integración al sistema.

Cualquier producto de ingeniería puede probarse de dos formas: primero conociendo la función específica para la que fue desarrollada el producto, se pueden desarrollar pruebas que aseguren que todas las piezas encajan, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada [Pressman, 05].

2.13.2 Pruebas de caja negra

Las pruebas de caja negra, también denominadas pruebas de comportamientos, se centran en los requisitos funcionales del software. Es decir la prueba de caja negra permite al ingeniero del software obtener conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.

Las pruebas de caja negra intentan encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores de estructuras de datos o en acceso a base de datos externas.
- Errores de rendimiento.
- Errores de inicio y fin.

2.13.3 Pruebas de caja blanca

Las pruebas de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control de diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca.

- Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejerciten las estructuras internas de datos para asegurar su validez [Pressman, 05].

CAPITULO III

MARCO APLICATIVO

En el Marco Aplicativo se detalla y aplica de forma clara los aspectos relacionados con los diferentes procesos que existen en la empresa.

Teniéndose como base la metodología seleccionada en el capítulo anterior, se estructuran los temas de este capítulo por las disciplinas que la metodología Programación Extrema (XP) define, además de hacer uso de algunas de las herramientas UML para documentar, ya que XP no utiliza diagramas para la documentación y es uno de los puntos en los que se apoya con herramientas UML.

En el presente capitulo se realiza una descripción de las características principales del sistema a desarrollar, de acuerdo al problema por el cual fue concebido. Se especifican las funcionalidades que se desean informatizar buscando satisfacer las necesidades de los clientes. Además, se describe la construcción de la propuesta con el desarrollo de las historias de usuario a través de iteraciones, se definen tareas y principios de diseño para su implementación. Se exponen pruebas de aceptación para garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable.

3.1 LOS CUATRO VALORES

Los cuatro valores que según Xtreme Programming (XP) se debe tomar en cuenta en el desarrollo del software son:

3.1.1 Comunicación

Los se nutre con este valor mediante la comunicación directa con el encargado del área, constantemente y cuando se tiene dudas, así también con los usuarios de otras áreas, contabilidad y trabajadores.

3.1.2 Coraje

Durante el desarrollo del presente proyecto se expuso las dudas, para poder captar la idea de los requerimientos del usuario mediante las reuniones o comunicación directa con el personal del área, debido a ello se tuvo que rehacer los puntos que no eran satisfactorios.

3.1.3 Simplicidad

La simplicidad en el diseño se refleja en las iteraciones del presente proyecto. Se puede evidenciar en la documentación que el sistema tiene solo la funcionalidad requerida en cada historia de usuario, no se incorporan funcionalidades que no son requeridas en las historias de usuario.

3.1.4 Continuo Seguimiento

Por medio de las pruebas de usuario y aceptación, reuniones con personal del área se mantiene una constante evolución del desempeño del sistema, adicionando al mismo tiempo nuevas funcionalidades que se veía en algunas de las iteraciones.

3.2 LOS DOCE PRINCIPIOS

A continuación se presentan los doce principios de la metodología XP que en el caso de la programación extrema son llevados a tal punto que hace que el proyecto tenga éxito, los cuales no son nuevos ya que son usados por otras metodologías.

3.2.1 Retroalimentación a escala fina

- El Principio de pruebas: Para cada historia de usuario se emplearon pruebas de aceptación, las cuales se realiza de manera manual, cuyos resultados se reflejan en las respectivas iteraciones propuestas.

- **Proceso de planificación:** Descrito en el punto 3.5 del presente capitulo, donde se detalla la planificación.
- **El cliente en el sitio:** Se implementó el cliente a través del desarrollo del proyecto en instalaciones de la Empresa For Security 4.5, la cual hace del cliente solo para efectos de la documentación.
- **Programación en parejas:** El presente proyecto se lo realizo solo por una persona, la cual es responsable del desarrollo del presente, pero con el constante intercambio de opiniones y trabajo conjunto con el personal encargado del área de sistemas.

3.2.2 Proceso continuo en lugar por lotes

- **Integración continua:** El sistema es integrado a medida que culmina el desarrollo de cada historia de usuario, antes de pasar las pruebas respectivas.
- **Refactorización:** Las historias de usuario que no superaron las pruebas respectivas se volvieron a programar.
- **Entregas pequeñas:** Se realizó pequeñas entregas a la conclusión de cada historia de usuario para que se implante y se entregue. Los oficiales se empaquetan al final de cada iteración.

3.2.3 Entendimiento compartido

- **Diseño simple:** al contar con la colaboración del encargado del área se mantiene la documentación del proyecto transparente y de fácil acceso a él, lo que hace que el diseño sea evidentemente simple.
- **Metáfora:** Se evidencian en la presentación de las iteraciones que son utilizadas para el proyecto, son sencillas.
- **Propiedad colectiva del código:** La propiedad del código será de propiedad de la empresa una vez culminado el sistema.

- **Estándar de codificación:** debido a que el presente proyecto es individual no se tuvo problemas con el estándar en la codificación.

3.2.4 Bienestar del programador

- Semana de 40 horas: El presente proyecto se lo trabajo cuatro días al mes en coordinación con la empresa (días sábados exclusivos para la implementación del sistema) con revisiones semanales con el que se supera las 40 horas de trabajo.

Descripción de los Actores de la Empresa

Los actores identificados son:

- **Gerente:** Es la persona encargada de realizar un control y seguimiento de la Empresa For Security 4.5.
- **Administrador:** Es la persona encargada que administra el sistema, asignado a los permisos de acceso según el rol de cada usuario, así como también es el que puede realizar modificaciones de los datos y registros.
- **Encargado de almacén:** Es la persona usuario del sistema, registra, modifica la adquisición y salida del producto.
- **Vendedor:** Es la persona usuario que tiene acceso al registro de ventas, stock de los productos y cotizaciones.

MODULO ALMACÉN

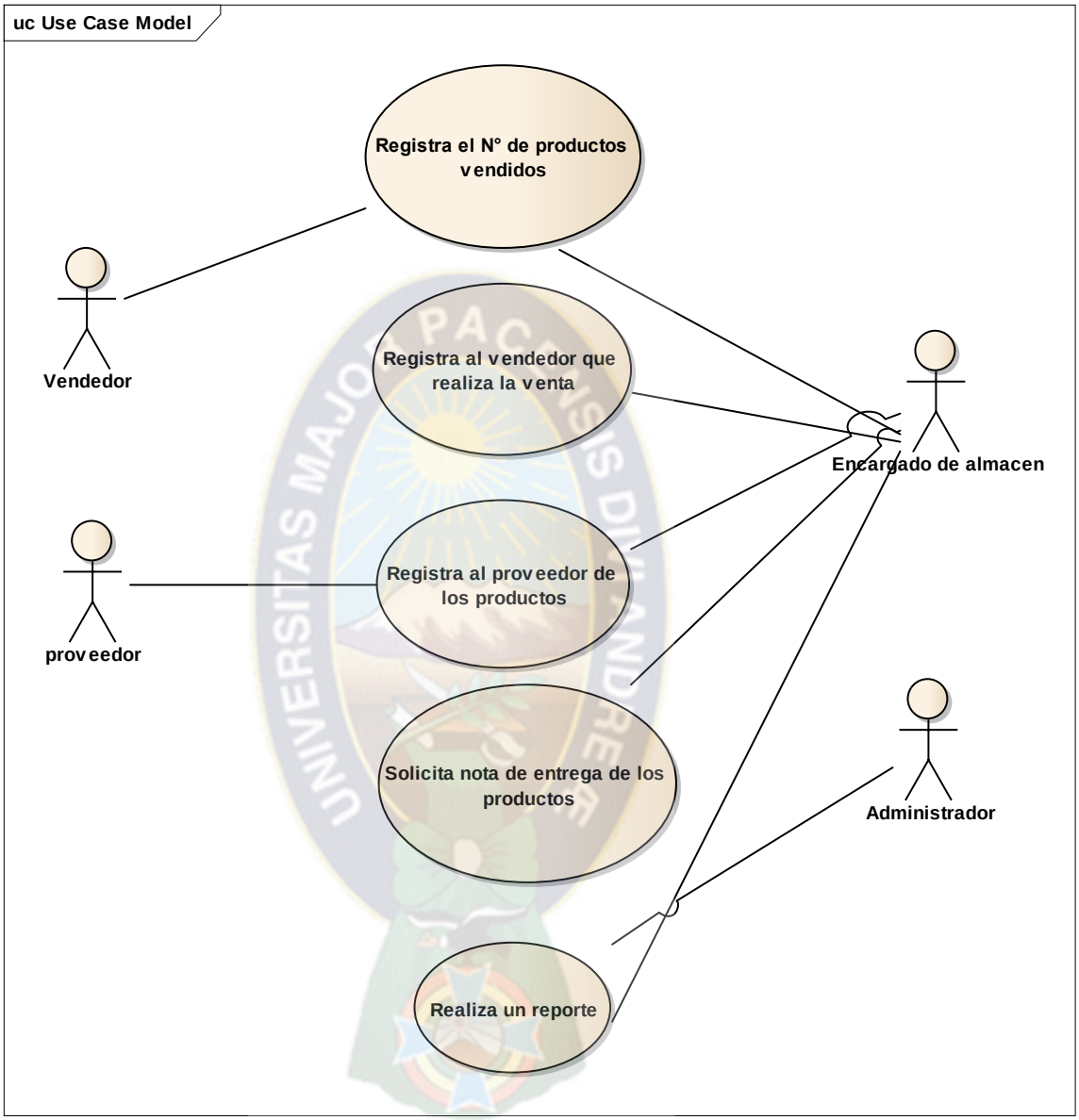


Figura 3.1 Diagrama de casos de uso general

Fuente: [Elaboración propia]

