SILC, Sistema de Información Lavandería Cristal

Andrés González López

Trabajo de grado para optar por el título de ingeniero de sistemas

Asesor:

Diego Hernán Montoya Bedoya

Departamento de Ingeniería de Sistemas

Escuela de Ingeniería

Universidad EAFIT

Medellín

2010

CONTENIDO

LISTA DE ILUSTRACIONES

Lista de anexos

0 PROLOGO

0.1 INTRODUCCION

En la actualidad, la lavandería CRISTAL es una de las empresas líderes en el sector de lavado tanto a nivel domestico como empresarial en el área metropolitana. Uno de los grandes fuertes que tiene la empresa y por lo cual es distinguido es el servicio al cliente que prestan desde el momento en que se recoge las prendas a lavar hasta el momento que vuelven a su dueño.

Además del servicio al cliente, se hace necesario para la empresa la implementación de las Tecnologías de Información (TI) y de Sistemas de Información (SI), siempre basándose en las mejores prácticas para el control de la calidad de los servicios que esta área presta a la compañía, fundamentadas en el mejoramiento continuo y logrando que los servicios de TI soporten y mejoren la actividad empresarial. Sin embargo, uno de los aspectos en los cuales hay más falencias en la empresa es irónicamente el área en mención, pues esta empresa gasto una considerable suma de dinero en TI, que no proporciono el soporte esperado a los procesos de la organización e incluso llego a incurrir en gastos innecesarios para la misma.

Es por esto que se decidió por parte de la directiva de la empresa crear un sistema de información hecho a la medida que llenara las necesidades que surgieron al momento de evaluar el proceso administrativo y de facturación de la empresa. Claro está, teniendo en cuenta la inversión previa que había realizado esta en TI.

Dado el contexto tecnológico, económico, político, social y cultural de la empresa, es posible que algunas de estas prácticas no sean aplicables al sistema de

información. La principal intención de este proyecto de grado será examinar, evaluar, proponer y desarrollar una solución que satisfaga tanto la necesidad de la empresa y tenga en cuenta las limitantes con respecto a las TI de la empresa.

0.2 JUSTIFICACION

Los sistemas de información en el área de contabilidad son en la actualidad una herramienta que bien implementada se convierte en un arma competitiva en los negocios, así como las empresas buscan diferenciarse de su competencia, los sistemas de información (SI) son una manera de hacerlo. Se pueden mencionar dos sistemas de información:

- ✓ Los sistemas transaccionales,
- ✓ Los sistemas de apoyo a la toma de decisiones.

El contar con infraestructura adecuada que garantice agilidad o rapidez en el procesamiento de datos conlleva a tomar decisiones oportunas para resolver los retos que se presentan[1], en ocasiones, inesperadamente y la complejidad de los negocios provoca la necesidad de integrar funciones dentro de la empresa y su entorno dinámico. En tal entorno lo importante es conocer lo que sucede dentro de la empresa y fuera de ella. De manera particular, lo que sucede con su producto y su cliente.

El que una empresa cuente con sistemas de información implementados producirá algunas ventajas como:

---

[1] *ERP, Columna vertebral del negocio*. Publicación: El Economista - Suplemento Tecnología, fecha elaboración: 19/08/2004, consultado: 15/08/2010.

- ✓ Integración de la administración de la empresa, las finanzas, la administración de las tesorerías, los recursos humanos entre otras.
- ✓ Es posible analizar la cadena de valor ya sea de un producto o un servicio para identificar que actividades no están generando valor, poder eliminarlas o implementar otras para disminuir los costos.
- ✓ Ayudar a incrementar la efectividad en la operación de las empresas.
- ✓ Proporcionar ventajas competitivas, si es que la competencia no cuenta con esta tecnología.
- ✓ Disponibilidad de la información para todos los usuarios en tiempo real.
- ✓ Eliminar la barrera de la distancia ya que se puede trabajar con un mismo sistema en puntos distantes.

Los problemas de adaptación de los sistemas de información, entre los problemas más significativos de estos sistemas, están el costo, el tiempo de implementación, la resistencia al cambio de los usuarios, también los problemas técnicos como por ejemplo la rapidez de respuesta del sistema.

Por definición general de lo que es un almacén podemos decir que es aquel lugar donde se guardan los diferentes tipos de mercancía de la empresa.

La formulación de una política de inventario para un departamento de almacén depende de la información que brinda un sistema respecto a tiempos de adelantos, disponibilidades de materiales, tendencias en los precios y materiales de compras.

La función básica, pero no la más importante, de un sistema de información en un almacén es que controla físicamente y mantiene todos los artículos inventariados, debiendo establecer resguardo físicos adecuados para proteger los artículos de algún daño de uso innecesario debido a procedimientos de rotación de inventarios incorrectos y a robos. Los registros se deben mantener actualizados constantemente en el sistema, lo cual facilitan la localización inmediata de los artículos.

Algunas de las funciones del sistema de información es que mantiene en constante información al departamento de compras, sobre las existencias reales de materia prima al contar con un perfecto enlace informativo interdepartamental, Llevar en forma minuciosa controles sobre las materias primas (entradas y salidas), Vigilar que no se agoten los materiales conservando los niveles permisibles en cuanto a máximos y mínimos.

En cuanto a las ventajas de contar con un sistema de información se puede mencionar los siguientes puntos:

- ✓ Evitar el retraso en el abastecimiento de materiales.
- ✓ Controlar el abastecimiento parcial.
- ✓ Compra o producción en totales económicos.
- ✓ Rapidez y eficacia en atención a las necesidades propias.

Función de Recepción**:**

La función de recepción, ya sea de una unidad de la compañía o de un transportador común, es la misma. Si el material se recibe de cualquier otra fuente u otro departamento de la compañía, las actividades de construcción, el procedimiento será el mismo.

Importancia de que los diferentes departamentos estén intercomunicados a través de red:

La recepción adecuada de materiales y de otros artículos es de vital importancia, ya que una gran parte de las empresas tienen como resultado de su experiencia centralizada la recepción total bajo un departamento único (el departamento bajo la responsabilidad es el departamento de compras), las excepciones principales son aquellas grandes empresas con plantas múltiples. La recepción está estrechamente ligada a la compra, ya que probablemente el 70% de los casos. Se observa pues que tanto el departamento de compras, almacén y recepción deben

hablar o manejar un mismo paquete o sistema para evitar conflictos en cuanto al intercambio de información.

Proceso: Algunos datos que se deben ingresar al sistema de información al recibir un envió: Se le someterá a verificación para comprobar si está en orden y en buenas condiciones, si el recipiente está dañado o no se recibió el número de paquetes requeridos realizando las anotaciones pertinentes. Se debe hacer la salvedad correspondiente inmediatamente y no se podrá dar recibo de conformidad por el envió, esto es esencial sin tomar en cuenta si el transporte es aéreo, marítimo o terrestre, como se podría exigir para dar fuerza a cualquier reclamo resultante sobre envíos ocultos. El material que recibe una instalación de la compañía también debe ser sometido a una inspección preliminar, antes de introducirles en el área de almacenamiento, en el caso de que en la inspección inicial se detecte materiales de calidad inferior o en malas condiciones se le debe rechazar.

Técnicas de Almacenamiento de Materiales: El almacenamiento de materiales depende de la dimensión y características de los materiales. Estos pueden exigir una simple estantería hasta sistemas complicados, que involucran grandes inversiones y complejas tecnologías. La elección del sistema de almacenamiento de materiales depende de los siguientes factores:

- ✓ Espacio disponible para el almacenamiento de los materiales.
- ✓ Tipos de materiales que serán almacenados.
- ✓ Número de artículos guardados.
- ✓ Velocidad de atención necesaria.
- ✓ Tipo de embalaje.
- ✓ El sistema de información debe nutrirse con información específica de las entradas como:
    - Datos generales del proveedor: Nombre, dirección, teléfono, RFC, representante o contacto, descuentos, historial de compras, número

de proveedor. Tiempos de entrega, calidad y costos del producto. Garantías y beneficios

- Datos generales de los clientes: Nombre, dirección, teléfono, RFC, representante o contacto, descuentos, historial de ventas, número de cliente.
- Políticas de compra: Limites, tipos de compra (crédito ó contado) y formas de pago (enganches, plazos fijos, variables o diferidos), descuentos y promociones.
- Políticas de venta: Límite de crédito, tipos de compra (crédito ó contado) y formas de pago (enganches, plazos fijos, variables o diferidos), descuentos y promociones.
- Mercancías: Origen de la mercancía (proveedor, nacional o extranjero), series, lote, tipo de empaque o embalaje, descripción del producto, observaciones (daños, mercancía incompleta o equivocada), número de piezas, código.

Los beneficios de contar con un sistema de información adecuado en el área de almacén otorga innumerables beneficios al arrojar una gran cantidad de información después de procesar los datos que se le insertan previamente a tal sistema[2], así mismo existen sistemas "inteligentes" que pueden avisarnos cronológicamente cuando se deben realizar o gestionar algunas acciones necesarias. La información que podemos obtener en el sistema sería entre otros ítems:

- ✓ Cálculo del saldo de un cliente.

---

[2] *Importancia de los sistemas de información en las áreas administrativas*, de la pagina: http://www.monografias.com/trabajos27/importancia-sistemas/importancia-sistemas.shtml, consultado: 13/08/2010

- ✓ Control de inventarios, hacer pedido si es necesario, hacer promociones de mercancía o enviarla a otra tienda
- ✓ Inicios y fines de temporadas por mercancía
- ✓ Control de venta de productos.
- ✓ Necesidades de compra por proveedor.
- ✓ Control de entradas, salidas y localización de la mercancía, requisición de mercancías.
- ✓ Pagos realizados a los proveedores y pagos realizados por los clientes (monto y fecha de pago).
- ✓ Movimientos del mes (pagos, depuraciones).
- ✓ Catálogo de clientes.
- ✓ Facturación.

Salidas:

- ✓ Reporte de pagos.
- ✓ Estados de cuenta.
- ✓ Disponibilidad de materiales o productos
- ✓ Tendencias en los precios
- ✓ Cantidad de clientes
- ✓ Tipo de clientes
- ✓ Cantidad de proveedores
- ✓ Tipo de proveedores
- ✓ Reporte de facturación
- ✓ Reporte de compras de mercancía o insumos
- ✓ Estados de cuenta
- ✓ Cobros o pagos realizados.

Resultados:

- ✓ Conocer los tiempos de entrega de mercancía al cliente (desde el pedido hasta la entrega)
- ✓ Historial crediticio de los clientes
- ✓ Historial crediticio de la empresa
- ✓ Seleccionar los mejores clientes
- ✓ Compras realizadas por cada cliente (producto mas vendido, frecuencia de la compra)
- ✓ Hacer promociones especiales (volumen de compra, por preferirnos, por pago anticipado, pago puntual, antigüedad de cliente, cliente frecuente)
- ✓ Detección de clientes morosos y canalización al área correspondiente.
- ✓ Mercancías más vendidas y rezagadas
- ✓ Análisis de tiempo de entrega de cada proveedor
- ✓ Análisis de precios y promociones del proveedor
- ✓ Rotación de productos
- ✓ Máximos y mínimos

Impacto: El resultado de implantar un sistema de información dentro de un almacén puede producir un incremento de las utilidades al tener un mejor control de los productos. Así mismo se considera que puede haber un impacto social al reducir la necesidad de empleados administrativos en el control de almacén.

0.3 OBJETIVOS

0.3.1 General

Definir y desarrollar un sistema de información basado en las buenas prácticas para el proceso de facturación e inventario de la Lavandería Cristal, implementando la metodología de programación extrema, que genere a su vez valor agregado para la organización.

0.3.2 Específicos

- ✓ Objetivo 1:
  Estudiar profundamente el proceso de facturación e inventario de la Lavandería Cristal para así mejorar el mismo e implementar una solución factible en un sistema de información sostenible.

- ✓ Objetivo 2:
  Estudiar la metodología de programación extrema para especificar en qué manera puede ser implementada durante el desarrollo de esta tesis.

- ✓ Objetivo 3:
  Estudiar cual sería la mejor tecnología que satisfaga la necesidad que presenta la Lavandería Cristal para el problema que ellos plantean teniendo en cuenta que la limitante de inversión y utilización de los equipos que ya posee la empresa.

- ✓ Objetivo 4:
  Desarrollar un prototipo que satisfaga las necesidades del cliente este con el fin de esclarecer cuales de las necesidades son factibles de solucionar con la implementación de un sistema de información.

1 Procesos

1.1 Objetivo.

Estudiar profundamente el proceso de facturación e inventario de la Lavandería Cristal para así mejorar el mismo e implementar una solución factible en un sistema de información sostenible.

1.2 Proceso de facturación.

La descripción de este proceso parte de diagramas de flujos ya existentes que tenia el cliente con base a un estudio previo realizado en la empresa.

