PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Sistema de apoyo a la administración de bar

Rodrigo Manuel Cacciuttolo Gatica

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO DE EJECUCIÓN EN INFORMÁTICA

Diciembre 2008

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

Sistema de apoyo a la administración de bar

Rodrigo Manuel Cacciuttolo Gatica

Profesor Guía: **Iván Mercado Bermúdez**
Profesor Co-referente: **Cristian Alexandru Rusu**
Carrera: **Ingeniería de Ejecución en Informática**

Dedico este logro primero que todo a Dios, que me dio la fe para hacer lo imposible, a mis Padres, que, sin ellos, no habría podido realizar mis metas y llegar hasta este instante tan importante de mi vida y por último a la Universidad que me dio las facilidades en todo sentido para poder permanecer en ella y sacar mi carrera adelante.

Gracias...

Rodrigo Cacciuttolo.

Resumen

Este trabajo de título, se lleva a cabo mediante la realización de un sistema que permita, en una empresa que funciona como Pub- discoteca, registrar y controlar el flujo de ventas en lo concerniente a tragos y demás, con el fin de evitar fugas de recursos y mantener un orden en lo que se refiere a transacciones; para lo cual fue solicitado que el arqueo de cajas y actualización de inventario fuese realizado de forma automática; es más, se pide que el sistema, dé aviso de los licores que deben ser comprados ante criterios pre-establecidos bajo las políticas que rigen el negocio. Se debe decir que dicho establecimiento cuenta con tres bares que funcionan en forma independiente y que el inventario de bodega es único para los tres.

Abstract

The title work, is performed by carrying out a system that would, in a company that operates as pub, record and monitor the sales pipeline in concerning the drinks and the like, to prevent leakage of resources and maintain order regard to transactions, for which it was requested that the tonnage of boxes and inventory update was done automatically and, moreover, calls for the system, give notice of the liquor must be purchased with pre-established criteria under the policies that govern the business. It must be said that the establishment has three bars that operate independently and that the warehouse inventory is unique to the three.

INDICE DE CONTENIDOS

INDICE DE IMÁGENES

INDICE DE TABLAS

Capítulo 1

Estudio preliminar del Sistema.

La fase de inicio es la primera fase del Proceso Unificado de Desarrollo de Software, paradigma seleccionado para el desarrollo del 'Sistema De Apoyo a Administración de Bar'. Como fase inicial, su objetivo principal es profundizar en la idea general sobre lo que se desea construir, esto, a través del análisis de las reglas del negocio (ver 1.2 Actualidad y problemática en la Organización), lo que permitirá justificar el posterior desarrollo de un Sistema o Solución Informática, identificando los diversos objetivos a alcanzar y los requerimientos necesarios para el logro de estos objetivos.

1.1 Estudio de la Organización.

1.1.1 Descripción de la Organización.

Si se habla de locales para diversión en la ciudad de La Calera, se debe detener un momento en el pub "CHIVATO", pues desde que fue inaugurado el día 24 de Marzo de 2006 es el lugar con mayor afluencia de público en dicha localidad, es más, la cantidad de público que asiste a dicho local ha ido en aumento, debido a la calidad del servicio que presta, que en comparación con los demás locales de la zona es destacable. Dicho local cuenta con servicio de bar, discoteca, y ofrece distintos servicios adicionales como estacionamiento publicación de fotografías y promociones de fiestas vía Internet, mismas fiestas o eventos que son organizados por el local.

1.2 Actualidad y problemática en la organización

1.2.1 Descripción de los procedimientos actuales.

A continuación se explicará la obtención y manejo de la información relevante para el administrador, necesaria tanto para un mejor control dentro del local, como también para las exigencias impuestas para seguir liderando el negocio en la localidad y sus alrededores.

- El procedimiento que se sigue para la venta de entradas, que es realizado por un encargado de la venta de ticket que son revisados en la entrada del local por guardias de seguridad que forman parte del personal de la empresa, cabe destacar que si bien el control de estos ticket no corresponde a una exigencia del sistema es algo que lo afecta pues este ticket va acompañado de un vale extendido para su canje por un trago dentro del local, éste proceso por el momento es visto como una excepción al funcionamiento normal del proceso de venta de tragos.
- El procedimiento por el cual se expende los productos a los clientes, que es realizado en su conjunto por un cajero y un barman, recibiendo el cajero por parte del cliente la información acerca del producto que desea y el dinero en pago por éste; luego, extendiendo este un vale que contiene la información necesaria para el barman y además se hace entrega de el documento legal que permite que la venta sea efectuada, que en éste caso es una boleta, luego el cliente hace llegar al barman el vale que contiene el nombre del producto que desea obtener, mismo que en base a ésta información ejecuta las acciones pertinentes para el cumplimiento de lo encomendado entregando el producto al cliente previa solicitud de aprobación (conformidad por parte del cliente acerca del producto solicitado).
- Proceso de actualización de stock de barras, este proceso es realizado por el administrador y el barman a cargo de la barra en cuestión, dando este ultimo un listado con lo que supone basado en su experiencia debe tener en las existencias para poder satisfacer los pedidos que serán efectuados por los clientes durante la jornada que comenzará, cabe destacar que este procedimiento se realiza el día mismo en que se comienza la atención. El administrador una vez que conoce el listado de pedido que le proporciona el barman procede a surtir de éste con los productos solicitados de la bodega, éste proceso queda documentado en un formulario que pretende ser algo como control de inventario (anexo Nº 1).
- Proceso de actualización de stock en bodega, este proceso es realizado por alguno de los socios, en este caso la mayor parte de las ocasiones es realizado por el socio quien cumple el rol de administrador, y se realiza en base a su experiencia en el negocio y es realizado sin un inventario actualizado, vale decir que no se documentan las existencias hasta que no sea estrictamente necesario, como por ejemplo por necesidades contables o algo así.
- El procedimiento para el pago de remuneraciones a cajeros, este es realizado en conjunto entre el cajero y el administrador en base a las boletas emitidas y el dinero en efectivo que posee en su poder el encargado de la caja en cuestión, se procede a cuadrar la suma de las boletas emitidas durante el día con el dinero y luego se efectúa

el pago por parte del administrador, cabe destacar que en caso de falta de dinero, este se descuenta de la remuneración del empleado.

1.2.2 Situación actual y problemas detectados.

En el presente ninguno los procesos descritos anteriormente esta automatizado los registros al estar solamente en papel generan un alto grado de inseguridad de los datos, ya que estos pueden ser extraviados y no hay medio de respaldo para ello. Sin tomar en cuenta el proceso de actualización de stock de bodega el cual ni siquiera es documentado.

Hoy en día el administrador no reutiliza la información obtenida de periodos anteriores ya que esto no es factible por la cantidad de tiempo que esto conlleva, por lo cual siempre hacen las cosas como si fuera la primera vez.

Cada vez que el administrador necesita algún documento o datos para generar algún informe este tiene que elaborarlo al menos con una semana de anticipación para poder hacer uso de el, lo cual es un proceso tedioso que se hace con frecuencia.

1.3 Solución propuesta.

En vista de los problemas presentados en la empresa, se plantea como solución la construcción de un Prototipo de Sistema de Gestión que logre terminar con la mayoría de los problemas existentes detallados en el punto anterior. La alternativa planteada es la construcción de un sistema que maneje una base de datos y permita realizar todos los procesos descritos de manera rápida y efectiva.

1.4 Objetivos del Proyecto.

1.4.1 Objetivo General.

Desarrollar un sistema informático que gestione los datos de importancia para la administración de la empresa “CHIVATO”, permitiendo la obtención segura y rápida y el acceso rápido y confiable a los datos; para apoyar una buena administración de recursos, tanto humanos como insumos.

1.4.2 Objetivos Específicos.

Para poder cumplir con el objetivo general del proyecto se definieron los siguientes objetivos específicos:

- Estudio de la operación del Pub “El chivato”, Con el fin de detectar falencias.
- Diseñar el modelo de una solución válida y factible a la problemática detectada en el cliente.
- Llevar a cabo la implementación de un prototipo al bajo el modelo previamente diseñado.
- Validar que el prototipo construido cumpla con todos los requerimientos obtenidos.

Capítulo 2

Análisis y Diseño del Sistema.

En el análisis del sistema comienza con la identificación de Necesidades, este proceso comienza con reuniones por parte del analista y el cliente, donde identifican las metas globales, se analiza las perspectivas del cliente, sus necesidades y requerimientos, sobre la planificación temporal y presupuestal.

Se suele llamar a esta parte ¨ Análisis de Requerimientos ¨ y algunos autores lo dividen en cinco partes:

- Reconocimiento del problema.
- Evaluación
- Modelado.
- Especificación.
- Revisión.

En teoría antes de su reunión con el analista, el cliente prepara un documento conceptual del proyecto, aunque generalmente se elabora durante reuniones Cliente – analista, ya que de hacerlo el cliente solo de todas maneras tendría que ser modificado, durante la identificación de las necesidades.

Muchas veces cuando se emprende el desarrollo de un proyecto de Sistemas los recursos y el tiempo no son realistas para su puesta en marcha sin tener pérdidas económicas y frustración profesional. La viabilidad y el análisis de riesgos están relacionados de muchas maneras, si el riesgo del proyecto es alto, la viabilidad de producir un software de calidad se reduce, sin embargo se deben tomar en cuenta cuatro áreas principales de interés:

- Viabilidad económica: Básicamente es la evaluación de los costos de desarrollo, comparados con los ingresos netos o beneficios obtenidos del producto o Sistema desarrollado.
- Viabilidad Técnica: Un estudio de funciones, rendimiento y restricciones que puedan afectar la realización de un sistema aceptable.
- Viabilidad Legal: Es determinar cualquier posibilidad de infracción, violación o responsabilidad legal en que se podría incurrir al desarrollar el Sistema.

Lo respectivo al Diseño de Sistemas se define el proceso de aplicar ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso o un Sistema, con suficientes detalles como para permitir su interpretación y realización física.

La etapa del Diseño del Sistema encierra tres etapas:

El Diseño de los datos: las estructuras de datos necesarios para implementar el Software.

El Diseño de la Interfaz: Describe como se comunica el software consigo mismo, con los sistemas que operan junto con él y con los operadores y usuarios que lo emplean.

El Diseño de procedimientos: Transforma elementos estructurales de la arquitectura del programa. La importancia del diseño del software se liga fuertemente a la Calidad del producto, dentro del diseño es donde se promueve la calidad del Proyecto. El Diseño es la única manera de plasmar con precisión los requerimientos del cliente.

El Diseño del Software es un proceso y un modelado a la vez. El proceso de Diseño es un conjunto de pasos repetitivos que permiten al diseñador describir todos los aspectos del Sistema a construir.

Cuando se va a diseñar un Sistema se debe tener presente que el proceso de un diseño incluye, concebir y planear algo en la mente, así como hacer un dibujo o modelo.

2.1 Definición de Requerimientos

Los requerimientos son una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. Debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal. Los requerimientos puedes dividirse en requerimientos funcionales y requerimientos no funcionales.

2.1.1 Características de los requerimientos

Las características de un requerimiento son sus propiedades principales. Un conjunto de requerimientos en estado de madurez, deben presentar una serie de características tanto individualmente como en grupo. A continuación se presentan las más importantes.

Necesario: Un requerimiento es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.

Conciso: Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.

Completo: Un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.

Consistente: Un requerimiento es consistente si no es contradictorio con otro requerimiento.

No ambiguo: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.

Verificable: Un requerimiento es verificable cuando puede ser cuantificado de manera que permita hacer uso de los siguientes métodos de verificación: inspección, análisis, demostración o pruebas [2].

2.1.2 Requerimientos Funcionales

Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Estos son los que describen lo que el sistema debe de hacer. Es importante que se describa el ¿Que? Y no el ¿Como? Estos requerimientos al tiempo que avanza el proyecto de software se convierten en los algoritmos, la lógica y gran parte del código del sistema.

Los requerimientos funcionales del sistema son los siguientes:

- Registrar datos de las ventas, tanto las ventas normales como los ticket que son emitidos bajo condiciones especiales.
- Registrar datos de la entrega de productos en forma directa a los clientes, en el momento en que esta es producida.
- Configurar herramienta para:

- 1.- Automatizar el proceso de arqueo de cajas.
- 2.- Apoyar el proceso de remuneración del personal encargado.
- 3.- Obtener información del dinero entrante entre periodos determinados.

- Mantener actualizada la información de inventario tanto de bodega como de cada barra.
- Realizar algún tipo de aviso, cuando sea necesario realizar compra de productos, para así mantener un stock mínimo en existencias.
- No se permitirá el ingreso de cualquier persona al sistema, esto se restringirá mediante un login y password por cada persona autorizada para utilizar el sistema.
- Cada funcionario tendrá asignada funciones dentro del sistema, las cuales serán definidas por el Administrador.

2.1.3 Requerimientos No Funcionales

Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, o restricciones, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc. En vista de estas características se identifican los siguientes requerimientos no funcionales del sistema:

- El sistema debe permitir que todos los equipos conectados al sistema puedan acceder de igual forma a la base de datos.
- Las interfaces deben permitir un uso fácil y práctico por parte de los usuarios y en vista de la capacidad de trabajo de los equipos computacionales con los que la Empresa cuenta, las interfaces deben ser simples y sencillas.

2.2 Evaluación de Paradigmas de Desarrollo.

Se entiende por paradigma de la Ingeniería de Software al método o estrategia que permite resolver un problema planteado. Los modelos del ciclo de vida o paradigmas de la Ingeniería de Software, estan compuesto por pasos que abarcan los métodos, herramientas y procedimientos, que basándose en los requerimientos, crearán un producto de software.

La elección del paradigma irá de acuerdo con la naturaleza del proyecto y de la aplicación, así también los métodos, herramientas a usar y los controles dependerán también de las entregas requeridas.

Los paradigmas más conocidos y que se han tratado y debatido ampliamente son los que se describen a continuación.

2.2.1 Paradigmas.

1.- Ciclo de Vida Clásico o Cascada: Este paradigma, también conocido como Modelo Lineal Secuencial, demanda un enfoque sistemático y secuencial del desarrollo del software, el seguimiento de este modelo significa transformar, en forma lineal el conjunto completo de requerimientos de manera que en cada paso sea más concreta, es decir, una etapa no se inicia si la anterior no se ha terminado [3], en la figura 1 se pueden observar las diferentes etapas de este paradigma.

El paradigma del ciclo de vida clásico abarca las siguientes actividades:

- Ingeniería y Análisis del Sistema.
- Análisis de los Requisitos del Software.
- Mantenimiento

Este paradigma presenta una serie de ventajas y desventajas, las que se pueden considerar en "Managing the Development of Large Software System" 1970 de la editorial Wescon.

