UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

REPORTES AUTOMÁTICOS WEB "RAW"

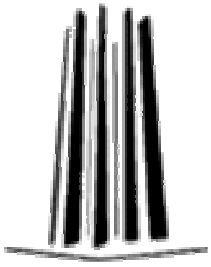
TRABAJO DE TITULACIÓN EN LA MODALIDAD DE DESARROLLO DE UN CASO PRÁCTICO PARA OBTENER EL TITULO DE INGENIERO EN COMPUTACIÓN

PRESENTA

CARLOS OMAR SÁNCHEZ CASTRO

ASESOR

ING. ENRIQUE GARCÍA GUZMÁN



SAN JUAN DE ARAGÓN, ESTADO DE MÉXICO, 2008.

Dedicatorias

Dedico principalmente este trabajo a mi madre Carmen Castro Castro a quien le debo todo lo que soy, quien me enseño a salir adelante, a superar todos los obstáculos que se me presentarán en la vida por muy difíciles que estos fueran y que me brindo la oportunidad de tener una educación profesional.

Y que por designios de Dios nos fuera arrebatada de mi familia y de mi poco tiempo antes de la presentación de este trabajo.

Con su ejemplo aún en sus últimos días, me enseño que nunca hay que rendirse y luchar hasta el final.

Tengo la seguridad de que en el lugar en donde ella se encuentre debe sentirse muy orgullosa de sus hijos.

¡Gracias mamá, por haber tenido la fortuna de ser tu hijo, por haberte conocido, por todos estos años de felicidad que vivimos junto a ti, por todo tu apoyo y amor incondicional!

A mi abuelita Catalina Castro Cornelia, que día a día nos da un ejemplo de fortaleza, amor y dedicación incondicional hacia su familia.

¡Nunca nos dejes, eres lo más importante que nos queda!

A mi hermana Carmen Sánchez Castro, quien siempre me ha dado su apoyo y cariño aún en los momentos más difíciles y obscuros de la vida.

¡Gracias por ser quien eres y nunca olvides que cuentas con tu hermano!

Agradecimientos

Al asesor de este trabajo Ing. Enrique García Guzmán, por su asesoria y apoyo para la realización de este documento.

A los revisores del trabajo:

Ing. Blanca Estela Cruz Luévano
Ing. Juan Gastaldi Pérez
Mat. Luís Ramírez Flores
Ing. Roberto Blanco Bautista

A la Universidad Nacional Autónoma de México y a la Facultad de Estudios Superiores Aragón, por haberme permitido formar parte de esta gran institución como alumno y como profesionista egresado.

Agradecimientos Adicionales

A Farmatel S.A. de C.V. empresa en la que trabajo desde hace varios años y en la que he tenido la oportunidad de desarrollarme como profesionista.

Al gerente de sistemas de Farmatel Ing. Jorge Mercado Ochoa, por todas las facilidades recibidas para la realización de este proyecto.

A mi compañero Ing. Gustavo López García, por el apoyo técnico recibido para la realización del proyecto.

A mi amigo y compañero de carrera Ing. Luís Enrique Montero Hernández, por el apoyo y motivación para iniciar y terminar este trabajo de titulación.

Al profesor y amigo Mat. Andrés Hernández Balderas, por la asesoria recibida para la documentación de este trabajo.

Contenido

Capitulo 3
Desarrollo de la Aplicación

Capitulo 4
Implantación del Sistema

Introducción

La idea principal de este trabajo es la de documentar un proyecto de software desarrollado en la empresa Farmacia Telefonica S.A. de C.V. (Farmatel) y en el cual se da una visión general del proyecto en donde se consideraron los siguientes aspectos:
-Las necesidades propias de la empresa.
-Los objetivos a cubrir.
-Las metodologías que guiaron el análisis y desarrollo.
-Las herramientas que se utilizaron para el desarrollo y documentación.
-Los estándares a los cuales se apego el software.

Farmatel, es una empresa relativamente joven, pero que con sus casi 10 años de haber sido fundada ha logrado posicionarse dentro del competitivo y cerrado sector de distribución de productos farmacéuticos. Sin embargo dentro del mercado en el cual tiene que competir es considerada una empresa pequeña en comparación con otros grandes distribuidores, los cuales por sus grandes volúmenes de ventas pueden darse el lujo de contratar consultorías para sus desarrollos, adquirir software especializado de alto costo y de mantener una gran infraestructura tecnológica.

Por su parte Farmatel al no tener tales volúmenes de venta le ha sido prioritario optimizar sus recursos tanto materiales como humanos. Esto da como resultado que le sea más accesible generar su propio software a la medida de sus necesidades. Y para realizar estos desarrollos de software se apoya en su departamento de sistemas el cual se encarga de analizar, documentar, programar, probar e implementar en un ambiente de producción los sistemas que les sean requeridos.

El sistema generador de Reportes Automáticos Web (RAW), fue un desarrollo a la medida de una necesidad en concreto de Farmatel y la cual fue cubierta satisfactoriamente, lo cual nos abrió una puerta hacia nuevas tecnologías de desarrollo utilizando software libre.

Considerando varios aspectos de este sistema tome la decisión de presentar este trabajo como un Desarrollo de un Caso Practico con la finalidad de obtener el titulo de Ingeniero en Computación.

Es importante mencionar que por cuestiones de confidencialidad y derechos de autor, fue imposible incluir código, datos de clientes o información procesada en forma de reportes.

Por ultimo es importante mencionar que este trabajo esta principalmente dirigido a lectores con conocimientos medio/avanzados en sistemas informáticos, por lo que muchos de los conceptos básicos que son mencionados en este texto no serán explicados.

Objetivos

Los objetivos que el sistema de Reportes Automáticos Web “RAW” deberá cubrir son los siguientes:

- Tener una aplicación 100% funcional en un tiempo corto.
- Hacer un diseño sencillo, que sea fácil de documentar.
- Que permita darle mantenimiento de forma sencilla y rápida.
- Proporcionar una herramienta sencilla de administrar y fácil de entender.
- Que cumpla con los requerimientos mínimos de información para generar los reportes solicitados.
- Que los reportes se puedan exportar a Excel.
- Realice la generación de reportes en tiempo real y sin la intervención humana.
- Que liberé de la actividad de generar reportes a los empleados de Farmatel.
- Información confiable

Resumen

El trabajo esta compuesto de cuatro capítulos que se resumen a continuación:

Capitulo 1: Análisis del sistema para la Generación Automática de Reportes Web: Tiene como objetivo obtener los requerimientos funcionales y no funcionales para desarrollar el sistema de generación de Reportes Automáticos Web, así como definir el documento de requerimientos del sistema en general. En este capítulo se describe la organización de la empresa y se enfatiza principalmente en el área que realiza los reportes que se entregan a los proveedores, también describe de manera general el proceso de requerimientos de software y por último plantea en forma específica el documento de requerimientos del sistema.

Capitulo 2: Diseño del Sistema: Tiene como objetivo proporcionar una visión general del Procesos Racional Unificado “RUP”, del Lenguaje de Modelado Unificado “UML” y del diseño y normalización de la Base de Datos. También en este capítulos se mostraran los diagramas modelados para el sistema de Reportes Automáticos Web como son:“Diagrama Casos de Uso”, “Diagrama de Secuencia”, “Diagrama de Actividades”, “Diagrama de Clases”, “Diagrama de Componentes”, “Diagrama de Paquetes” y por último el “Diagrama Físico de la Base de Datos”

Capitulo 3: Desarrollo de la Aplicación: Tiene como objetivo describir las herramientas y estándares utilizados en el desarrollo del sistema de Reportes Automáticos Web, así como el desarrollo de las interfaces con las que el usuario interactúa. Este capítulo describe el lenguaje de programación utilizado para el desarrollo del sistema, las herramientas utilizadas durante el desarrollo de la aplicación, el entorno donde el sistema se estará ejecutando y el desarrollo de la interfaz grafica de usuario.

Capitulo 4: Implantación del Sistema: Tiene como objetivo describir la manera en como el sistema se migra a producción para que pueda ser utilizado por todos los usuarios. Este capítulo describe los pasos a seguir dentro de la organización, para realizar una migración a un ambiente productivo para una aplicación, desde lo elemental hasta la publicación de la misma en la página Web de la empresa.

Capitulo 1 - Análisis del sistema para la generación de Reportes Automáticos Web

1.1 Antecedentes

1.1.1 Organización

La empresa Farmacia Telefónica S.A. de C. V. “Farmatel”, fue fundada en 1997, con el propósito de dar solución al abasto privado e institucional de los medicamentos especializados, crónicos, de difícil localización, de alto precio o para enfermedades crónicas.