1.2.1 Procedimientos

1.2.1.1 Diagramas de flujo

Nivel de contexto

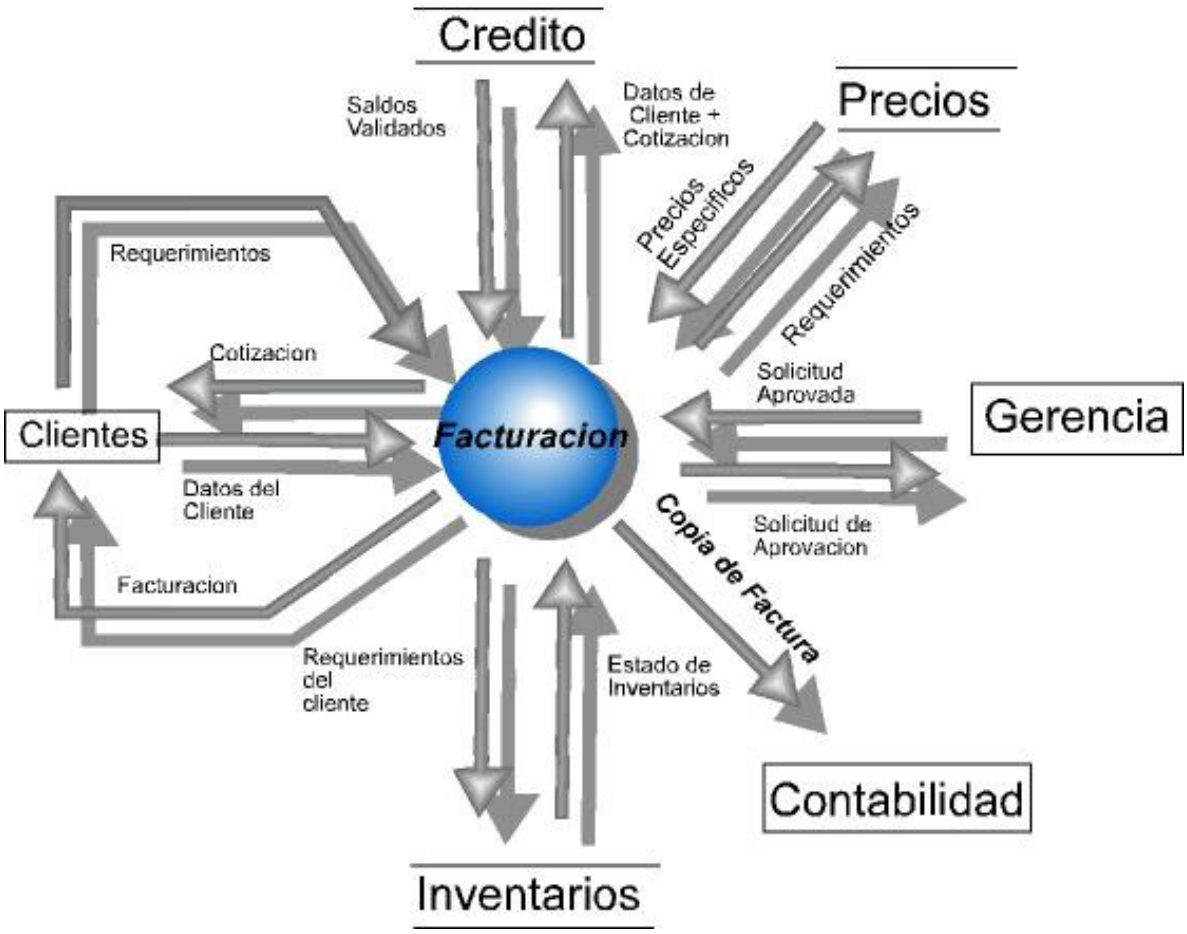

Ilustración 1 – Subnivel i del Proceso de Facturación

Ilustración 2 – Continuación del Subnivel i del Proceso de Facturación

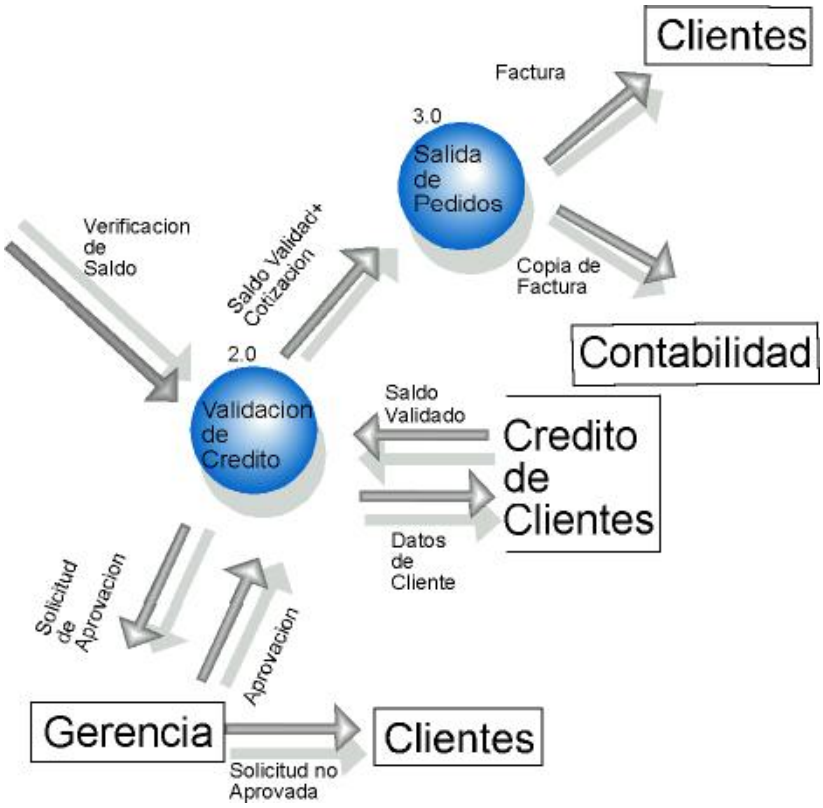

Ilustración 3 – Subnivel ii del Proceso de Facturación

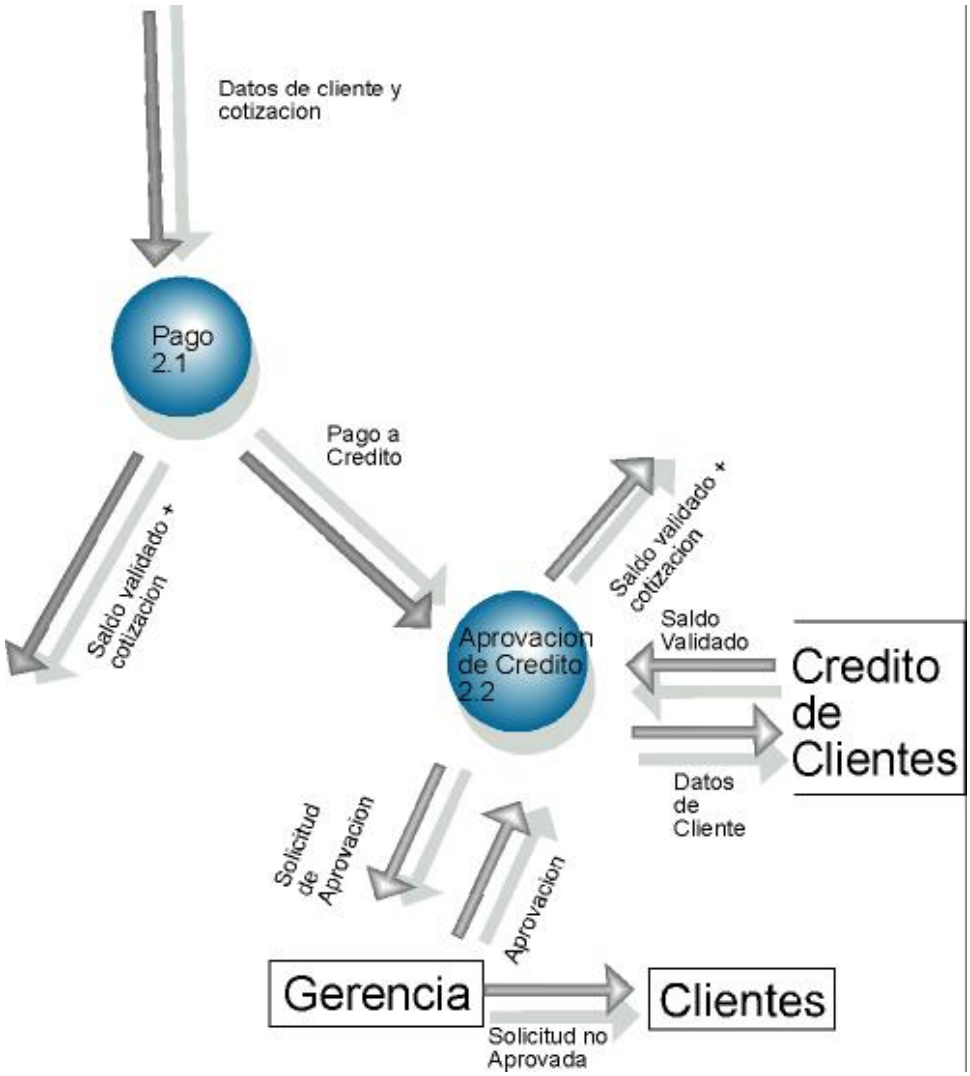

Ilustración 4 - Subnivel iii del Proceso de Facturación

1.2.1.2 Documentación de los diagramas de flujo

Requerimientos de Cliente
Descripción El cliente proporciona los requerimientos del producto que necesita.
Procesos-Entidades El cliente Realiza una Solicitud Para la compra de materiales que requiere a través de una solicitud.
Estructura de datos Los datos que el cliente proporciona son: características del producto, precios aceptables, Cantidad x Unidad y fecha de entrega.
Cotizaciones
Descripción Es un documento donde se especifica los precios actuales de los productos que el cliente requiere y esperar que el cliente los acepte.
Procesos-Entidades La cotización se da desde el sistema de y ventas hacia la entidad cliente.
Estructura de datos Los datos que contiene la cotización son: descripción del producto cotizado, cantidades, precios, fecha de cotización, fecha de entrega, datos del vendedor y cualquier otra información adicional necesaria.
Cotización aprobada

Descripción

Es la decisión que el cliente toma de acuerdo a la cotización que se le entrego, puede tener dos diferentes tipos que son: cotización aceptada, cotización rechazada.

Procesos-Entidades

Esta es la respuesta desde la entidad cliente al proceso de control y venta.

Estructura de datos

La estructura esta defina por cotización aceptada o rechazada.

Requerimientos de Precios

Descripción

Es un documento que se emite con relación a los precios especificados que el cliente requiere.

Procesos-Entidades

El documento es realizado por el departamento de ventas hacia la entidad cliente.

Estructura de datos

La estructura del documento es: fecha, nombre de cliente, precios de producto requerido, nombre de producto.

Precios específicos

Descripción

Es la información actual de los precios de los productos contra los precios que el cliente requiere.

Procesos-Entidades Es proporcionado por la entidad cliente hacia el departamento de ventas.
Estructura de datos La estructura del documento es: fecha, nombre de cliente, precios de producto confrontados (precios requeridos contra precios actuales), nombre de producto. Este se anexa a la cotización.
Requerimientos de Cliente(Existencias)
Descripción Comprende la verificación del producto que el cliente requiere.
Procesos-Entidades Este va desde persona que factura hacia almacenamiento producto.
Estructura de datos La estructura de este flujo comprende: fecha, numero de pedido, descripción de producto, cantidad de producto.
Estado de Inventario
Descripción Es la confirmación de los productos requeridos cumplen con lo mínimo de existencia requerida de la entidad cliente.
Procesos-Entidades Este se da de almacenamiento de productos hacia el Cliente.

Estructura de datos Los datos que recibe la gerencia general son: numero de cotización, nombre del cliente, descripción de productos, cantidad a elaborar, materias primas y sus costos respectivos.
Solicitud de aprobación
Descripción Es un documento que se le envía a la Gerencia General, cuando el cliente rebasa su crédito, esta puede tomar dos aspectos: rechazo o aprobación de crédito.
Procesos-Entidades El gerente general recibe esta solicitud de la persona que factura con relación al crédito del cliente
Estructura de datos Los datos que componen este flujo son: numero de cotización, fecha, datos del cliente, saldo de Cliente, cotización (inf. De precios y productos)
Solicitud aprobada
Descripción Es la respuesta de la solicitud de aprobación del crédito de un cliente específico, y esta puede ser negativa o positiva.
Procesos-Entidades El gerente general **Envía** esta solicitud aprobada o rechazada a la persona que factura con relación al crédito del cliente.
Estructura de datos

Los datos que componen este flujo son: numero de cotización, fecha, datos del cliente, saldo de Cliente, cotización (inf. De precios y productos), firma de aprobación o Negación de Crédito.