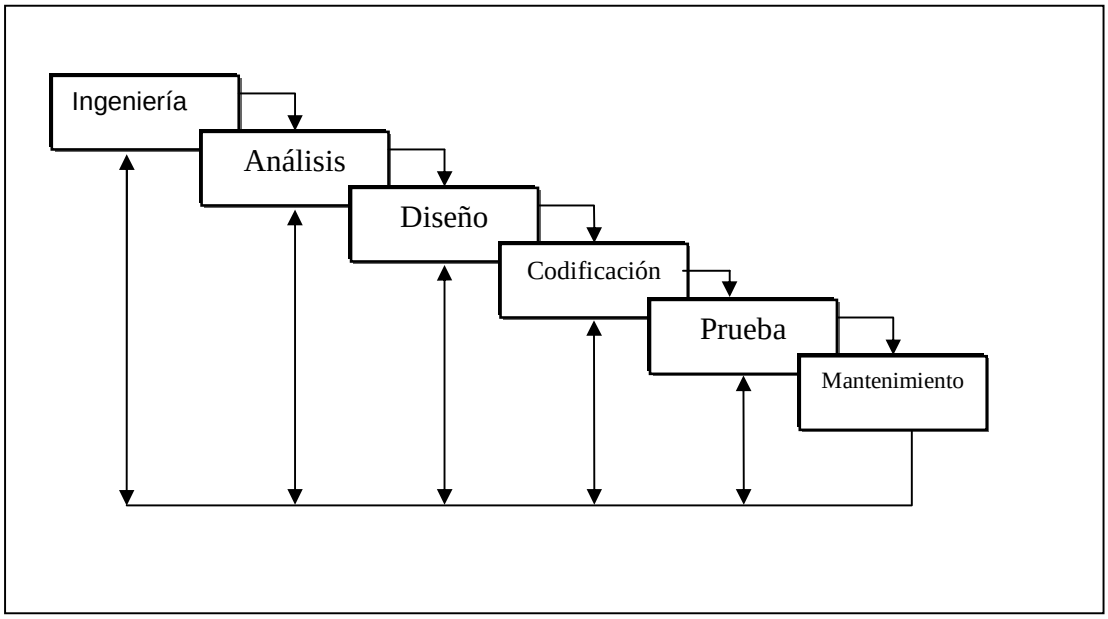


Figura 1: Paradigma Clásico o Cascada.

2.- Construcción de Prototipos: Este paradigma se basa primero en la producción operacional del software. La construcción de prototipos facilita al programador la creación de un modelo de software a construir. Es útil cuando el cliente no tiene claros los requisitos que necesita o estos no son específicos, en la figura 2 se pueden observar las partes de este paradigma.

Dentro de este modelo existen dos metodologías a considerar:

- Desechable: en el cual el prototipo es desechado y no es usado en el producto final.
- Evolutivo: una parte o el total del prototipo es usado en el producto entregado.

Entre las diferentes ventajas y desventajas que presenta este modelo se pueden considerar en "Managing the Development of Large Software System" 1970 de la editorial Wescon.

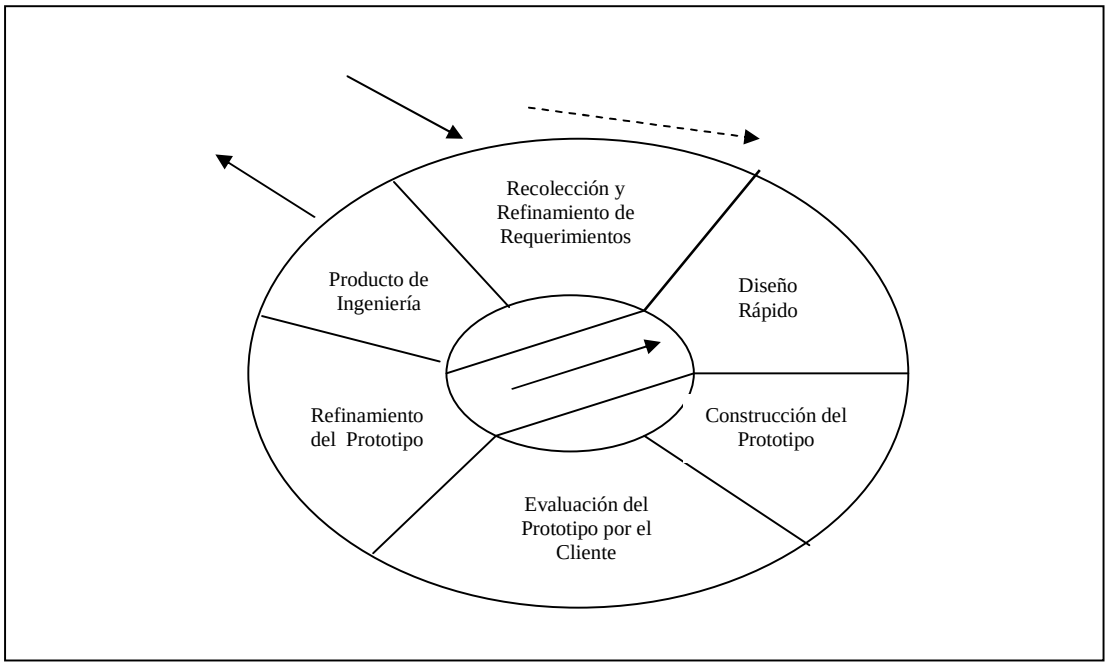


Figura 2: Construcción de Prototipos.

3.- Modelo Espiral: El modelo espiral ha sido desarrollado para cubrir las mejores características tanto del Ciclo de Vida Clásico, como la Construcción de Prototipos, añadiendo un nuevo elemento: el análisis de riesgo, que no se encuentra en los paradigmas antes mencionados, el modelo espiral se puede apreciar en la figura 3.

Este modelo define seis actividades principales:

- Comunicación con el cliente.
- Planificación.
- Análisis de riesgos.
- Ingeniería.
- Construcción y adaptación.
- Evaluación del cliente.

Ventajas que este paradigma posee:

- Demanda una consideración directa de los riesgos técnicos en todas las etapas del proyecto, con el fin de reducirlos antes de que se conviertan en problemas.
- Permite la creación de prototipos como mecanismo para la reducción del riesgo. A demás permite utilizar el enfoque de creación de prototipo en cualquier etapa de evolución del producto.

Desventajas que este paradigma posee:

- Es un modelo que no se ha utilizado tanto como los modelos anteriores, por lo que no se puede determinar con certeza la eficacia de este paradigma.
- Requiere de una considerable habilidad para la valoración del riesgo.

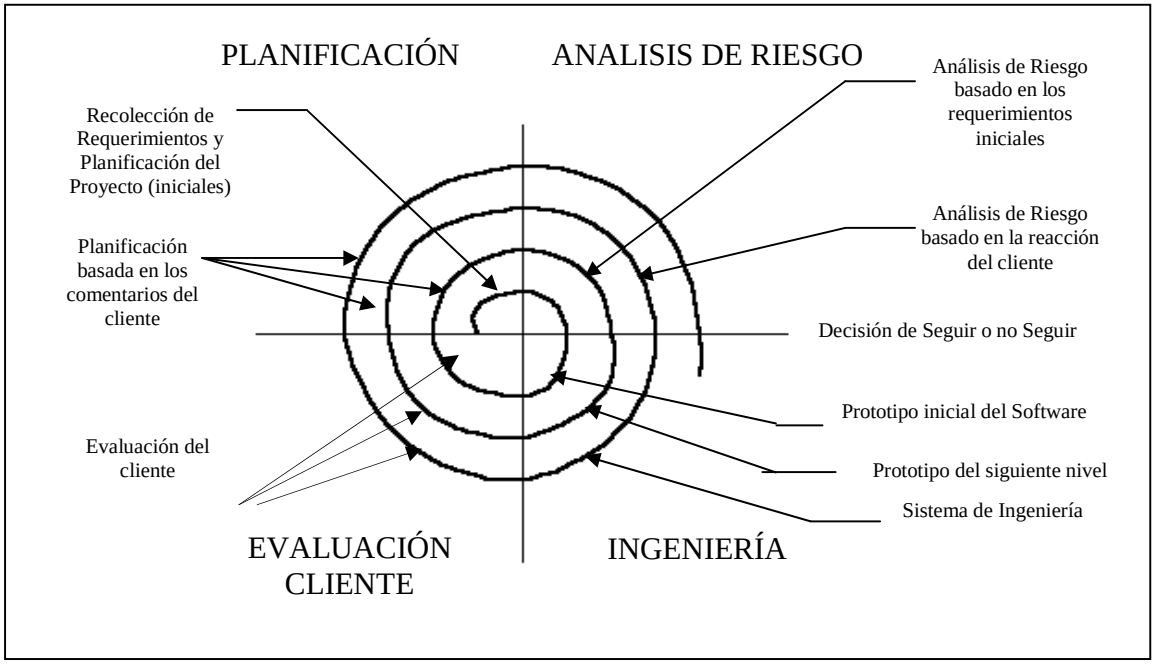


Figura 3: Modelo Espiral

4.- Técnicas de Cuarta Generación: Estas técnicas, también conocidas como T4G son un amplio conjunto de herramientas que permiten al desarrollador especificar algunas características del software de alto nivel a través de lenguaje especializado o notaciones gráficas que describan el problema de manera que sea entendible por el cliente.

Todo comienza con la recolección de requisitos del cliente que luego son llevados a un prototipo operativo. En las aplicaciones pequeñas se puede pasar directo de la recolección de requisitos a la implementación, lo cual permite centrarse en la representación de los resultados esperados, lo que se traduce en un código fuerte que produce dichos resultados. Para transformar una implementación en un producto se debe realizar una prueba completa, desarrollar la documentación y realizar las actividades de integración que son requeridas en los otros paradigmas de la ingeniería de software.

El paradigma de cuarta generación comprende los siguientes pasos: Recolección de requerimientos, Estrategia de diseño, Implementación en lenguajes de cuarta generación y Pruebas, que se pueden apreciar en la figura 4.

Dentro de las Herramientas de T4G se pueden contar ciertos modelos, los cuales presentan alguna variación o una mezcla de los modelos antes descritos, es modelos son:

- Modelo de ensamblaje de componentes.
- Modelo de desarrollo concurrente.

- Modelo Incremental.

Los problemas que se presentan en las Técnicas de Cuarta Generación se pueden apreciar en la bibliografía utilizada [3].

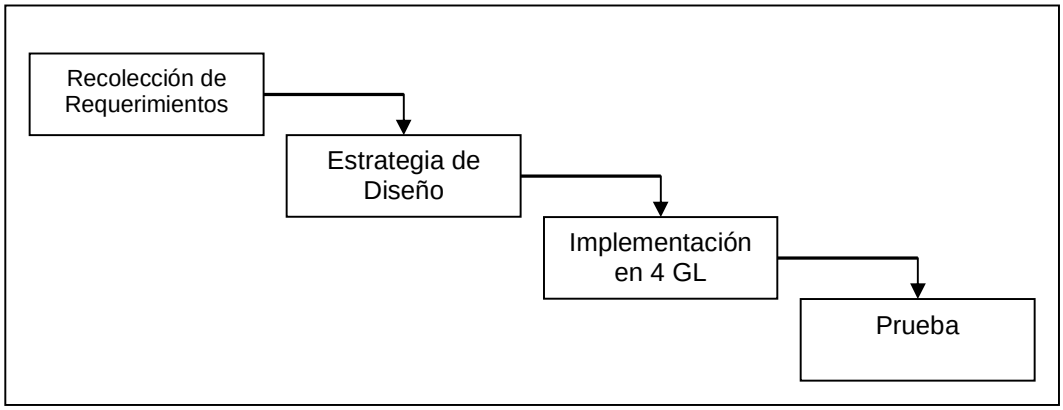


Figura 4: Técnica de Cuarta Generación

5.- Proceso Unificado de Desarrollo: Este paradigma entrega un marco de trabajo genérico que es posible aplicar y adaptar para una gran cantidad de sistemas de diferentes tamaños, aplicación, que pueden pertenecer a distintos niveles de una misma institución e incluso de diferentes instituciones.

Este paradigma se encuentra formado por componentes de software que están intercomunicados por interfaces, además de involucrar una metodología orientada a objetos, de la cual incorpora los conceptos de los casos de uso, clases y objetos.

El Proceso de Desarrollo de Software define en forma clara quien debe de hacer qué y también el cómo y cuando hacerlo.

Las características principales de este paradigma son: ser un proceso iterativo e incremental, centrado en la arquitectura y dirigido por los casos de uso.

Es iterativo en el sentido de iterar las tareas para desarrollar el software e incremental en que tales iteraciones en las tareas generan un incremento del mismo. El ciclo de vida iterativo se basa en la producción de prototipos ejecutables, además de que en cada iteración se reproduce un ciclo de vida en cascada a menor escala, como se observa en la figura 5. Todas estas iteraciones se controlan a través de una selección y ejecución planificada.

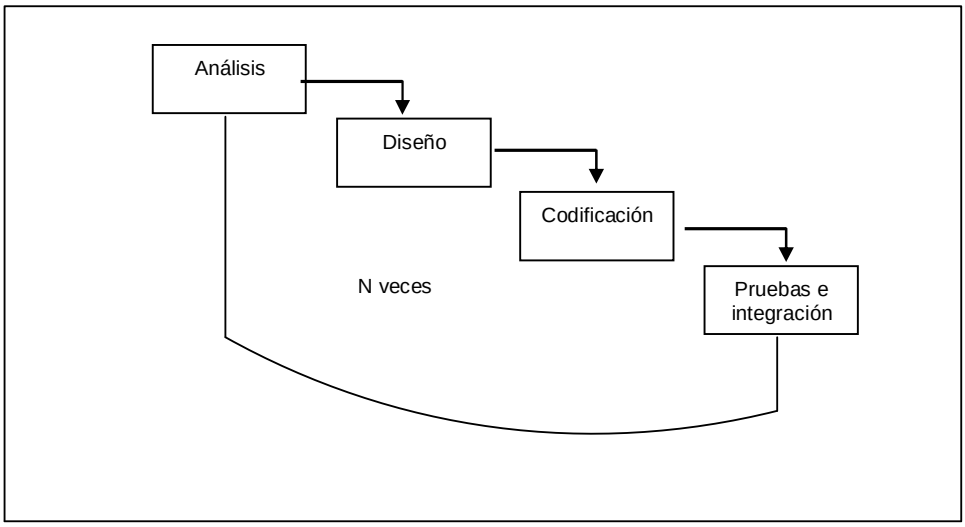


Figura 5: Iteración en un Proceso Unificado de Desarrollo

Es centrado en la arquitectura debido a que toma en cuenta la organización o estructura de sus partes más relevantes. Se entiende por arquitectura ejecutable a la implementación parcial del sistema que demuestra algunas funciones y propiedades, el proceso de desarrollo del software establece refinamientos sucesivos de una arquitectura ejecutable, construida como prototipos evolutivos.

Es un proceso centrado en los casos de uso ya que es necesario capturar, definir y validar los casos de uso que se utilizan en el desarrollo del software, luego se deben realizar para validar que se satisfagan.

El flujo de trabajo fundamental del proceso Unificado de Desarrollo se puede observar en la figura 6 en donde están presentes 4 fases y en cada una de ellas un mini ciclo de vida clásico, en cada una de estas fases se puede dar iteraciones.