La misión es poner al alcance del paciente los productos de avanzada tecnología, que representan nuevas alternativas para su tratamiento y una mejor calidad de vida.

La visión es ser el mejor sistema personalizado para comercializar y distribuir productos farmacéuticos especializados, proporcionando información y seguimiento oportuno con la finalidad de lograr un mayor apego al tratamiento.

Enfoque

- Facilitar a Clínicas, Hospitales, Centros Médicos, Aseguradoras, Médicos y Pacientes la adquisición de medicamentos de difícil localización
- Brindar apoyo a la población en casos de emergencia
- Lograr un mayor apego a los tratamientos prescritos
- Contribuir a minimizar el impacto social de enfermedades crónicas
- Facilitar al cuerpo médico la prescripción de medicamentos, utilizando el sistema de Receta Digital

Nuestros Servicios

Farmatel suministra medicamentos a diversos segmentos del mercado, optimizando de esta manera los servicios que las instituciones brindan.

Cobertura Nacional en:

- Mercado Privado (publico en general).
- Sector Salud.
- Aseguradoras.
- Hospitales.
- Ventas Institucionales.
- Bancos.

Pacientes

- Le ayuda a cumplir su tratamiento, evitando interrupciones por falta de medicamento.
- Le proporciona material informativo y educativo sobre su padecimiento.
- Ofrece mejores precios permanentemente, a diferencia de las farmacias convencionales.
- Lleva el medicamento a cualquier lugar de la República Mexicana, sin cargo adicional.
- Entrega el pedido entre las 24 y 48 horas posteriores a la confirmación del pago.
- Recibe pagos con tarjeta de crédito, depósito bancario y efectivo.

Médicos

- Le facilita prescribir a sus pacientes telefónicamente a través del sistema Receta Digital.
- Evita el cambio de medicamento por el farmacéutico.

- Le informa mensualmente sobre el apego al tratamiento de cada uno de sus pacientes.
- Le recuerda periódicamente a su paciente la terminación del medicamento y lo motiva a hacer nuevos pedidos, aumentando significativamente el promedio de días de cumplimiento del tratamiento.
- Le da acceso a la información estadística sobre el apego al tratamiento de sus pacientes.

Laboratorios

Farmatel, vende y distribuye medicamento de más de 35 laboratorios de prestigio y da servicio tanto al mercado privado individual, como al sector institucional público.

Laboratorios y sus respectivas áreas terapéuticas:

ELI LILLY
- Diabetes
- Deficiencia en el crecimiento
- Parkinson
- Depresión
- Osteoporosis
- Oncología
- Esquizofrenia

SERONO :
- Salud reproductiva
- Oncología
- Deficiencia en el crecimiento
- Esclerosis múltiple

MERCK SHARP & DOHME
- Anti-inflamatorios
- Antihipertensivos(Presión Alta)
- Osteoporosis
- Hiperplasia prostática
- Asma crónica
- Presión intra-ocular

SCHERING PLOUGH
- Infecciones oseas
- Artritis reumatoide

ASTRA ZENECA
- Asma Bronquial
- Oncología
- Esquizofrenia
- Antimigrañoso

HOLLISTER
- Ostomía
- Urostomía
- Colostomía
- Cuidado de Heridas
- Obstetricia
- Pediatría

FARMASA SCHWABE
- Gastroenterología

BRISTOL-MYERS SQUIBB
- Esquizofrenia

GENZYME
- Enfermedades genéticas

UCB
- Epilepsia

JOHNSON & JOHNSON
- Diabetes

CITIZEN
- Diabetes
- Presión arterial

SERVIER
- Cardiovascular
- Diabetes
- Insuficiencia Venosa

ABBOTT
- Diabetes

AVENTIS
- Artritis reumatoide
- Enfermedades tromboembólicas

ALLERGAN
- Glaucoma

NUTRICIA
- Complementos Alimenticios Pediátricos

PIERRE FABRE
- Oncología

NOVO NORDISK
- Deficiencia en el crecimiento

ASOFARMA
- Osteoartritis

ARMSTRONG
- Alcoholismo

3M DE MÉXICO
- Infecciones por el virus del papiloma humano (VPH)

WYETH
- Artritis reumatoide

MERZ
- Demencia

SCHERING MEXICANA
- Esclerosis múltiple

PFIZER
- Disfunción eréctil
- Presión arterial
- Demencia
- Esquizofrenia

EXAKTA
- Colesterol

GLAXOSMITHKLINE
- Oncología

Promociones

Promociones Vigentes comprando medicamentos a través de Farmatel.

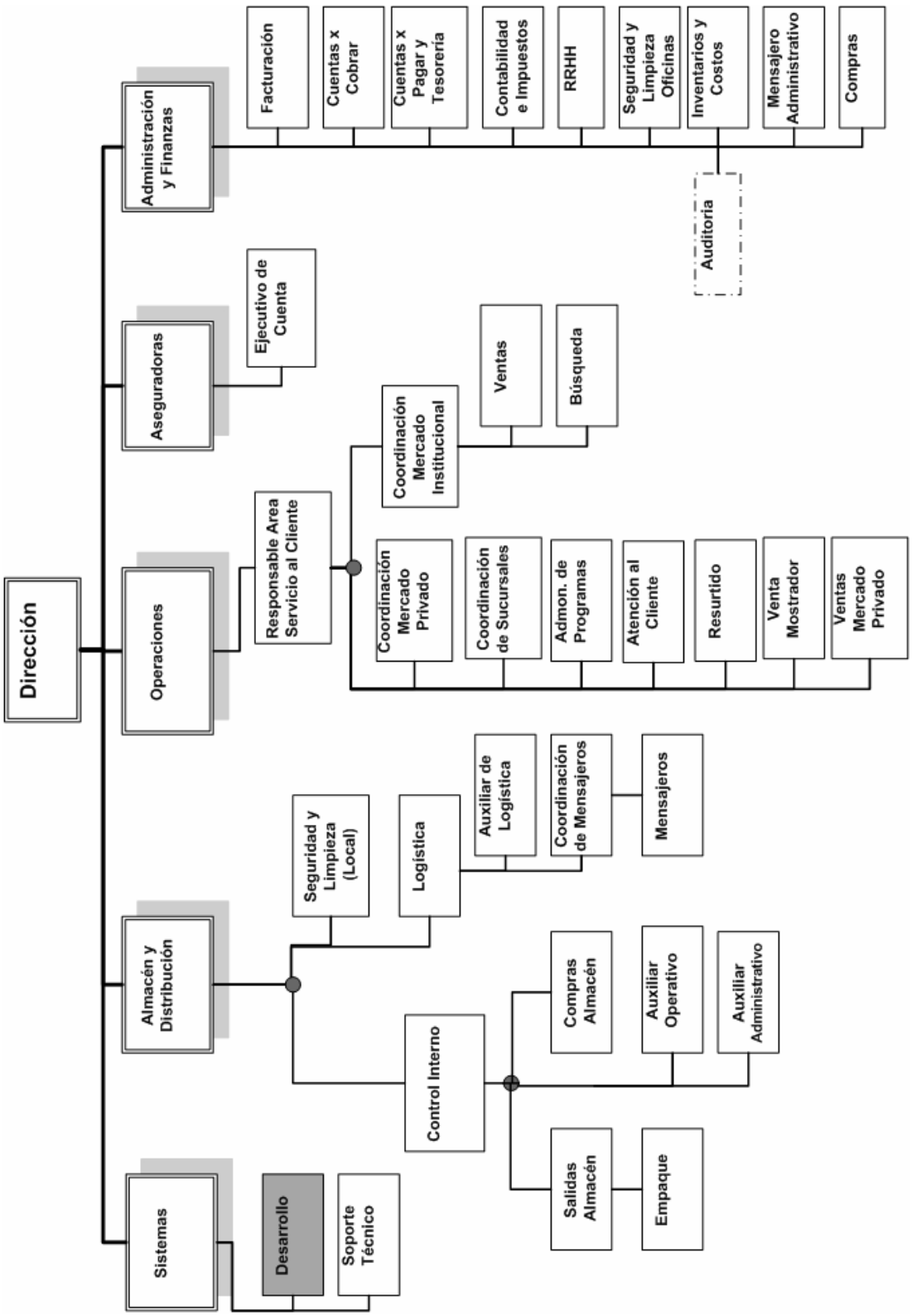
Los descuentos se aplican de acuerdo al siguiente esquema:
- Descuentos por Volumen: Entre más productos se adquieran, mayor es el descuento.
- Descuentos por Producto: Dependiendo del producto, se aplica un descuento.
- Descuentos por Compras Acumuladas: Se regala un producto, o se otorga un descuento por comprar un determinado número de productos.
- Promoción por Paquetes: Se hace un descuento por comprar un paquete de productos, o un tratamiento.