Datos de Cliente + cotización

Descripción

Se refiere a la validación que se verifica cuando se tiene los requerimientos necesarios (precios y existencias) + datos del cliente.

Procesos-Entidades

La validación del crédito de cliente de la persona encargada de facturar, ya sea aprobada o denegada

Estructura de datos

Esta comprende: fecha, datos del cliente, detalle del producto, precios, cantidad y total de cotización aprobada por el cliente.

Factura

Descripción

Es el documento de factura o crédito fiscal que se emite en forma total o parcial del pedido del cliente.

Procesos-Entidades

La emisión de este comprobante es entregada a la entidad cliente

Estructura de datos

Esta comprende: numero de factura, fecha, datos del cliente, detalle del producto, precios, cantidad y valores y total.

Saldo Validado

Descripción Es la verificación del total del cliente no sobrepase él limite de crédito estipulado.
Procesos-Entidades Esta información se da de cliente a través de la cotización que el mismo aprueba a la persona que factura para verificar saldo.
Estructura de datos Esta comprende: fecha, datos del cliente, detalle del producto, precios, cantidad y total de cotización aprobada por el cliente mas una verificación de saldo aprobado.

Copia de factura
Descripción Es una copia de la factura expedida al cliente.
Procesos-Entidades Este se da de la salida de pedidos a la entidad Contabilidad.
Estructura de datos Esta comprende: numero de factura, fecha, datos del cliente, detalle del producto, precios, cantidad y valores y total.

ENTIDADES.

CLIENTE.
Descripción.

El cliente llega a la empresa en búsqueda de su producto, El cliente solicita un producto, y lo transmite a la persona que factura. Al cliente se le presenta la cotización de ventas y él es quien determina al final la aceptación o rechazo de la misma.

Flujos de datos.

Internos.

- Solicitud de compra.
- Datos de pago.
- Respuesta de la cotización.

Externos.

- Cotización.
- Datos de entrega.

Resumen de Lógica.

- Existe una solicitud de compra hacia la persona que factura.
- El cliente recibe una cotización de acuerdo a sus requerimientos.
- El cliente toma la decisión de aceptar la cotización o rechazarla.
- Si el cliente acepta la cotización se le factura.
- Se le entrega un recibo de factura.

GERENCIA GENERAL.

Descripción.

Esta compuesta por una sola persona, la que tiene como función principal la administración de la empresa. Esta entidad es la encargada de decidir si dar un crédito mayor a sus clientes o no.

Flujos de datos.

Internos.

- Solicitud de aprobación de Crédito.

CONTABILIDAD.

Descripción.

Es el Departamento encargado de realizar las gestiones financieras, control contable y manejo de archivos de documentos legales.

Flujos de datos.

Internos.

(No posee)

Externos.

- Copia de Factura

PROCESOS.

VALIDACIÓN DE CREDITO.

Descripción.

Se realiza con los datos de pago del cliente, es decir se evalúa él limite de crédito de

Flujos de datos.

Internos.

- Datos de pago

Externos.

- Respuesta

ELABORACIÓN DE RECIBO DE PAGO.

Descripción.

Se elabora un recibo de pago, en base a los pagos del cliente cuando el cliente paga el anticipo para la elaboración del producto, o los pagos del cliente.

Flujos de datos.

Internos.

- Recibo ingreso.
- Datos de pago.

Externos.

- Cotización aceptada.
- Datos de pago validados.

SALIDA DE PEDIDOS

Descripción.

Se elabora una factura que detalla los productos despachados al cliente.

Flujos de datos.

Internos.

- Pedido.

Externos.

REQUERIMIENTO DE PRODUCTO

Descripción.

Es un proceso donde se obtienen los datos + requerimientos y se verifican los precios y existencias requeridas por el cliente.

Flujos de datos.

Internos.

- Datos + Requerimientos

Externos.

- Cotización

NIVEL UNO

FLUJOS DE DATOS

Requerimientos mas Datos de Cliente

Descripción

El cliente proporciona los requerimientos del producto más los Datos que se necesita.

Procesos-Entidades

El cliente Realiza una Solicitud Para la compra de materiales que requiere a través de una solicitud.

Estructura de datos

Los datos que el cliente proporciona son: características del producto, precios aceptables, Cantidad x Unidad, fecha de entrega y datos de cliente

Cotizaciones

Descripción

Es un documento donde se especifica los precios actuales de los productos que el cliente requiere y esperar que el cliente los acepte.

Procesos-Entidades

La cotización se da desde el sistema de y ventas hacia la entidad cliente.

Estructura de datos

Los datos que contiene la cotización son: descripción del producto cotizado, cantidades, precios, fecha de cotización, fecha de entrega, datos del vendedor y cualquier otra información adicional necesaria.

Cotización Aprobada

Descripción

Es la decisión que el cliente toma de acuerdo a la cotización que se le entrego, puede tener dos diferentes tipos que son: cotización aceptada,

cotización rechazada.
Procesos-Entidades Esta es la respuesta desde la entidad cliente al proceso de control y venta.
Estructura de datos La estructura esta defina por cotización aceptada o rechazada.
Requerimientos de Precios
Descripción Es un documento que se emite con relación a los precios especificados que el cliente requiere.
Procesos-Entidades El documento es realizado por el departamento de ventas hacia la entidad cliente.
Estructura de datos La estructura del documento es: fecha, nombre de cliente, precios de producto requerido, nombre de producto
Precios específicos
Descripción Es la información actual de los precios de los productos contra los precios que el cliente requiere.
Procesos-Entidades Es proporcionado por la entidad cliente hacia el departamento de ventas.
Estructura de datos

La estructura del documento es: fecha, nombre de cliente, precios de producto confrontados(precios requeridos contra precios actuales),nombre de producto,

Este se anexa a la cotización.

Requerimientos de Producto(Existencias)

Descripción

Comprende la verificación del producto que el cliente requiere.

Procesos-Entidades

Este va desde persona que factura hacia almacenamiento producto.

Estructura de datos

La estructura de este flujo comprende: fecha, numero de pedido, descripción de producto, cantidad de producto.

Estado de Inventario

Descripción

Es la confirmación de los productos requeridos cumplen con lo mínimo de existencia requerida de la entidad cliente.

Procesos-Entidades

Este se da de almacenamiento de productos hacia el Cliente.

Estructura de datos

Los datos que recibe la gerencia general son: numero de cotización, nombre del cliente, descripción de productos, cantidad a elaborar, materias primas y sus costos respectivos.

Solicitud de aprobación
Descripción Es un documento que se le envía a la Gerencia General, cuando el cliente rebasa su crédito, esta puede tomar 2 aspectos: rechazo o aprobación de crédito.
Procesos-Entidades El gerente general recibe esta solicitud de la persona que factura con relación al crédito del cliente
Estructura de datos Los datos que componen este flujo son: numero de cotización, fecha, datos del cliente, saldo de Cliente, cotización (inf. De precios y productos)
Solicitud Aprobada
Descripción Es la respuesta de la solicitud de aprobación del crédito de un cliente específico, y esta puede ser negativa o positiva.
Procesos-Entidades El gerente general **Envía** esta solicitud aprobada o rechazada a la persona que factura con relación al crédito del cliente.
Estructura de datos Los datos que componen este flujo son: numero de cotización, fecha, datos del cliente, saldo de Cliente, cotización (inf. De precios y productos), firma de aprobación o Negación de Crédito.
Saldo Validado + Cotización Aprobada

Descripción

Se refiere a la validación que se verifica cuando se tiene los requerimientos necesarios (precios y existencias) + datos del cliente.

Procesos-Entidades

La validación del crédito de cliente de la persona encargada de facturar, ya sea aprobada o denegada.

Estructura de datos

Esta comprende: fecha, datos del cliente, detalle del producto, precios, cantidad y total de cotización aprobada por el cliente.

Factura

Descripción

Es el documento de factura o crédito fiscal que se emite en forma total o parcial del pedido del cliente.

Procesos-Entidades

La emisión de este comprobante es entregada a la entidad cliente

Estructura de datos

Esta comprende: numero de factura, fecha, datos del cliente, detalle del producto, precios, cantidad y valores y total.

Saldo Validado

Descripción

Es la verificación del total del cliente no sobrepase el limite de crédito estipulado.

Procesos-Entidades Esta información se da de cliente a través de la cotización que el mismo aprueba a la persona que factura para verificar saldo.
Estructura de datos Esta comprende: fecha, datos del cliente, detalle del producto, precios, cantidad y total de cotización aprobada por el cliente mas una verificación de saldo aprobado.
Copia de factura
Descripción Es una copia de la factura expedida al cliente.
Procesos-Entidades Este se da de la salida de pedidos a la entidad Contabilidad.
Estructura de datos Esta comprende: numero de factura, fecha, datos del cliente, detalle del producto, precios, cantidad y valores y total.

1.3 Proceso de inventario.

Independientemente del método de inventario, el proceso de inventario puede dividirse en tres fases:

- Preparación de inventario.
    - Creación de un documento para inventario.
    - Bloqueo de materiales para contabilización.

    - Impresión y distribución del documento para inventario.
- Recuento de inventario.
    - Recuento de stocks.
    - Introducción del resultado del recuento en la impresión del documento para inventario.
- Análisis del inventario.
    - Introducción del resultado del recuento en el sistema.
    - Inicio de un nuevo recuento, en caso necesario.
    - Contabilización de las diferencias de inventario.

Etapas de tratamiento

En la preparación del inventario, deben realizarse los siguientes pasos básicos:

- Creación de un documento para inventario
- Introducción del recuento de inventario
- Compensación de las diferencias de inventario

Además, también es posible agrupar varias fases individuales y llevarlas a cabo en una única etapa, tal como se indica a continuación:

- Contabilización del resultado del recuento de inventario sin referencia a un documento para inventario. En esta etapa, se combina lo siguiente:
    - Se crea un documento para inventario.
    - Se contabiliza el resultado del recuento.
- Contabilización del resultado del recuento y compensación de diferencias de inventario. Si existe un documento para inventario, en esta etapa se realiza lo siguiente:
    - Se contabiliza el resultado del recuento.
    - Se compensa cualquier diferencia de inventario.

- Compensación de diferencias de inventario sin referencia a un documento para inventario. En esta etapa, se combina lo siguiente:
    - Se crea un documento para inventario.
    - Se contabiliza el resultado del recuento.
    - Se compensa cualquier diferencia de inventario.

Status de inventario

Con el fin de supervisar el proceso de inventario, en cada documento para inventario se registran las etapas llevadas a cabo. Esta información puede hallarse en una posición o cabecera de documento.

- Historial de inventario de una posición

  En el historial de inventario pueden determinarse las etapas que se han realizado en lo que respecta a una posición.

- Cabecera del documento

  En la cabecera del documento, los campos *Status de recuento, Status de compensación* y *Status de borrado* informan de si algunas o todas las posiciones se han contado, compensado (diferencias de inventario) o borrado.

También es posible obtener estadística en el documento; la estadística proporciona un listado del número de posiciones abiertas, contadas, compensadas (diferencias de inventario), contadas de nuevo o borradas.

1.3.1 Preparación de inventario

En la preparación del inventario, deben seguirse las siguientes etapas:

1.3.1.1 Creación de un documento para inventario

Un documento para inventario contiene, entre otras cosas, los siguientes datos:

- El centro y el almacén en los que se realiza el recuento
- El momento en el que se realiza el recuento
- Los materiales objeto de recuento
- En el material sujeto a lote: Los lotes objeto de recuento
- En la valoración separada: Los substocks objeto de recuento
- Los tipos de stocks objeto de recuento

1.3.1.2 Bloqueo de materiales para contabilización

Debido al desfase temporal entre un movimiento de material y la contabilización del mismo, se produce una diferencia a corto plazo entre el stock en almacén real y el stock teórico. Con el fin de evitar dicha diferencia en el inventario, SAP recomienda bloquear los materiales a contabilizar mientras se hace inventario. El bloqueo de la contabilización puede realizarse de dos modos:

- Se bloquean los materiales relevantes cuando se introduce el documento para inventario. Esto es recomendable si el documento para inventario se crea inmediatamente antes del recuento.
- Se bloquean los materiales relevantes posteriormente, mediante la modificación del documento para inventario contabilizado. Esto es recomendable si el documento para inventario no se crea inmediatamente antes del recuento.

El bloqueo de contabilización se cancela automáticamente cuando se contabilizan los resultados del recuento del documento para inventario.

1.3.1.3 Impresión y distribución del documento para inventario.

Se debe imprimir el documento para inventario correspondiente al recuento de inventario y pasarlo a los responsables del recuento.