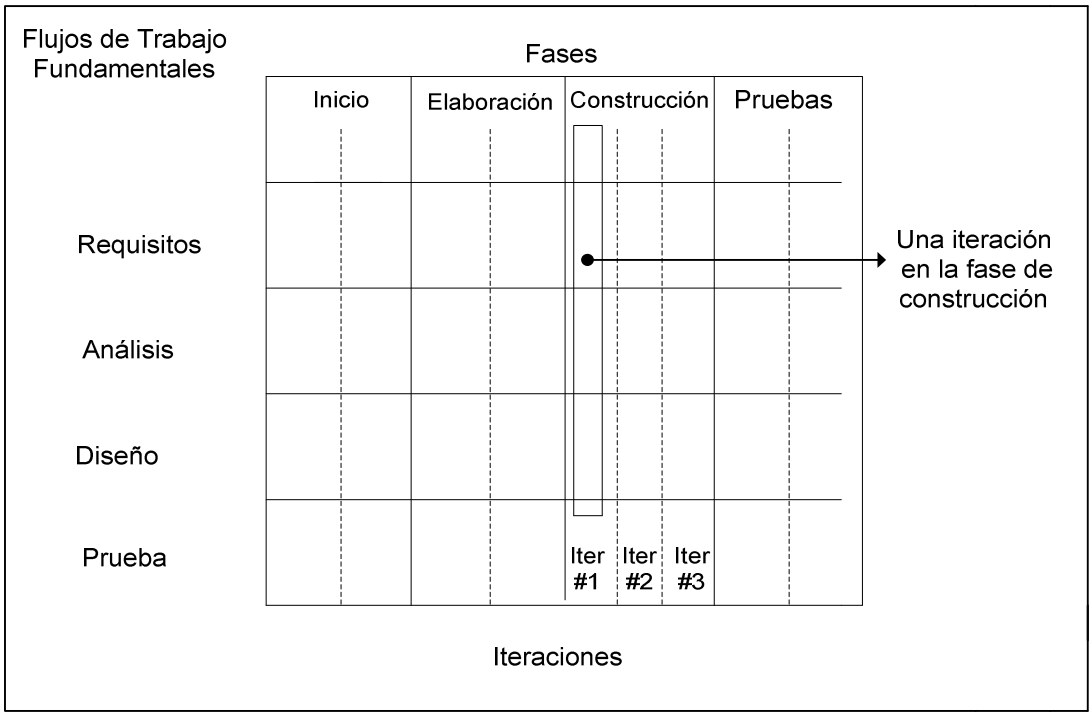


Figura 6: Proceso Unificado de Desarrollo

2.2.2 Elección del Paradigma de Ingeniería.

Después de analizar los principales paradigma de la Ingeniería de Software se puede concluir que cada uno de estos es aplicable al desarrollo de un software, sin embargo, cada uno cuenta con ventajas y desventajas que son decisivas en el momento de su elección.

Atendiendo a las características de cada paradigma y las del proyecto a desarrollar se estima conveniente la utilización del "Paradigma de Proceso Unificado de Desarrollo", el cual realiza una evolución del trabajo a través del uso de prototipos, los que favorecen una interacción entre el cliente y sus requisitos con el desarrollo del software, lo cual posibilita realizar las correcciones que el cliente que estime conveniente.

También hay que destacar que este paradigma permite la creación de "mini proyectos" los cuales al mismo tiempo de permitir la refinación del proyecto refinan los requisitos del usuario e incorporan requisitos nuevos a lo largo del desarrollo del software.

Otro punto importante de citar es la participación de los casos de uso, como se observa en la figura 7, en las diferentes fases que permiten tener una visión mas clara de lo que se quiere representar en el software.

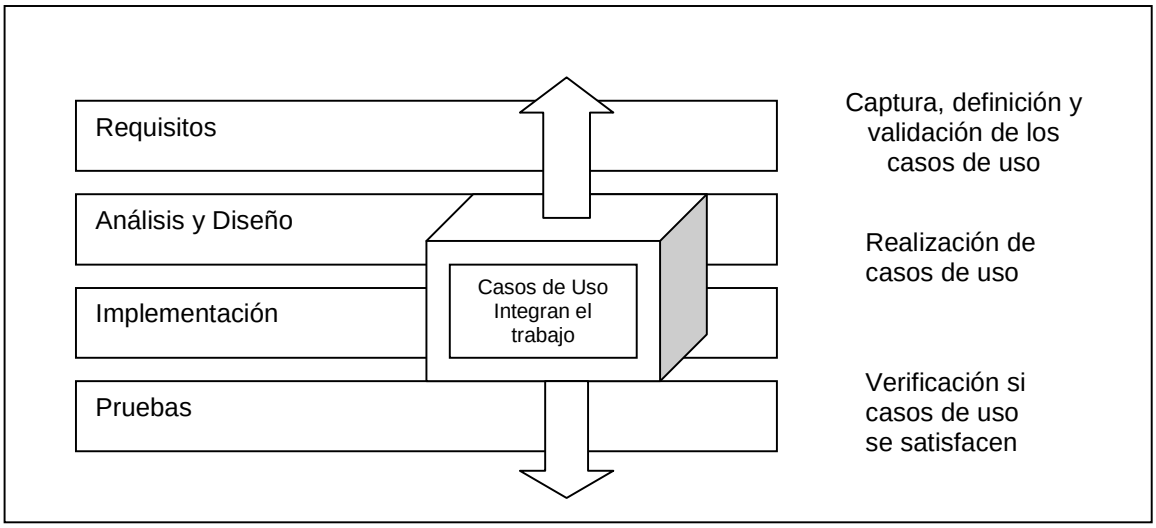


Figura 7 Participación de Casos de Uso.

2.3 Estudio de las Metodologías alternativas de Análisis y Diseño.

Existen diversas metodologías que permiten desarrollar un software, debido a que en el capítulo anterior se ha elegido el "Paradigma de Proceso Unificado de Desarrollo", la Metodología de Análisis se basará en el Análisis Orientado a Objeto siendo éste el análisis relacionado al paradigma antes mencionado.

2.3.1 Análisis y Diseño alternativa Orientado a Objeto.

Propuesta que permite "modelar la realidad del sistema en desarrollo, representado por clases, objetos, atributos y métodos. Con estos objetos es posible representar y modelar casi cualquier aspecto identificable en el sistema (entidades externas, sucesos, unidades organizacionales), como objetos que generalizan datos y procesos" [3].

El propósito de esta metodología es definir todas las clases, objetos, atributos y métodos que serán relevantes para el buen desarrollo de la aplicación, además permite trabajar con distintos niveles de abstracción al agrupar los diferentes objetos en clases.

Las clases son un miembro real o una entidad abstracta del sistema y agrupan un conjunto de objetos con comportamiento y características similares.

Los objetos son una instancia o variable de una clase, cada objeto se encuentra definido por un estado y comportamiento, que lo distinguen de los otros objetos.

Para el desarrollo del software se trabajará con el Análisis Orientado a Objeto, esto debido a ser una herramienta relativamente nueva cuyo uso se ha ido masificando, además por presentar las siguientes características:

- El Análisis Orientado a Objeto permite descomponer el sistema global en clases, atributos y operaciones, permitiendo así modular y encapsular los datos.

- Entrega la posibilidad de ampliar las funcionalidades de la aplicación en desarrollo, además de permitir, a través de un buen análisis, generar funcionalidades de la aplicación que pueden ser reutilizables, ya sea por otros módulos de la misma aplicación o de otra similar.
- Permite crear diferentes modelos que representan la relación entre objetos, clases y jerarquías.
- Presenta una forma de programar las cosas según como se expresan en la vida real, identificar e integrar las diferentes entidades que participan en un sistema, tratándolos como un todo, reduciendo la posibilidad de trabajar con sistemas complejos.
- Permite encapsular o empaquetar variables de los objetos, con el fin de proteger sus métodos. Esta característica aporta dos beneficios: Modularidad (el código fuente de un objeto se puede escribir y mantener independiente del código fuente de otros objetos) y Ocultamiento de Información (un objeto tiene Interfaz Pública que permite su comunicación con otros objetos, pero también puede tener información y métodos privados que pueden cambiar sin alterar a los demás objetos).
- Mecanismo de Herencia, con el que se puede definir una clase en términos de otra clase, es decir, sólo se especifican las diferencias entre éstas, además permite apreciar relaciones de jerarquía entre clases.

2.4 Análisis de Riesgos.

En el desarrollo de un proyecto de software una tarea muy importante es la anticipación de los riesgos, pues estos podrían afectar la programación del proyecto o la calidad del mismo. Los resultados de este análisis de riesgos deben ser documentados durante todo su desarrollo, además del análisis de lo que podría pasar ante la eventualidad de un riesgo. El ser capaz de identificar estos riesgos y crear un plan de prevención se denomina Administración de Riesgos, dando como resultado el Análisis de Riesgos.

Existen 4 etapas importantes para la Administración de riesgos:

- Identificación de Riesgos del Sistema.
- Análisis de Riesgos.
- Planeación de Riesgos.
- Supervisión de Riesgos.

En la tabla 1 se describe la identificación de riesgos en el proyecto.

Riesgo	Tipo Riesgo	Descripción
Cambio Administración	Proyecto	Cambio de la administración organizacional

		con diferentes prioridades.
Cambios en los Requerimientos	Proyecto y Producto	Dependiendo de la evolución del proyecto irán apareciendo otros requerimientos en forma anticipada.
Retrasos en la especificación	Proyecto y Producto	Las especificaciones principales del sistema no van a estar listas antes de la fecha de entrega.
Sub-utilización del soporte del sistema	Proyecto y Producto	El SABD que sustenta la Base de Datos del sistema no se utiliza de buena forma.
Bajo desempeño con la utilización de PHP	Producto	El lenguaje con el que se ha desarrollado el proyecto no tiene el desempeño adecuado.
Bajo desempeño de la herramienta CASE	Producto	La herramienta CASE que ayuda al proyecto no tiene el desempeño anticipado.
Cambio de Tecnología	Negocio	La tecnología en la cual se pretendía implementar el sistema ha sido cambiada por nueva tecnología.
Competencia del producto	Negocio	Un producto de la competencia sale al mercado antes de la finalización del proyecto.

Tabla 1: Identificación de riesgos del Sistema.

La siguiente etapa de la Administración de Riesgos es el análisis de riesgos, el cual no tiene una valoración con números fijos, sino que se evalúa de acuerdo a conceptos, los cuales son:

- Probabilidad: Esta se puede subdividir en muy bajo, bajo, moderado, alto, y muy alto.
- Efectos: Se pueden subdividir en catastrófico, serio, tolerable o insignificante.

En la tabla 2 se describe el resultado del Análisis de Riesgos.

Riesgo	Probabilidad	Efectos
Cambio de prioridades por causa de un cambio en la administración organizacional.	Baja	Serio
Cambios en los requerimientos del proyecto hacen que el diseño se reestructure.	Moderada	Serio
Las especificaciones principales del sistema no van a estar listas antes de la fecha de entrega.	Alta	Serio
El tiempo requerido para el desarrollar el software ha sido subestimado	Alta	Serio
Un producto de la competencia sale al mercado antes de la finalización del proyecto.	Moderada	Serio
El SABD que sustenta la Base de Datos del sistema no es bien	Moderada	Serio

utilizado.		
El tamaño del software esta subestimado	Alta	Tolerable
El lenguaje con el que se esta desarrollado el proyecto no tiene el desempeño adecuado.	Alta	Tolerable
Las herramientas CASE no se pueden integrar	Alta	Tolerable
La tecnología en la cual se pretendía implementar el sistema ha sido cambiada por nueva tecnología.	Baja	Tolerable
Es ineficiente el uso de la herramienta CASE	Moderada	Insignificante

Tabla 2: Análisis de Riesgo del Sistema.

Continuando con el análisis corresponde realizar una planeación de riesgos, en donde se pueden observar tres categorías:

Estrategias de Disminución: Se reduce el impacto de riesgo.

Estrategias de Anulación: Se reducen la probabilidad de que el riesgo surja

Planes de Contingencia: Estos se establecen para poder contar con una estrategia en el caso de alguna eventualidad que ocurra un riesgo.

Esto se observa en la tabla 3.

Riesgo	Estrategia
Reestructuración Organizacional	Se debe de preparar un documento al nuevo administrador para que observe las contribuciones que otorga el proyecto a la empresa.
Cambio en los Requerimientos	Rastrear la información para valorar el impacto de los requerimientos, maximizar la información oculta en ellos
Pobre desempeño en la finalización de hitos	Reorganizar las labores y potenciar las virtudes.
Desempeño del SABD	Estudiar la posibilidad de compra de una base de datos con un desempeño más alto.
Lenguaje defectuoso	Investigar los lenguajes que se han adquirido, y las posibilidades que ofrece el mercado frente a la disponibilidad de lenguajes similares.
Pobre desempeño de la Herramienta CASE	Elegir una nueva herramienta desechando la que se ocupo.
Cambio de Tecnología	Averiguar las posibilidades que ofrecen los distintos equipos que se encuentran en el mercado para la implementación del proyecto.

Tabla 3: Planeación de riesgo del Sistema.

La Supervisión de Riesgos valora cada uno de los riesgos identificados para decidir si éste es más o meno probable y cuándo los efectos del mismo han cambiado.

La tabla 4 refleja la Supervisión de Riesgos del Sistema.

Tipo de Riesgo	**Indicadores Potenciales**
Ausencia de Personal Experimentado	La especialización de las tareas dentro de la empresa siempre es un obstáculo cuando hay ausencia laboral.
Reestructuración Organizacional	Falta de acciones por parte de la plana administrativa.
Problemas Financieros de la Organización	Falta de acciones por parte de la plana administrativa.
Falta de Apoyo Técnico	Entrega retrasada del hardware previsto para el sistema. Exceso de problemas técnicos registrados.
Falta de Capacitación	Fracaso en el cumplimiento de los tiempos acordados.
Cambio en los Requerimientos	Demasiadas peticiones de cambios en los distintos departamentos de la empresa.
Demoras Temporales de las Especificaciones	Fracaso en el cumplimiento de los tiempos acordados.
Sistema muy grande	Falta de análisis de las capacidades tecnológicas pertenecientes a la empresa.
Desempeño de la Base de Datos	Tecnología atrasada, defectuosa, mal aprovechamiento de los recursos.
Lenguaje Defectuoso	Software defectuoso, falta de conocimiento de los desarrolladores frente a un lenguaje, poca familiaridad del usuario frente a los lenguajes propuestos por los desarrolladores.
Cancelación del Proyecto	Baja expectativa de expansión del negocio considerando una inversión innecesaria.

Tabla 4: Supervisión de Riesgos del Sistema.

La Supervisión de Riesgos debe ser un proceso continuo. En cada revisión todos los riesgos identificados deben ser considerados en forma individual y sometidos a discusión [5].

2.5 Estudio de Factibilidad.

Según lo citado por (James Martin, James J.Odell.: “Métodos Orientados a Objetos: consideraciones practicas”), para todos los sistemas nuevos, el proceso de ingeniería de requerimientos empieza con el estudio de factibilidad. La entrada de éste es una descripción resumida del sistema y de cómo se utilizará dentro de una organización.

El estudio de factibilidad en un proyecto, consiste en descubrir cuales son los objetivos de la organización, luego determinar si el proyecto es útil para que la empresa logre sus objetivos. La búsqueda de estos objetivos deben contemplar los recursos disponibles o aquellos que la empresa puede proporcionar, nunca deben definirse con recursos que la empresa no es capaz de dar. La factibilidad se apoya en cuatro aspectos básicos:

- Técnico.
- Operativo.
- Legal
- Económico.