Sucursales

- Guadalajara
- Aguascalientes
- Monterrey
- Puebla
- Mérida

Organigrama de la Empresa

En el siguiente imagen se muestra como esta formada la estructura de departamentos de la empresa Farmatel y en la cual se puede observar el área de sistemas a la cual pertenece el desarrollo del sistema.



1.1.2 Descripción general de las funciones del Departamento de Sistemas

El Departamento de Sistemas pertenece estructuralmente al grupo de Soporte/Apoyo de la empresa, y busca mediante una filosofía orientada a resultados y con vocación de servicio brindar lo mejor a su alcance para contribuir con los objetivos empresariales.

Nuestros principales objetivos son:

1) Brindar soporte técnico/operativo oportuno a todas las áreas de la empresa y a las sucursales concesionadas. Atendiendo el 100% de las solicitudes, comunicando oportunamente las resoluciones.
2) Mantener la infraestructura tecnológica de la empresa en óptimas condiciones.
3) Resguardar y respaldar los sistemas de Información, periódicamente.
4) Llevar los beneficios de la tecnología con implementaciones innovadoras que cubran las necesidades de cada área de la empresa, apoyando en el logro de los objetivos empresariales.
5) Aplicar metodologías de análisis de los procesos que permitan automatizar y brindar sistemas de información que satisfagan las necesidades de la empresa.
6) Contar con personal calificado en cada área del departamento para cumplir con los resultados de cada actividad
7) Optimizar los recursos materiales y financieros que se nos otorguen.

Canales y Formas de Comunicación

Los canales oficiales de comunicación hacia el departamento de sistemas, son:

1) Solicitud “Vía Intranet”.
2) Solicitud “Personal” al responsable de la sección involucrada en el requerimiento.
3) Solicitud “Vía Correo Electrónico” a la sección involucrada en el requerimiento.
4) “Aviso telefónico” de incidencia urgente.
5) Solicitud “Vía mensaje” al responsable de la sección involucrada en el requerimiento.

Observaciones:

- Las solicitudes recibirán un seguimiento de acuerdo a su naturaleza, hasta que queden atendidas (atendida puede significar: no procede o resuelta)

- La máxima autoridad del área es la Gerencia y es quién autoriza las acciones inherentes a una solicitud compleja o que implique afectaciones a los Sistemas, Infraestructura o Servicios a su cargo.

- En toda reunión que implique cambios, soluciones, requerimientos; se generará una minuta estándar. Esto permitirá dar seguimiento adecuado y oportuno.

- Los sistemas de información nuevos se documentarán de acuerdo a la Metodología del Departamento de Sistemas (basada en los principios de “Rational Unified Process” y utilizando diagramas de “Unified Modeling Language”). Está puede ser de dos tipos:

1) Documentación Lite, contempla: Minutas, Documento de Visión General, Casos de Uso de Alto Nivel, Manual de Usuario, Diagramas de Bases de Datos y Arquitectura, Pruebas de usuario, Capacitación.

2) Documentación Completa, contempla: Minutas, Documentación de Visión General, Plan de Trabajo, Plan de Riesgos, Especificación complementaria, Casos de Uso, Diagramas de Bases de Datos, Documento de Arquitectura, Documento de diseño, Esquema de Capacitación por Nivel, Documentación de Pruebas y Control de Errores, Manuales de Usuario, Manual de Requerimientos e Instalación.

- En la sección de Anexos, se especifican los Formatos existentes para Soporte, Documentación, Desarrollo, Requerimientos, Actividades, Asuntos Pendientes y Nuevos Proyectos.

Responsabilidades inherentes al Departamento de Sistemas

Actualmente el Departamento de Sistemas, es responsable de:

- Mantener en óptimas condiciones el equipo de cómputo.
- Administrar y dar Soporte Técnico al Hardware/Software, Comunicaciones, Sistemas de Información.
- Controlar, Soportar y apoyar los requerimientos de Telefonía.
- Respaldar y Resguardar los datos de los Sistemas de Información y archivos electrónicos de las áreas críticas.
- Resguardar el software y cumplir con licenciamiento.
- Analizar requerimientos de hardware/software y dar solución.
- Analizar los procesos de las áreas de la empresa para contar con Sistemas de Información que resuelvan sus necesidades; en base a datos y objetivos concretos.
- Administrar el Sitio Web, Correo Electrónico, Usuarios, Dominios.
- Administrar la comunicación e intercambio de información electrónica con las Sucursales concesionadas.

1.1.3 Que son y como funciona la generación de reportes para Laboratorios/Proveedores

Los reportes para Laboratorios/Proveedores están formados con información que se muestra agrupada en columnas y renglones, creando así una tabla. Alguna información es mostrada con una fila o columna de totales al pie de los datos, y en algunos casos los reportes deben ir acompañados con gráficos de tipo “Pastel” o “Barras”.

Los reportes se generan periódicamente dependiendo del Laboratorio/Proveedor. Los periodos que se utilizan son semanales, quincenales, mensuales o anuales (aunque la mayoría se realizan mensualmente).

La información que se utiliza para la elaboración de los reportes se obtiene de los sistemas con los que cuenta la empresa:

Sistema Farmatel

- Toma de pedidos para clientes
- Administración productos
- Administración de promociones, obsequios y bonificaciones.
- Genera sus propios reportes

Sistema Remoto Farmatel

- Administración de Inventario
- Administración de Recetas
- Administración de compras a proveedores
- Surtido a clientes
- Surtido a sucursales
- Facturación para sucursales
- Autorización de pedidos
- Genera sus propios reportes

Sistema de Ventas

- Facturación de Pedidos locales y de pedidos Institucionales
- Genera sus propios reportes

Sistema de Entregas

- Asignación de pedidos a mensajeros
- Asignación de pedidos a compañías de Mensajería
- Registra los tiempos de entrega a los clientes
- Controla los reenvíos de pedidos
- Genera sus propios reportes

Los reportes que se generan para los Laboratorios/Proveedores son los siguientes:

Bonificaciones entregadas
Bonificaciones entregadas - Efexor
Bonificaciones entregadas / Aseguradoras
Compras por Paciente
Compras por Paciente - Efexor
Compras por Paciente / Aseguradoras
Compras por Paciente por Período
Distribución de producto por sucursal
Distribución de producto por sucursal - Efexor
Eventos Adversos
Existencia de producto
Material Educativo/Informativo, entregado
Material Educativo/Informativo, entregado - Efexor
Médicos
Médicos - Efexor
Movimiento de piezas
Movimiento de piezas - Efexor
Pacientes
Pacientes - Efexor
Pedidos por Estado
Pedidos por Estado - Efexor
Pedidos y productos por Estado
Pedidos y productos por Estado - Efexor
Piezas vendidas por representante
Piezas vendidas por representante - Efexor
Piezas vendidas por representante / Aseguradoras
Productos Vendidos
Prueba
Reporte
Reporte de Altas de Folios
Reporte de pacientes activos
Reporte de precios
Reporte de Ventas Año y Mes
Venta de producto por representante
Venta de producto por representante - Efexor
Venta de producto por representante / Aseguradoras
Venta en Mostrador de Sucursales
Ventas a detalle DDD
Ventas a detalle DDD - Efexor
Ventas a detalle DDD / Aseguradoras
Ventas por Estado
Ventas por Estado - Efexor
Ventas por Estado y por mes
Ventas por Estado y por mes - Efexor
Ventas por mes
Ventas por mes - Efexor

Ventas por presentación
Ventas por presentación - Efexor
Ventas por Producto x Mes
Ventas por producto y por mes
Ventas por producto y por mes - Efexor
Ventas por producto y por mes / Aseguradoras
Ventas por representante
Ventas por representante - Efexor
Ventas por representante / Aseguradoras

1.2 Planteamiento del Problema

Periódicamente la Empresa debe entregar a sus “Laboratorios/Proveedores” información de diferente índole, la cuál les es de utilidad para conocer y evaluar sus ventas, clientes, productos, promociones, etc. La generación de dicha información es de vital importancia para la Empresa, ya que es parte de las ventajas que se ofrece frente a la competencia. Es decir los reportes que se generan para los Laboratorios/Proveedores son parte del servicio que se vende.

La información que se maneja es mucha y su uso es muy delicado, por lo que se debe tener en cuenta que la confidencialidad de la información es de alta prioridad, ya que por cuestiones legales y de Competencia Desleal (Piratería de Información), se debe tener mucho cuidado con la información que es entregada y sólo a las personas autorizadas. Ya que si se hace entrega de información que viole la privacidad de los clientes y/o a personas no autorizadas, esto pondría a la Empresa en una posición de alto riesgo.