1.3.2 Análisis de inventario

El análisis de inventario cubre las siguientes tareas:

- Introducción del resultado del recuento en el sistema. Una vez realizado el recuento, deben introducirse los resultados del recuento en el sistema. Se introducen los stocks contados de cada posición de un documento para inventario.

1.3.2.1 Inicio de un nuevo recuento

Es posible iniciar nuevos recuentos de las posiciones individuales de un documento para inventario. Esto es recomendable si se sospecha de la existencia de un error en el recuento. Cuando se inicia un nuevo recuento, se crea un documento para inventario nuevo.

1.3.2.2 Compensación de diferencias

Si el inventario difiere del stock teórico, se deben compensar las diferencias para corregir el stock teórico. Esta etapa pone fin al inventario.

Cuando se compensa la diferencia de inventario, el sistema crea un documento de material en el que se registran los stocks teóricos compensados y un documento contable que contiene las actividades de cuentas necesarias.

Existe un período limitado para compensar la diferencia:

- El **período contable** se fija de modo automático durante el recuento. De este modo, la diferencia debe compensarse en el mismo período o en el período siguiente, si se permiten compensaciones en el período anterior.

El **ejercicio** se fija especificando una fecha de recuento prevista cuando se crea un documento para inventario. Todas las compensaciones posteriores a dicho documento deben realizarse en este ejercicio y/o en el primer período del siguiente ejercicio, si se permiten compensaciones en el período anterior.

2 Programación Extrema

2.1 Objetivo

Estudiar la metodología de programación extrema para especificar en qué manera puede ser implementada durante el desarrollo de esta tesis.

2.2 Definición

La programación extrema o *eXtreme Programming* (XP) 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)[3]. Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

Los Valores originales de la programación extrema son: simplicidad, comunicación, retroalimentación (*feedback*) y coraje. Un quinto valor, respeto, fue añadido en la

---

[3] Beck, K. 2002 *Una explicación de la Programación Extrema. Aceptar el cambio*. Addison-Wesley, 2002

segunda edición de *Extreme Programming Explained*. Los cinco valores se detallan a continuación.

2.3 Simplicidad

La simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. Un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hace que la complejidad aumente exponencialmente. Para mantener la simplicidad es necesaria la refactorización del código, ésta es la manera de mantener el código simple a medida que crece. También se aplica la simplicidad en la documentación, de esta manera el código debe comentarse en su justa medida, intentando eso sí que el código esté autodocumentado. Para ello se deben elegir adecuadamente los nombres de las variables, métodos y clases. Los nombres largos no decrementan la eficiencia del código ni el tiempo de desarrollo gracias a las herramientas de autocompletado y refactorización que existen actualmente. Aplicando la simplicidad junto con la autoría colectiva del código y la programación por parejas se asegura que cuanto más grande se haga el proyecto, todo el equipo conocerá más y mejor el sistema completo.

2.4 Comunicación

La comunicación se realiza de diferentes formas. Para los programadores el código comunica mejor cuanto más simple sea. Si el código es complejo hay que esforzarse para hacerlo inteligible. El código autodocumentado es más fiable que los comentarios ya que éstos últimos pronto quedan desfasados con el código a medida que es modificado. Debe comentarse sólo aquello que no va a variar, por ejemplo el objetivo de una clase o la funcionalidad de un método. Las pruebas

unitarias son otra forma de comunicación ya que describen el diseño de las clases y los métodos al mostrar ejemplos concretos de como utilizar su funcionalidad. Los programadores se comunican constantemente gracias a la programación por parejas. La comunicación con el cliente es fluida ya que el cliente forma parte del equipo de desarrollo. El cliente decide que características tienen prioridad y siempre debe estar disponible para solucionar dudas.

2.5 Retroalimentación (*feedback*)

Al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real. Al realizarse ciclos muy cortos tras los cuales se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en lo que es más importante. Considérense los problemas que derivan de tener ciclos muy largos. Meses de trabajo pueden tirarse por la borda debido a cambios en los criterios del cliente o malentendidos por parte del equipo de desarrollo. El código también es una fuente de retroalimentación gracias a las herramientas de desarrollo. Por ejemplo, las pruebas unitarias informan sobre el estado de salud del código. Ejecutar las pruebas unitarias frecuentemente permite descubrir fallos debidos a cambios recientes en el código.

2.6 Coraje o valentía

Los puntos anteriores parecen tener sentido común, entonces, ¿por qué coraje? Para los gerentes la programación en parejas puede ser difícil de aceptar, porque les parece como si la productividad se fuese a reducir a la mitad ya que solo la mitad de los programadores está escribiendo código. Hay que ser valiente para confiar en que la programación por

parejas beneficia la calidad del código sin repercutir negativamente en la productividad. La simplicidad es uno de los principios más difíciles de adoptar. Se requiere coraje para implementar las características que el cliente quiere ahora sin caer en la tentación de optar por un enfoque más flexible que permita futuras modificaciones. No se debe emprender el desarrollo de grandes marcos de trabajo (*frameworks*) mientras el cliente espera. En ese tiempo el cliente no recibe noticias sobre los avances del proyecto y el equipo de desarrollo no recibe retroalimentación para saber si va en la dirección correcta. La forma de construir marcos de trabajo es mediante la refactorización del código en sucesivas aproximaciones.

2.7 Respeto

El respeto se manifiesta de varias formas. Los miembros del equipo se respetan los unos a otros, porque los programadores no pueden realizar cambios que hacen que las pruebas existentes fallen o que demore el trabajo de sus compañeros. Los miembros respetan su trabajo porque siempre están luchando por la alta calidad en el producto y buscando el diseño óptimo o más eficiente para la solución a través de la refactorización del código. Los miembros del equipo respetan el trabajo del resto no haciendo menos a otros, si no orientándolos a realizarlo mejor, obteniendo como resultado una mejor autoestima en el equipo y elevando el ritmo de producción en el equipo.

2.8 Características fundamentales

Las características fundamentales del método son:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.

- Pruebas unitarias: continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.

- Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.

- Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.

- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.

- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.

- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y

tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Cuanto más simple es el sistema, menos tendrá que comunicar sobre éste, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores.

2.9 Las instalaciones

Para poder hacer algo es necesario tener el lugar adecuado para hacerlo. La programación extrema necesita de mucho espacio en común.

Hay que tener en cuenta que esta metodología es una disciplina comunal de desarrollo de *software*. Los miembros del equipo necesitan poder verse unos a otros, y aceptar que otros van a escuchar los problemas personales que puedan surgir dentro del grupo de desarrollo.

La estructura recomendada para un equipo de programación extrema es:

- Tener un espacio central común.
- Tener pequeños cubículos en los laterales de dicho espacio.
- Colocar las mejores máquinas en el espacio central común.

Los miembros del equipo pueden tener sus efectos personales en los cubículos y pueden ir a ellos cuando no quieran ser interrumpidos.

Al tener las máquinas en el área central común, se obligará a una persona que desee codificar, a ir dicho sector, de tal forma que todo el mundo pueda ver lo que

está haciendo. Las parejas se pueden formar fácilmente y cada pareja puede atraer la energía de las otras parejas que también están desarrollando al mismo tiempo.

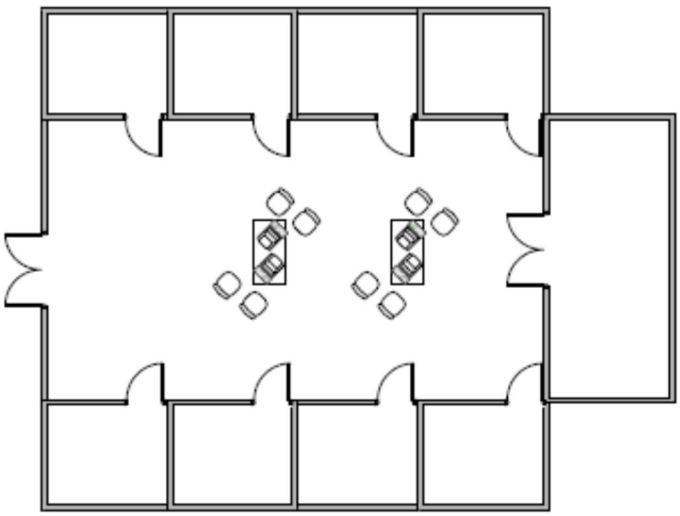

Ilustración 5 – Instalación ideal para un equipo extremo

Si el equipo debe trabajar en el edificio del cliente, hay que tener cuidado, ya que lo primero es la comodidad del equipo de programación extrema. Ellos deben tener la luz adecuada, el mobiliario adecuado, un movimiento fluido, un ambiente agradable que permita el desarrollo de la creatividad.

Tomar el control del entorno físico envía un mensaje convincente al equipo de no permitir que intereses opuestos irracionales en la organización impidan conseguir el éxito.

2.10 Planificación

Se determina el ámbito de la siguiente versión, combinando las prioridades del negocio y las estimaciones técnicas. El desarrollo del *software* es siempre un dialogo que evoluciona entre lo posible y lo deseable.

El cliente debe tener claro que es imprescindible y que su presencia no es opcional, el cliente debe establecer una prioridad ya que únicamente él es el que conoce lo que tiene más valor para el negocio.

El cliente establece una propuesta del contendido de la versión, la cual es discutida con el equipo de desarrollo para poder establecer la verdadera urgencia de cada parte del proyecto propuesta para dicha versión y la fecha de entrega.

Una vez se ha establecido lo necesario para la versión, se realiza una programación detallada divida en iteraciones. Los programadores necesitan la libertad de realizar las partes con mayor riesgo primero para reducir el riesgo global del proyecto.

Se deben realizar versiones pequeñas que contengan los requisitos más importantes para el negocio. Éstas versiones son muchas historia juntas que hacen que el negocio tenga sentido. Dicha versión debe ser atómica por lo que no se puede implementar a medias y luego presentarla como una versión completa. Se debe recordar que entre más pequeña es la versión, más rápido es evaluada por el cliente y se tiene una pronta retroalimentación.

2.10.1 Planificación de la versión

La versión se planifica con un dialogo entre el cliente y el equipo de desarrollo. Este dialogo se debe dar en un ambiente de confianza mutua y respeto a la contraparte. Tanto el cliente como el equipo de desarrollo deben permitir que su contraparte realice su trabajo.

Hay que enfatizar que el objetivo de esta planificación es de maximizar el valor del *software* producido por el equipo. Del valor del *software*, se debe descontar el coste de su desarrollo y el riesgo contraído durante el desarrollo.

Se obtiene una mejor planificación si se tiene en cuenta que hay que invertir lo menos posible para poner la mayor funcionabilidad válida en producción tan rápidamente como sea posible.

En la planificación de versión se utilizan fichas de historias, las cuales maneja la siguiente información:

- Fecha.
- Tipo de actividad (nueva, arreglar, agregar).
- Número de historia.
- Prioridad (para el cliente y para el equipo de desarrollo).
- Descripción de la historia.
- Notas extras.
- Bitácora de labores (fecha, estado, por hacer, comentario).

2.10.1.1 Fase de exploración

Tiene como propósito dar a conocer una estimación de lo que el sistema realizará. Se puede dividir esta fase en tres pasos:

- **Escribir una historia**: el cliente escribe una historia en la ficha ya mencionada, se incluye el nombre y un párrafo que explique el objetivo de la historia.

- **Estimar una historia**: el equipo de desarrollo calcula el tiempo que le llevará implementar dicha historia, si no se puede estimar la historia, se le pide al

cliente que aclare o divida la historia. Los tiempos se estiman con el tiempo ideal de ingeniería (TII). Antes de comprometerse a una programación, se mide la relación entre el tiempo ideal y el calculado.

- **Dividir una historia**: si el equipo de desarrollo no puede estimar una historia global, o si el cliente se da cuenta qué parte de una historia es más importante que el resto, el cliente puede dividir la historia en dos o más historias.

2.10.1.2 Fase de compromiso

Tiene como propósito que el cliente establezca el ámbito y la fecha de la próxima versión y el equipo se comprometa a entregarla. Se divide en cuatro fases:

- **Clasificar por valor**: el cliente tiene la obligación de clasificar las historias entre aquellas que son cruciales para el sistema, las que son menos importantes y las que le agradaría tener.

- **Clasificar por riesgo**: el equipo de desarrollo clasifica las historias en aquellas que puede estimar con precisión, aquellas que se pueden estimar razonablemente bien y aquellas que no pueda estimar.

- **Establecer velocidad**: el equipo de desarrollo le dice al cliente la rapidez con que el equipo pude trabajar en el TII por mes o semana.

- **Escoger el ámbito**: el cliente escoge el conjunto de fichas de la versión, bien poniendo o estableciendo una fecha de finalización y escogiendo las fichas sobre la base de sus estimaciones y la velocidad del proyecto o bien escogiendo las fichas y calculando la fecha.