MODULO COTIZACIONES

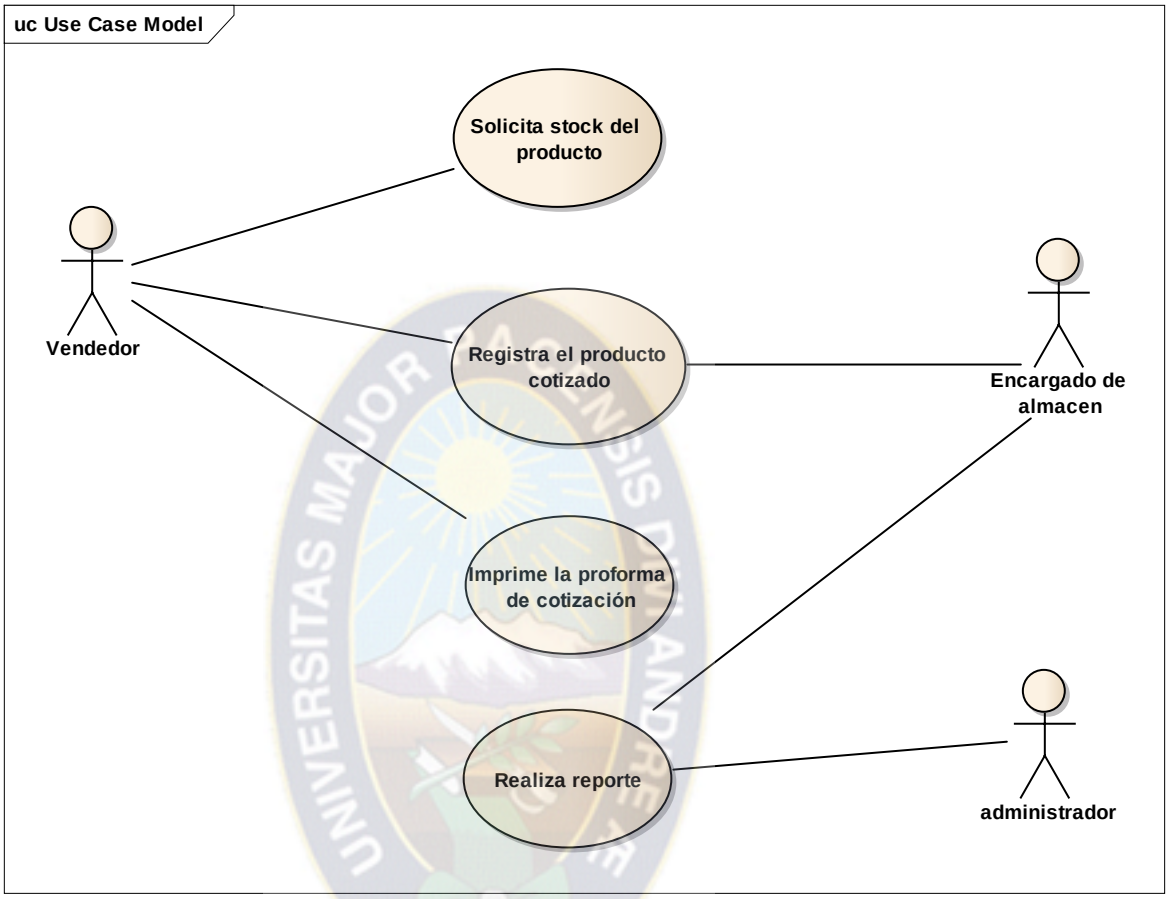


Figura 3.2 Diagrama de casos de uso general

Fuente: [Elaboración propia]

3.3 DESCRIPCION DE LOS ESCENARIOS

MODULO ALMACÉN

Título: Registro de ingreso y salida de los productos	**Propósito:** Se ejecuta cada vez que se adquiere y vende un producto.
Actores: Encargado de Almacén	
Episodios: 1. El encargado de almacén efectúa el registro del producto que el vendedor retira de almacén e ingresa por parte del proveedor. 2. El sistema registra los datos del sistema. 3. El registro es almacenado en la base de datos.	

Tabla 3.1 Descripción de escenarios para registro de productos vendidos.

Fuente: [Elaboración propia]

Título: Registro del vendedor.	**Propósito:** Se ejecuta cada vez que el vendedor retira el producto del almacén.
Actores: Encargado de Almacén	
Episodios: 1. El encargado de almacén registra al agente de ventas por la venta del producto. 2. El sistema registra los datos del sistema. 3. El registro es almacenado en la base de datos.	

Tabla 3.2 Descripción de escenarios para registro del agente de ventas.

Fuente: [Elaboración propia]

Título: Registro de proveedor.	**Propósito:** Se ejecuta cada vez que el proveedor entrega el producto a almacén.
Actores: Encargado de Almacén	
Episodios: 1. El encargado de almacén registra al proveedor una vez entregado los productos. 2. El sistema registra los datos del sistema. 3. El registro es almacenado en la base de datos.	

Tabla 3.3 Descripción de escenarios para el registro del proveedor.

Fuente: [Elaboración propia]

Título: Registro del producto comprado y vendido.	**Propósito:** Se ejecuta cada vez que el usuario compra el producto al proveedor y lo vende al cliente.
Actores: Administrador	
Episodios: 1. El sistema procederá a la base de datos para el registro del producto comprado y vendido. 2. El sistema registra los datos del sistema. 3. El registro es almacenado en la base de datos.	

Tabla 3.4 Descripción de escenarios para el registro del producto comprado y vendido.

Fuente: [Elaboración propia]

MODULO COTIZACIONES

Título: Generar stock del producto	**Propósito:** Se ejecuta cada vez que el agente de ventas realiza una cotización.
Actores: Vendedor	
Episodios: 1. El sistema procederá a la base de datos para verificar la existencia del producto. 2. El sistema registra los datos del sistema. 3. El registro es almacenado en la base de datos.	

Tabla 3.5 Descripción de escenarios para la solicitud de stock del producto.

Fuente: [Elaboración propia]

Título: Generar cotización.	**Propósito:** Se ejecuta cada vez que el vendedor realiza la proforma del producto al cliente.
Actores: Vendedor	
Episodios: 1. El vendedor procederá al llenado de la proforma de cotización. 2. El sistema registra los datos del sistema. 3. El registro es almacenado en la base de datos.	

Tabla 3.6 Descripción de escenarios para la proforma de la cotización.

Fuente: [Elaboración propia]

Título: Reportes de impresión.	**Propósito:** Se ejecuta cada vez que el usuario lo requiere.
Actores: Administrador	
Episodios: 1. El sistema procederá a la base de datos para la emisión del reporte. 2. El sistema registra los datos del sistema. 3. El registro es almacenado en la base de datos.	

Tabla 3.7 Descripción de escenarios para imprimir los reportes.

Fuente: [Elaboración propia]

3.4 FASE DE EXPLORACIÓN

En esta fase se define el alcance del proyecto y al mismo tiempo el equipo de trabajo se familiariza con las herramientas y tecnologías que se utilizaran en el mismo, realizándose las historias de usuarios. La duración de esta etapa puede variar en dependencia de la familiarización que tengan los desarrolladores con las herramientas.

3.4.1 Historias de Usuario.

Las historias de usuario (HU) son el medio mediante el cual se logra una especificación de los requisitos que conformarán el sistema. Éstas son generadas por el cliente contando con alguna ayuda del desarrollador, en caso de ser necesario. El nivel de detalle de las HU debe de ser el mínimo posible que permita hacerse una idea de cuánto costará realizar la implementación del sistema.

Historia de Usuario	
Número: 1	**Usuario:** Encargado de Almacén
Nombre historia: Registro de ingreso y salida de productos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Una vez que se procede con la adquisición y salida del producto se registra y almacena en la base de datos.	
Observaciones:	

Tabla 3.8 Historia de usuario "Registro de productos".

Historia de Usuario	
Número: 2	**Usuario:** Encargado de Almacén.
Nombre historia: Registro del vendedor.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Permite ingresar el nombre del vendedor que realizó el retiro de productos del almacén.	
Observaciones:	

Tabla 3.9 Historia de usuario "Registro del vendedor".

Historia de Usuario	
Número: 3	**Usuario:** Encargado Almacén.
Nombre historia: Registro del proveedor.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Permitir ingresar el nombre del proveedor que realizó la entrega de productos a almacén.	
Observaciones:	

Tabla 3.10 Historia de usuario "Registro del proveedor".

Historia de Usuario	
Número: 4	**Usuario:** Encargado de Almacén
Nombre historia: Registro de compra y venta de productos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Una vez que se procede con la venta y compra del producto se registra y almacena en la base de datos.	
Observaciones:	

Tabla 3.11 Historia de usuario "Registro de productos vendidos y comprados".

Historia de Usuario	
Número: 5	**Usuario:** Vendedor.
Nombre historia: Gestionar stock del producto.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Se desarrolló una ventana en la cual se verificará la existencia del producto requerido en almacenes.	
Observaciones:	

Tabla 3.12 Historia de usuario "Gestionar stock del producto".

Historia de Usuario	
Número: 6	**Usuario:** Vendedor.
Nombre historia: Generar cotización.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Se desarrolló una plataforma de cotización para ingresar los datos de productos solicitados por el cliente.	
Observaciones:	

Tabla 3.13 Historia de usuario "Generar cotizaciones".

Historia de Usuario	
Número: 7	**Usuario:** Administrador.
Nombre historia: Generar reportes.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: El administrador podrá imprimir los reportes de los sucesos ocurridos en almacenes y la de cotizaciones.	
Observaciones:	

Tabla 3.14 Historia de usuario "Generar reportes".