También se debe considerar que la Empresa no tiene asignado a un recurso que se dedique exclusivamente a la generación de estos reportes, por lo tanto estos son elaborados por empleados de diferentes áreas (Departamento de Mercado Institucional, Departamento Mercado Privado y Departamento de Sistemas). Los reportes son generados de forma programada y excepcionalmente a petición de los Laboratorio/Proveedores, pero en cualquier caso esta actividad se realiza descuidando sus actividades principales, lo que deriva en retrasos y en que los reportes tengan un margen de error significativo.

Lista de beneficios:

- Automatizar una tarea de vital importancia.
- Mejorar el servicio hacia los clientes.
- Aprovechar de mejor forma los recursos de la Empresa.
- Obtener más provecho de Internet para promover y mejorar la imagen de la Empresa.

1.3 Análisis de Requerimientos

1.3.1 Definición de Requerimientos Funcionales y No Funcionales.

1.3.1.1 ¿Qué son los requerimientos Funcionales?

Los Requerimientos Funcionales definen el tipo de servicio esperado del Sistema, así como el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos, validaciones y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica.

Los requerimientos funcionales cubren las funciones y operaciones a realizar para proporcionar un sistema que operará de acuerdo a las necesidades del usuario y de la compañía. Al elaborar una lista completa de las percepciones de los usuarios respecto a sus requerimientos,

se definen las funciones que tendrán que ser realizadas por el sistema a desarrollar y son complementadas por los requerimientos no funcionales.
.
En resumen 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.

1.3.1.2 ¿Qué son los requerimientos No Funcionales?

Los requerimientos no funcionales tienen que ver con las características que de una u otra forma puedan limitar al sistema como son: el rendimiento (en tiempo y espacio), confiabilidad, interfaces, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.

Es importante aclarar que no todos los "Requerimientos No Funcionales" fueron considerados y/o aplicados para este desarrollo, debido a diferentes motivos, tales como: tiempo de desarrollo, poca complejidad del sistema y tamaño moderado del sistema de "Reportes Automáticos Web". Pero que de todas formas serán expuestos a continuación para dar un mejor panorama de este tema tan amplio.

1.3.1.2.1 Usabilidad

La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.
A partir de la conceptualización llevada a cabo por la ISO, se infieren los principios básicos en los que se basa la usabilidad:

- **Facilidad de Aprendizaje**: facilidad con la que nuevos usuarios desarrollan una interacción efectiva con el sistema o producto. Está relacionada con la predictibilidad, sintetización, familiaridad, la generalización de los conocimientos previos y la consistencia.

- **Flexibilidad**: relativa a la variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información. También abarca la posibilidad de diálogo, la multiplicidad de vías para realizar la tarea, similitud con tareas anteriores y la optimización entre el usuario y el sistema.

- **Robustez**: es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos. Está relacionada con la capacidad de observación del usuario, de recuperación de información y de ajuste de la tarea al usuario.

1.3.1.2.2 Confiabilidad

La confiabilidad de software significa que un programa particular debe de seguir funcionando en la presencia de errores. Los errores pueden ser relacionados al diseño, a la implementación, a la programación, o el uso de errores. Así como los sistemas llegan a ser cada vez más complejos, aumenta la probabilidad de errores.

1.3.1.2.3 Seguridad

La seguridad en el software se enfoca principalmente en cubrir los siguientes puntos:

- **Privacidad**. Es la habilidad de controlar quien puede (o no puede) ver la información. Las transacciones deben permanecer privadas e inviolables en el sentido de que entidades no autorizadas no puedan descifrar el contenido de los mensajes.
- **Integridad**. Es la seguridad de que los datos almacenados o transmitidos no son alterados. Se debe asegurar que las transmisiones no son alteradas o interferidas.

- **Autenticidad**. Es la habilidad de determinar la identidad de las partes que se comunican.
- **No repudiación**. No debe ser posible que un emisor de un mensaje pueda alegar que no envió una comunicación segura o que no realizo una compra.

1.3.1.2.4 Desempeño

El análisis de desempeño es requerido en todas las etapas del ciclo de vida de un sistema, incluyendo la etapa de funcionamiento en la que éste ayuda a determinar que tan bien está funcionando y que mejoras (en caso que haya lugar) deben hacerse a un sistema.
Se debe especificar el tiempo de respuesta para una transacción (promedio), capacidad (número de usuarios y transacciones), rendimiento del procesamiento (ej. Transacciones x segundo) y cuando el sistema se ha degradado cuál es el modo aceptable de operación.

1.3.1.2.5 Mantenimiento y Actualización (Modificabilidad)

La capacidad de mantenimiento es la habilidad que se tiene para realizar cambios al producto en el tiempo y la capacidad que se tiene para entregar las versiones del producto a bajo costo a los clientes con un mínimo de tiempo. Una característica clave para apoyar este objetivo es la descarga automática de parches o actualizaciones y actualizaciones del equipo del usuario final. También debemos utilizar formatos para archivos de datos que incluyan suficientes metadatos para permitirnos trasformar con seguridad la información existente del usuario durante una actualización.

1.3.1.2.6 Soportabilidad y Operabilidad

La soportabilidad es la habilidad de proveer soporte técnico eficiente y a buen precio y la operabilidad es la habilidad que se tiene de hospedar y operar el software como un ASP (Proveedor de Servicios de Aplicaciones).

1.3.1.2.7 Restricciones de Diseño

Este apartado se refiere a cualquier limitación de diseño en el sistema que se ha construido, es decir todas las decisiones de diseño que se han asignado de forma obligatoria. Por ejemplo: lenguajes de software, requerimientos del proceso de software, uso de herramientas de desarrollo, componentes comprados, etc.

1.3.1.2.8 Requerimientos de Documentación en Línea y de Sistemas de Ayuda

En caso de que exista se debe describir los requerimientos, para la documentación en línea del usuario, sistemas de ayuda, ayuda sobre avisos, etc.

1.3.1.2.9 Interfaz

Este apartado se refiere a las interfaces que deben apoyar a la aplicación, como son: las interfaz de usuario, interfaz de software, interfaz de hardware. Para que el software pueda ser creado y probado contra los requerimientos de la interfaz, debe incluir la especificidad adecuada protocolos, puertos, direcciones lógicas, etc.
En caso de que aplique es conveniente hacer referencia a estándares de la aplicación o Corporativos.

- **Interfaz de Usuario**

  Son las descripciones de las interfaces de usuario que van a hacer ejecutadas por el software.

- **Interfaz de Software**

  Son las interfaces de software hacia otros componentes del sistema. Pueden ser: componentes comprados, reutilizados o realizados para subsistemas fuera del alcance de este documento.

- **Interfaz de Hardware**

Son las descripciones y comentarios de cualquier interfaz de hardware que debe ser apoyada por el software, esto incluye: comportamiento, estructura lógica, etc.

- **Interfaz de Comunicaciones**

Son las definiciones de las interfaces de comunicaciones a los demás sistemas o dispositivos como: redes LAN y dispositivos seriales remotos.

1.3.1.2.10 Aspectos Legales

Este punto define las cuestiones legales que pudieran afectar esta entrega. No considerar cuidadosamente estas cuestiones puede poner a la organización de desarrollo en riesgo de una acción legal. Se deberá buscar consejo profesional si es necesario.

- **Políticas de la Organización**

En este punto se deben responder la siguiente pregunta: ¿El producto satisface las políticas de la organización (por ejemplo, de privacidad y seguridad)?
Sí. Describa cómo se satisfacen cada una de estas políticas.
No. Describa los pasos a tomar para hacer que el producto las cumpla.
No. No hay políticas que apliquen.

- **Contratos con Otras Organizaciones**

En este punto se debe responder la siguiente pregunta:
¿Fue algún componente o información producida por otra organización bajo contrato?
Sí. Revise los detalles del contrato para derechos de propiedad y licenciamiento.
No. No se requiere hacer nada al respecto.

- **Propiedad Intelectual**

En este punto se deberán especificar los aspectos correspondientes al software utilizado en nuestro sistema:

*Componente
*Dueño
*Licencia
*Estado
*Comentarios

1.3.1.2.11 Estándares Aplicables

Son las descripciones por referencia cualquier estándar aplicable y las secciones específicas de dichos estándares que apliquen al sistema, como son: estándares de calidad, aspectos legales, interoperabilidad, internacionalización, estándares de seguridad de la información, compatibilidad del sistema operativo, etc.

1.4 Documento de Requerimientos para el Sistema de Reportes Automáticos Web “RAW”

A partir de este punto y hasta el final del capitulo 1, se presenta el documento de “Especificación de Requerimientos”, es necesario mencionar que se trata de un documento de uso común dentro del área de desarrollo de Farmatel y el cual fue diseñado tiempo atrás.