2.10.1.3 Fase de dirección

En esta fase es en donde se actualiza el plan basándose en lo que han aprendido tanto el cliente como el equipo de desarrollo.

- **Iteración**: al comienzo de cada iteración, el cliente escoge la historia más valiosa a ser implementada en una iteración. Las historias para la primera iteración deben producir un sistema que funciona de punta a punta.

- **Recuperación**: si el equipo de desarrollo se da cuenta que ha sobreestimado su capacidad creación de código, puede preguntarle al cliente qué grupo de historias son más valiosas para mantener en la versión actual basándose en la nueva velocidad y estimaciones.

- **Nueva historia**: es muy probable que el cliente desee agregar una nueva historia a mitad del desarrollo de una versión, se agrega una nueva ficha, el equipo de desarrollo estima la historia y el cliente elimina en el resto del plan de historias que tengan una estimación equivalente y se agrega la nueva historia a la versión.

- **Volver a estimar**: si el equipo de desarrollo se da cuenta que el plan no proporciona un panorama claro para el desarrollo, puede volver a estimar el resto de historias que quedan y la velocidad.

2.10.2 Planificación de la iteración

La planificación de la iteración es muy parecida a la planificación de la versión. La primera modificación se puede apreciar con las personas involucradas ya que son únicamente los programadores individuales los que afectan esta planificación.

Otra variante es que se utilizan fichas de tarea en vez de fichas de historias. Se sigue teniendo tres fases identificadas con los mismos nombres pero las acciones de cada una cambian.

2.10.2.1 Fase de exploración

- **Escribir una tarea**: se selecciona una historia para la iteración y el programador la transforma en tareas. Esta tarea es más pequeña que la historia y puede que existan algunas que no estén directamente relacionadas con alguna historia particular.

- **Dividir una tarea / combinar tareas**: si no se puede estimar una tarea en unos pocos días, se divide en tareas más pequeñas, por lo contrario si varias tareas necesitan unas horas, hay que combinarlas para formar una tarea mayor.

2.10.2.2 Fase de compromiso

- **Acepta una tarea**: una vez que se tienen las tareas, es responsabilidad del programador el seleccionar una tarea y comprometerse a terminarla.

- **Estimar una tarea**: el programador estima el TII para cada tarea. Esto esta condicionado a obtener ayuda de otro programador que puede estar más familiarizado con el código a elaborar.

- **Conjunto de factores de carga**: cada miembro del equipo de desarrollo elige su factor de carga para cada iteración, éste número indica el tiempo que

dedicará realmente al desarrollo. Este número no debe ser muy alto ya que representaría que no va a dedicar tiempo a ayudar a sus compañeros.

- **Balanceo**: cada programador suma sus estimaciones de las tareas y las multiplica por su factor de carga. Los programadores que se comprometen en exceso deben ceder algunas tareas, si todo el equipo está comprometido en exceso, se debe encontrar la forma de volver al equilibrio.

2.10.2.3 Fase de dirección

- **Implementar una tarea**: un programador coge una ficha de tareas, encuentra un compañero, escriben los casos de prueba para la tarea, realizan todo el trabajo, lo integran y generan nuevo código cuando el juego de pruebas globales funciona.

- **Registrar el progreso**: cada tres o dos días, un miembro del equipo pregunta a cada programador cuánto tiempo ha utilizado a cada una de sus tareas y cuanto le hace falta.

- **Recuperación**: si un programador está comprometido en exceso, está obligado a pedir ayuda y evitar reducir el ámbito de algunas tareas, pedir al cliente que reduzca el ámbito de algunas historias, quitar tareas no esenciales, obtener una mejor ayuda o pedir al cliente que retrase algunas historias a una iteración posterior.

- **Verificar la historia**: cuando ya se tiene las pruebas funcionales preparadas y completadas las tareas para una historia, se ponen en funcionamiento las pruebas funcionales para verificar que la historia funciona. Los casos

interesantes que surjan durante la implementaron pueden añadirse al conjunto de pruebas funcionales.

2.11 Diseño

Cuando se elabora el diseño hay que recordar que es más fácil dar a entender un diseño sencillo que uno complicado, por lo que hay que crear estrategias de diseño que propongan el diseño más sencillo posible, y que tenga relación con el resto de los objetivos. Estos diseños deben ser bien claros para que las personas puedan entenderlos con bastante claridad sin tener que perder mucho tiempo tratando de entender lo quiere decir.

El diseño debe encontrar de una forma rápida una técnica para comprobarse, esto se logra consultando con los otros miembros del equipo si le parece el diseño y escuchar los cambios propuestos. El bueno retroalimentarse de lo que se aprende en el diseño.

Crear el diseño, probarlo, mejorarlo y aprender de él, es un proceso que se debe hacer de una forma cíclica y lo más pronto posible con tal de mejorar la calidad del diseño.

El éxito del diseño de la metodología de programación extrema se basa en asumir la simplicidad, ya que como los proyectos están sujetos a cambios es muy probable que el tiempo invertido en un diseño complejo sea tirado a la basura.

Es necesario tener la mentalidad del cambio incremental, se debe diseñar un poco cada vez, hay que estar consientes que el sistema nunca estará completamente diseñado, ya que como hay cambios, existirán partes del sistema que nunca cambiarán y otras que evolucionarán a lo largo del tiempo.

Ya consientes que el diseño irá cambiando a lo largo del tiempo, entonces se debe enfocar el esfuerzo en lo que se desea hacer. El diseño debería de ser suficientemente completo para adaptarlo a los objetivos actúales pero no más.

Una estrategia de diseño recomendada por el creador de la programación extrema es la siguiente:

1. "Comenzar con una prueba, así sabremos lo que estamos haciendo. Tenemos que hacer una cierta cantidad de diseño mínimo para poder escribir la prueba: ¿Cuáles son los objetivos y sus métodos visibles?
2. Diseñar e implementar justo lo suficiente para conseguir que la prueba funcione. Tendrás que diseñar una cantidad suficiente de la implementación para que estas pruebas y todas las anteriores funcionen.
3. Repetir.
4. Si siempre ves la posibilidad de hacer el diseño más simple, hazlo."[4]

2.12 Desarrollo

El desarrollo conlleva una estrategia que comienza con la planificación de la iteración. La integración continua reduce los conflictos del desarrollo y crea un final natural para un episodio del desarrollo. La propiedad colectiva anima a todo el equipo a mejorar el sistema y finalmente la programación en parejas mantiene unido todo el proceso.

[4] Beck, K. *Una explicación de la Programación Extrema. Aceptar el cambio.* Addison-Wesley, 2002.

2.12.1 Integración continua

El código no puede permanecer sin integrarse por mucho tiempo. Al final de cada episodio de programación, el código debe ser integrado con la última versión y todas las pruebas deben funcionar al 100%.

No sería posible trabajar con este estilo de integración, si el tiempo para realizar la misma fuera de un par de horas. Es importante tener herramientas que soporten un ciclo rápido de integración/construcción/pruebas. También es necesario tener un conjunto de pruebas que se ejecuten en unos pocos minutos.

La integración continua reduce significativamente los riesgos del proyecto. Si dos personas tienen diferentes ideas sobre la forma o el funcionamiento de una parte del código, se sabrá en horas. No se dará el caso de dedicarle varios días a encontrar un error que fue cometido en algún momento de las últimas semanas.

2.12.2 Propiedad colectiva

Es el concepto de permitir que cualquier persona pueda cambiar cualquier parte del código del sistema en cualquier momento. Esta propiedad del desarrollo se puede mantener si las pruebas bien documentadas y su calidad es muy buena.

Este concepto de propiedad colectiva permite que el código complejo no dure mucho tiempo, esto se debe a que, como todo el mundo puede ver cualquier parte del sistema, este código complejo se encontrará tarde o temprano. Al encontrarlo, alguien intentará simplificarlo. Una vez el nuevo código haya pasado las pruebas, el código complejo será desechado.

Cuando el programador sabe que el código que se está realizando va a ser utilizado por un compañero, se detiene a pensar dos veces antes de agregar un código complejo al sistema.

La propiedad colectiva tiende a esparcir el conocimiento del sistema sobre todo el equipo lo cual reduce los riegos del proyecto.

2.12.3 Programación en parejas

La programación en parejas es un dialogo entre dos personas que intentan simultáneamente programar (y analizar, diseñar y probar) y comprender juntos como programar mejor.

2.12.4 Recodificación

Los programadores reestructurarán el sistema sin cambiar su comportamiento para eliminar la duplicidad, mejorar la comprensión, la sencillez o añadir flexibilidad del código fuente.

El éxito de la recodificación proviene de los siguientes puntos.

- Se debe estar acostumbrado a la propiedad colectiva del código y no se tenga inconvenientes en hacer cambios si son necesarios.

- Tener estándares de codificación, para que no se tenga que cambiar el formato del código antes de hacer la recodificación.

- Codificar en parejas para que se tenga más valentía a la hora de afrontar una mejora difícil, y sea menos probable que se dañe algo sin querer.

- Se debe tener un diseño sencillo para que la recodificación no tenga dificultad.

- Tener pruebas, de esta forma de disminuye la probabilidad de dañar algo sin querer.

- Tener integración continua, ya que si se ha dañado algo, que no está directamente asociado con el diseño que se ha tocado, o la recodificación entra en conflicto con algo que funciona se sepa en cuestión de horas.

2.12.5 Estándares de codificación

Ya que se va a tener a varios programadores cambiando de una parte del sistema a otra distinta, intercambiando compañeros varias veces al día y haciendo recodificación en otras partes del código de forma constante, no es posible tener diferentes tipos de normas de codificación.

El estándar debería exigir la menor cantidad de trabajo posible y ser consistente. Un estándar de codificación muy recomendado y utilizado es de PHP.

2.13 Pruebas

Las pruebas que se deben escribir en la programación extrema deben ser aisladas y automatizadas. Cada prueba no debe interactuar con las otras pruebas que se escriben, para evitar el problema que una prueba falle y cause otros fallos.

Las pruebas tienen mayor valor cuando el estrés crece, ya que el juicio humano empieza a fallar, al automatizarlas se evita el factor humano, Las pruebas deben devolver una condición de aprobado o rechazado del comportamiento del sistema.

Es muy difícil probar absolutamente todo, por lo que se debe probar cosas que podrían fallar. Si el código es tan sencillo que posiblemente no puede fallar, entonces no se debe escribir una prueba para esto. La experiencia es lo que ayuda al programador a identificar que clase de pruebas tienden a merecer la pena y cuales no.

Las pruebas provienen tanto del programador como del cliente, ellos deben trabajar en conjunto para lograr el éxito de las pruebas.

2.13.1 Los programadores

El programador escribe pruebas método-a-método bajo las siguientes circunstancias.

- Si la interfaz para un método no está clara, debe escribir una prueba antes de escribir el método.

- Si la interfaz está clara, pero imagina que la implementación podría ser un poco menos complicada, debe escribir una prueba antes de escribir el método.

- Si piensa en una circunstancia no usual en la cual el código debería funcionar conforme está escrito, debe escribir una prueba para comunicar la circunstancia.

- Si encuentra un problema posterior, debe escribir una prueba que aísla el problema.

- Si está haciendo recodificación en algún código, y no está seguro de cómo debe ser su comportamiento y todavía no hay una prueba para dicho comportamiento, debe escribir una primera prueba.

Estas, también son conocidas como pruebas de unidad y no se puede decir que algo medio pasó la prueba, la unidad o funciona o no funciona, no hay términos medios.

2.13.2 Los clientes

Los clientes escriben pruebas historia-a-historia. La pregunta que necesitan hacer es "¿Qué tendría que probar antes de estar seguro que esta historia se ha completado?". El cliente establece diferentes escenarios con los cuales se puede probar el producto.

Estas son pruebas funcionales y aunque cada escenario tiene unidades en común otras no lo son, por lo que, la prueba funcional es un éxito o un fracaso, no es discreto. Conforme se vaya cerrando una versión, el cliente necesitará clasificar las pruebas funcionales que fallan. Algunas serán más importantes de corregir que otras. La idea, es llegar al valor más cercano del 100% de aceptabilidad.

El equipo XP debe tener al menos una persona dedicada a hacer pruebas, sin importar el tamaño del equipo. El tiene que ayudar al cliente a escribir las pruebas funcionales y traducir los datos de prueba a prueba. El, también utiliza las pruebas ideadas por el cliente como punto de partida para las variaciones que probablemente harán que falle el *software*.

2.13.3 Otras pruebas

Las pruebas de unidad y funcionales son el centro de la estrategia de pruebas de la programación extrema. El equipo de programación extrema debe reconocer cuando va por más camino y debe utilizar cualquier otro tipo de prueba que pueda ayudar. Algunas que se utilizan se describen a continuación:

2.13.3.1 Prueba paralela

Es una prueba diseñada para probar que el nuevo sistema funciona exactamente como el viejo. La prueba muestra cómo el nuevo sistema difiere del viejo sistema.