3.5 FASE DE PLANIFICACIÓN

Durante la fase de planificación el usuario establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto se considera como una semana ideal de trabajo, donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción. Esta estimación incluye todo el esfuerzo asociado a la implementación de la historia de usuario.

3.5.1 Estimación de esfuerzos por historias de usuario.

Para el desarrollo de la aplicación propuesta en este trabajo se realizó una estimación del esfuerzo para cada una de las historias de usuario identificadas, permitiendo tener una medida real de la velocidad de progreso del proyecto y brindando una guía razonable a la cual ajustarse, llegándose así a los resultados que se muestran en la siguiente tabla.

No	Historia de usuario	Prioridad	Riesgo	Esfuerzo	Iteración
1	Registro de ingreso y salida de productos	Alta	Medio	1	1
2	Registro vendedor	Alta	Alto	2	1
3	Registro proveedor	Alta	Alto	2	1
4	Registro de compra y venta de productos	Alta	Medio	1	1
5	Gestionar stock del producto	Alta	Medio	1	1
6	Generar cotización	Alta	Medio	2	1
7	Generar reportes	Alta	Alto	1	1

Tabla 3.15 Estimación de esfuerzos por historia de usuario.

3.5.2 Plan de duración de las iteraciones.

Una vez identificadas las historias de usuario del sistema y estimado el esfuerzo dedicado a la realización de cada una de estas se procede a la planificación de la etapa de implementación del proyecto.

Para ello XP define la elaboración del Plan de Entrega, compuesto por iteraciones en semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio).

De acuerdo a lo mencionado anteriormente se decidió realizar dicha planificación en dos iteraciones detalladas a continuación.

Iteración	Orden de las historias de usuario	Duración de la iteración
Primera Iteración	1-Registro de ingreso y salida de productos 2- Registro vendedor 3-Registro proveedor 4-Registro de compra y venta de productos	8 Semanas y 3 días
Segunda Iteración	5-Gestionar stock del producto 6-Generar cotización 7- Generar reportes	3 semanas y 2 días

Tabla 3.16 Plan de duración de las iteraciones.

3.5.3 Plan de entrega.

Después de determinar qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas se realiza el cronograma de entregas que establece la fecha acordada con el cliente para la liberación de las diferentes versiones. En la siguiente tabla se muestra el Plan de duración de entregas en el cual se especifican un aproximado de las fechas para cada iteración.

Iteración	**Iteración 1**	**Iteración 2**
Entrega	Final 1ra Iteración 1ra semana de Octubre	Final 2da Iteración 2da semana de Noviembre

Tabla 3.17 Plan de Duración de la Entrega.

3.6 ITERACIONES.

Esta es la fase principal en el ciclo de desarrollo de XP, incluye varias iteraciones sobre el sistema antes de ser entregado. Las funcionalidades son desarrolladas en esta fase,

generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el usuario todos los datos que sean necesarios. El usuario, por lo tanto, también debe participar activamente durante esta fase del ciclo. Las iteraciones son también utilizadas para medir el progreso del proyecto. Una iteración terminada sin errores es una medida clara de avance.

A continuación se muestra el desarrollo de las tareas por cada iteración:

Iteración 1.

El objetivo de esta iteración es darle cumplimiento a las historias de usuarios 1, 2, 3, 4, las cuales están relacionadas con los datos que se gestionan la Empresa de Seguridad Electrónica For Security 4.5 y resultan ser las de más importancia.

Iteración 2.

Esta iteración se centra en darle solución a las historias de usuario 5, 6, 7, las mismas se basan en visualización de información y reportes.

3.6.1 Tareas.

Todo el trabajo de las iteraciones es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable. Estas historias de usuario son divididas en tareas de entre 1 y 5 días de duración asignadas a la programación.

A continuación se presentan algunas de las tareas pertenecientes a las diferentes historias de usuario definidas.

Tarea	
Número tarea: 1.1	**Nombre de historia:** Registro de ingreso y salida de productos.
Nombre tarea: Diseño y programación del registro de productos.	
Tipo de tarea: Diseño- Desarrollo	**Puntos estimados:** 1
Fecha inicio: 1/08/2015	**Fecha fin:** 8/08/2015
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Se diseñará una página para que el usuario pueda registrar tanto el ingreso como la salida de los productos de almacenes, se programarán métodos.	

Tabla 3.18 Tarea 1.1 de la historia de usuario "Registro de productos".

Tarea	
Número tarea: 2.1	**Nombre de historia:** Registro del vendedor
Nombre tarea: Diseño de interfaz de HU y programación de insertar vendedor.	
Tipo de tarea: Diseño- Desarrollo	**Puntos estimados:** 2
Fecha inicio: 9/08/2015	**Fecha fin:** 23/08/ 2015
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Se diseñará una página para que el usuario pueda insertar el nombre del vendedor, se programarán métodos que permitan insertar nombres.	

Tabla 3.19 Tarea 2.1 de la historia de usuario "Registro del vendedor".

Tarea	
Número tarea: 3.1	**Nombre de historia:** Registro del proveedor.
Nombre tarea: Diseño de interfaz de HU y programación de insertar proveedor.	
Tipo de tarea: Diseño - Desarrollo	**Puntos estimados:** 2
Fecha inicio: 24/08/2015	**Fecha fin:** 6/09/2015
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Se diseñará una página para que el usuario pueda insertar el nombre del proveedor, se programarán métodos que permitan insertar nombres.	

Tabla 3.20 Tarea 3.1 de la historia de usuario "Registro del proveedor".

Tarea	
Número tarea: 4.1	**Nombre de historia:** Registro de compra y venta de productos.
Nombre tarea: Diseño de interfaz de HU y programación de insertar compra y venta.	
Tipo de tarea: Diseño-Desarrollo	**Puntos estimados:** 1
Fecha inicio: 7/09/2015	**Fecha fin:** 14/09/2015
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Se diseñará una página para que el usuario pueda insertar venta y compra, se programarán métodos que permitan insertar ventas y compras.	

Tabla 3.21 Tarea 4.1de la historia de usuario "Registro de productos vendidos y comprados".

Tarea	
Número tarea: 5.1	**Nombre de historia:** Gestionar stock del producto.
Nombre tarea: Diseño de interfaz de HU y programación de búsqueda del producto.	
Tipo de tarea: Diseño - Desarrollo	**Puntos estimados:** 1
Fecha inicio: 1/10/2015	**Fecha fin:** 8/10/2015
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Se programarán métodos que permitan buscar cierto producto.	

Tabla 3.22 Tarea 5.1 de la historia de usuario "Gestionar venta".

Tarea	
Número tarea: 6.1	**Nombre de historia:** Generar cotización.
Nombre tarea: Diseño de interfaz de HU y programación que generara cotizaciones.	
Tipo de tarea: Diseño-Desarrollo	**Puntos estimados:** 2
Fecha inicio: 9/10/2015	**Fecha fin:** 23/10/2015
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Se diseñará una plataforma para que el usuario pueda generar una cotización, se programarán métodos que permitan realizar aquello.	

Tabla 3.23 Tarea 6.1 de la historia de usuario "Generar cotización".

Tarea	
Número tarea: 7.1	**Nombre de historia:** Generar reportes.
Nombre tarea: Diseño de interfaz de HU y programación que generara reportes.	
Tipo de tarea: Diseño-Desarrollo	**Puntos estimados:** 1
Fecha inicio: 24/10/2015	**Fecha fin:** 1/11/2015
Programador responsable: Jaime Junior Rada Aranibar	
Descripción: Se diseñará una plataforma para que el usuario pueda generar reportes, se programarán métodos que permitan realizar reportes.	

Tabla 3.24 Tarea 7.1 de la historia de usuario "Generar reportes".

3.7 PRODUCCIÓN

En esta fase se realizan las tareas planificadas por iteración. Para ello se diseña, se codifica y se prueba. Se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

3.7.1 Diseño

Este sistema fue diseñado para facilitar la gestión de la información en la Empresa de Seguridad Electrónica For Security 4.5, de forma dinámica y agradable al usuario. Para lograrlo se emplearon algunos principios de diseño visual en las páginas web que la conforman.

El diseño visual define la apariencia del sistema y es de gran importancia para lograr que el usuario se sienta satisfecho con la información que obtiene y con la forma en que lo hace por eso la aplicación presenta un diseño simple y sencillo, sin muchas complicaciones, orientado al entorno de trabajo del usuario para que se sienta identificado con la aplicación.

Se eligieron los colores en la gama de los azules pues se considera que estos colores contribuyen a construir una interfaz agradable a la vista del usuario. Se utilizó el negro para las letras garantizando una lectura favorable de los textos. Se usa la familia de letras verdana, arial, helvetica, sans-serif para los textos de las páginas. Este tipo de letra permite una lectura rápida y cómoda. Es mínimo el uso de imágenes y animaciones para evitar largos tiempos de espera a la hora de cargar la página y visualizarlas.

Para la construcción del sistema se tomaron en cuenta algunos de los estándares de implementación propuestos: un header o banner, donde se muestra la información general de sistema como logo, el nombre del sistema e imágenes y textos que muestren de manera general el contenido de la aplicación, un menú en la parte frontal izquierda donde se encuentran los diferentes vínculos de acceso a las secciones del sistema, la sección del contenido donde se muestra la información que se desea buscar.

Una muestra de las pantallas del sistema es la de registro de productos, que puede ser consultada en Anexos.

El diseño de la base de datos fue realizado, la cuales están normalizadas, cumpliendo con las normas establecidas para el diseño de bases de datos. El modelo físico de la base de datos de la aplicación puede ser visto en Anexos.