Reportes Automáticos Web “RAW”
Especificación de requerimientos

Versión 1.0

Control de versiones

Fecha	Versión	Descripción	Autor
20/02/2006	1.0	Primer iteración	OSC

La tabla anterior se utiliza para llevar un control de las versiones del documento de “Especificación de Requerimientos” y es actualizado por el analista cada vez que sufre un cambio.

A continuación se muestra el índice correspondiente al documento de requerimientos y para evitar que se confunda con el índice general del trabajo, se opto por cambiar la numeración a números romanos.

Especificación de requerimientos

I. Introducción.

I.1 Propósito.

El propósito de este documento es el de plasmar los requerimientos Funcionales y no Funcionales del sistema de Reportes Automáticos Web “RAW”.

I.2 Alcance.

El sistema de Reportes Automáticos Web “RAW” esta dividido básicamente en 2 secciones, la administración en un ambiente Windows y la generación propia de los reportes que se visualizan desde un navegador de Internet.

- Administración del sistema “RAW”: Este módulo se encarga del control, configuración, diseño, etc. de los reportes que se generan, así como de la administración de usuarios, permisos, etc.

- Generación de Reportes: Este módulo se encarga de la generación de la información que se visualizara en el reporte, así como de la presentación final (aplicando formato, colores, imágenes, graficas, títulos, resúmenes, etc.) que se le entregara al cliente.

I.3 Definiciones, acrónimos y abreviaturas.

Cliente: empresa, institución o persona que requiere información.

Usuario: usa el sistema, y genera reportes con su información. Está asociado a un cliente.

Administrador: Usuario con privilegios el cual se encarga de administrar las definiciones de las Plantillas/Reportes así como de los usuarios y demás catálogos del sistema.

Plantilla: Formato base con el cual se generan los Reportes Web

I.4 Referencias.

RUP: Proceso Unificado de Rational

UML: Lenguaje unificado de Modelado

SQL Server: Servidor de Base de Datos

MySQL: Base de Datos relacional libre

JAVA: Lenguaje de Programación

I.5 Visión general.

La especificación de requerimientos para el sistema de Reportes Automáticos Web “RAW”, va a servir para definir todos los puntos clave que el software debe cubrir. Esto también puede servir para que en un futuro las personas que tengan que darle mantenimiento al sistema, tenga una visión clara de cómo y por que se diseño el sistema de esta forma. Hay que dejar bien claro que nada que no este especificado como un requerimiento, no se deberá desarrollar.

II. Casos de uso.

II.1 Iniciar el sistema de administración.

El Usuario Administrador iniciará el módulo de administración del sistema RAW, para lo cual deberá ingresar su usuario y contraseña. Entonces el sistema validará los datos proporcionados y si todo es correcto, entonces se iniciará la aplicación.

II.2 Administrar Usuarios.

El usuario Administrador ingresa a la aplicación administrativa del sistema RAW, después ingresa al módulo de Catálogo de usuarios, en donde podrá dar de alta usuario, editar usuarios existentes y por supuesto borrar usuarios.

II.3 Administrar Clientes.

El usuario Administrador ingresa a la aplicación administrativa del sistema RAW, después ingresa al modulo de Catálogo de Clientes, en donde podrá dar de alta, editar y por supuesto borrar.

II.4 Administrar Plantilla de Reporte.

El usuario Administrador inicia la aplicación de Administración del sistema RAW, ingresa validando sus datos. Después ingresa al módulo de Plantillas en donde podrá generar una nueva plantilla, para lo cual ingresará la consulta SQL con la cual se generará el reporte, después definirá el formato de cada una de las columnas, los títulos de las columnas, se definirán los parámetros, se seleccionará el tipo de grafica, y una vez que se concluyen todos estos pasos, entonces se podrá guardar para lo cual será necesario ingresar el nombre de la nueva plantilla.

II.5 Agregar plantillas a Cliente.

El usuario Administrador inicia la aplicación de Administración del sistema RAW, ingresa validando sus datos. Después ingresa al modulo de Plantilla/Clientes, consulta a los Clientes, selecciona de la lista a un Cliente en especial y este se le asigna una plantilla en especial. Después de que se hayan hecho todas las asignaciones necesarias se guardan los cambios, para lo cual el sistema pregunta si se esta seguro de los cambios, si se responde que si entonces el sistema actualizará los datos en la base de datos y cerrará la pantalla de Plantillas/Clientes.

II.6 Firmarse en el sistema vía Web.

Un cliente se conecta al sistema vía web con la siguiente dirección “https//www.farmatel.com.mx/reportes”, para lo cual su navegador le mostrara una

página web en donde se le solicitarán sus datos “usuario” y “contraseña”, después de que haya capturados sus datos el cliente presionará el botón de “Login” y el sistema web validará los datos y si son correctos el navegador le desplegará una nueva pantalla en donde se le da la bienvenida, se le indica que ha iniciado una nueva sesión y se le mostrara una serie de links para generar los reportes que en su caso tenga asignados según su perfil.

Después de que haya generado los reportes pertinentes y para cerrar su sesión sólo deberá presionar el botón que dice “Logout” y el navegador le mostrará una pantalla en donde se le indicará que la sesión a terminado.

II.7 Generar reporte vía Web.

Después de que el Cliente se ha firmado en la página de “https://www.farmatel.com.mx/reportes” y el sistema por su parte ha validado sus datos y le ha creado una nueva sesión, entonces el Cliente podrá generar el/los reporte(s) que desee, lo único que tiene que hacer es seleccionar de una lista de reportes disponibles el reporte requerido. Una vez que se haya seleccionado el reporte, el sistema mostrará una nueva página en donde se desplegarán los campos con los criterios para generar el reporte (esto en caso de que el repote lo necesite), después de que el Cliente ingresó los criterios de búsqueda, se deberá hacer clic en el botón de “Generar Reporte”, para los cual el sistema iniciará la generación del reporte y lo desplegará en pantalla.

Cabe mencionar que el tiempo para la generación del reporte dependerá en gran medida de la cantidad de información a consultar, la velocidad de la conexión a Internet y por supuesto de las capacidades propias del equipo local.

II.8 Exportar reporte Web a Excel.

Después de que el Cliente haya generado el reporte vía web, entonces se habilitará el botón para exportarlo a Excel, una vez que se haga clic sobre este botón, el sistema desplegará una ventana de diálogo en donde el Cliente deberá capturar el nombre que recibirá el archivo y la ruta en donde se guardara localmente. Una vez terminada la exportación a Excel, el sistema mostrará una pantalla de diálogo en donde indicará que el archivo de Excel se generó con éxito.

III. Funcionalidad.

III.1 El usuario siempre debe saber el estado que guarda su reporte, aún si este no es generado.

Identificador	1
Características	El sistema debe mandar algún tipo de notificación en caso de error, progreso y finalización del proceso de generación del Reporte.
Prioridad	Media

III.2 Sólo se necesita un usuario administrador.

Identificador	2
Características	El sistema no es muy grande ni complejo por lo que un solo Administrador podrá manejar el sistema.
Prioridad	Media

III.3 Para poder ingresar al módulo de Administración, el Usuario Administrador deberá introducir una contraseña.

Identificador	3
Características	Debido a que la información que se genera en de uso confidencial, es importante que se deba validar el ingreso al sistema por medio de un Usuario y Contraseña.
Prioridad	Media

III.4 En el módulo de Administración se podrá cambiar la contraseña de los Usuarios y del Usuario Administrador.

Identificador	4
Características	Por seguridad es importante que se puedan cambiar las contraseñas de los Clientes y del Administrador.
Prioridad	Media

III.5 Los Usuarios podrán cambiar su password una vez que estén firmados en el sistema vía Web.

Identificador	5
Características	Por seguridad es importante que los Clientes y Usuarios tenga la facilidad de poder cambiar su Contraseña en el momento que ellos consideren necesario.
Prioridad	Baja

III.6 Los reportes se deben mostrar en línea, y ofrecer un método para guardar la información localmente en Excel.

Identificador	6
Características	Se debe poder almacenar la información generada de forma local, por lo que es conveniente poderla guardar en archivos de Excel.
Prioridad	Media

III.7 Se debe contar con el funcionamiento para clonar reportes.

Identificador	7
Características	Para hacer mas eficiente la creación de reportes, se debe contar con una herramienta que facilite copiar una plantilla para así sólo hacer los cambios necesarios.
Prioridad	Media

III.8 Se podrá definir si un Reporte llevara una gráfica.

Identificador	8
Características	Los reportes podrán ser generados con gráficas que permitan un mejor análisis.
Prioridad	Media

III.9 Cuando el Reporte lleve gráfica se podrá elegir el tipo y tamaño de la gráfica.