El éxito de un proyecto esta determinado por el grado de factibilidad que se presente en cada una de los cuatro aspectos anteriores.

Un estudio de factibilidad requiere ser presentado con todas la posibles ventajas para la empresa u organización, pero sin descuidar ninguno de los elementos necesarios para que el proyecto funcione. Para esto dentro de los estudios de factibilidad se complementan dos pasos en la presentación del estudio:

- Requisitos Óptimos.
- Requisitos Mínimos.

Como primer paso se debe realizar un pequeño estudio con los requisitos óptimos que el proyecto requiera, estos elementos deberán ser los necesarios para que las actividades y resultados del proyecto sean obtenidos con la máxima eficacia.

El segundo paso consiste en un estudio de requisitos mínimos, el cual cubre los requisitos mínimos necesarios que el proyecto debe ocupar para obtener las metas y objetivos, este paso trata de hacer uso de los recursos disponibles de la empresa para minimizar cualquier gasto o adquisición adicional.

2.5.1 Factibilidad Técnica.

El estudio de factibilidad técnica consiste en determinar si se cuenta con los recursos tales como herramientas, conocimientos, habilidades, experiencia, etc., que son necesarios para efectuar las actividades o procesos que requiere el proyecto. Generalmente se refieren a elementos tangibles. El proyecto debe considerar si los recursos técnicos actuales son suficientes o deben complementarse. El departamento Administrativo cuenta con lo siguiente:

Recursos de software

- Cuenta con sistema operativo Microsoft Windows XP
- Microsoft Office 2003.

Recursos de hardware

- 2 Equipos HP, Intel Celeron 2.8 Ghz, 512 Mb RAM.

De acuerdo a las especificaciones técnicas con las cuales la empresa "CHIVATO" cuenta, se determina que para la implementación del sistema se requiere de:

- Un sistema administrador de base de datos (SABD), PostgreSQL-8.0.2 para Microsoft Windows XP.
- Un lenguaje que permita generar y procesar la información de formularios. Para lo cual se utilizaría en un comienzo JAVA, pero debido al corto tiempo se ha decidido utilizar un lenguaje que es mas conocido por el desarrollador que es PHP.
- 4 Lectores de Código de barras.
- 1 switch Dlink 8 bocas 1008D 10/100.
- 3 impresoras TM 222.
- 6 monitores. (3 pare caja y 3 para barras).
- 6 computadores Sempron 2800+ ,Con 256Mb de Memoria Ram DDR400, Disco duro de 80GB 7200 RPM, CD ROM 52X, Disketera, Placa Madre ASROCK KVM800, Monitor Samsung 15 CRT Plano, Teclado, Parlantes, multimedia y Mouse Óptico
- Cable.

De lo anteriormente visto se concluye que la empresa "CHIVATO" contara con los requerimientos mínimos (requisitos mínimos) en cuanto a Hardware y Software para la implementación del sistema. Por lo tanto técnicamente es factible de llevar a cabo el proyecto.

2.5.2 Factibilidad Operacional.

El estudio de factibilidad operacional se refiere a todos aquellos recursos donde interviene algún tipo de actividad (Procesos), depende de los recursos humanos que participen durante la operación del proyecto. Durante esta etapa se identifican todas aquellas actividades que son necesarias para lograr el objetivo y se evalúa y determina todo lo necesario para llevarla a cabo. En pocas palabras investiga si será utilizado el sistema, si los usuarios usarán el sistema, para obtener beneficios.

El sistema que como objetivo final se pretende implementar se ajustará a la forma de trabajar en el proceso normal usado actualmente, sin tener que realizar cambios severos en la manera de trabajar que tiene la organización. Para los usuarios el sistema no hará más difícil las tareas específicas de cada uno sino facilitará la realización de éstas.

Dentro de la organización, la mayoría de los empleados que tendrá manejo del sistema que se implementará cuenta con conocimientos a nivel de usuario en el manejo de computadores. Además el personal tiene ideas claras referentes al funcionamiento de sus tareas, lo cual facilitará la implantación del nuevo sistema con un mínimo de capacitación

Por lo tanto es factible llegar a realizar el sistema cumpliendo operacionalmente la forma de trabajar en la organización ya mencionada.

2.5.3 Factibilidad Legal.

El estudio de la factibilidad legal consiste en realizar una investigación de las licencias de herramientas y software que se van a utilizar, pretendiendo verificar que al implantar un nuevo sistema dentro de una empresa, no se incurra en algún delito contra la propiedad intelectual de otros desarrolladores de software. Para ello se debe considerar la legislación vigente en el ámbito nacional correspondiente a leyes informáticas. También es importante considerar las políticas internas de la empresa, tratando de afectar de la menor forma posible el accionar o el desarrollo interno de la empresa y que al utilizar el sistema incurra en algún delito.

La empresa “CHIVATO”. Cuenta con las licencias necesarias de sistema operativo y no será necesario contar con licencias para el software a utilizar ya que se utilizará software libre para la implementación del sistema.

2.5.4 Factibilidad Económica.

El estudio de factibilidad económica busca determinar los recursos económicos y financieros necesarios para desarrollar o llevar a cabo las actividades o procesos. También se utiliza para determinar los recursos económicos básicos que deben considerarse como son el costo del tiempo, el costo de la realización y el costo de adquirir nuevos recursos.

Generalmente la factibilidad económica es el elemento mas importante, ya que a través de él se solventan las demás carencias de otros recursos, es lo mas difícil de conseguir y requiere de actividades adicionales cuando no se posee.

Para la implementación de este sistema, la empresa “CHIVATO” se supone deberá contar con los requerimientos mínimos.

Para el desarrollo de este sistema, se ha decidido utilizar herramientas de libre distribución puesto a que la organización no cuenta con la disponibilidad de herramientas de desarrollo, y dada la naturaleza y características del proyecto no se ve necesidad real de realizar inversión en este respecto.

El costo incurrido en el desarrollo del sistema será asumido por los desarrolladores, puesto a que se trata de un proyecto de carácter académico. Por lo tanto se concluye que el sistema es económicamente factible de realizar.

2.6 Gestión de cambio y configuraciones.

Lo que se pretende en este flujo de trabajo de soporte es realizar una metodología que permita controlar los cambios y mantener la integridad de los artefactos de un proyecto, entendiendo por artefacto a una pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar distintas actividades, éstos pueden ser un modelo, elementos de un modelo o un documento.

Es importante destacar que los cambios son inevitables en un proceso de desarrollo, pero es importante evaluar si estos cambios son necesarios e investigar el impacto que causarán.

Al trabajar con un paradigma que está basado en el desarrollo iterativo, nos propone una comprensión incremental del problema a través de refinamientos sucesivos sobre artefactos y un crecimiento incremental de una solución efectiva a través de varias versiones las cuales contiene mejoras o la inserción de nuevas funcionalidades[3].

Para la realización de esto se fijan las siguientes medidas:

2.6.1 Plan de Iteración.

En el cual se pretende fijar tareas relacionados con el control de cambios sobre las posteriores etapas del Proceso Unificado de Desarrollo.

Etapa de elaboración

- Analizar los riesgos que son de mayor prioridad para el proyecto y que pueden poner en peligro la realización de éste.
- Fijar una arquitectura base, la cual consiste en tener una visión mas específica para la realización del proyecto (plataformas, modelos, etc.), en otras palabras fijar el esqueleto del sistema a desarrollar y que cumpla con todo lo planteado.
- En caso de que se necesite realizar un cambio se debe hacer un estudio del impacto que causaría en otros artefactos relacionados con éste.
- En relación con el punto anterior, se debe agregar la actualización de documentos relacionados, junto con además modificar las interfaces creadas en una primera etapa.
- Fijar un plan de construcción con la asignación de ciertos artefactos para que sean revisados antes de empezar a construir el software.

Etapa de construcción

- Verificar que no existan nuevas modificaciones en los artefactos.
- Asegurarse de que todos los artefactos fueron revisados para comenzar con la construcción.

Etapa de transición

- Reflejar los errores encontrados en las etapas de pruebas del software.

2.6.2 Gestión del Proyecto.

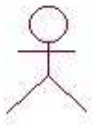
Al igual que la gestión del cambio, la gestión de proyecto se enmarca dentro de lo que son los flujos de trabajo de soporte y que básicamente lo que se pretende es constar con una estrategia del proyecto, definición de las actividades a desarrollar, las estimaciones de las actividades y lógicamente la planificación del proyecto, en cuanto a las estimaciones, éstas se realizan por medio de Bottom up (de abajo hacia arriba) con juicio de expertos, debido a que otras técnicas se basan en la cantidad de líneas de códigos o fórmulas pero éstas son relativas en cuanto a lo de líneas de código se refiere, porque para un mismo componente pueden existir diferentes cantidades de líneas.

2.7 Casos de Uso.

El diagrama de casos de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso)[10].

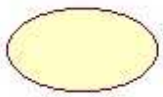
Elementos

- **Actor**:



Una definición previa, es que un Actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

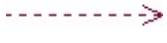
- **Caso de Uso**:



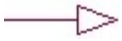
Es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

- **Relaciones**:
  - **Asociación** 

Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

  - **Dependencia o Instanciación** 

Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.

Generalización 

Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de Uso (<<uses>>) o de Herencia (<<extends>>).

Este tipo de relación esta orientado exclusivamente para casos de uso (y no para actores).

extends: Se recomienda utilizar cuando un caso de uso es similar a otro (características).

uses: Se recomienda utilizar cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica.

2.7.1 Diagramas de Casos de Uso

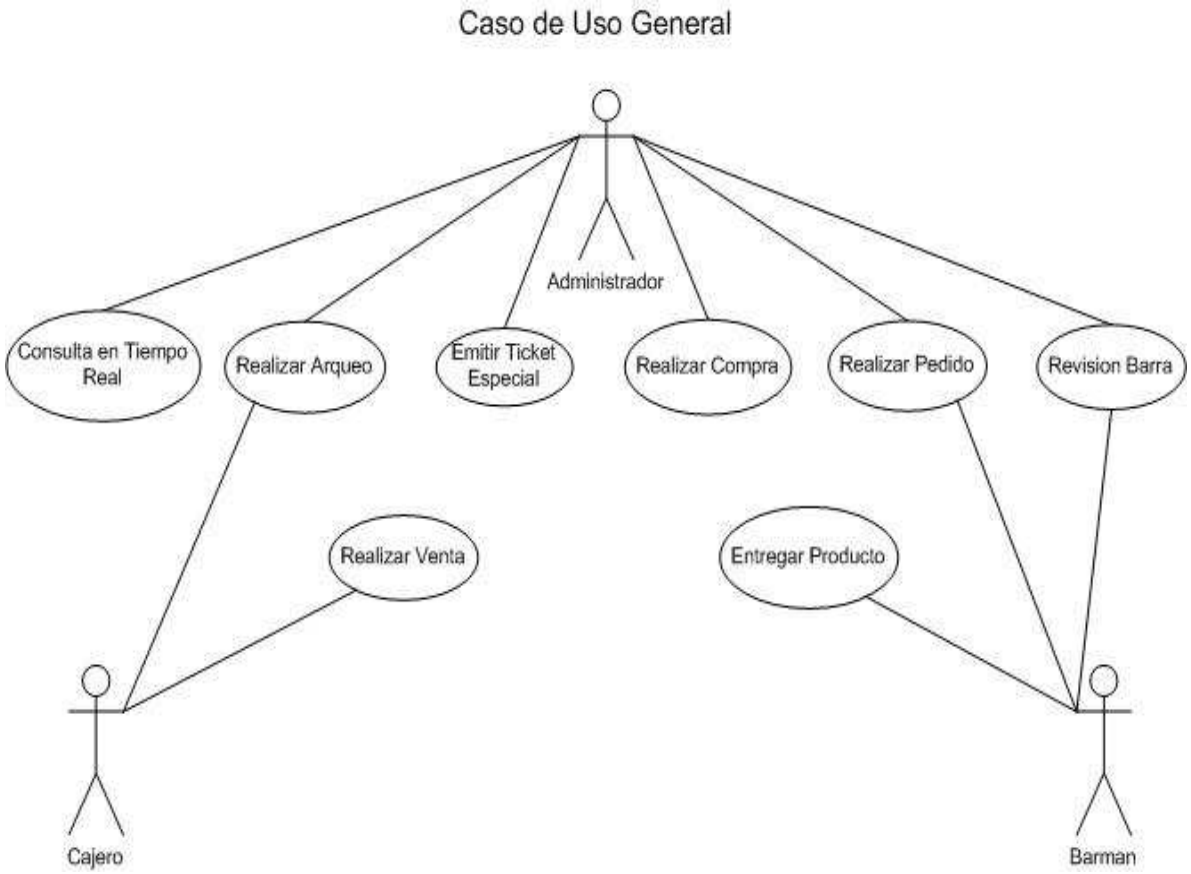


Figura 8: Diagrama General de Casos de Uso.

CASO DE USO : ADMINISTRADOR

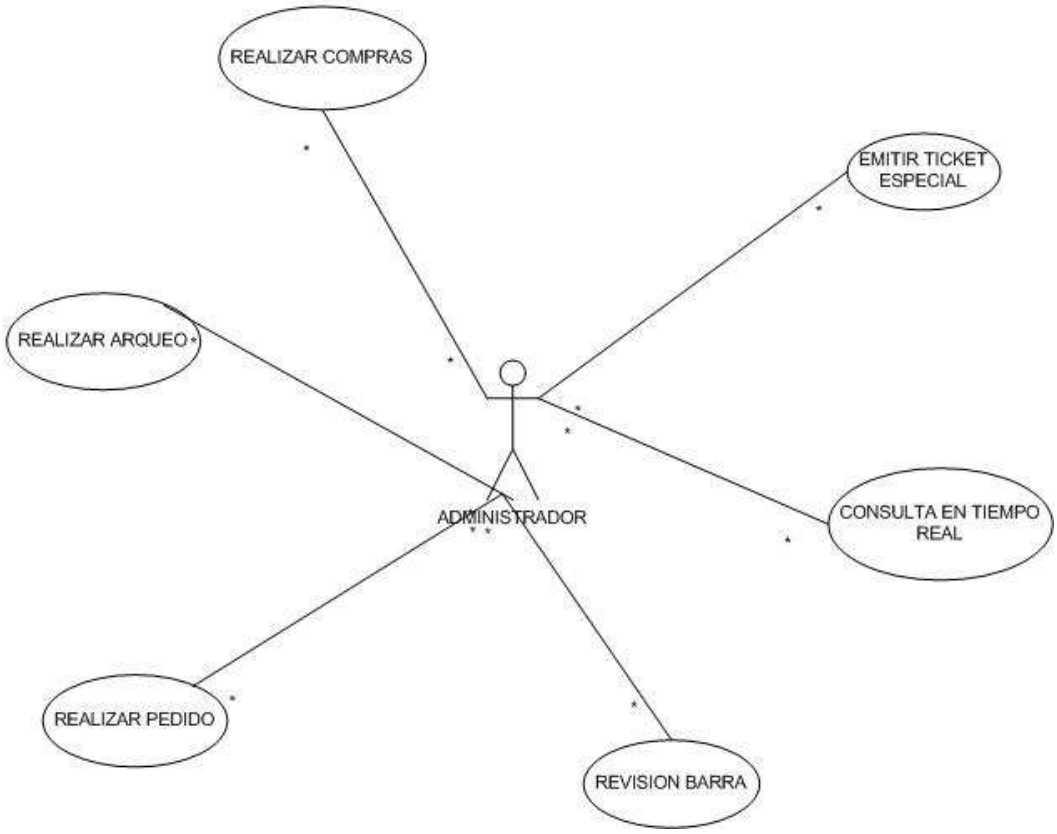


Figura 9: Diagrama de Casos de Uso para Administrador.