2.13.3.2 Prueba de tensión

Diseñada para simular la peor carga posible. Estas son recomendadas para sistemas complejos donde las características de rendimiento no son fáciles de predecir.

2.13.3.3 Prueba de monkey

Está diseñada para asegurar que el sistema actúa de forma sensible frente a entradas sin sentido.

2.13.3.4 Pruebas recuperación

Diseñada para forzar el fallo del *software* de muchas formas y verifica que la recuperación se lleva a cabo apropiadamente.

2.14 La gestión

Hay que tener cuidado con la estrategia que se utilice, ya que no es posible que una persona sea la que tome todas las decisiones. No existe una persona que sepa lo suficiente para hacer un buen trabajo y tomar todas las decisiones. Pero tampoco se puede dejar al equipo de programación extrema sin supervisión. Es necesario tener a alguien con una visión superior que pueda influir al equipo cuando éste se desvíe.

Se debe tener un director de proyecto que exponga de una forma clara lo que se necesita hacer y no asignar trabajo. Este director de proyecto debe mantener una relación basada en la sinceridad con los programadores ya que ellos tienen la voluntad de hacer un buen trabajo.

El director debe tener una mentalidad de "estoy consiguiendo ayudar a estos tipos a hacer su trabajo" y no de "estoy intentando conseguir que mi equipo haga un trabajo decente". El debe proporcionar continuamente indicadores de lo que ha sucedido en cada incremento en vez de dar un manual de políticas al principio.

El director de proyecto necesita tomar la iniciativa en la adaptación de la programación extrema a las condiciones particulares, debe saber cómo la cultura de la programación extrema choca con la cultura de la empresa y encontrar la forma de resolver aquello que no se adapte.

2.14.1 Control

En cualquier proyecto es necesario controlar lo que se está haciendo. Se puede hacer un sin fin de estimaciones, pero si no se mide lo que realmente sucede frente a lo que se estimó no se aprenderá nada.

Se deben reunir lo que las métricas estén controlando en un momento dado y asegurarse que el equipo es consiente de lo que se estaba midiendo realmente sin olvidar lo que había estimado.

El control necesita llevarse a cabo sin muchos costes, la distracción continua del equipo puede llegar a perjudicar el proyecto. Se recomienda que se recojan los datos reales de desarrollo dos veces por semana.

2.14.2 40-horas semanales

Para mantener el ánimo del personal, se debe tener como regla no trabajar más de 40 horas a la semana. Una persona puede permanecer concentrada durante 35 horas, otra durante 45. Pero ninguna puede ser capaz de hacerlo 60 horas semanales durante varias semanas y, todavía sentirse fresco, creativo, prudente y confiado.

Las horas extras son un síntoma de un problema en el proyecto. El problema puede residir en que la estimación de trabajo fue mal hecha, por lo que es necesario reevaluar si la asignación de recursos fue correcta. Si el problema reside en el personal, se tiene que evaluar la razón por la cual no rinde el equipo y hacer lo posible para que no se retrase el proyecto.

2.14.3 El cliente *in-situ*

El cliente debe ser miembro del equipo de desarrollo, y estar disponible para responder a las preguntas, resolver discusiones, y fijar prioridades a pequeña escala. Este cliente debe ser alguien que utilizará realmente el sistema cuando esté en producción Si se está construyendo un sistema de servicios al cliente, éste será

un cliente representativo del servicio. Si se está haciendo un sistema de gestión de bonos, el cliente debe ser un agente comercial de bonos.

Hay que ser un buen negociador ya que los usuarios del sistema bajo desarrollo son muy valiosos para cederlos al equipo de desarrollo. Se debe decidir que es más valioso, tener el *software* trabajando pronto y mejor o tener el rendimiento de una o dos personas. Analizando esto, se puede llegar a la conclusión, que si el sistema no aporta más valor al negocio que tener una persona trabajando más tiempo, quizás el sistema no debería ser construido.

2.15 Roles del personal

La programación extrema está basada en el trabajo en equipo y éste equipo está formado por programadores, el cliente, el encargado de pruebas, el controlador, el preparador, el consultor, y el jefe.

2.15.1 El programador

Su principal preocupación es la de ser comunicativo con otras personas, ya que la metodología requiere la programación en parejas y resultaría difícil si no se comunicara con ella.

Debe tener una disposición de aprender como enseñar ya que no hay que ser egoístas con el conocimiento del lenguaje de programación.

El, debe tener presente que, si algún componente de comunicación esencial no se ha hecho, no se ha terminado. Debe tomar como base que las pruebas le van a indicar cuando termina con la codificación. Estas pruebas deben mostrar algún aspecto esencial del *software*.

Debe ser lo más simple posible y que funcione, se puede ayudar afrontando el problema con un enfoque de "divide y vencerás".

El programador debe dejar a un lado el concepto de propiedad individual y aceptar que otra persona cambie o modifique lo que el había escrito.

2.15.2 El cliente

Es un factor muy importante ya que únicamente él conoce qué hay que programar. El cliente deba aprender a escribir buenas historias y a estar cómodo influyendo en un proyecto sin ser capaz de controlarlo.

Debe tener la capacidad de tomar decisiones y conocer de qué forma se podrá dar más valor a su negocio.

El cliente es un factor muy importante para hacer las pruebas ya que él es el encargado de escribir las pruebas funcionales, debe trabajar estrechamente con el equipo para aprender qué clase de cosas son útiles para probar.

2.15.3 El encargado de pruebas

Es responsable de ayudar al cliente a escoger y escribir pruebas funcionales, también debe hacer funcionar las pruebas funcionales regularmente y poner los resultados en un lugar destacado.

No hay que olvidar que el encargado de pruebas no es una persona aislada, dedicada a detectar fallos en el sistema, también debe asegurarse que las herramientas de pruebas funcionen correctamente.

2.15.4 El controlador

Se puede tomar como la conciencia del equipo. Debe hacer muchas estimaciones y observar como la realidad conforma esas suposiciones.

Para verificar, debe observar al conjunto del sistema y dar información al equipo si en algún momento van en el rumbo incorrecto o si deben hacer algo para adaptarse a lo estimado para dicha iteración.

Tiene que recaudar datos para conocer como va funcionado el equipo, los resultados de las pruebas funcionales y de las estimaciones que ya se habían hecho. Todo para poder aprender más del equipo y preparase mejor para futuras iteraciones y/o proyectos.

2.15.5 El preparador

Es el responsable del proceso en conjunto y de entender con mucha mayor profundidad la aplicación, debe identificar que practicas alternativas pueden ayudar al conjunto actual de problemas; como están utilizando otros equipos XP y como relacionan con la situación actual.

Hace su mejor trabajo cuando interviene indirectamente, si ve un error en el diseño, primero evalúa si es lo suficientemente importante como para que intervenga.

Lo primero que se tiene que decidir, es si el problema que se ve de tal tamaño necesite aceptar el riesgo de intervenir. Cada vez que se guía, el equipo se vuelve menos independiente. Demasiada dirección hace que se pierda la habilidad del

equipo de trabajar sin el, dando lugar a una baja productividad, una baja calidad y una baja moral.

Debe ser una persona con bastante conocimiento XP para poder detectar cuando está fallando el proceso. El rol del preparador disminuye conforme madura el equipo.

2.15.6 El consultor

El consultor es un especialista, es necesario cuando el equipo necesita un conocimiento técnico profundo en un área.

Tiene como objetivo enseñar al equipo a resolver un problema, debe estar dispuesto a que lo cuestionen ya que las propiedades de la programación extrema hace que se busque la solución más sencilla posible.

2.15.7 El gestor

Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Tanto del gestor como el preparador deben ser líderes transformacionales, deben inspirar a los demás con la visión correcta del proyecto, deben promoverla en contra de la resistencia. Las características comunes de los líderes transformacionales son:

- Visionario.
- Inspirador.
- Amable.
- Considerado.

- Digno de confianza.
- Seguro de si.

3 PHP

3.1 Objetivo

Estudiar cual sería la mejor tecnología que satisfaga la necesidad que presenta la Lavandería Cristal para el problema que ellos plantean teniendo en cuenta que la limitante de inversión y utilización de los equipos que ya posee la empresa.

3.2 ¿Por qué utilizar PHP como lenguaje para desarrollar un sistema?

Es un lenguaje flexible, es rápido de utilizar, se puede programar en cualquier IDE (desde notepad en adelante, siendo mi preferido notepad++ en windows y Geany en Gnome corriendo en Ubuntu GNU/Linux).

Lo interesante de PHP es que es fácil de aprender, rápido de acoplarse, y existe diversidad de "frameworks" bajo los cuales se puede desarrollar código en PHP de manera sencilla.

Alternativas a PHP podrían ser Ruby (con el framework Ruby on Rails), Python, CGI, ASP.NET (Utilizando C# o VB.NET como Codebehind) y Java utilizando WebObjects como framework.

La razón por la que considero a PHP como una opción excelente es porque consume pocos recursos, los scripts corren considerablemente rápido **cuando**

están bien desarrollados, incluye orientación a objetos, y se puede depurar (usando herramientas como PHP.NET) o de manera tradicional usando "*echo*".

Hay comunidades enteras que consideran que PHP en realidad no es tan buena opción como parece. Inclusive ciertos miembros de la comunidad **#PHP** en el servidor IRC freenode (irc.freenode.net) argumentan que usan PHP por razones de trabajo pero que en realidad no les gusta tanto el lenguaje.

A final de cuentas, cada lenguaje tiene sus ventajas y desventajas, pero compararlas es como comparar **manzanas** y **naranjas**. El sabor es distinto, la textura es distinta, la velocidad de consumo puede variar un poco (o notablemente) sin embargo los dos sirven para lo mismo: Alimentarse.

Como ejemplo: Yahoo usa PHP, Google usa Python. Los dos corren de manera adecuada, y cumplen con las necesidades de los usuarios. Por lo tanto, los dos son lenguajes de programación aceptables.

3.3 10 razones para usar PHP

3.3.1 La Comunidad PHP

PHP tiene una comunidad muy grande de desarrolladores, existen miles de lugares donde se pueden encontrar: documentación, tutoriales, ejemplos de código, foros. Si se tiene un problema con PHP puedes encontrar la respuesta en muchos sitios en donde los usuarios comparten el conocimiento adquirido en el proceso de desarrollo.

3.3.2 Aprender PHP es fácil

PHP es fácil de aprender comparado con otros lenguajes de programación. El lenguaje es semejante a C y Java pues la sintaxis primaria esta basada en Perl. Además si conoces *Javascript* o *ActionScript* verás la semejanza entre estos lenguajes por ejemplo en sus estructuras de control. Otro punto es que PHP tiene librerías especializadas en determinados trabajos por lo cual solo necesitas conocer la sintaxis, aplicarla y lograrás grandes resultados.

3.3.3 Rendimiento

El rendimiento de PHP es muy bueno y verdaderamente eficiente, utilizando un servidor modesto puedes atender millones de peticiones al día. Además de ello si necesitas mejorar este rendimiento Zend Technologies ha desarrollado versiones especiales para incrementar este rendimiento.

3.3.4 Bajo costo

El precio para utilizar PHP es cero, PHP es *gratuito* y lo puedes descargar desde www.php.net. Incluso si contratas un hosting verás que sale mas barato uno con soporte *PHP* comparado con el que tiene soporte *ASP* o *ASP.NET*.

3.3.5 Es Open Source, lo puedes modificar

PHP es *Open Source* es decir que se tiene acceso al código fuente. Si deseas agregar o modificar algo para obtener un funcionamiento de acuerdo a tus necesidades puede hacerlo con total libertad. Esto a diferencia de las aplicaciones comerciales en las cuales solo queda esperar versiones mejoradas de la empresa desarrolladora. Este punto es importante también pues teniendo acceso al código

miles de desarrolladores detectan *bugs* y van corrigiendo y mejorando PHP, logrando tener una aplicación muy segura y constantemente mejorada.

3.3.6 Librerías Incluidas

PHP fue diseñada para trabajar sobre la web por ello trae un conjunto muy amplio de funciones para ser utilizadas en diferentes tares relacionadas con la web. Se puede conectar con bases de datos, conectar a *web services*, parsear XML, enviar email, generar PDFs, generar imágenes, etc. Basadas en estas librerías existes clases implementadas para facilitar el trabajo de los desarrolladores. Otro punto es que hay desarrolladores que agregan librerías especializadas para extender las funcionalidades de PHP.

3.3.7 Portabilidad

PHP esta disponible para la mayoría de sistemas operativos existentes. Desde Unix, Linux, Microsoft Windows, MAC, entre otros. Una vez desarrollado tu aplicación PHP esta puede funcionar cualquiera de estos sistemas operativos sin necesidad de modificar el código.