Identificador	9
Características	Las gráficas podrán ser de tipo Pastel o de Barras, los colores serán aleatorios.
Prioridad	Media

III.10 Se podrá deshabilitar la exportación de un Reporte a Excel.

Identificador	10
Características	Esta opción no deberá estar presente para todos los Clientes, ya que en algunos casos los reportes serán sólo de lectura.
Prioridad	Baja

III.11 A los Reportes se les podrá asignar una imagen para el logo del Cliente/Proveedor.

Identificador	11
Características	Es importante poder personalizar los reportes para cada Laboratorio o Proveedor con su respectivo logotipo o imagen empresarial.
Prioridad	Media

III.12 A las columnas del reporte se les podrá asignar un formato en especial.

Identificador	12
Características	Para darle mayor presentación a la información, es necesario que las columnas de los reportes vayan formateadas según su tipo de dato.
Prioridad	Alta

IV. Usabilidad.

IV.1 El usuario debe poder usar el sistema sin necesidad de capacitación.

V. Confiabilidad (*Reliability*).

V.1 El sistema debe estar disponible 5 días por semana.

V.2 El sistema debe estar disponible 5 días sin detenerse.

VI. Seguridad.

VI.1 La información de los clientes es confidencial, por lo que no se debe mostrar ninguna información que no se haya especificado

de manera explícita.

VI.2 Sólo el usuario administrador se puede firmar para poder manejar las secciones administrativas.

VI.3 Los clientes solo podrán acceder a los reportes que tengan previamente asignados

VI.4 Los clientes podrán cambiar su password en el momento que consideren necesario.

VI.5 Los clientes sólo podrán ver los reportes e información del Laboratorio/Proveedor asignado.

VII. Desempeño (*Performance*).

VII.1 El reporte se debe generar en máximo 30 segundos (un reporte mensual), aunque el usuario se tarde más en poder visualizarlo (ya que depende de su equipo de cómputo y su explorador).

VIII. Mantenimiento y Actualización (Modificabilidad).

VIII.1 Se piensa cambiar de proveedor de base de datos, de Microsoft SQL Server a Oracle (tal vez MySQL).

VIII.2 Puede que se requiera que, a un mismo cliente se le puedan agregar varios usuarios con distintos perfiles.

IX. Soporte y Operabilidad (*Supportability*).

IX.1 Se debe notificar por correo electrónico al administrador del sistema de cualquier problema.

X. Restricciones de diseño.

X.1 Se debe utilizar tecnología Java.

X.2 No se deben usar funciones nativas (para mantener la portabilidad).

X.3 Fuera de la base de datos, todos los componentes que se usen deben ser gratuitos.

XI. Documentación *online* y requerimientos de ayuda.

XI.1 En el cliente Web se dispondrá de una breve sección de ayuda

para generar los reportes.

XII. Interfaces.

XII.1 Interfaces de usuario.

XII.2 Interfaces de hardware.

XII.3 Interfaces de software.

XII.4 Interfaces de comunicación.

XIII. Aspectos Legales, Copyright y otros.

Componente	Dueño	Licencia	Estado	Comentario
Java JDK 6	SUN Microsystem	GNU General Public License	Open Source	Maquina virtual de java
SQL Server 2000	Microsoft	Microsoft Licencia	Software Propietario	Base de datos oficial
MySQL Community Edition	MySQL AB	GPL license	Open Source	Base de datos alternativa
Apache TomCat	Apache Software Fundation	Apache 2.0 Licence	Open Source	Manejador contenido web para Java
NetBeans 5.0	SUN Microsystem	GNU General Public License	Open Source	IDE para desarrollo de aplicaciones en Java
Magic Draw. UML	No Magic, Inc.	Licencia Academica	Software Propietario	Modelador de UML
Erwin CA	AllFusion	AllFusion License	Software Propietario	Modelador de Base de Datos

XIV. Componentes de terceros.

En sistema no se tiene contemplado utilizar ningún componente comercial desarrollado por otra empresa.

XV. Estándares aplicables.

XV.1 Java Code Conventions (ver Anexos del trabajo).

XVI. Requerimientos de licenciamiento.

Ninguno ya que se contempla su uso interno para la empresa y para los Clientes con quienes se tiene convenio comercial, por lo que no se les cobrará nada.

XVII. Anexo A.

No aplican en este caso para el documento de Especificación de Requerimientos.

Capitulo 2 – Diseño del Sistema

2.1 Modelado de Sistema

Para el desarrollo del sistema de Reportes Automáticos Web, se utilizó de forma básica el proceso de desarrollo de software RUP (Rational Unified Process) y auxiliado con diagramas en UML (Unified Modeling Language). Por razones de tiempo y complejidad del sistema se tomó la decisión de sólo utilizar lo elemental del proceso RUP y sólo los diagramas de UML que nos fueran de mayor utilidad. Ya que al ser demasiado amplios estos dos temas, se volvería muy pesado, tardado y poco efectivo el desarrollo de este sistema.

2.2 El Proceso Racional Unificado (RUP)

RUP es uno de los procesos más generales que existen actualmente, ya que en realidad esta pensado para adaptarse a cualquier proyecto y no sólo de software. Junto con UML conforman la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El ciclo de vida RUP es una implementación del desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto terminado al final de cada ciclo, y a su vez un ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- Concepción: se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos
- Elaboración: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos
- Construcción: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario
- Transición: se Instala el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

Fases de RUP

En cada fase se ejecutarán una o varias iteraciones (de tamaño variable según el proyecto), y dentro de cada una de ellas seguirá un modelo en cascada (waterfall) para los flujos de trabajo que requieren las actividades que se describen a continuación.

RUP define nueve actividades a realizar en cada fase del proyecto:

1. Modelado del negocio
2. Análisis de requisitos
3. Análisis y diseño
4. Implementación
5. Pruebas
6. Distribución
7. Gestión de configuración y cambios
8. Gestión del proyecto
9. Gestión del entorno

A continuación se muestra un diagrama general de las fases de RUP en donde podemos apreciar de forma grafica el flujo de procesos y como pueden cambiar al avanzar las iteraciones.

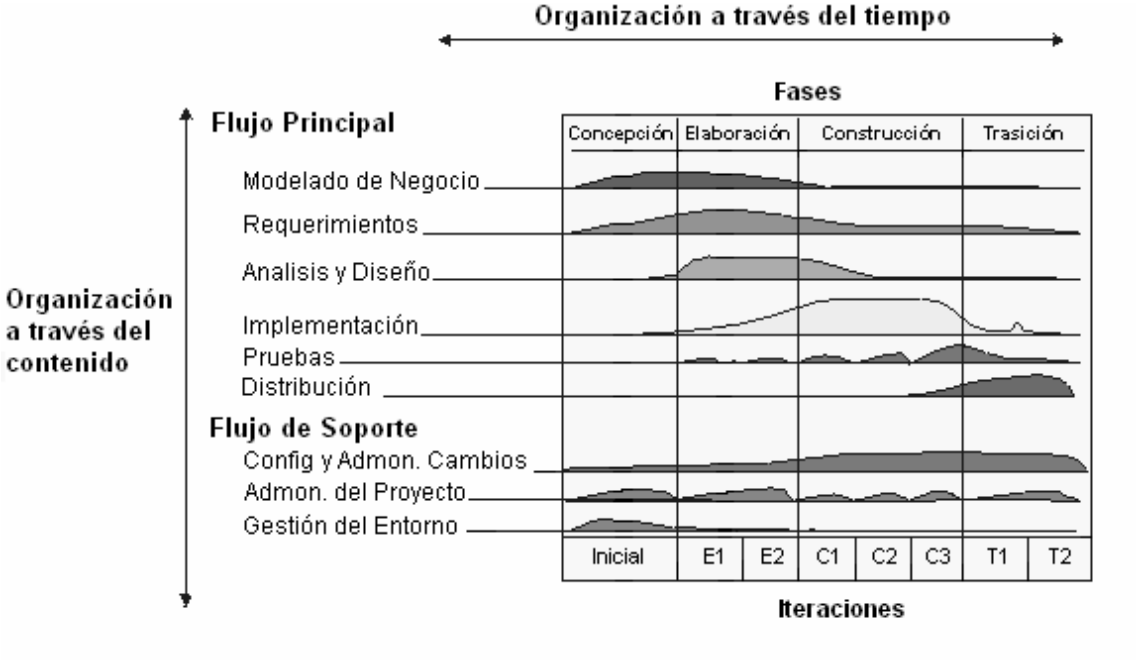
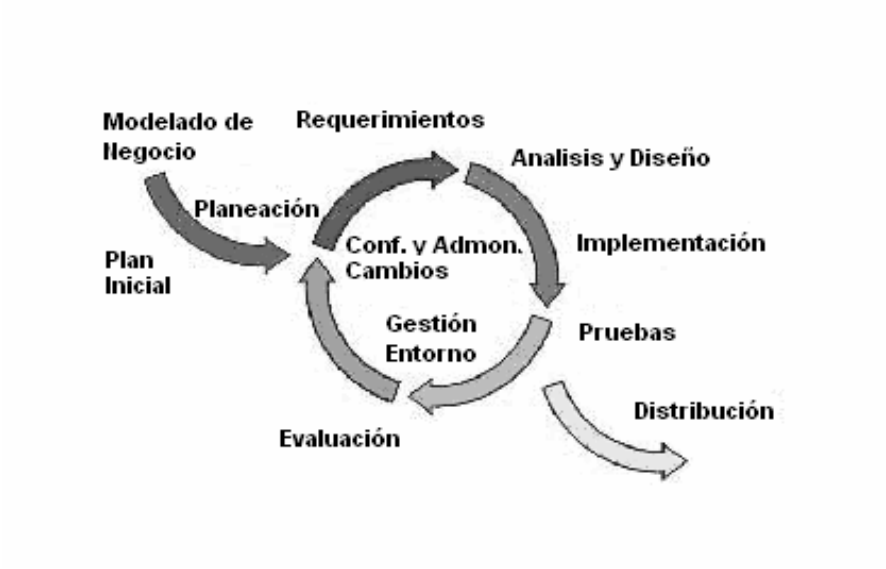