CASO DE USO : CAJERO

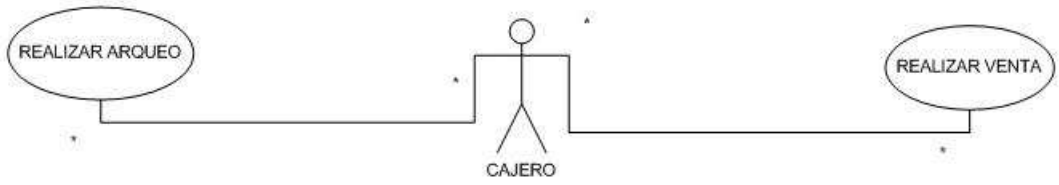


Figura 10: Diagrama de Casos de Uso para Cajero.

CASO DE USO : BARMAN

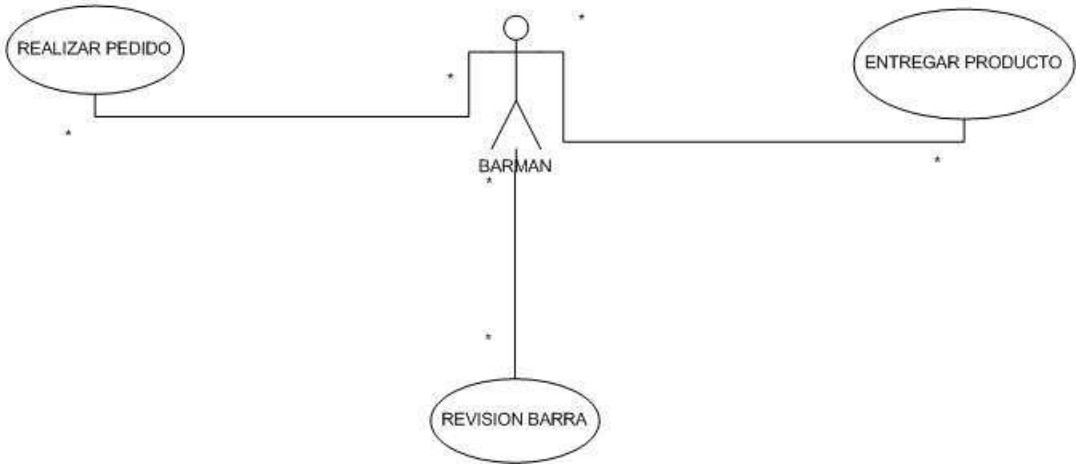


Figura 11: Diagrama de Casos de Uso para Barman.

2.7.2 Casos de uso Descripción Formal

Caso de Uso: Realizar venta

Caso de Uso	Realizar venta.
Actor Primario	Cajero
Participantes e Intereses	Cajero: Registra venta en el sistema, realiza petición de ticket(s) y entrega de éste.
Precondiciones	El cajero ha ingresado al sistema en forma satisfactoria.
Poscondiciones	Se ingresa al sistema el(los) ticket(s) expendidos por el cajero.
Escenario Principal	1.-El cajero recibe por .parte del cliente la solicitud de ser surtido de uno o mas productos. 2.-El cajero selecciona la opción “realizar venta” de su menú. 3.-El sistema muestra pantalla para que sea elegido el o los productos. 4.-El cajero proporciona la información en base a la(s) solicitud(es) del cliente. 5.-El sistema muestra el total de la venta (en todo momento). 6.-El cajero efectúa el cobro de lo solicitado. 7.-El cajero emite la boleta correspondiente. 8.-El cajero selecciona imprimir ticket. 9.-El Sistema despliega un mensaje confirmando la operación. 10.-El cajero confirma la venta. 11.- El sistema imprime el(los) ticket(s). 12.-El cajero hace entrega al cliente de el(los) ticket(s) y de la boleta por la venta.
Extensiones	8.-El cajero no entrega información de productos al sistema. 1.-El sistema despliega un mensaje que indica que no existen productos para realizar la operación y vuelve a pantalla anterior 10.-El cajero después de confirmar quiere ingresar más productos. 1.-El cajero debe realizar una nueva venta por los productos faltantes.
Frecuencia de ocurrencia	Continuo.

Caso de Uso: Entregar producto

Caso de Uso	Entregar producto
Actor Primario	Barman
Participantes e Intereses	Barman: surte al cliente de su pedido que es informado mediante ticket.
Precondiciones	El cliente posee ticket.
Poscondiciones	Se ingresa al sistema el ticket surtido por el barman.
Escenario Principal	1.-El barman recibe por parte del cliente la solicitud de ser atendido en forma verbal. 2.-El barman introduce al sistema el ticket del cliente. 3.-El sistema muestra por pantalla el producto solicitado por el cliente con algún detalle de preparación o ingredientes. 4.-El barman solicita confirmación al cliente acerca del pedido. 5.-El barman confirma ticket. 6-El barman procede a preparar el producto solicitado. 7.-El barman hace entrega del producto al cliente.
Extensiones	5.-El ticket tiene información errónea acerca del producto solicitado. 1.-El barman selecciona la opción “rechazar ticket” y este no es ingresado como producto entregado. Y devuelve ticket al cajero, la venta debe ser efectuada nuevamente obviamente con el descuento del valor del ticket.
Frecuencia de ocurrencia	Continuo.

Caso de Uso: Emitir ticket especial

Caso de Uso	Emitir ticket especial.
Actor Primario	Administrador
Participantes e Intereses	Administrador: Registra un ticket que no será ingresado en la sesión del cajero.
Precondiciones	El administrador solicita permiso para hacer uso de la caja correspondiente.
Poscondiciones	Se ingresa al sistema el(los) ticket(s) expendidos por el administrador.
Escenario Principal	1.-El administrador inicia una sesión en el equipo asignado a la caja. 2.-El administrador selecciona la opción “expender ticket especial” de su menú. 3.-El sistema muestra pantalla para que sea elegido el o los

	productos. 4.-El administrador proporciona la información de el(los) ticket(s). 5.-El administrador selecciona imprimir ticket. 9.-El Sistema despliega un mensaje confirmando la operación. 10.-El administrador confirma la operación. 11.- El sistema imprime el(los) ticket(s).
Extensiones	8.-El administrador no entrega información de productos al sistema. 1.-El sistema despliega un mensaje que indica que no existen productos para realizar la operación y vuelve a pantalla anterior 10.-El administrador después de confirmar quiere ingresar más productos. 1.-El administrador debe realizar una nueva operación.
Frecuencia de ocurrencia	Continuo.

Caso de Uso: Realizar compra

Caso de Uso	Realizar compra.
Actor Primario	Administrador.
Participantes e Intereses	Administrador: efectúa adquisición de productos que faltan en stock de bodega.
Precondiciones	El sistema da aviso de alerta al administrador de la falta de stock en productos de bodega.
Poscondiciones	Inventario queda actualizado con el ingreso de los datos de la compra.
Escenario Principal	1.-El Sistema da aviso de la falta o próxima falta de productos en stock de bodega. 2.-El administrador Solicita productos a sus proveedores. 3.-El administrador hace ingreso de los productos al sistema mediante la opción “ingresar compra”. 4.-El administrador ingresa datos acerca de los productos comprados. 5.-El administrador utiliza la opción “ingreso de productos”. 6.-El sistema muestra un listado con los productos ingresados y solicita confirmación. 7.-El administrador confirma ingreso de los productos.
Extensiones	6.-La información es errónea. 1.-El administrador debe realizar nuevamente la operación.

Frecuencia de ocurrencia	Continuo.

Caso de Uso: Realizar pedido

Caso de Uso	Realizar pedido.
Actor Primario	Administrador.
Participantes e Intereses	Administrador: surtido de productos que faltan en stock de barra.
Precondiciones	El barman solicita el surtido de stock de productos de barra.
Poscondiciones	Inventario queda actualizado con el ingreso de los datos del pedido.
Escenario Principal	1.-El barman solicita al administrador los productos que faltan en el stock de la barra. 2.-El administrador ingresa a la opción “ingresar pedido”. 3.-El sistema despliega información acerca del stock en bodega. 4.-El administrador indica las cantidades de los productos que serán entregados al barman. 5.-El administrador ingresa la opción “efectuar pedido”. 6.-El sistema muestra el listado de los productos solicitador y pide confirmación. 7.-El administrador confirma el pedido.
Extensiones	6.-La información es errónea. 1.-El administrador debe realizar nuevamente la operación.
Frecuencia de ocurrencia	Continuo.

Caso de Uso: Revisión barra

Caso de Uso	Revisión barra.
Actor Primario	Administrador.
Participantes e Intereses	Administrador: efectúa la revisión de los productos que quedan en la barra.
Precondiciones	Al final de la jornada el barman solicita revisión de su barra.
Poscondiciones	Inventario queda actualizado con el ingreso de los datos del stock en barra.
Escenario Principal	1.-El administrador pide al sistema la información del stock de la barra en cuestión. 2.-El administrador procede a comparar los datos entregados

	por el sistema con los datos reales. 3.-El administrador toma las medidas del caso. 4.-El administrador actualiza la información del sistema con los datos reales 5.-El sistema pide conformación de los datos mostrándolos por pantalla. 6.-El administrador confirma actualización del inventario de la barra.
Extensiones	5.-La información es errónea. 1.-El administrador debe realizar nuevamente la operación desde paso 4, donde lo devuelve el sistema tras rechazar la confirmación.
Frecuencia de ocurrencia	Continuo.

Caso de Uso: Realizar arqueo.

Caso de Uso	Realizar arqueo.
Actor Primario	Administrador, cajero.
Participantes e Intereses	Administrador: realiza el conteo de los dineros que ingresaron en loa caja en cuestión durante la jornada, y actualiza los datos.
Precondiciones	El cajero solicita el arqueo de su caja.
Poscondiciones	El sistema efectúa el ingreso de los datos reales de las ventas.
Escenario Principal	1.-El cajero solicita al sistema el arqueo de caja desde su caja. 2.-El sistema imprime en la oficina del administrador el arqueo 3.-El cajero hace llegar los tickets con carácter de rechazados al administrador. 4.-El administrador efectúa el descuento de los tickets rechazados. 5.-El sistema actualiza la información de la caja. 6.-El administrador y el cajero cuentan el dinero. 7.-El administrador toma las medidas pertinentes dependiendo de la situación. 8.-El administrador solicita la información actualizada. 9.-El sistema muestra la información actualizada y pide confirmación. 10.-El administrador confirma el arqueo.
Extensiones	9.-La información es errónea.

	1.-El administrador debe realizar nuevamente la operación.
Frecuencia de ocurrencia	Continuo.

Caso de Uso: Consulta en tiempo real

Caso de Uso	Consulta en tiempo real
Actor Primario	Administrador.
Participantes e Intereses	Administrador visualiza la situación actual de los datos en línea
Precondiciones	El administrador ingresa al sistema vía Web.
Poscondiciones	
Escenario Principal	El administrador ingresa al sistema vía Web mediante su password. El administrador ingresa la opción "Consulta en tiempo real". El sistema muestra los datasen tiempo real. El administrador sale de opción a su menú.
Extensiones	
Frecuencia de ocurrencia	Continuo.

2.8 Descripción técnica de las herramientas.

2.8.1 Herramienta de Modelado: UML (Unified Modeling Language).

Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. UML es una especificación de notación orientada a objetos. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representa la arquitectura del proyecto.

Con UML no se debe olvidar el protagonismo excesivo que se le da al diagrama de clases, este representa una parte importante del sistema, pero sólo representa una vista estática, es decir, muestra al sistema parado. Se sabe de su estructura, pero no acerca de qué le sucede a sus diferentes partes cuando el sistema empieza a funcionar. UML introduce nuevos diagramas que representa una visión dinámica del sistema. Es decir, gracias al diseño de la parte dinámica del sistema se tiene conocimiento en la fase de diseño, de problemas de la estructura al propagar errores o de las partes que necesitan ser sincronizadas, así como del estado de cada una de las instancias en cada momento. El diagrama de clases continúa siendo

muy importante, pero se debe tener en cuenta que su representación es limitada, y que ayuda a diseñar un sistema robusto con partes reutilizables, pero no a solucionar problemas de propagación de mensajes ni de sincronización o recuperación ante estados de error. En resumen, un sistema debe estar bien diseñado, pero también debe funcionar bien.

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los desarrollos se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo.

UML es ahora un standard, no existe otra especificación de diseño orientado a objetos, ya que es el resultado de las tres opciones existentes en el mercado. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otro ramo.

UML permite la modificación de todos sus miembros mediante estereotipos y restricciones. Un estereotipo permite indicar especificaciones del lenguaje al que se refiere el diagrama de UML. Una restricción identifica un comportamiento forzado de una clase o relación, es decir, mediante la restricción se está forzando el comportamiento que debe tener el objeto al que se le aplica.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo.

UML no es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguaje de programación, así como construir modelos por ingeniería inversa a partir de programas existentes.

Existían diversos métodos y técnicas Orientadas a Objetos, con muchos aspectos en común pero utilizando distintas notaciones, se presentaban inconvenientes para el aprendizaje, aplicación, construcción y uso de herramientas, etc., además de pugnas entre enfoques, lo que generó la creación del UML como estándar para el modelamiento de sistemas de software principalmente, pero con posibilidades de ser aplicado a todo tipo de proyectos [5].

Objetivos del UML

- UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática.
- UML no pretende ser un método de desarrollo completo. No incluye un proceso de desarrollo paso a paso. UML incluye todos los conceptos que se consideran necesarios para utilizar un proceso moderno iterativo, basado en construir una sólida arquitectura para resolver requisitos dirigidos por casos de uso.
- Ser tan simple como sea posible, pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesita construir. UML necesita ser lo suficientemente expresivo para manejar todos los conceptos que se originan en un sistema moderno, tales como la concurrencia y distribución, así como también los mecanismos de la ingeniería de software, como son la encapsulación y componentes.
- Debe ser un lenguaje universal, como cualquier lenguaje de propósito general.
- Imponer un estándar mundial.

2.8.2 Lenguaje de Programación: PHP (Hypertext Preprocessor)

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje "Open Source" interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor, a diferencia de Java o Java Script, que se ejecutan en el navegador. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas Web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP [7].

PHP es uno de los lenguajes más extendidos en la web. Nacido en 1994, se trata de un lenguaje de creación relativamente creciente que ha tenido una gran aceptación en la comunidad de webmasters debido sobre todo a la potencia y simplicidad que lo caracterizan.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluido HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS y probablemente alguno más. PHP soporta la mayoría de servidores Web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape y iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. PHP tiene componentes disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía Web para una base de datos es una tarea simple con PHP. PostgresSQL es el motor de base de datos que se utilizará en el desarrollo de este proyecto, el cual está soportado actualmente, [8]

2.8.3 Motor de Bases de Datos: PostgreSQL

El Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos conocido como PostgreSQL (y brevemente llamado Postgres95) está derivado del paquete Postgres escrito en Berkeley. Con cerca de una década de desarrollo tras él, PostgreSQL es el gestor de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl, tcl, pitón, PHP y muchos más) [9].