3.7.2 Codificación.

En la implementación del sistema se utiliza el lenguaje de programación web PHP (Personal Home Page), el cual es un lenguaje del lado del servidor y es diseñado originalmente para la creación de aplicaciones web dinámicas. Se emplean clases en el código fuente porque según las características del sistema se considera que es necesario utilizar la programación orientada a objetos (POO). Además se hace evidente el uso de la reutilización de código pues las funcionalidades del sistema presentan características en común.

3.7.3 Pruebas de sistema.

Uno de los pilares fundamentales de XP es el proceso de prueba, la cual constituyen el último bastión desde el que se puede evaluar la calidad de forma pragmática y descubrir los errores. Las pruebas son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente.

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el usuario.

3.7.3.1 Pruebas de Aceptación.

Las pruebas de aceptación son pruebas de caja negra que se realizan a partir de las historias de usuarios. Durante las iteraciones las historias de usuarios escogidas serán traducidas a prueba de aceptación. En ella se especifican, la perspectiva del usuario, y los escenarios para probar que la historia de usuario ha sido implementada correctamente. Una historia de usuario puede tener todas las pruebas de aceptación que desee para asegurar su funcionamiento.

El objetivo específico de esta prueba es garantizar que los requerimientos han sido cumplidos y que el sistema ha sido aceptable. A continuación se muestran algunas de las pruebas de aceptación propuestas a realizase.

MODULO ALMACÉN

Pruebas de aceptación para la historia de usuario 1

- Identificar todos los posibles resultados observables de la historia:
    - Se tiene la interfaz para registrar el ingreso y salida de los productos.
- Identificar los resultados que terminan la historia y los que permiten continuar dentro de la historia:
    - La historia termina cuando se llena el formulario de registro de productos.
- Identificar los caminos de ejecución posibles:
    - El formulario pide el registro de los movimientos de productos en el día.
    - Valida los datos.
    - La historia termina a petición del encargado de almacén, cuando no se desea registrar mayor información.
- Asignar un conjunto de valores Validos y valores del entorno a cada camino de ejecución para obtener resultados esperados:
    - El conjunto de valores valiosos está dado por el conjunto de datos compuestos por la información que se introduce sobre los datos.
- Eliminación de caminos redundantes:
    - No existen caminos redundantes.

Pruebas de aceptación para la historia de usuario 2

- Identificar todos los posibles resultados observables de la historia:
    - Se tiene la interfaz para el registro de datos personales del vendedor.
- Identificar los resultados que terminan la historia y los que permiten continuar dentro de la historia:
    - La historia termina cuando se llena el formulario de registro de datos personales del vendedor.
- Identificar los caminos de ejecución posibles:
    - El formulario pide el registro de los datos del vendedor.
    - Valida los datos.
    - La historia termina a petición del encargado de almacén, cuando no desea registrar mayor información.
- Asignar un conjunto de valores Validos y valores del entorno a cada camino de ejecución para obtener resultados esperados:
    - El conjunto de valores valiosos está dado por el conjunto de datos compuestos por la información que se introduce sobre los datos.
- Eliminación de caminos redundantes:
    - No existen caminos redundantes.

Pruebas de aceptación para la historia de usuario 3

- Identificar todos los posibles resultados observables de la historia:

- Se tiene la interfaz para el registro de datos personales del proveedor.

- Identificar los resultados que terminan la historia y los que permiten continuar dentro de la historia:

  - La historia termina cuando se llena el registro de datos personales del proveedor.

- Identificar los caminos de ejecución posibles:

  - El formulario pide el registro de los datos del vendedor.
  - Valida los datos.
  - La historia termina a petición del encargado de almacén, cuando no desea registrar mayor información.

- Asignar un conjunto de valores Validos y valores del entorno a cada camino de ejecución para obtener resultados esperados:

  - El conjunto de valores valiosos está dado por el conjunto de datos compuestos por la información que se introduce sobre los datos.

- Eliminación de caminos redundantes:

  - No existen caminos redundantes.

Pruebas de aceptación para la historia de usuario 4

- Identificar todos los posibles resultados observables de la historia:

  - Se tiene la interfaz para registrar el producto comprado y vendido.

- Identificar los resultados que terminan la historia y los que permiten continuar dentro de la historia:

    - La historia termina cuando se llena el formulario de registro de compra y venta de productos.

- Identificar los caminos de ejecución posibles:

    - El formulario pide el registro del producto comprado y vendido.
    - Valida los datos.
    - La historia termina a petición del encargado de almacén, cuando no se desea registrar mayor información.

- Asignar un conjunto de valores Validos y valores del entorno a cada camino de ejecución para obtener resultados esperados:

    El conjunto de valores valiosos está dado por el conjunto de datos compuestos por la información que se introduce sobre los datos.

- Eliminación de caminos redundantes:

    - No existen caminos redundantes.

MODULO COTIZACIONES

Pruebas de aceptación para la historia de usuario 5

- Identificar todos los posibles resultados observables de la historia:

    - Se tiene la interfaz para generar el stock del producto solicitado por el vendedor.

- Identificar los resultados que terminan la historia y los que permiten continuar dentro de la historia:

    - La historia termina cuando se llena el formulario de registro de stock del producto.

- Identificar los caminos de ejecución posibles:

    - El formulario pide registros de stock de los productos en el día.
    - Valida los datos.
    - La historia termina a petición del vendedor, cuando no se desea registrar mayor información.

- Asignar un conjunto de valores Validos y valores del entorno a cada camino de ejecución para obtener resultados esperados:

    - El conjunto de valores valiosos está dado por el conjunto de datos compuestos por la información que se introduce sobre los datos.

- Eliminación de caminos redundantes:

    - No existen caminos redundantes.

Pruebas de aceptación para la historia de usuario 6

- Identificar todos los posibles resultados observables de la historia:

    - Se tiene la interfaz para generar la cotización.

- Identificar los resultados que terminan la historia y los que permiten continuar dentro de la historia:

- La historia termina cuando se llena el formulario de registro de la cotización.

- Identificar los caminos de ejecución posibles:

  - El formulario pide el registro de los productos para generar la cotización.
  - Valida los datos.
  - La historia termina a petición del vendedor, cuando no se desea registrar mayor información.

- Asignar un conjunto de valores Validos y valores del entorno a cada camino de ejecución para obtener resultados esperados:

  - El conjunto de valores valiosos está dado por el conjunto de datos compuestos por la información que se introduce sobre los datos.

- Eliminación de caminos redundantes:

  - Non existen caminos redundantes.

Pruebas de aceptación para la historia de usuario 7

- Identificar todos los posibles resultados observables de la historia:

  - Se tiene la interfaz para generar los reportes de los productos en general.

- Identificar los resultados que terminan la historia y los que permiten continuar dentro de la historia:

  - La historia termina cuando imprime el reporte.

- Identificar los caminos de ejecución posibles:

- El formulario pide la descripción de los datos almacenados.
- Valida los datos.
- La historia termina a petición del administrador, cuando no se desea registrar mayor información.

- Asignar un conjunto de valores Validos y valores del entorno a cada camino de ejecución para obtener resultados esperados:

    - El conjunto de valores valiosos está dado por el conjunto de datos compuestos por la información que se introdujo en el proceso.

- Eliminación de caminos redundantes:

    - No existen caminos redundantes.

3.8 MANTENIMIENTO

Mientras la primera versión se encuentra en producción, el proyecto XP mantiene el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para esto se realizan tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. Se puede requerir además de nuevo personal dentro del equipo y cambios en su estructura.

Como parte del proceso de perfeccionamiento de las funcionalidades del sistema y con el objetivo de satisfacer las solicitudes e inconformidades del usuario una vez que fueron entregadas cada iteración se realizaron mantenimientos al sistema.

CAPITULO IV

METRICAS DE CALIDAD

4.1 INTRODUCCIÓN

En este capítulo se determina la calidad del sistema, la gestión de riesgos y los procedimientos de seguridad.

La calidad es uno de los aspectos más importantes dentro del desarrollo de software, la medición se realizará mediante métricas que la ISO 9126 hace referencia y que fueron detalladas en el Capítulo II.

El objetivo no es alcanzar una calidad perfecta, sino la necesaria y suficiente para cada contexto de uso a la hora de la entrega a los usuarios. Se detallará la gestión de riesgos, los procedimientos más imprescindibles de seguridad que deberían tener en el sistema.

4.2 FUNCIONALIDAD

La funcionalidad mide el grado en el que el software satisface las necesidades indicadas por la adecuación, exactitud, interoperabilidad y seguridad de acceso [Pressman, 05].

Es por tanto que la eficacia en la eliminación de defectos (EED). Dentro del ámbito de Desarrollo de Sistema Virtual, la EED se define de la siguiente manera.

$$EED = E / (E + D)$$

Dónde:

E = Número de errores antes de la entrega del software al usuario final.

D = Número de errores encontrados después de la entrega al usuario final.

Si el número de defectos presentados después de la entrega al usuario final es 0, entonces el valor de EED es 1. La correspondiente ponderación la presentaremos en la siguiente tabla:

Nivel de Calidad	Valor EED
Primer nivel	Entre 1.15 y 1.10
Segundo nivel	Entre 1.10 y 1.05
Tercer nivel	Entre 1.05 y 1.00

Tabla 4.1 Nivel de calidad según valor del EED

Fuente: [Pressman, 05]

Reemplazando tenemos:

- Realizando pruebas al sistema antes de la entrega detectamos 40 errores: E = 40
- Realizando pruebas después de la entrega al usuario se encontraron 8 fallas: D = 8

$$EED = \frac{40}{40 + 8} \equiv 90\%$$

Si observamos los resultados de la EED, tenemos un 0.90, comparando con la ponderación de la tabla 4.1 tenemos que nuestro valor no se encuentra dentro de los tres niveles de calidad formulados, pero es necesario hacer notar que estamos cerca al tercer nivel.