Diagrama de flujos de trabajo de RUP.



Principales características

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software.

RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

RUP es un proceso muy general y muy grande, por lo que antes de usarlo se tuvo que adaptar a las características de la empresa. Por suerte ya hay muchos procesos alternativos mucho más ligueros y fáciles de llevar, como “Extreme Programing”, el cual valdría mucho la pena considerarlo para futuros sistemas en Farmatel y que por supuesto es tema para analizar.

2.3 El Lenguaje Unificado de Modelado (UML)

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido concebido por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa: Booch, OMT y OOSE. UML ha puesto fin a las llamadas “guerras de métodos” que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

El objetivo principal cuando se empezó a gestar UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común. Hay que tener en cuenta que el estándar UML no define un proceso de desarrollo específico, tan solo se trata de una notación.

En UML 2.0 hay 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta se suelen dividir en tres grupos: Diagramas de Estructura, Diagramas de Comportamiento y Diagramas de Interacción.

En este caso solo trabajaremos con los diagramas que fueron utilizados con el Sistema de Reportes Automáticos Web “RAW” los cuales se muestran a continuación.

2.3.1 Diagrama de Caso de Uso

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

Elementos: Los elementos que pueden aparecer en un Diagrama de Casos de Uso son: Actores, casos de uso y relaciones entre casos de uso.

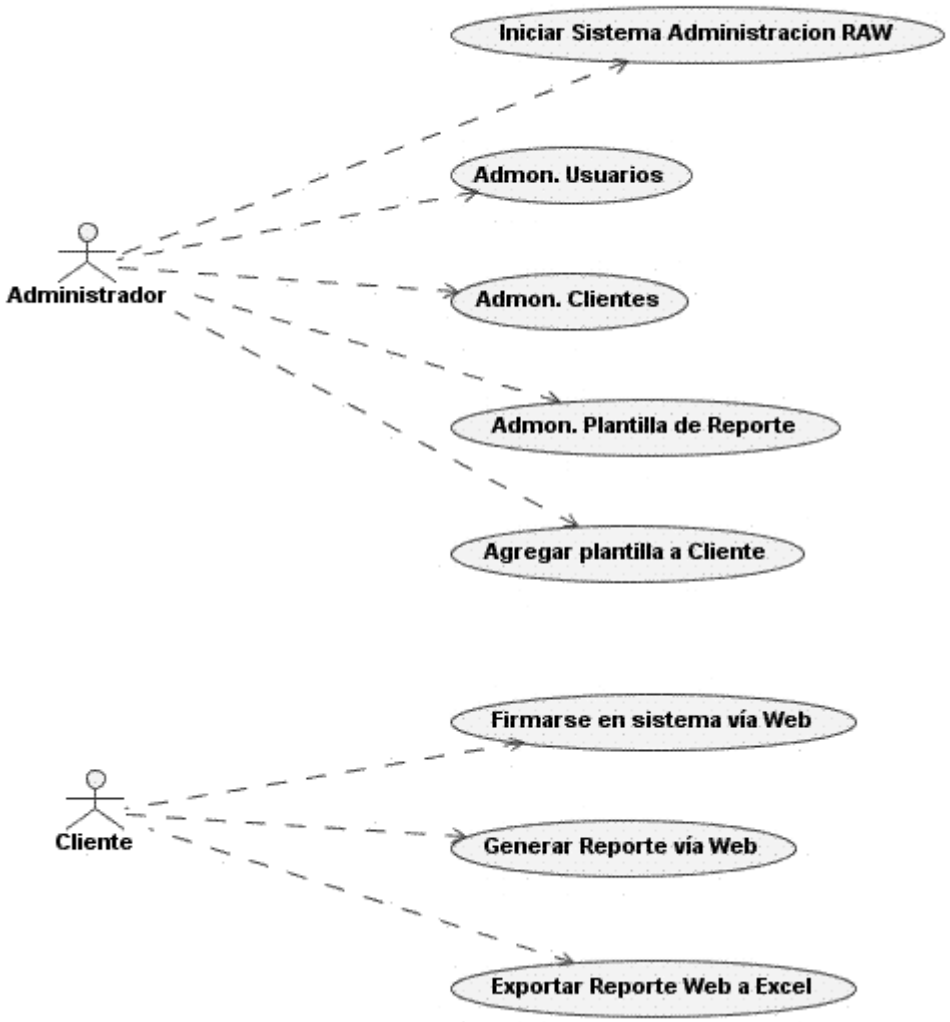
Actores: Un actor es una entidad externa al sistema que realiza algún tipo de interacción con el mismo. Se representa mediante una figura humana dibujada con palotes. Esta representación sirve tanto para actores que son personas como para otro tipo de actores (otros sistemas, sensores, etc.).

Casos de Uso: Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el Diagrama de Casos de Uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

Relaciones entre Casos de Uso

Entre dos casos de uso puede haber las siguientes relaciones: Desarrollo Orientado a Objetos con UML
• **Extiende**: Cuando un caso de uso especializa a otro extendiendo su funcionalidad.
• **Usa**: Cuando un caso de uso utiliza a otro.
Se representan como una línea que une a los dos casos de uso relacionados, con una flecha en forma de triángulo y con una etiqueta <<extiende>> o <<usa>> según sea el tipo de relación. En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea.

A continuación se muestra el diagrama de Casos de Uso del sistema "RAW".



2.3.2 Diagrama de Secuencia

Un diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo.

El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo.

Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

Diagramas de secuencia del sistema “RAW”.
Diagrama de Secuencia del caso de uso “Iniciar Sistema Administración RAW”

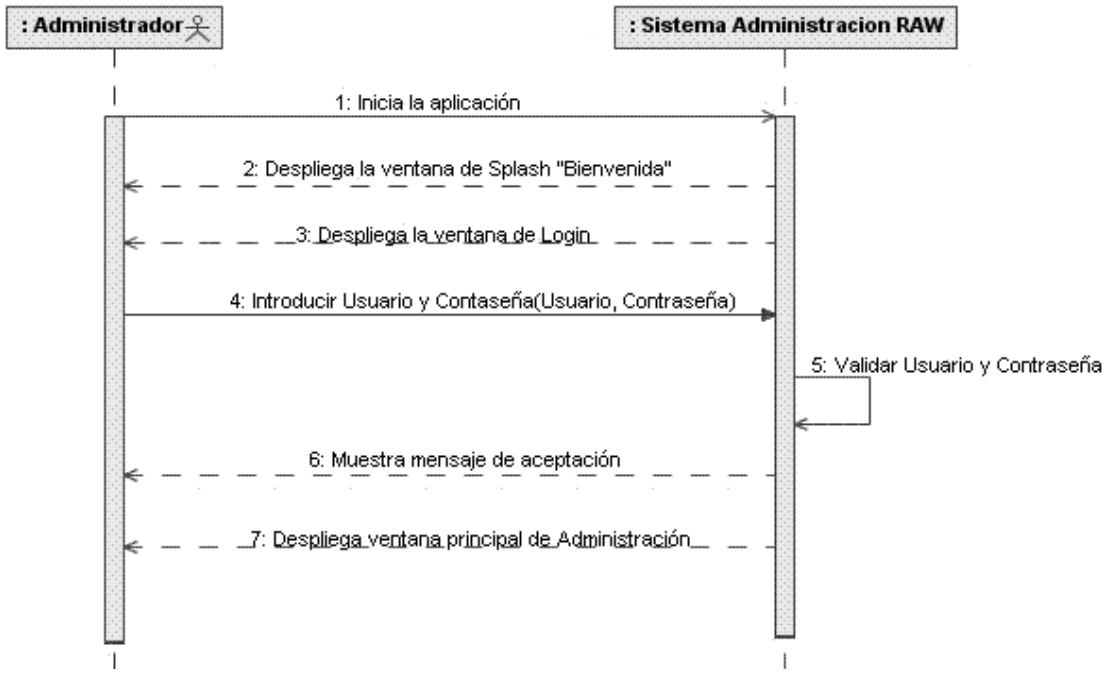


Diagrama de Secuencia del Caso de Uso “Administración Usuarios”