En 1996, se hizo evidente que el nombre "Postgres95" no resistiría el paso del tiempo. Se eligió un nuevo nombre, PostgreSQL, para reflejar la relación entre el Postgres original y las versiones más recientes con capacidades SQL. Al mismo tiempo, se hizo que los números de versión partieran de la 6.0, volviendo a la secuencia seguida originalmente por el proyecto Postgres.

PostgreSQL es un proyecto Open Source, esto significa que se puede obtener el código fuente, usar y modificar el programa sin las restricciones que comúnmente encuentra con el software propietario.

Características:

A continuación se enumeran las principales características de este gestor de bases de datos:

- Implementación del estándar SQL92/SQL99.
- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

2.9 Diagrama de Actividad

Un diagrama de actividades es un caso especial de un diagrama de estados en el cual casi todos los estados son estados de acción (identifican que acción se ejecuta al estar en él) y casi todas las transiciones son enviadas al terminar la acción ejecutada en el estado anterior. Puede dar detalle a un caso de uso, un objeto o un mensaje en un objeto. Sirven para representar transiciones internas, sin hacer mucho énfasis en transiciones o eventos externos. Generalmente modelan los pasos de un algoritmo [11].

- **Estado de acción**

Representa un estado con acción interna, con por lo menos una transición que identifica la culminación de la acción (por medio de un evento implícito). No deben tener transiciones internas ni transiciones basadas en eventos (Si este es el caso, represéntelo en un diagrama de estados). Permite modelar un paso dentro del algoritmo.

Se representan por un rectángulo con bordes redondeados.

- **Transiciones**

Las flechas entre estados representan transiciones con evento implícito. Pueden tener una condición en el caso de decisiones.

- **Decisiones**

Se representa mediante una transición múltiple que sale de un estado, donde cada camino tiene un label distinto. Se representa mediante un diamante al cual llega la transición del estado inicial y del cual salen las múltiples transiciones de los estados finales.

Ejemplo de diagrama de Actividad

Diagrama de Actividad Realizar venta

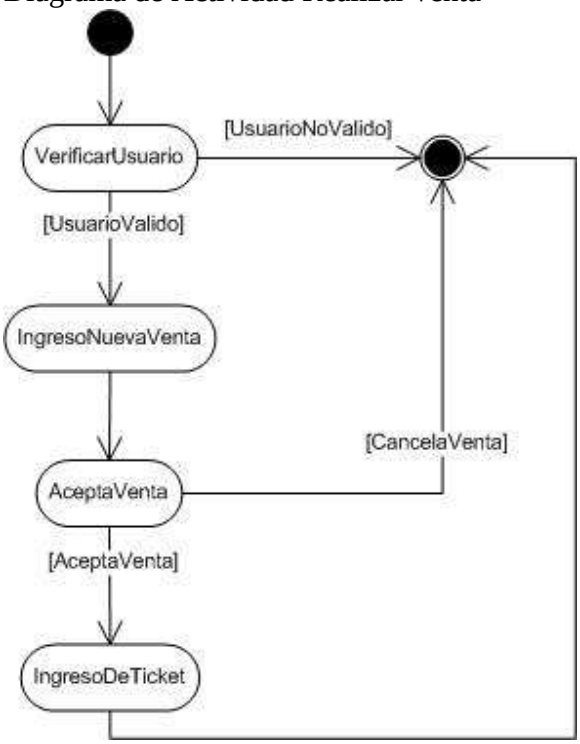


Figura 12: Diagrama de Actividad Realizar venta.

Diagrama de Actividad emitir ticket especial

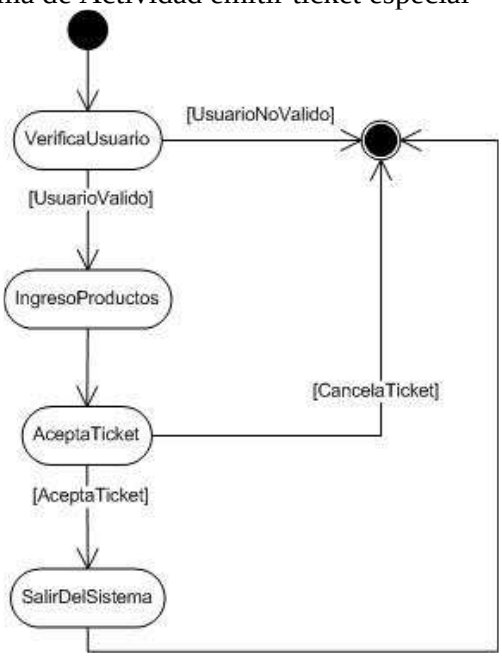


Figura 13: Diagrama de Actividad emitir ticket especial.

Diagrama de entrega producto

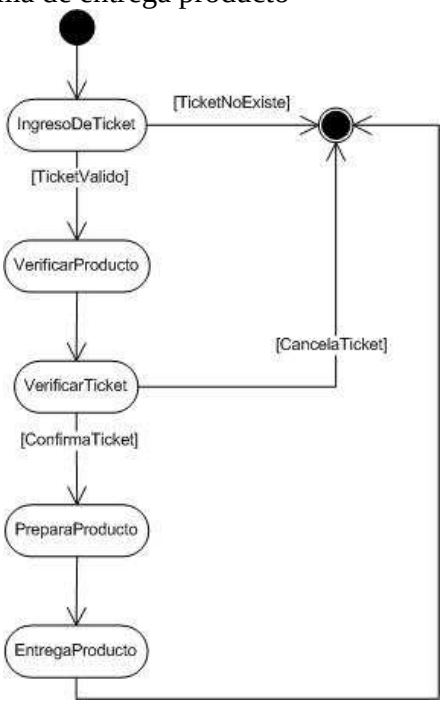


Figura 14: Diagrama de Actividad emitir ticket especial.

Ejemplo de diagrama de Colaboración

Diagrama de Colaboración Mantenedor de usuario

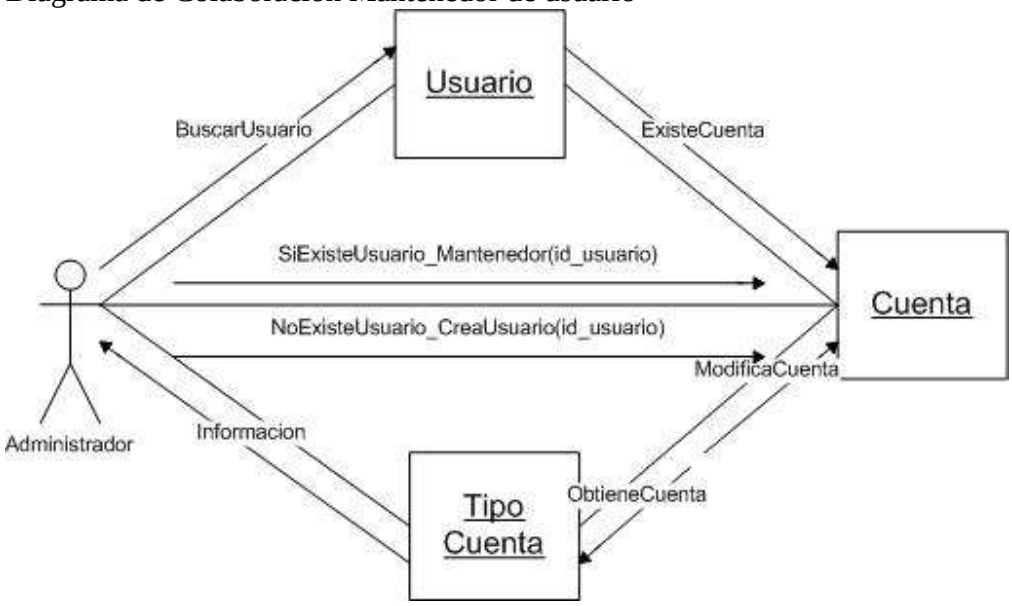


Figura 15: Diagrama de Colaboración Mantenedor de usuario

Diagrama de Colaboración Proceso de Login

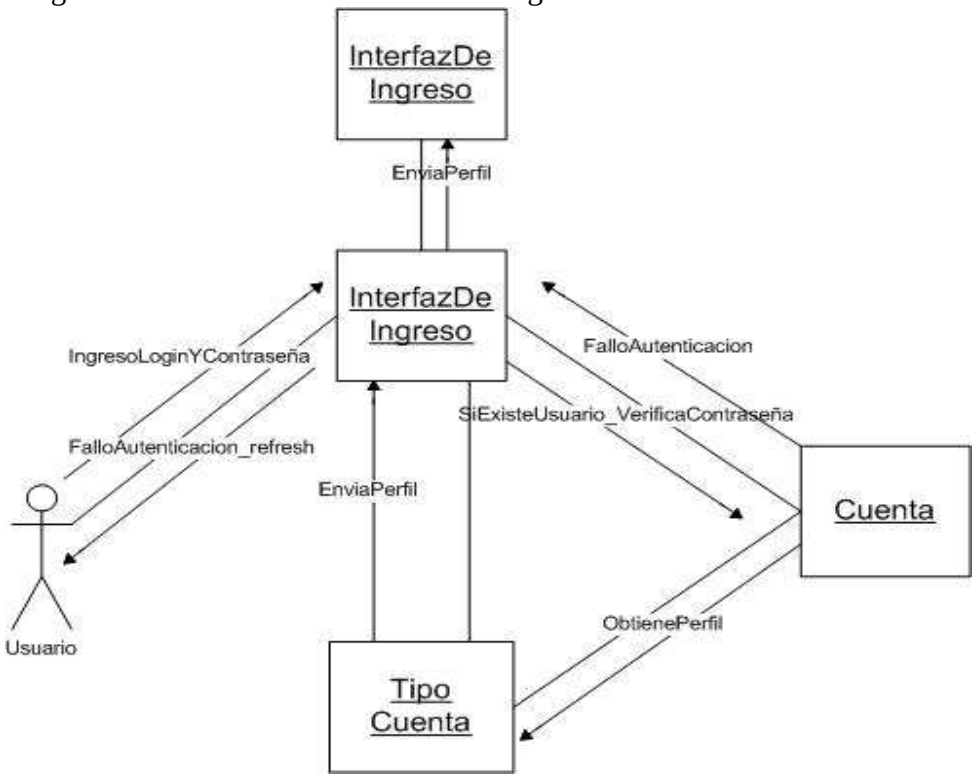


Figura 16: Diagrama de Colaboración Proceso de Login

Ejemplo de diagrama de secuencia

Diagrama de secuencia Realiza venta

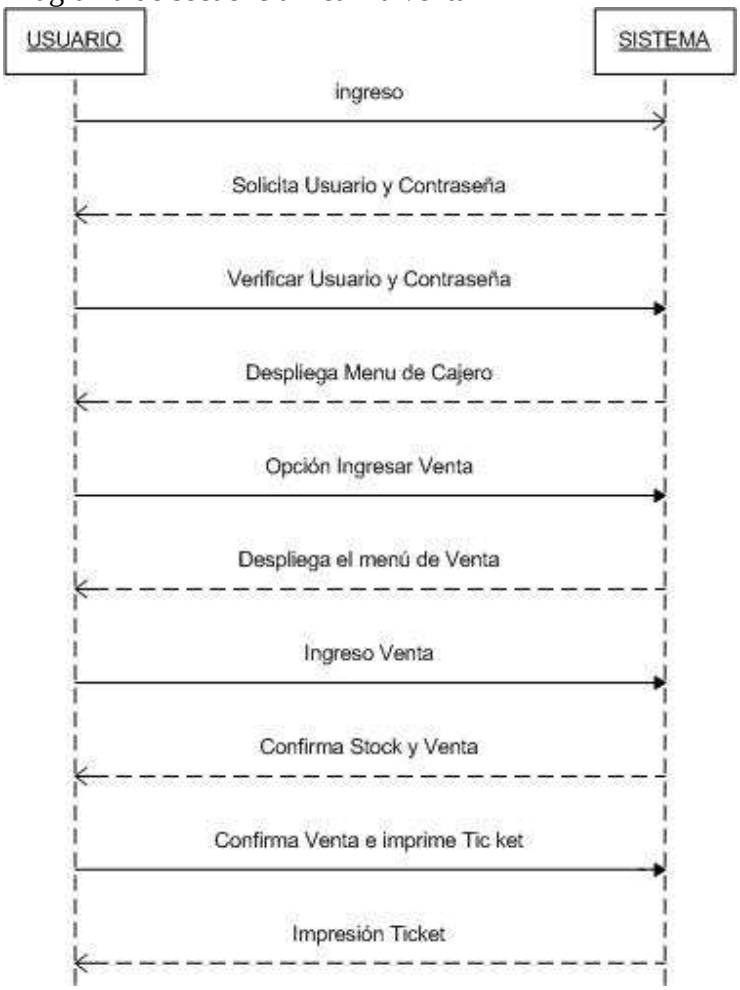


Figura 17: Diagrama de secuencia Realiza venta

Modelo Relacional

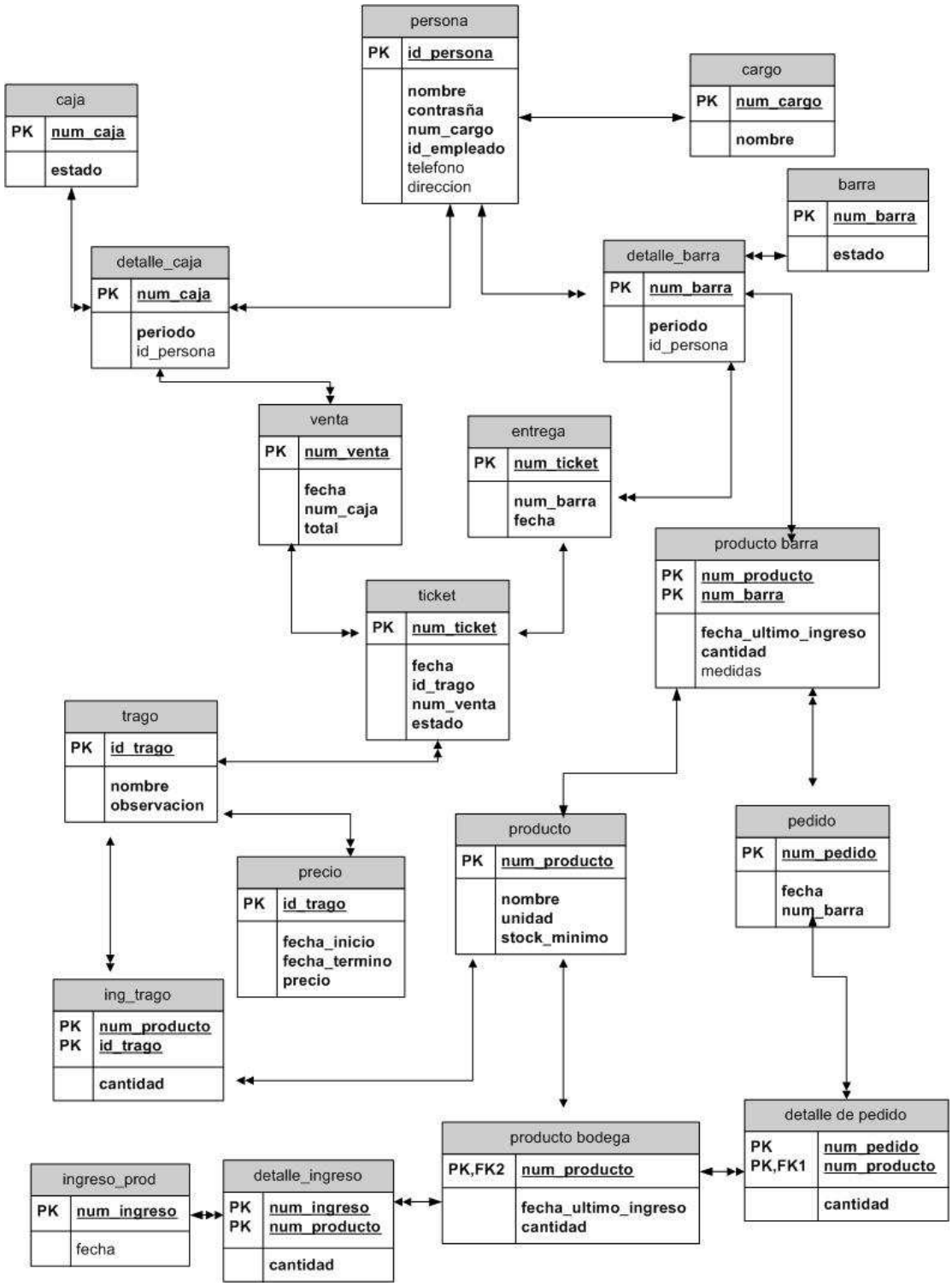


Figura 18: Modelo Relacional

Capítulo 3

Construcción del Sistema.