3.3.8 Soporte para OOP

La versión 5 de PHP esta diseñada para soporte de características de *programación orientada a objetos*. Características como herencia, métodos y atributos públicos o privados, clases y métodos abstractos, constructores, interfaces y destructores. Si tienes conocimientos de *C++* o *Java* estas características te serán muy familiares con una sintaxis muy similar.

3.3.9 Soporte para gran variedad de Bases de Datos

PHP tiene soporte para conectarse a una gran variedad de base de datos como: MySQL, PostgreSQL, mSQL, Oracle, dbm, FilePro, HyperWave, Informix, InterBase, Sybase entre otras. Las base de datos hacen que una aplicación sea mas robusta y con este soporte tu aplicación puede conectarse con facilidad a tu base de datos existente.

3.3.10 Soporte

Si lo que necesitas es soporte, Zend Technologies la empresa que patrocina PHP, ofrece versiones comerciales con todo el soporte que puedas necesitar.

4 Prototipo

4.1 Objetivo

Desarrollar un prototipo que satisfaga las necesidades del cliente este con el fin de esclarecer cuales de las necesidades son factibles de solucionar con la implementación de un sistema de información.

4.2 Índex

Ilustración 6 - pagina Índex

Esta página no cuenta con ninguna funcionalidad especifica.

4.3 Login

Ilustración 7 - Pagina de login

Esta página permite al usuario el ingreso a información reservada para usuarios gerenciales, esta información es:

- Lista de usuarios registrados en el sistema.
- Reporte de mercadeo de la lavandería.
- Reporte de ventas de la lavandería.
- Reporte de contabilidad de la lavandería.

4.3.1 Descripción de los campos

Campo	Descripción

usuario	Campo en el cual se ingresa el nombre del usuario que ingresara a la zona de información reservada.
password	Contraseña del usuario que ingresara a la zona de información reservada.
Ingresar (botón)	Este botón permite validar la información del usuario al presionarlo y de ser satisfactorio le permite ingresar a la zona de información reservada. En caso que la validación no autorice al usuario entonces será re direccionado a una página de error.

4.3.2 Errores

Ilustración 8 - pagina de errores

Esta página muestra un mensaje al usuario del error de la aplicación.

4.3.3 Descripción de los campos

Campo	Descripción
Mensaje de error	Este mensaje especifica el error que ocurrió con la aplicación en el momento que el usuario realizo una acción sobre el mismo.

4.4 Listar Clientes

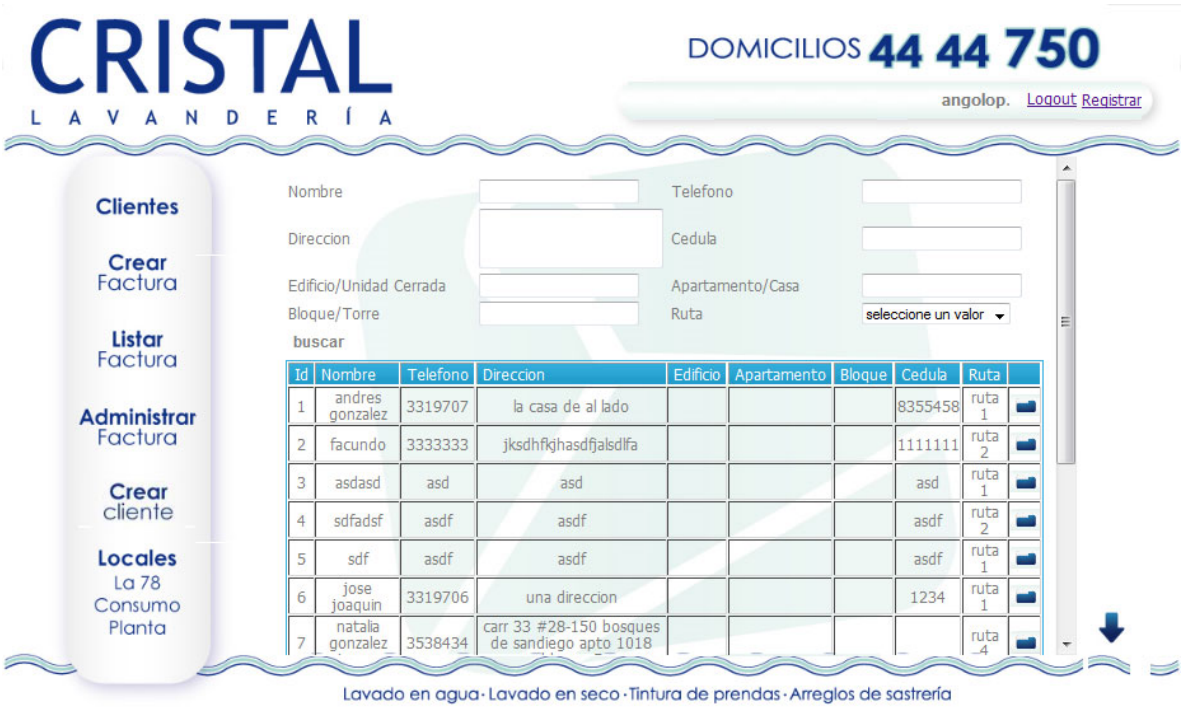

Ilustración 9 - Pagina lista de clientes

Esta pagina permite por medio de unos filtros establecidos por el usuario buscar los registros que cumplan con los mismos ya sea uno especifico o varios que

cumplen con el criterio de búsqueda, esta pagina utiliza un método para paginar los registros devueltos de la base de datos cuando estos sean un numero mayor a 15 registros por consulta.

4.4.1 Descripción de los campos

Campo	Descripción
Nombre	Nombre del cliente que desea buscar.
Teléfono	Teléfono del cliente que desea buscar.
Dirección	Dirección del cliente que desea buscar.
Cedula	Cedula del cliente que desea buscar.
Edificio	Nombre del edificio del cliente que desea buscar.
Apartamento	Numero del apartamento del cliente que desea buscar.
Bloque	Numero del bloque del cliente que desea buscar.
Ruta	Numero de la ruta que le recoge la ropa al cliente.
Consultar (botón)	Este botón permite hacer la consulta de todos los clientes que cumplan con los parámetros establecidos en los criterios de búsqueda.

4.5 Crear factura

Ilustración 10 - Ingreso a la pagina Crear Factura

Para acceder a esta opción el usuario tiene la opción de seleccionar, desde la página de listado de clientes, un cliente especifico al cual se le desea crear una factura o también seleccionar la opción desde el menú y ya en la pagina de la creación de la factura el usuario tiene que seleccionar el usuario al cual se le desea crear la factura.

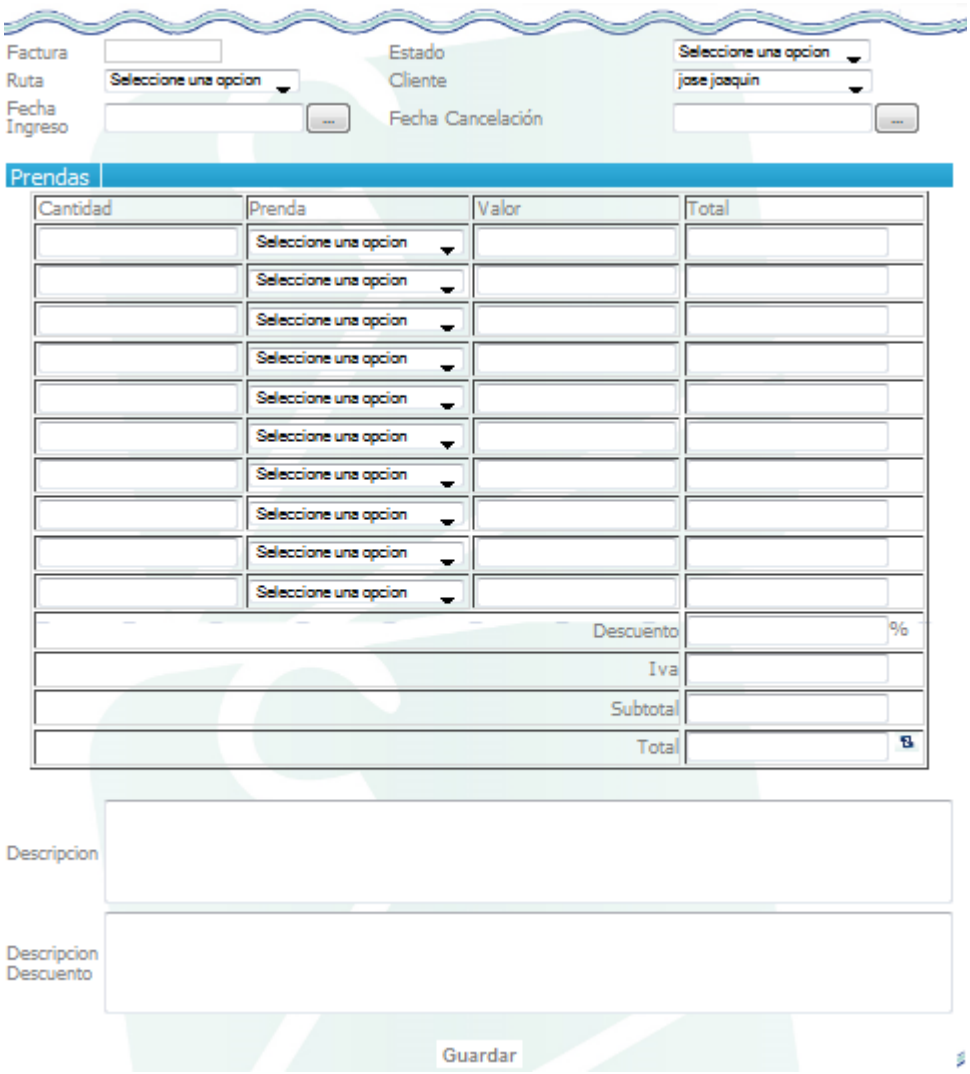

Ilustración 11 - Pagina Crear Factura

Esta pagina permite al usuario crear una nueva factura dentro del sistema de información SILC, en la cual además de la información básica de la factura el usuario también tiene como opción ingresar los detalles dela misma como lo son las prendas relacionadas con la misma, las cantidades de cada una de las referencias escogidas, los descuentos con los que cuenta el usuario, el total de la facturación, además que de cuanto corresponde al IVA como tal de la factura.

4.5.1 Descripción de los campos

Campo	Descripción
Factura	En este campo se ingresa manualmente el número de la factura.
Estado	Lista de posibles estados que puede tener una factura dentro del sistema de información SILC. Este campo es una lista desplegable.
Ruta	Descripción de la ruta que tiene a cargo el cliente seleccionado para la creación de la factura. Este campo es una lista desplegable.
Cliente	Descripción de los clientes registrados en el sistema de información SILC que pueden seleccionarse para la creación de la factura. Este campo es una lista desplegable.
Fecha de ingreso	Este campo permite al usuario ingresar la información de la fecha en la cual se ingreso la factura al sistema de información SILC. Este campo utiliza un calendario y reloj para la selección de la fecha y hora exacta de registro.
Fecha de cancelación	Este campo permite al usuario ingresar la información de la fecha en la cual se cancelo la factura al sistema de información SILC. Este campo utiliza un calendario y reloj para la selección de la fecha y hora exacta de registro.
Cantidad de prendas	Este campo permite al usuario registrar el numero de prendas definidas para un tipo de prenda en especifico.
Prenda	Este campo permite al usuario registrar el tipo de prenda que el cliente esta

	entregando a la lavandería.
Valor de las prendas	Este campo permite al usuario registrar el valor individual de la prenda que el cliente esta entregando a la lavandería.
Total de las prendas	Este campo registra automáticamente después de ingresar la información de los campos "cantidad de prendas" y "valor de las prendas", multiplicando estos dos campos y llenando el campo con el resultado de esta operación.
Descuento	Este campo permite al usuario registrar si el cliente cuenta con algún descuento para la factura que se esta llenando, además que tiene que especificar el valor porcentual por el cual la persona recibe el descuento.
IVA	Este campo se registra al hacer clic en el botón actualizar de la aplicación, en donde se calcula el IVA resultante de la suma de todas prendas registradas en la factura.
Subtotal	Este campo se registra al hacer clic en el botón actualizar de la aplicación, en donde se calcula el subtotal resultante de la suma de todas prendas registradas en la factura menos el valor del IVA.
Total	Este campo se registra al hacer clic en el botón actualizar de la aplicación, en donde se calcula el total resultante de la suma de todas prendas registradas en la factura más el valor del IVA.
Descripción	Este campo permite ingresar al usuario una breve descripción de particularidades de

	factura como tal.
Descripción descuento	Este campo permite ingresar al usuario una breve descripción de la razón por la cual el cliente recibe un descuento sobre una factura o prenda.