4.3 CONFIABILIDAD

La confiabilidad del software, es la probabilidad de que el sistema opere sin fallas bajo determinación condicional para un intervalo de tiempo dado.

Los usuarios quieren que el software que van a utilizar tenga un funcionamiento correcto y los desarrolladores esperen que el número de defectos que existan puedan der solucionados sin causar problemas a medida que lo van probando, esto hace que la confiabilidad en el software y en los tiempos entre fallas se hace cada vez más grandes [Pressman, 05].

Se examinará la confiabilidad, que refleja la incertidumbre sobre la utilización del sistema. Por lo tanto, en cualquier instante el tiempo hasta la próxima falla es incierto y se lo puede considerar como una variable aleatoria.

Para estudiar esa variable aleatoria estudiaremos una distribución exponencial encausada en sus pruebas con cero fallas, que derivan de una función de cifras de falla, se supone que el número de fallas en el instante t es igual a:

$$Fallas = a * e^{-b(t)}$$; donde a, b son constantes

Por lo tanto el modelo requiere tres entradas: el número proyectado promedio de fallas como objetivo (fallas), el número total de fallas observadas en las pruebas (fallas probadas) y el número total de horas de ejecución de pruebas hasta la última falla (horas hasta la última falla).

El cálculo de horas necesarias de prueba para cero fallas es:

$$\frac{Ln[(fallas)/(0.5 + fallas)] * (horas\ hasta\ ultima\ falla)}{\text{Ln } [(0.5 + \text{fallas})/(\text{fallas probadas} + \text{fallas})]}$$

En el sistema se probaron aproximadamente 10.000 líneas de código del programa y han ocurrido 40 fallas sobre un total de tiempo de prueba de 200 horas. La última falla ocurrió en la hora 180. Durante las últimas 20 horas no han ocurrido fallas. La tabla 4.2 presenta las fallas ocurridas por cada mil líneas de código.

Fallas	**Fallas por cada mil líneas de código**
12	12/1000 = 0.012
8	8/1000 = 0.008
6	6/1000 = 0.006
4	4/1000 = 0.004

Tabla 4.2 Fallas de línea de código

Fuente: [Elaboración propia]

Basados en la información, el promedio máximo de fallas por cada mil líneas de código es:

$$(0.012 + 0.008 + 0.006 + 0.004)/4 = 0.0075$$

El número proyectado promedio de fallas es:

$$(\text{Promedio máximo de falla} * \text{líneas de código})/100 = 0.0625$$

El número de horas de prueba necesarias para alcanzar la confiabilidad del sistema.

$$\frac{Ln[(0.0625)/(0.5 + 0.0625)] * (180)}{\mathrm{L}n\,[(0.5 + 0.0625)/(40 + 0.0625)]} = 92.71$$

Por lo tanto el sistema es confiable en un 93%.

4.4 USABILIDAD

Grado en el que el software es fácil de usar. Viene reflejado por la facilidad de comprensión, facilidad de aprendizaje y operatividad [Pressman, 05].

Es decir determinar esta medida, pero se hallaron tres métricas que servirán para decidir cuan usable es el sistema SAC. Adicionalmente también se detallaran dos criterios (entrenamiento, operatividad) para usabilidad del sistema. Primero empezaremos por las tres métricas estas son:

La completitud de la descripción

Viene dada por la siguiente formula:

$$X = A / B$$

Dónde:

A es el número de Funciones o tipo de Funciones descritas en la descripción del producto.

B es el número total de Funciones.

Remplazando tenemos:

$$X = 9 / 15$$
$$X = 0.6$$

Por tanto existe un 60 % de entendimiento por parte de los usuarios con respecto a la capacidad del producto.

Consistencia operacional

Viene dada por la siguiente formula:

$$X = 1 – A / B$$

Dónde:

A es el número de instancias de operaciones con comportamiento inconsistente.

B el número total de operaciones

Remplazando tenemos:

$$X = 1 – 4 / 20$$
$$X = 0.8$$

Por tanto existe un 80% del sistema que no tiene instancias de operaciones con comportamiento inconsistente.

Consistencia operacional en el uso

Viene dada por la siguiente formula:

$$X = 1 – A / B$$

Dónde:

A es el número de funciones que el usuario encontró inaceptable inconsistente según sus expectativas.

B es el número de funciones usadas por el usuario durante el periodo de prueba.

Remplazando tenemos:

$$X = 1 – 2 / 14$$
$$X = 0.8$$

Por tanto el usuario encuentra un 20% del sistema inaceptable en el periodo de prueba.

También se tiene la siguiente formula:

$$X = A / OUT$$

Dónde:

A es el número de funciones que el usuario encontró inaceptable inconsistente según sus expectativas.

OUT es el tiempo de operación del usuario durante el periodo de observación.

Luego de haber resuelto la inconsistencia del sistema se ve que el usuario se encuentra satisfecho por la consistencia operacional del uso del sistema. Podemos concluir que el resultado obtenido en el factor de calidad de usabilidad en el sistema SAC es bueno.

4.5 MANTENIMIENTO

El estándar IEEE sugiere que el índice de madurez del software (IMS) que proporciona una indicación de la estabilidad de un producto de software basada en los cambios que ocurren con cada versión del producto se determina con la siguiente formula:

$$IMS = [Mt – (Fa + Fc + Fd)] / Mt$$

Dónde:

Mt es el número de módulos de la versión actual.

Fc es el número de módulos en la versión actual que se han cambiado.

Fa es el número de módulos en la versión actual que se han añadido.

Fd es el número de módulos en la versión que se han borrado en la versión actual.

A medida que el IMS se aproxima a 1, el producto se empieza a estabilizar [Pressman, 05].

Esta fase de mantenimiento se centra en los cambios que va a sufrir el sistema a lo largo de su vida útil, estos cambios pueden deberse a la corrección de errores, a cambios en el entorno inmediato del sistema o a cambios en los requisitos del usuario, dirigidos normalmente a ampliar el sistema. Para calcular el IMS se establecerá los cambios que ocurren con cada versión del producto y se calculará para los mismos.

Estos datos están detallados en la tabla 4.4

Versión del sistema	**Mt**	**Fa**	**Fd**	**Fc**	**IMS**
Versión 1.1	4	0	0	1	0.75
Versión 1.2	4	0	1	0	0.75
Versión 1.3	5	0	0	0	1
Versión 1.4	5	0	0	0	1

Tabla 4.3 Métricas IMS (Índice de madurez del software)

Fuente: [Elaboración propia]

4.6 PORTABILIDAD

La portabilidad es la factibilidad con que el software puede ser llevado de un entorno a otro, con una facilidad de instalación, facilidad de ajuste y facilidad de adaptación al cambio.

El grado de portabilidad está dado por:

$$GP = 1 – [CT / CRD]$$

Dónde:

GP es el grado de portabilidad.

CT es el costo de portabilidad.

CRD es el costo de re-desarrollo

Si GP = 1, la portabilidad es perfecta.

Si GP < 0, el re-desarrollo es más rentable que la portabilidad.

Remplazando tenemos:

GP = 1 – [10$ / 1500]

GP = 0.99

Por tanto se concluye que el sistema es portable un 99%.

Definición del árbol de requisitos de calidad	
1. Usabilidad	Comprensibilidad global
	Facilidad de aprendizaje
	Aspectos de interfaz y estilo
2. Funcionalidad	Facilidad de navegación
	Aspectos de búsqueda
3. Confiabilidad	Errores de enlace
	Errores o deficiencias varias
4. Portabilidad	Instalabilidad
	Independencia de hardware y software
5. Mantenimiento	Estabilidad
	Expandibilidad

Tabla 4.4 Definición del árbol de requisitos de calidad

Fuente: [Elaboración propia]

Característica	%
1. Usabilidad	80
2. Funcionalidad	90
3. Confiabilidad	93
4. Portabilidad	99
5. Mantenimiento	99

Tabla 4.5 Resultados obtenidos

Fuente: [Elaboración propia]

4.7 ANALISIS DE COSTOS

4.7.1 Costo de software desarrollado

Para el análisis de costo se hace uso del modelo COCOMO intermedio. Se considera un tipo de proyecto orgánico.

- Cálculo del esfuerzo de desarrollo del software en función de la estimación de las líneas de código fuente.

$$E = aKLDC^{b} * FAE$$

Dónde:

E es el esfuerzo aplicado en personas – mes.

KLDC es el número estipulado de líneas de código en miles.

a, b son los valores mostrados en la tabla 4.6

FAE es el factor de ajuste del esfuerzo que depende de 15 atributos.