El objetivo de la fase de construcción de sistema es llegar a definir el plan de construcción.

3.1Plan de Construcción.

El objetivo del plan de construcción es establecer una serie de iteraciones para la etapa de construcción, asignando casos de uso a cada una de éstas. Es importante mencionar que al momento de estimar el tiempo de duración de una iteración, se debe tener en cuenta que por cada una de ellas se debe efectuar análisis, diseño, codificación, pruebas, integración y documentación.

El plan de construcción finaliza sólo cuando todos los casos de uso han sido asignados a una iteración y se han identificado las fechas de inicio para cada una de ellas.

La manera de seleccionar los casos de uso es identificando cuales de estos casos de uso son de mayor prioridad para que un sistema pueda funcionar, o cuales de ellos pueden ocasionar demoras en el desarrollo del proyecto, es por esta razón que los casos de uso seleccionados deben asignarse a una primera iteración y así sucesivamente hasta llegar a la última iteración que es aquella en la cual el sistema podría funcionar por un determinado tiempo sin esas funciones.

3.1.1 Asignación de Casos de Usos.

3.1.1.1Primera Iteración.

- Mantenedor usuario
- Mantenedor producto
- Mantenedor trago

3.1.1.2Segunda Iteración.

- Realizar venta.
- Entregar producto
- Emitir ticket especial.
- Realizar compra.
- Realizar pedido.
- Revisión barra.
- Realizar arqueo.
- Consulta en tiempo real

Es importante mencionar que al estar trabajando con un paradigma que promueve el desarrollo iterativo e incremental, habrá que nuevamente realizar las etapas de análisis, diseño, construcción, etc. para los casos de uso que tienen relación con el componente a construir y verificar si existe algún cambio o alguna agregación en los casos de uso seleccionados que no fue considerada en la etapa de Concepción del Proceso Unificado de Desarrollo.

La idea de revisar todos los diagramas nuevamente es para estar seguro de que se puede pasar a la codificación del sistema, ya que cualquier modificación debiese realizarse en esta instancia (antes de la codificación), y así evitar volver atrás en etapas posteriores, en donde el tiempo empleado es mucho mayor y más costoso.

En cuanto al análisis de las iteraciones ya mencionadas, es importante destacar que mediante una revisión de los casos de uso seleccionados, no se encontraron modificaciones ni agregaciones, por lo que se mantiene lo presentado en el análisis del sistema.

Para el diseño de las iteraciones se revisaron los diagramas relacionados con esta etapa y no existen cambios o modificaciones, por lo que el diagrama de clases presentado en la etapa de diseño se mantiene. A continuación se presentará el modelo de datos del sistema.

3.1.2 Directrices de prueba.

Para lograr un correcto desarrollo de Software, es necesario realizar pruebas con el fin de comprobar la funcionalidad del sistema y con ésto afirmar que lo que se está construyendo es realmente lo propuesto en los requerimientos descritos en etapas anteriores. Una prueba con éxito es aquella que encuentra un error.

Para que estas pruebas cumplan con su finalidad, es necesario encontrar errores o falencias no detectadas hasta ese momento. Lo ideal es que este proceso se realice en la etapa de desarrollo de software, ya que posterior a ésta los costos puede llegar a ser muy elevados.

Existen dos métodos para probar un producto de software:

- Prueba de Caja Blanca (las pruebas se eligen mirando la especificación de lo que se va a probar).
- Prueba de Caja Negra (las pruebas se eligen mirando la estructura de lo que se va a probar).

En el flujo de trabajo de la prueba verificamos el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas a terceros.

Objetivos de las Pruebas:

- Planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las pruebas de sistema son necesarias sólo al final de la iteración.

- Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, creando los procedimientos de prueba que especifican cómo realizar las pruebas.
- Realizar las diferentes pruebas y manejar los resultados de cada prueba sistemáticamente. Las construcciones que detectan defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados.

El resultado final de las pruebas especifica:

- Casos de Prueba: especifican qué probar en el sistema
- Procedimientos de Prueba: especifican cómo realizar los casos de prueba

La prueba da también como resultado un plan de pruebas, evaluaciones de las pruebas realizadas y los defectos que pueden ser pasados como entrada a flujos de trabajo anteriores, como el diseño y la implementación.

3.1.3 Prueba de Caja Blanca.

Este método de casos de prueba se basa en un estudio exhaustivo de los detalles procedimentales, con ésto se pretende probar todos los caminos lógicos de cada modulo abarcando bucles y decisiones lógicas entre otras. En la realidad, ésto es difícil de probar, ya que por muy pequeño que sea el programa, éste consta de un gran número de caminos lógicos.

Existen distintas técnicas de prueba de caja blanca que se detallan a continuación:

- Prueba de camino básico: Esta técnica define un conjunto básico de caminos de ejecución, en donde los casos de pruebas garantizan la ejecución de por lo menos una vez, cada sentencia del programa. Utiliza una notación de grafo de flujo para representar el flujo de control, en donde los nodos del grafo representan una o más sentencias y las aristas representan el flujo de control.
- Prueba de estructura de control: Son variantes que amplían la cobertura de las pruebas y mejoran la calidad de la prueba de caja blanca.
  - Prueba de condiciones: Se definen casos de prueba que ejecuten las condiciones lógicas contenidas en un programa. Existen condiciones simples (una variable lógica o una expresión relacional) y condiciones compuestas (formada por dos o más condiciones simples, operadores lógicos y paréntesis). Si una condición es incorrecta, entonces al menos uno de los componentes está incorrecto, por lo que podemos encontrar errores en operador lógico, paréntesis lógico, operador relacional y expresión aritmética.
  - Prueba de flujo de datos: Selecciona caminos de pruebas de un programa de acuerdo con la ubicación de definiciones y los usos de las variables del programa. Es útil para seleccionar caminos de prueba de un programa que contengan sentencias condicionales o de bucles anidadas. Su efectividad se encuentra en la protección contra errores. Sin embargo la selección de caminos de prueba es más difícil.
  - Prueba de bucles: Se basa en la realización de pruebas a los bucles de un programa. Se pueden definir 4 clases diferentes de bucles:
    - Bucles simples

- Bucles anidados
- Bucles concatenados
- Bucles no estructurados

3.1.4 Prueba de Caja Negra.

Este método de prueba intenta encontrar errores en funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externa, errores de rendimiento, errores de inicialización y de terminación.

Se centran en los requisitos funcionales del software, para lo cual se utiliza un conjunto de condiciones de entrada.

En este método se tienen las siguientes técnicas:

- Partición equivalente: Divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de pruebas. Consta de una clase de equivalencia, la cual representa un conjunto de estados válidos o no válidos para condiciones de entrada.
    - Rango, una clase de equivalencia válida y dos inválidas.
    - Valor, una clase de equivalencia válida y dos inválidas.
    - Conjunto, una clase de equivalencia válida y una inválida.
    - Lógica, una clase de equivalencia válida y una inválida.

- Análisis de valores limites: Complementa la técnica de partición equivalente y la selección de los casos de pruebas se lleva a cabo en los extremos de las clases.
    - Si una condición de entrada especifica un rango delimitado por los valores a y b, se diseñan casos de prueba con valores justo por encima y debajo de a y b.
    - Si una condición de entrada especifica un número de valores, se diseñan casos de pruebas para los valores máximos y mínimos.
    - Los puntos anteriores se aplican a las condiciones de salida.

- Técnica de grafo causa-efecto: El intento de traducir un determinado procedimiento en un lenguaje natural a un algoritmo basado en software conduce a errores, para ello esta técnica proporciona una representación de las condiciones lógicas y sus correspondientes acciones. Se listan las causas (condiciones de entrada) para un componente y los efectos (acciones), asignando un identificador a cada uno de ellos.
- Pruebas de comparación: Consiste en verificar los resultados de aplicaciones que se construyen en forma paralela y por separado para que tengan concordancia.

3.1.5 Selección de las Directrices de Pruebas a Utilizar.

Después de haber descrito cada uno de estos métodos de prueba, ya sea de caja banca o caja negra, se está en condición de seleccionar el enfoque que se va a utilizar para la realización de las pruebas del software.

Se ha elegido el enfoque de caja negra, ya que éste se basa en los requerimientos funcionales del sistema, y se considera que para el sistema a desarrollar las pruebas principales que se deben llevar a cabo para detectar los posibles errores que puede tener el programa, son las pruebas de lo resultados que arroja el sistema dada cierta información.

Para esto se ha decidido utilizar la técnica de grafo causa-efecto, que refleja que con información entregada al sistema, éste, entrega los resultados esperados. Para ello, se debe probar con datos válidos, inválidos o incorrectos, y así verificar que se está procesando la información correctamente. Con el plan de pruebas que se encuentra detallado en el anexo 1 de este documento se verifica lo expuesto anteriormente, y con ésto comprobar que la técnica seleccionada es la más adecuada.

3.1.6 Diseño de Interfaces.

El diseño de las interfaces del sistema no es un tema menor en el desarrollo del proyecto, ya que como se señala en capítulos anteriores y más específicamente en los objetivos planteados, las interfaces deben estar diseñadas de tal manera que el Usuario no tenga mucho más trabajo del que tiene actualmente y también deben ser de fácil manejo. La idea es que el Usuario se sienta parte de este sistema para que posteriormente lo utilice en forma correcta y no se transforme en algo tedioso de utilizar.

Por lo expuesto anteriormente, las interfaces serán las siguientes:

Interfaz de autenticación de usuario:

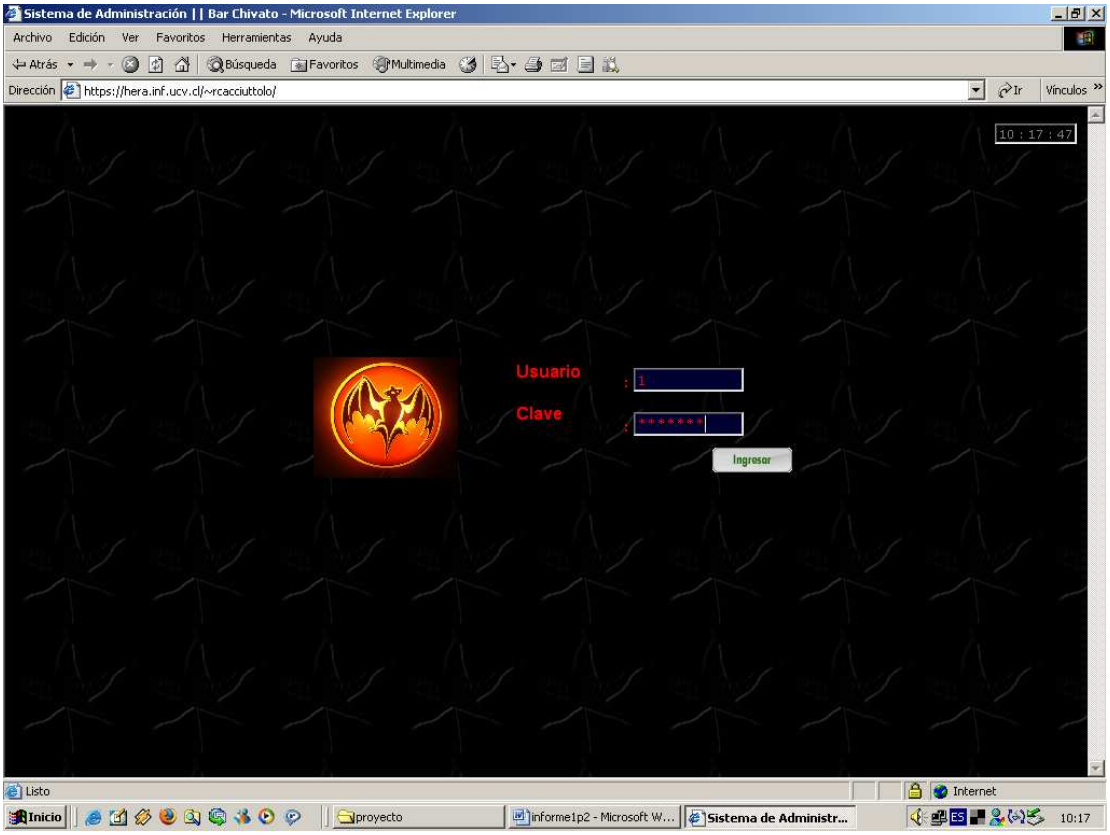


Figura 19: Interfaz de autenticación de usuario

Menu de usuario tipo Administrador.