4.6 Crear cliente

A esta opción se puede ingresar desde el menú del sistema de información, esta permite ingresar nueva información de los clientes en la base de datos del sistema de información SILC.

Ilustración 12 - Pagina Crear Cliente

4.6.1 Descripción de los campos.

Campo	Descripción
Nombre	En este campo el usuario ingresa el nombre del cliente que quedara registrado en el sistema de información SILC.
Teléfono	En este campo el usuario ingresa el teléfono del cliente que quedara registrado en el sistema de información SILC.
Dirección	En este campo el usuario ingresa la dirección del cliente que quedara registrado en el sistema de información SILC.
Edificio / Unidad cerrada	En este campo el usuario ingresa el nombre de edificio / unidad cerrada del cliente que quedara registrado en el sistema de información SILC.
Apartamento / Casa	En este campo el usuario ingresa el numero del apartamento / casa del cliente que quedara registrado en el sistema de información SILC.
Bloque / Torre	En este campo el usuario ingresa el numero del bloque / torre del cliente que quedara registrado en el sistema de información SILC.
Cedula	En este campo el usuario ingresa la cedula de ciudadanía del cliente que quedara registrado en el sistema de información SILC.
Ruta	En este campo el usuario ingresa la ruta según el vendedor con la cual el cliente quedara registrado en el sistema de

	información SILC.

4.7 Informe de Prendas

4.7.1 Descripción de los campos.

Campo	Descripción
Fecha inicial	Este campo permite al usuario filtrar la información de las prendas según la fecha en la que se ingreso la prenda dentro del sistema.
Fecha final	Este campo permite al usuario filtrar la información de las prendas según la fecha en la que se ingreso la prenda dentro del sistema.
Ruta	Este campo permite al usuario filtrar la

	información de las prendas según la ruta para las que se ingreso la prenda dentro del sistema.
Prenda	Este campo permite al usuario filtrar la información de las prendas según el tipo ingresado en el sistema.
Cantidad de prendas	Este campo permite al usuario filtrar la información de las prendas según la cantidad ingresada en el sistema.
Cliente	Este campo permite al usuario filtrar la información de las prendas según el cliente al que se le ingresada en el sistema.
Buscar (botón)	Este botón permite el usuario buscar los registros en el sistema que cumplan con los parámetros de búsqueda.
Id factura	Id que tiene la factura para la cual se registro la prenda.
Numero factura	Numero que tiene la factura para la cual se registro la prenda.
Cantidad	Cantidad de prendas que se registraron para una factura en particular.
Prenda	Tipo de prenda que se registro para una factura en particular.
Cliente	Cliente que se registro para una factura en particular.
Fecha ingreso	Fecha de ingreso de la factura en la cual se registro la prenda.
Valor	Valor de la prenda que se registro para una factura en particular.
Total	Valor total de la factura en la cual se registro la prenda.

5 Conclusiones

1. No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto (recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema, etc.).

2. La lluvia de ideas que se genera con la programación en parejas crea diseños de mayor calidad en menor tiempo.

3. Un equipo de *software* puede reducirse en tamaño sin reducir su productividad total.

4. Debe existir una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

5. PHP es una opción excelente para la implementación de proyectos web porque consume pocos recursos, los scripts corren considerablemente rápido **cuando están bien desarrollados**, incluye orientación a objetos y se deja depurar.

6. La complejidad de los negocios provoca la necesidad de integrar funciones dentro de la empresa y su entorno dinámico.

7. El contar con infraestructura adecuada que garantice agilidad o rapidez en el procesamiento de datos conlleva a tomar decisiones oportunas para resolver los retos que se presentan, en ocasiones, inesperadamente.

8. La metodología XP fue empleada en respuesta del problema de cambio de requerimientos. Debido a que es muy frecuente que el cliente no tenga una clara idea de lo que el sistema debe hacer. Se tiene un sistema cuya funcionabilidad cambia cada pocos meses. En muchos sistemas la única constante es el cambio dinámico de los requerimientos.

Anexo A – Prototipo de la aplicación

Anexo B – Código de la base de datos

```
-- PhpMyAdmin SQL Dump
-- Version 3.2.4
-- http://www.phpmyadmin.net
--
-- Servidor: local host
-- Tiempo de generación: 17-11-2010 a las 22:44:03
-- Versión del servidor: 5.1.47
-- Versión de PHP: 5.3.1

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";


/*! 40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*! 40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*! 40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Base de datos: `cristaldb`
--


-- --------------------------------------------------------


--
-- Estructura de tabla para la tabla `cliente`
```

```
--

CREATE TABLE IF NOT EXISTS `cliente` (
  `ID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `NOMBRE` varchar(100) NOT NULL,
  `TELEFONO` varchar(45) DEFAULT NULL,
  `DIRECCION` varchar(255) DEFAULT NULL,
  `EDIFICIO` varchar(100) DEFAULT NULL,
  `APARTAMENTO` varchar(20) DEFAULT NULL,
  `BLOQUE` varchar(10) DEFAULT NULL,
  `CEDULA` varchar(45) DEFAULT NULL,
  `ID_RUTA` int(10) unsigned NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `FK_cliente_1` (`ID_RUTA`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=14 ;


--
-- Estructura de tabla para la tabla `estados`
--

CREATE TABLE IF NOT EXISTS `estados` (
  `ID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `NOMBRE` varchar(45) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;
```

```
--
-- Estructura de tabla para la tabla `factura`
--

CREATE TABLE IF NOT EXISTS `factura` (
  `ID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `NUMERO_FACTURA` varchar(20) DEFAULT NULL,
  `IVA` float(15,1) NOT NULL,
  `SUBTOTAL` float(15,1) NOT NULL,
  `TOTAL` float(15,1) NOT NULL,
  `DESCUENTO` float(3,1) NOT NULL,
  `OBSERVACIONES` varchar(400) NOT NULL,
  `OBSERVACIONES_DES` varchar(400) NOT NULL,
  `FECHA_INGRESO` datetime NOT NULL,
  `FECHA_CANCELACION` datetime DEFAULT NULL,
  `ID_CLIENTE` int(10) unsigned NOT NULL,
  `ID_ESTADO` int(10) unsigned NOT NULL,
  `ID_VENDEDOR` int(10) unsigned NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `FK_factura_2` (`ID_ESTADO`),
  KEY `FK_factura_3` (`ID_VENDEDOR`),
  KEY `ID_CLIENTE` (`ID_CLIENTE`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=36 ;



--
-- Estructura de tabla para la tabla `historicos_parametros`
--
```

```
CREATE TABLE IF NOT EXISTS `historicos_parametros` (
  `ID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `NOMBRE` varchar(45) NOT NULL,
  `VARIABLE1` varchar(45) NOT NULL,
  `VARIABLE2` float NOT NULL,
  `FECHA` datetime NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

--
-- Estructura de tabla para la tabla `pedidos`
--

CREATE TABLE IF NOT EXISTS `pedidos` (
  `ID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `ID_FACTURA` int(10) unsigned NOT NULL,
  `ID_PRENDA` int(10) unsigned NOT NULL,
  `VALOR` float(10,1) NOT NULL,
  `CANTIDAD` int(10) unsigned NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `FK_pedidos_2` (`ID_FACTURA`),
  KEY `FK_pedidos_3` (`ID_PRENDA`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=29 ;



--
-- Estructura de tabla para la tabla `prendas`
```

```
--

CREATE TABLE IF NOT EXISTS `prendas` (
  `ID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `NOMBRE` varchar(45) NOT NULL,
  `DESCRIPCION` varchar(250) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=18 ;


--
-- Estructura de tabla para la tabla `rutas`
--

CREATE TABLE IF NOT EXISTS `rutas` (
  `ID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `NOMBRE` varchar(45) NOT NULL,
  `DESCRIPCION` varchar(450) NOT NULL,
  `ID_VENDEDOR` int(10) unsigned NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `FK_rutas_1` (`ID_VENDEDOR`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=11 ;


--
-- Estructura de tabla para la tabla `usuarios`
--

CREATE TABLE IF NOT EXISTS `usuarios` (
```

```
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `usuario` varchar(20) NOT NULL,
  `password` varchar(10) NOT NULL,
  `descripcion` text CHARACTER SET utf8 COLLATE utf8_spanish_ci,
  `email` varchar(45) CHARACTER SET utf8 COLLATE utf8_spanish_ci DEFAULT NULL,
  `fecha` date NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;



--
-- Estructura de tabla para la tabla `vendedor`
--

CREATE TABLE IF NOT EXISTS `vendedor` (
  `ID` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `NOMBRE` varchar(45) NOT NULL,
  `CEDULA` int(10) unsigned NOT NULL,
  `PERFIL` varchar(5) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;



--
-- Filtros para las tablas descargadas (dump)
--

--
-- Filtros para la tabla `cliente`
```

```
--

ALTER TABLE `cliente`

  ADD CONSTRAINT `FK_cliente_1` FOREIGN KEY (`ID_RUTA`) REFERENCES `rutas` (`ID`);



--

-- Filtros para la tabla `factura`

--

ALTER TABLE `factura`

  ADD CONSTRAINT `factura_ibfk_1` FOREIGN KEY (`ID_CLIENTE`) REFERENCES `cliente`
(`ID`),

  ADD CONSTRAINT `FK_factura_2` FOREIGN KEY (`ID_ESTADO`) REFERENCES `estados`
(`ID`),

  ADD CONSTRAINT `FK_factura_3` FOREIGN KEY (`ID_VENDEDOR`) REFERENCES `vendedor`
(`ID`);



--

-- Filtros para la tabla `pedidos`

--

ALTER TABLE `pedidos`

  ADD CONSTRAINT `FK_pedidos_2` FOREIGN KEY (`ID_FACTURA`) REFERENCES `factura`
(`ID`),

  ADD CONSTRAINT `FK_pedidos_3` FOREIGN KEY (`ID_PRENDA`) REFERENCES `prendas`
(`ID`);



--

-- Filtros para la tabla `rutas`

--

ALTER TABLE `rutas`

  ADD CONSTRAINT `FK_rutas_1` FOREIGN KEY (`ID_VENDEDOR`) REFERENCES `vendedor`
(`ID`);


/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

```
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Anexo C – Manual de instalación

Anexo D – Manual de usuario

6 Bibliografía

6.1 Clásica

- BECK, Kent. *Una explicación de la programación extrema. Aceptar el cambio* España 2002, Ed. Pearson Educación S.A. . ISBN 84-7829-055-9.

- BOOCH, Grady. *Análisis y diseño orientado a objetos con aplicaciones* 2ª ed. E.E.U.U 1996, Ed. Addison-Wesley Iberoamericana, S.A. ISBN 0-201-60122-2.

- HELLRIEGEL, JACKSON Y SLOCUM. *Administración, un enfoque basado en competencias*. 9ª. ed. México: International Thompson Editores, S.A. ISBN 970-686-197-1.

- ROJAS, Claudia, *Curso de Análisis y Diseño 1*, notas del Curso 2002 U.S.A.C. Guatemala.

- SENN, James. *Análisis y diseño de sistemas de información*. 2a ed. México: Ed. McGraw Hill 2002. ISBN 968-422-991-7.

- Welling, Luke; Thompson, Laura. *Desarrollo web con php y mysql*. 1ª ed. España 2005: Anaya Multimedia. ISBN 844-151-818-1.

- Pavón, J. *Creación de un portal con php y mysql*. 2a ed. España 2006: Ra-ma. ISBN: 847-897-690-6

6.2 Internet

- Anonimo@2010 Tomado de la página de modelado de procesos de SAP, web: *http://help.sap.com/saphelp_40b/helpdata/es/4d/2b8cde43ad11d189410000e829fbbd/content.htm*.

- Anónimo @2005 tomado de la página de opinión krillz, web: *http://www.krillz.com/10-reasons-why-i-use-php/*.

- ERP al alcance de las Pymes, Jaime Guerrero, consultado; 16/08/2010; *http://www.gestiopolis.com/recursos/documentos/fulldocs/ger1/erppymes.htm*

- "Importancia de los sistemas de información en las áreas administrativas", de la pagina: *http://www.monografias.com/trabajos27/importancia-sistemas/importancia-sistemas.shtml*, consultado: 13/08/2010

7 Programas utilizados

- EXCEL. Microsoft® Excel 2007. Copyright © Microsoft Corporation, USA, 1985 – 2010.
- WORD. Microsoft® Word 2007. Copyright © Microsoft Corporation, USA, 1985 – 2010.
- FIREFOX. Mozilla® Firefox 2010. Copyright © Mozilla Foundation, USA, 1998 – 2010.
- XAMPP. Apache Friends XAMPP (Basis Package) version 1.7.3
- ASTAH* COMMUNITY 6.2.1 (Model Version: 33). Copyright © 2006-2010 Change Vision, Inc.
- NOTEPAD++ v 5.7 GNU General Public License.