MODO	A	b
Orgánico	3.20	1.05
Semiacoplado	3.00	1.12
Empotrado	2.80	1.20

Tabla 4.6 Constantes para el modelo COCOMO intermedia

Fuente: [Elaboración propia]

De acuerdo a la evaluación de los factores de ajuste de los atributos se tiene el siguiente resultado visto en la tabla 4.7

FAE = 1.15 *1.08*1*1*1*1*1*1*1*1*1*1*0.91*0.91*1.04 = 1,070

$$E = 3.20(10)^{1.05} * (1.070) = 38.42$$

- Para el cálculo del tiempo desarrollo del proyecto se hace uso de:

$$D = cE^{d}$$

Dónde: c = 2.5 y d = 0.38 para proyectos orgánicos

$$D = 2.5(38.42)^{0.38} = 10$$

- Entonces el número de personas necesarias para realizar el proyecto es:

$$Np = E/\mathrm{D}$$

$Np = \frac{38.42}{10} = 3.8 = 4$; este caso es 1

- El costo total del proyecto:

$$Np * Salario\ medio\ de\ programadores\ y\ analistas$$

Cp = 1 (300) = 300 $us

Atributos	Valor					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extra alto
Atributos de software						
Fiabilidad	0.75	0.88	1.00	1.15	1.40	
Tamaño de la base de datos		0.94	1.00	1.18	1.16	
Complejidad	0.70	0.85	1.00	1.15	1.30	1.65
Atributos de hardware						
Restricciones de tiempo de ejecución			1.00	1.11	1.30	1.66
Restricciones de memoria virtual			1.00	1.06	1.21	1.56
Volatilidad de la máquina virtual		0.87	1.00	1.15	1.30	
Tiempo de espera		0.87	1.00	1.07	1.15	
Atributos de personal						
Capacidad de análisis	1.46	1.19	1.00	0.86	0.71	
Experiencia en la aplicación	1.29	1.13	1.00	0.91	0.82	
Calidad de los programadores	1.42	1.17	1.00	0.86	0.70	
Experiencia en la máquina virtual	1.21	1.10	1.00	0.90		
Experiencia en el lenguaje	1.14	1.07	1.00	0.95		
Atributos del proyecto						
Técnicas actualizadas de programación	1.24	1.10	1.00	0.91	0.82	
Utilización de herramientas de software	1.24	1.10	1.00	0.91	0.83	
Restricciones de tiempo de desarrollo	1.23	1.08	1.00	1.04	1.10	

Tabla 4.7 Factores de ajuste del esfuerzo

Fuente: [Elaboración propia]

4.7.2 Costo de hardware y software

Descripción de costos	Total $us	Observaciones
Costo de hardware		
Servidor	1.000	Aplicación intranet
Equipo de red	0	Cuenta cable utp, hub, cortapico
Subtotal	1.000	
Costo de software	0	Ya dispone de licencia
Costo implantación		
Costo de instalación	10	Disco de instalación
Costo de capacitación	0	Posee de un manual de funciones
Subtotal	10	
TOTAL	1010	

Tabla 4.8 Costo de hardware y software

Fuente: [Elaboración propia]

Por lo tanto el costo total es de 1310 $us que incluye el costo de software desarrollado, el costo de hardware y el costo de implantación.

Descripción de costos	Total $us
Costo total de software desarrollado	300
Costo de software y hardware	1010
Costo de implantación	0
TOTAL	**1310**

Tabla 4.9 Costo de hardware y software

4.8 ANÁLISIS DE BENEFICIOS

Los beneficios del sistema son de tipo intangible, entre los cuales se pueden distinguir beneficios de tipo estratégico.

Beneficios	Descripción
Optimización de los procesos de registro.	Se optimiza y mejoró el registro de ingreso y salida de productos.
Reducción de errores en los registros de los productos.	Se redujo los errores con respecto al registro de compra y venta de los productos.
Reducción de tiempo en la búsqueda de stock de los productos.	El tiempo de búsqueda se mejoró obteniendo en pocos segundos el producto requerido.
Optimización del proceso de registro del vendedor y proveedor.	El tiempo y llenado de los datos personales es más corto.
Optimización del proceso de cotización	El tiempo en realizar una cotización es más efectiva y rápida a la hora que el cliente lo solicita.
Centralización de información de almacenes y cotizaciones.	Permite al administrador ver los reportes y datos en cualquier momento.

Tabla 4.10 Beneficios del sistema

4.9 SEGURIDAD

4.9.1 Seguridad a nivel de base de datos

Se hace uso del manejador de base de datos MySQL que proporciona estabilidad, confiabilidad y alto rendimiento ya que no existen reportes de caídas en varios años de operación, además brinda extensiones para distintas funcionalidades como ser encriptación de datos.

MySQL garantiza la integridad y consistencia de la base de datos mediante procedimientos almacenados. Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

4.9.2 Seguridad a nivel de aplicación

Tomando en cuenta las recomendaciones especificadas en la norma ISO17799 con respecto a la presentación de las características de confidencialidad, integridad y disponibilidad de la información se ha incorporado las siguientes medidas de seguridad en el sistema de administración.

Recomendaciones ISO17799	Medidas de seguridad incorporadas en el sistema de almacenes y cotizaciones.
Proteger la confidencialidad, autenticidad o integridad de la información mediante sistemas y técnicas criptográficas.	Se implementaron niveles de roles con permisos de acceso a la información, encriptación de la contraseña de los usuarios.
Realizar controles como la validación de datos de entrada.	Se controla valores fuera de rango, caracteres inválidos como campos nulos.
Se debe poder determinar las responsabilidades de todo el personal involucrado en el proceso de entrada de datos.	Se realiza el registro de cada usuario en cada proceso realizado por este.

Tabla 4.11 Seguridad a nivel de la aplicación

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

El análisis del funcionamiento del sistema de gestión de la Empresa de Seguridad Electrónica "For Security 4.5" permitió definir las características necesarias para la creación de una aplicación web acorde a las peticiones del usuario, todo esto a través de la metodología de ingeniería del software escogida. En este capítulo se detallan los resultados de cada una de las fases que XP propone.

Se construyó la aplicación web que introdujo una nueva vía para gestionar la información generada en la Empresa de Seguridad Electrónica For Security 4.5 que brindará al cliente conformidad y seguridad ante las funcionalidades del sistema.

La utilización de herramientas informáticas en la solución de problemas relacionados con la gestión de la información se ha hecho muy popular en la actualidad, lo que permite encontrar soluciones de alta calidad para este tipo de problemas.

Con la realización del presente trabajo se ha logrado:

- Caracterizar la situación existente en la Empresa de Seguridad Electrónica For Security 4.5, demostrándose así la necesidad de desarrollar un sistema que fuese capaz de gestionar almacén y cotizaciones.
- Implementar un "Sistema web de Almacenes y Cotizaciones para la Empresa de Seguridad Electrónica For Security 4.5, herramienta que permite el control de la compra-venta de productos, existencia de los mismos y atención al cliente, así como otros servicios con fines comerciales.

De esta forma se ha cumplido con los objetivos planteados en la presentación de esta investigación, pues se ha logrado de forma eficiente la implementación del sistema propuesto.

5.2. RECOMENDACIONES

Al mismo tiempo que se han cumplido los objetivos involucrados en el desarrollo del presente trabajo se realizan las siguientes recomendaciones:

- Realizar una investigación más profunda para determinar nuevas funcionalidades que se puedan agregar a la aplicación.
- Se recomienda que este trabajo sea usado como material de estudio en la realización de alguna aplicación similar.
- Definir las políticas de seguridad para la implantación del sistema web de almacenes y cotizaciones la Empresa de Seguridad Electrónica For Security 4.5.
- Se recomienda la aplicación del Sistema web de almacenes y cotizaciones de Ventas en otras pequeñas empresas con características similares la Empresa de Seguridad Electrónica For Security 4.5.

BIBLIOGRAFIA

[Pressman, 05] Roger S. Pressman R., 2005: Ingeniería de Software: Un enfoque Práctico, Quinta Edición, Editorial Mcgra-Hill, Mexico.

[Chin, 04] Chin, Gary, 2004, Agile Project Management: How to Succed in the Face of Changing Project Requirements. Amacom.

[Aguilar Ramos, 05] Aguilar Ramos, 2005: Aplicación de Conceptos de Gestión de Proyectos de Riesgo en el Desarrollo de Productos Nuevos en el Campo de Tecnología de información, Universidad de Puerto Rico.

[Beck, 01] Beck Kent, 2001: Extreme Programming Explained Embrace Change

[Jacobson, 00] Jacobson P., 2000: El Proceso Unificado de Desarrollo de Software, Madrid, España: Pearson Education.

[Hans, 02] Hans V., 2002: Principios y Prácticas de la Ingeniería de Software, (Tercera Edición), USA.

[Knoow.Net, 08] Knoow.Net, 2008: Ciencias Económicas y Comerciales.

[Laurence Pleeger, 02] Laurence Pleeger, 2002: Ingeniería de Software. Teoría y Práctica, (Primera Edición), Argentina: Pearson Education.

[ISO9126, 00] Aspectos de la Ingeniería de ISO/IEC 9126 y 14598, 2000, Un punto de vista Brasileño. El segundo congreso sobre calidad del Software, Yokohama, Japón.