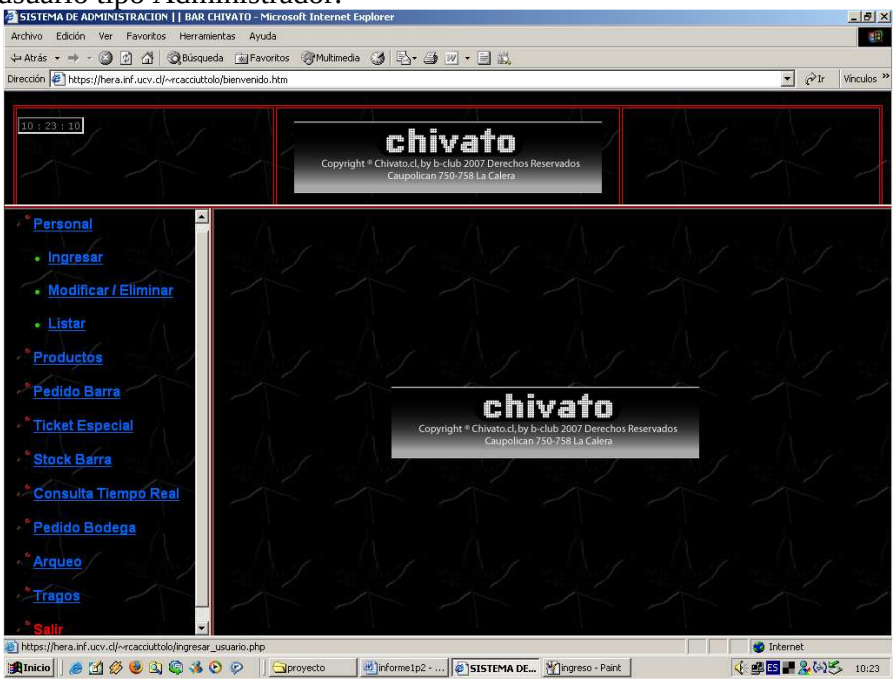


Figura 20: Menu de usuario tipo Administrador.

Interfaz de tipo modificación o eliminación de usuarios

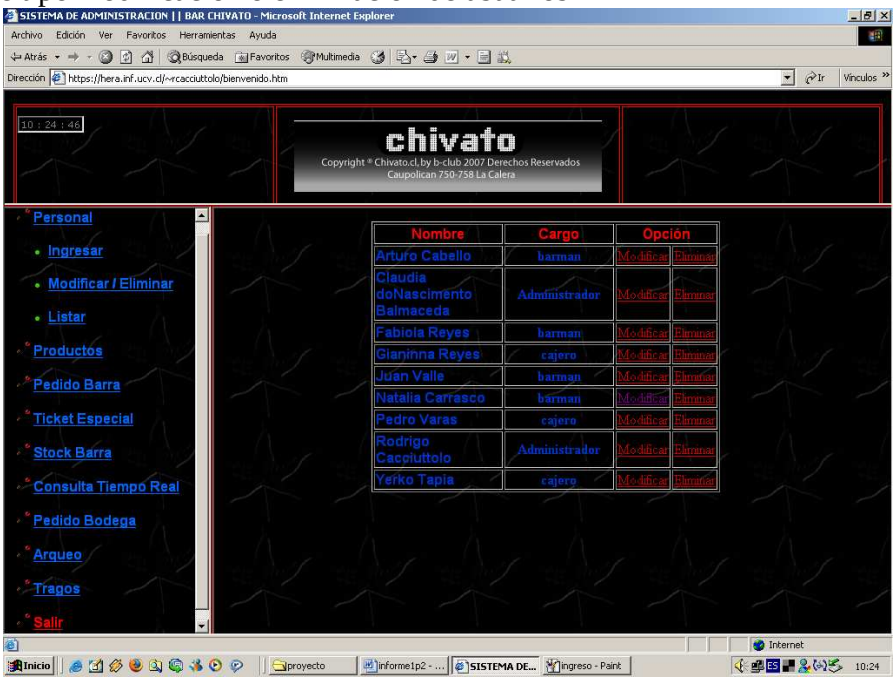


Figura 21: Interfaz de tipo modificación o eliminación de usuarios

Interfaz para el ingreso de usuario

Figura 22: Interfaz para el ingreso de usuario

Interfaz tipo para pedido de producto a bodega desde una barra

· Figura 23: Interfaz tipo para pedido de producto a bodega desde una barra

Interfaz tipo para pedido de producto para bodega

Figura 24: Interfaz tipo para pedido de producto para bodega

Interfaz tipo para modificación de tragos.

Figura 25: Interfaz tipo para modificación de tragos.

Menu de usuario tipo cajero.

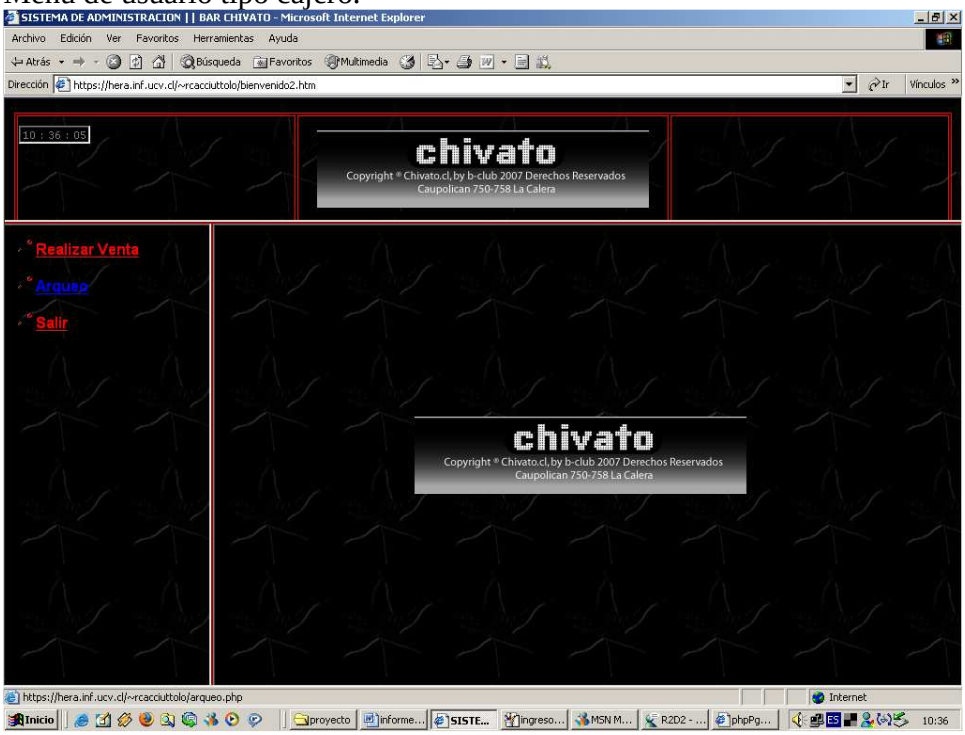


Figura 26: Menu de usuario tipo cajero.

Interfaz que permite la realización de una venta.

Figura 27: Interfaz que permite la realización de una venta

Interfaz de consulta por venta a ingresar

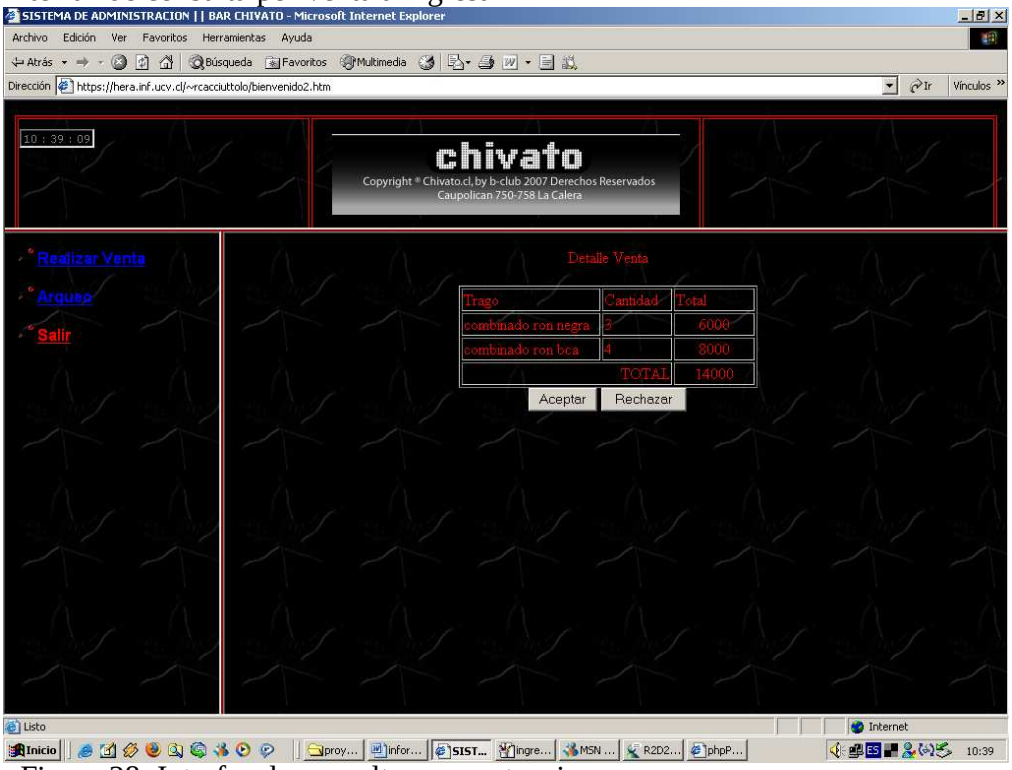


Figura 28: Interfaz de consulta por venta a ingresar

Interfaz de respuesta en consulta de arqueo.

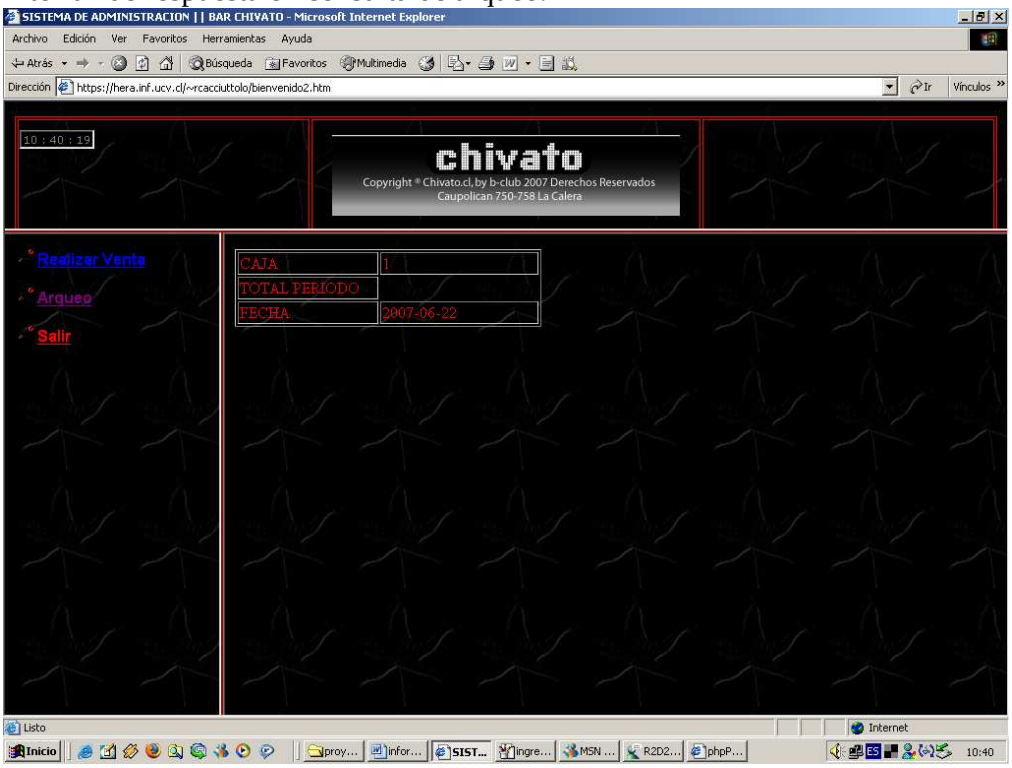


Figura 29: Interfaz de respuesta en consulta de arqueo.

Capítulo 4

Conclusiones y anexos.

4.1 Conclusión.

Al desarrollar un proyecto Informático se definen grandes procesos metódicos que con una buena utilización se llega al producto final. Se debe destacar el importante papel de los desarrolladores en la correcta aplicación de cada una de las herramientas que entregan los diferentes paradigmas para el desarrollo del análisis y diseño. En el caso particular de la metodología OO, junto con UML, ofrecen estándares mundiales o con la idea de llegar a ello, para el desarrollo de las distintas etapas de un proyecto informático, para el análisis y diseño, se destaca la gran calidad gráfica que dan los diferentes tipos de modelos, proporcionando cada uno de ellos una vista de cada componente según el diagrama que sea implementado, permitiendo manejar el sistema en construcción de una manera muy detallada para cada caso, permite un acercamiento claro, preciso y conciso de la arquitectura del proyecto, pudiendo así abarcarlo en etapas y por partes más pequeñas otorgando un mayor entendimiento. Es así como el diagrama de casos de uso permitió establecer una mejor comunicación y un mejor entendimiento de los requerimientos establecidos por el cliente, a su vez fue ventajoso para que el cliente pudiera entender el desarrollo del sistema.

En cuanto a la etapa de construcción del Proceso Unificado de Desarrollo, en donde los principales flujos de trabajo corresponden a la implementación y a la vez a la aplicación de casos de prueba para los componentes del proyecto, las pautas de trabajo fueron bien establecidas desde un principio y nos permiten estar claramente posicionados en esta fase de desarrollo, es muy importante contar con una planificación clara y precisa, para no caer en desajustes que puedan poner en peligro el desarrollo del sistema.

Al trabajar directamente con una empresa el desarrollo del estudio de factibilidad se realizó bajo a información real y fidedigna.

La etapa de requerimientos es una actividad sumamente importante para el desarrollo de un sistema; si esta etapa inicial no es clara y precisa, los resultados posteriores pueden traer consigo grandes riegos incluso el término del proyecto, por lo tanto el buen entendimiento que debe haber entre el cliente y el recolector de requerimientos es fundamental para lograr un sistema que cumpla las características requeridas.

De esta manera se considera que las características del sistema brinda un gran apoyo a las gestiones administrativas dentro la empresa

4.2 Referencias.

[1] Roger S. Pressman, Ingeniería de Software, un enfoque práctico, Quinta Edición.

[2] Héctor Ricardo Acevedo A, Análisis de Sistemas: Una mirada desde el rediseño de procesos organizacionales, Primera Parte (1999), Universidad Técnica Federico Santa María.

[3] Martin Fowler, UML Gota a Gota. Addison-Wesley, 1999.

[4] G.Booch, J.Rumbaugh, I. Jacobson, El Proceso Unificado de Desarrollo de Software. Addison-Wesley, 1999.

[5] G.Booch, J.Rumbaugh, I. Jacobson, El Lenguaje Unificado de Modelado, Addison-Wesley, 1999.

[6] G.Booch, J.Rumbaugh, I. Jacobson, El Lenguaje Unificado de Modelado. Manual de Referencia. Addison-Wesley, 1999.

[7] The PHP Group, http://www.php.net/msnusl/es/preface.php, 19 de Abril de2003.

[8] Joaquín Gracia, http://www.webestilo.com/php/php00.phtml, 17 de Noviembre de 2001.

[9] PostgreSQL., http://advocacy.postgresql.org/advantages/?lang=es.

[10] Diagramas de Uml, http://www.dcc.uchile.cl/~psalinas/uml/casouso.html#casouso.

[11] Diagramas de Uml, http://www.cs.ualberta.ca/~pfiguero/soo/uml.

[12] Diagramas de Uml, http://es.tldp.org/Tutoriales/doc-modebdo-sistemas-uml/multiple-html/x208.html.