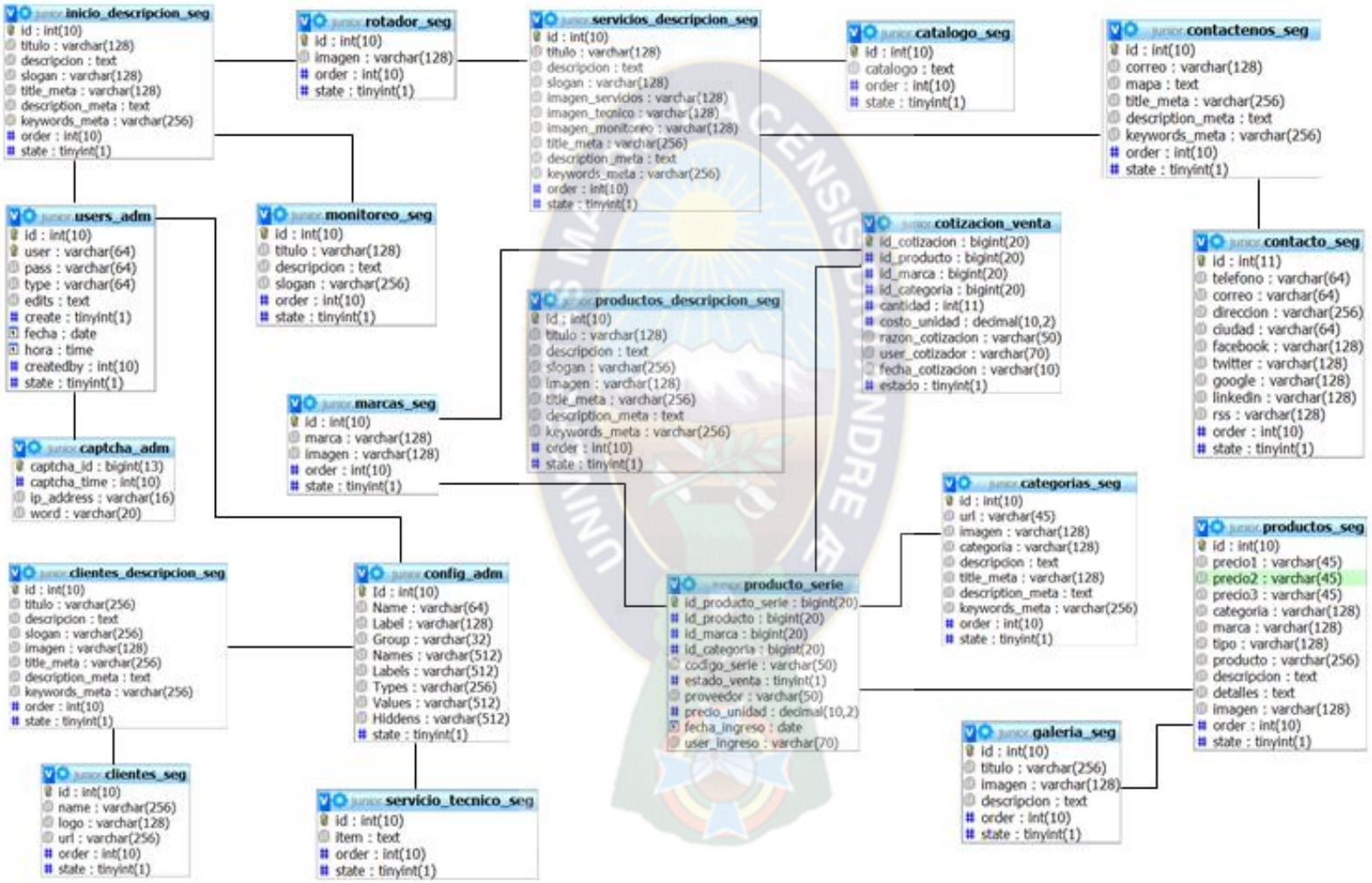
[Lewis, 94] Lewis G., 1994: What is Software Ingineering.

[Sommerville, 02] Sommerville I., 2002: Ingeniería de Software, (Sexta Edición), Pearson Education.

[Fernández, 02] Fernández D., 2002: Definición de una Arquitectura de Software para el Diseño de Aplicaciones Web, Universidad de Oviedo, Italia.

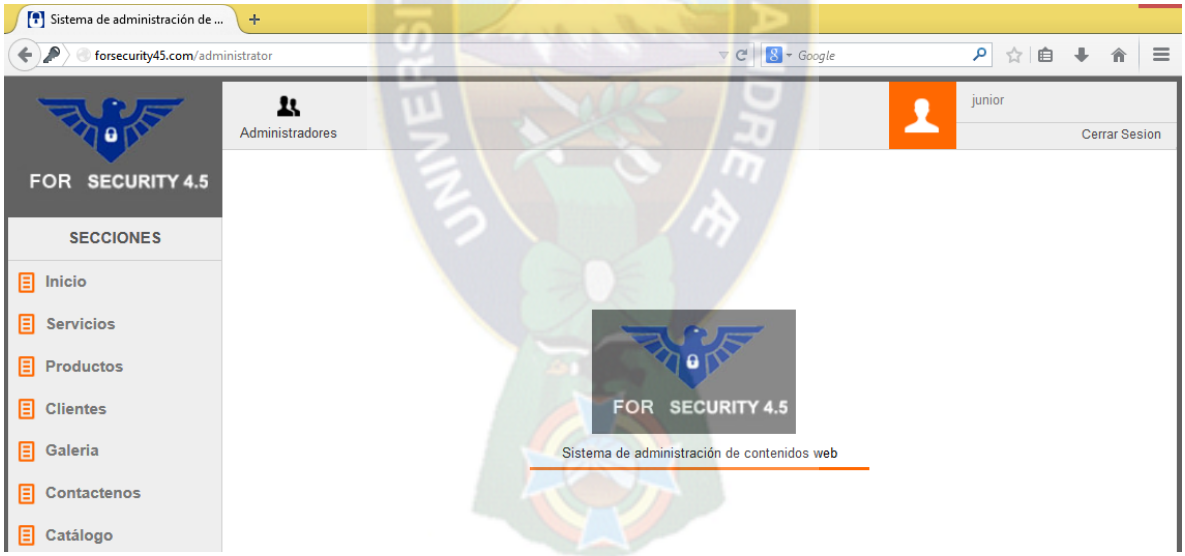
[Delgado, 08] Delgado A., 2008: Metodologías de Desarrollo para Aplicaciones con enfoque SOA, Uruguay.

MODELO FISICO DE LA BASE DE DATOS



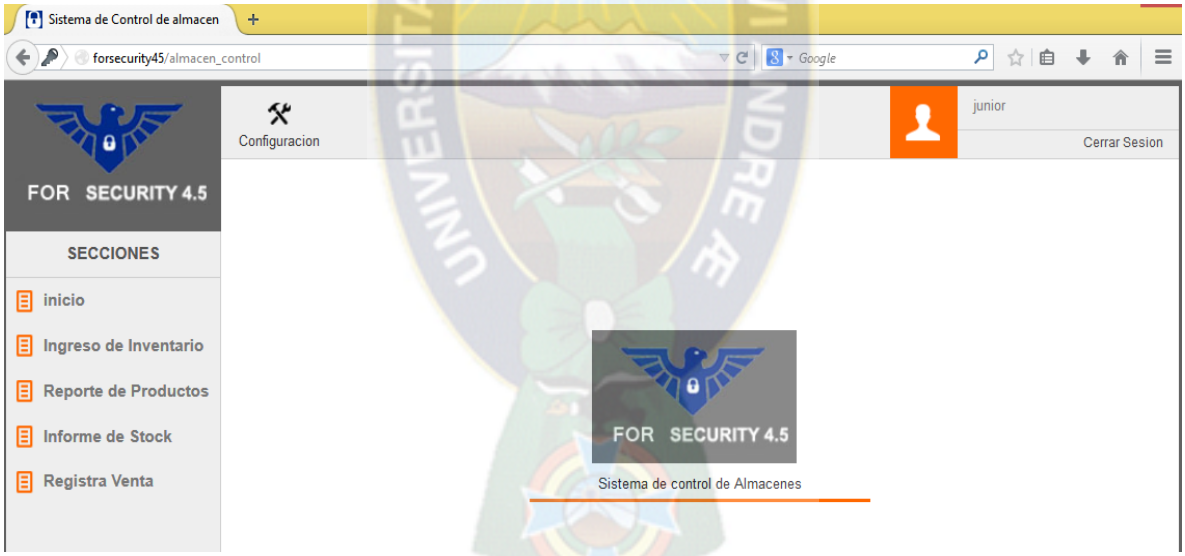
CAPTURA DE PANTALLAS

INGRESO AL SISTEMA DE ADMINISTRACION

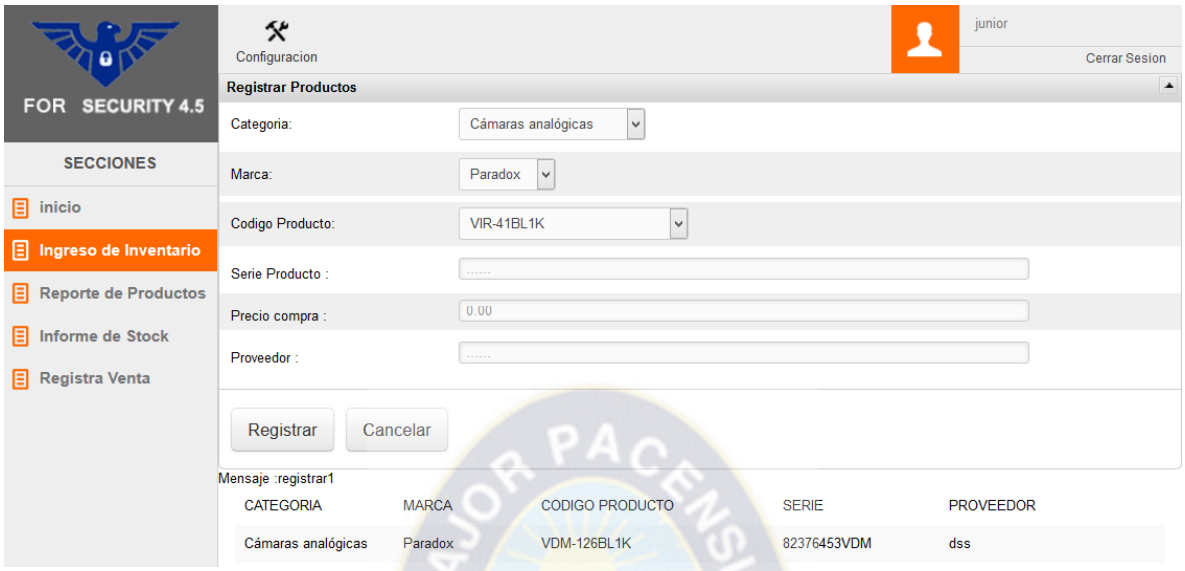


PANEL DE ADMINISTRACIÓN PAGINA WEB

PORTAL PAGINA WEB



MÓDULO ALMACEN



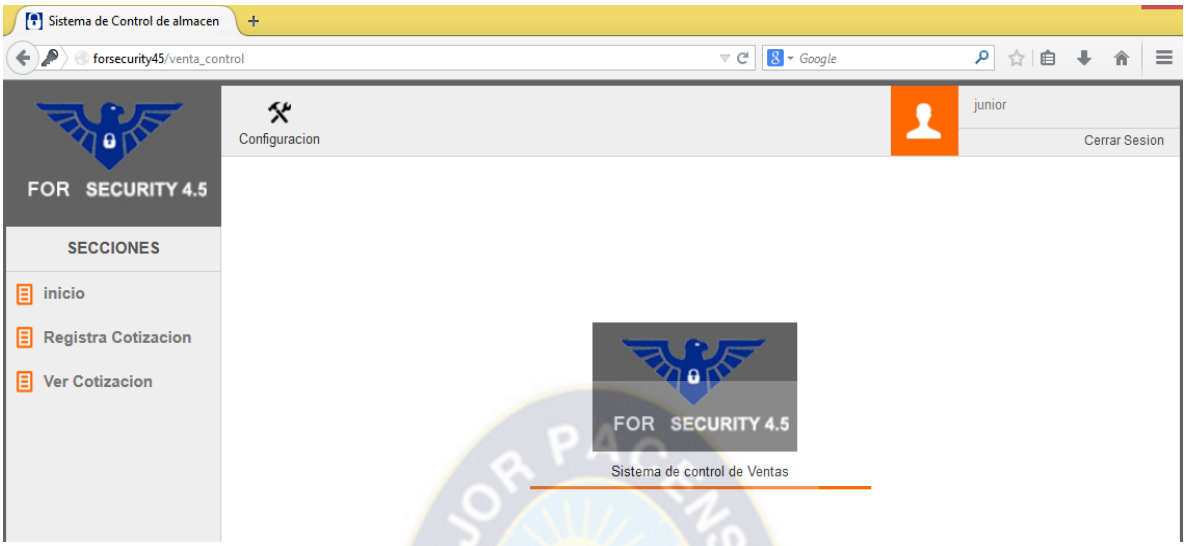
REGISTRO DEL INGRESO DE PRODUCTOS

INFORME DE STOCK DEL PRODUCTO SOLICITADO

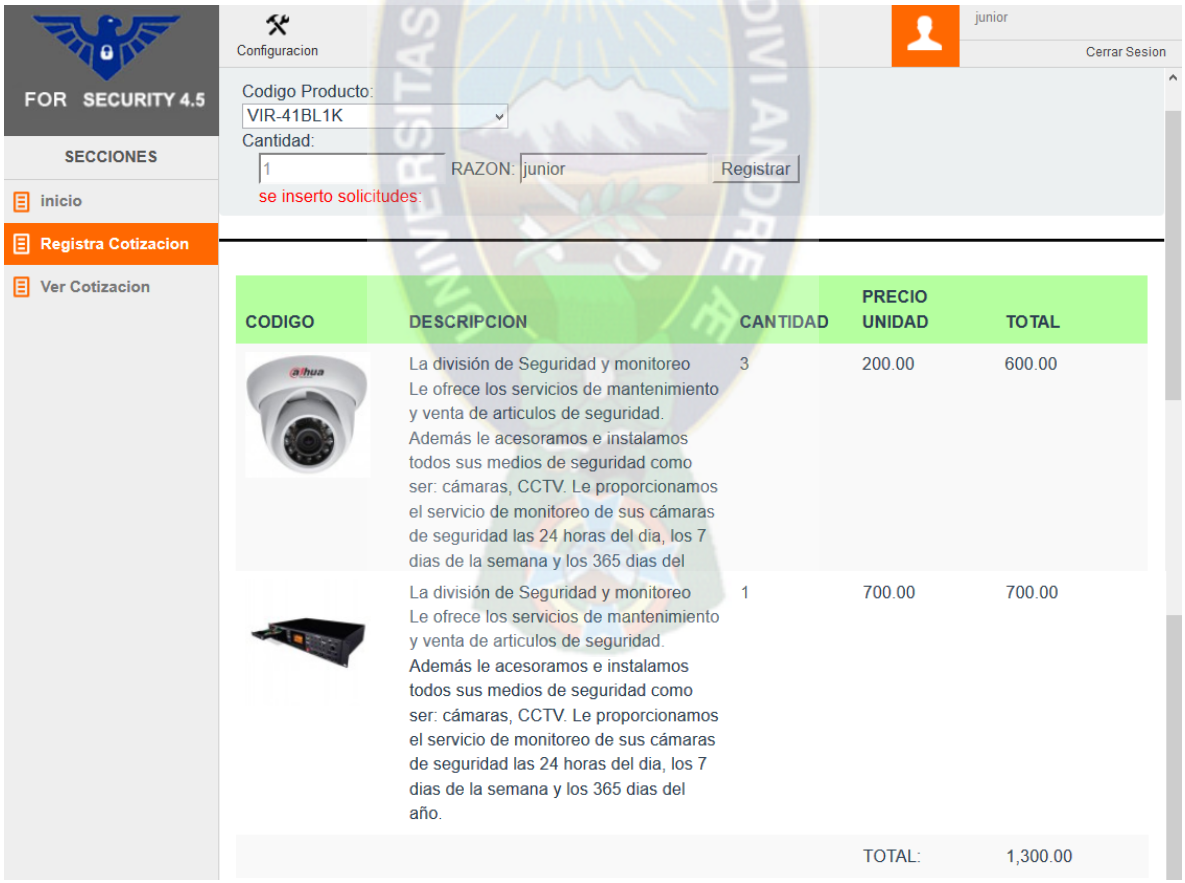
REGISTRO DE VENTA DEL PRODUCTO

CATEGORIA	MARCA	CODIGO PRODUCTO	CANTIDAD	ESTADO
Cámaras analógicas	Paradox	VIR-41BL1K	4	EXISTENTE
Cámaras analógicas	MikroTik	VIR-233BL1K	0	no EXISTENTE
Grabadores Analógicos	Paradox	VDM-126BL1K	1	EXISTENTE
Grabadores Digitales	Ubiquiti	VDM-406BL1K	1	EXISTENTE
Cámaras analógicas	MikroTik	CA-FW191JN-IR-0360B	0	no EXISTENTE
Grabadores Digitales	Paradox	DH-CA-DW191EN-IR-0280B	0	no EXISTENTE
Grabadores Analógicos	Paradox	CA-FW181GN-IR-0280B	0	no EXISTENTE
Grabadores Analógicos	MikroTik	CA-DW181HN-IR-0280B	1	EXISTENTE
Grabadores Analógicos	MikroTik	CA-DW181HN-IR-0280B	1	EXISTENTE
Grabadores Analógicos	Paradox	VIR-41BL1K1	0	no EXISTENTE
Grabadores Analógicos	MikroTik	VIR-233BL1K2	0	no EXISTENTE
Grabadores Analógicos	Paradox	VDM-126BL1K3	0	no EXISTENTE
Grabadores Analógicos	Ubiquiti	VDM-406BL1K4	1	EXISTENTE
Grabadores Analógicos	MikroTik	CA-FW191JN-IR-0360B5	0	no EXISTENTE
Grabadores Analógicos	Paradox	DH-CA-DW191EN-IR-0280B6	0	no EXISTENTE
Grabadores Analógicos	Paradox	CA-FW181GN-IR-0280B7	1	EXISTENTE
Grabadores Analógicos	MikroTik	CA-DW181HN-IR-0280B8	0	no EXISTENTE
Grabadores Analógicos	MikroTik	CA-DW181HN-IR-0280B9	0	no EXISTENTE
Grabadores Analógicos	Paradox	VIR-41BL1K11	0	no EXISTENTE

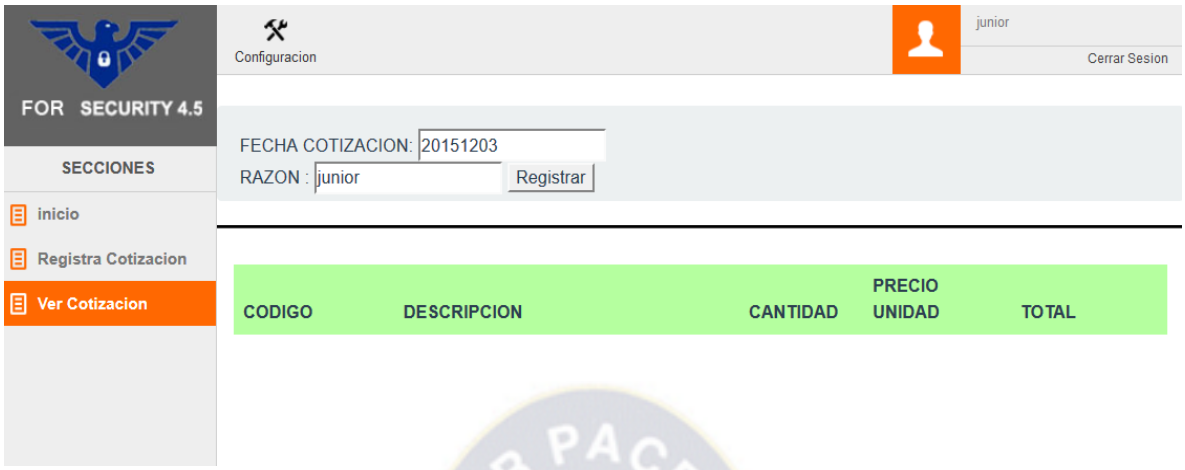
REPORTE DE PRODUCTOS



MÓDULO COTIZACIONES



GENERA COTIZACIÓN DE PRODUCTOS



REPORTE DE COTIZACIONES