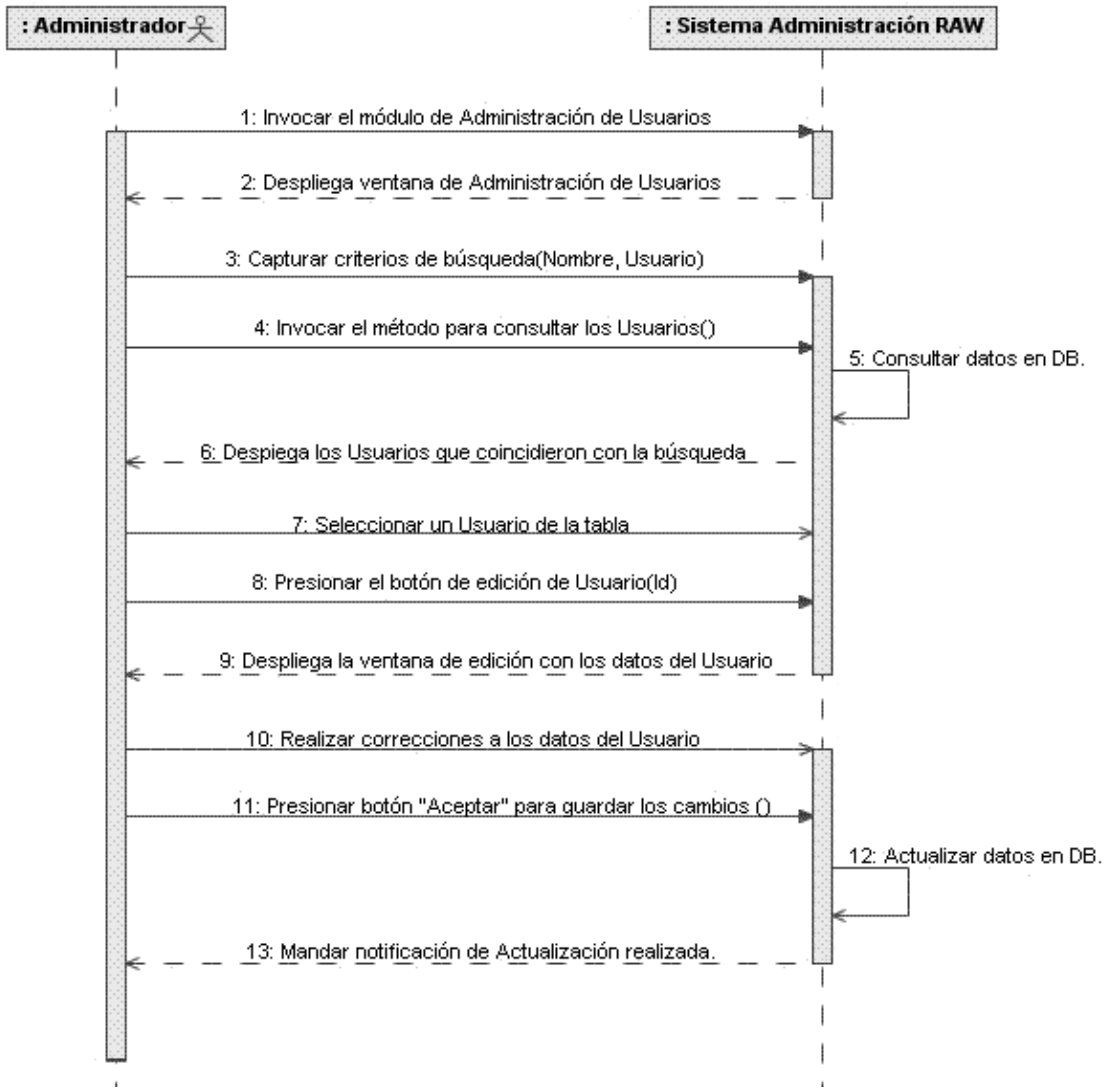


Diagrama de Secuencia del Caso de Uso “Administración de Clientes”

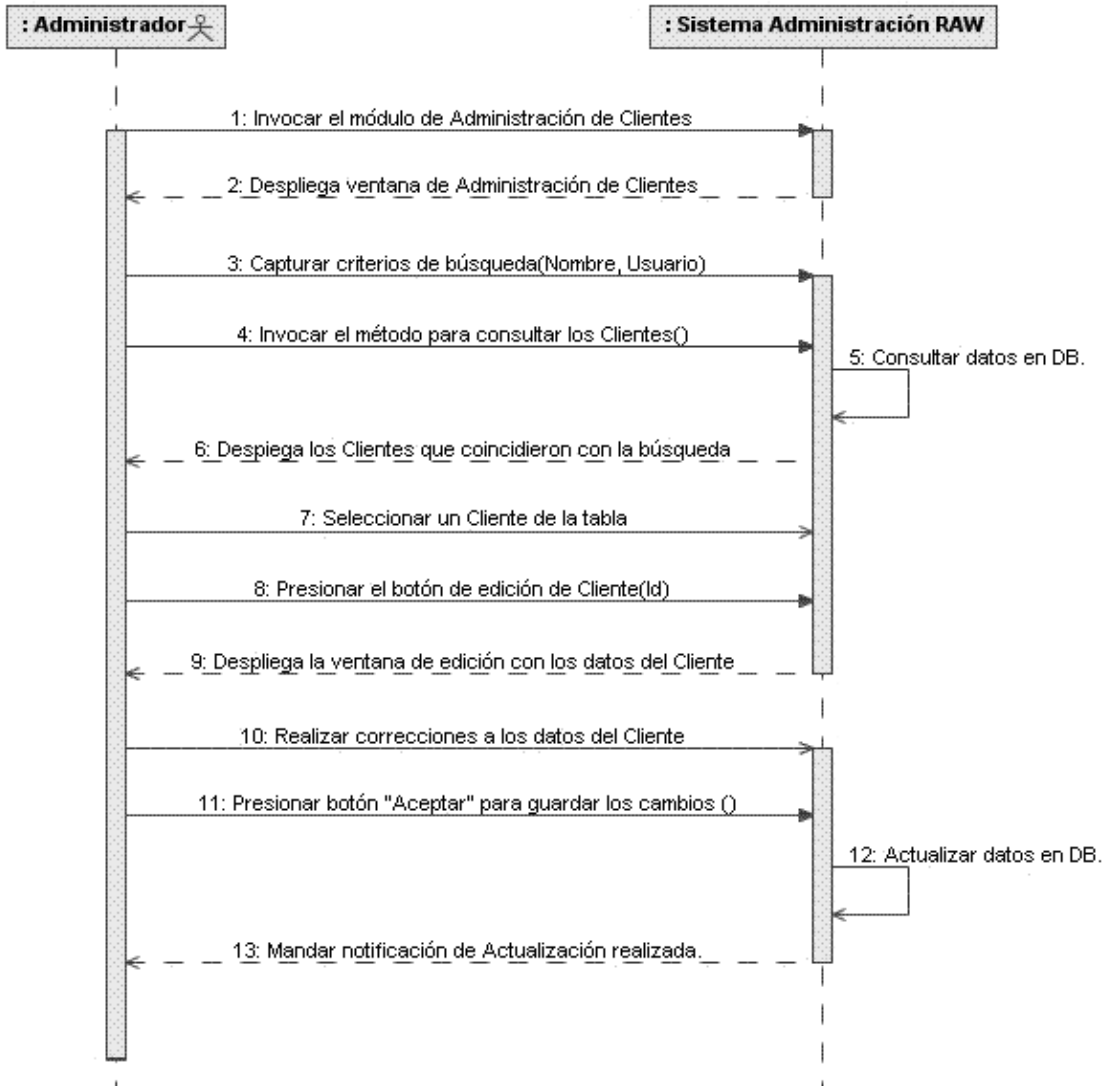


Diagrama de Secuencia del Caso de Uso “Administración de Plantillas (Edición)”

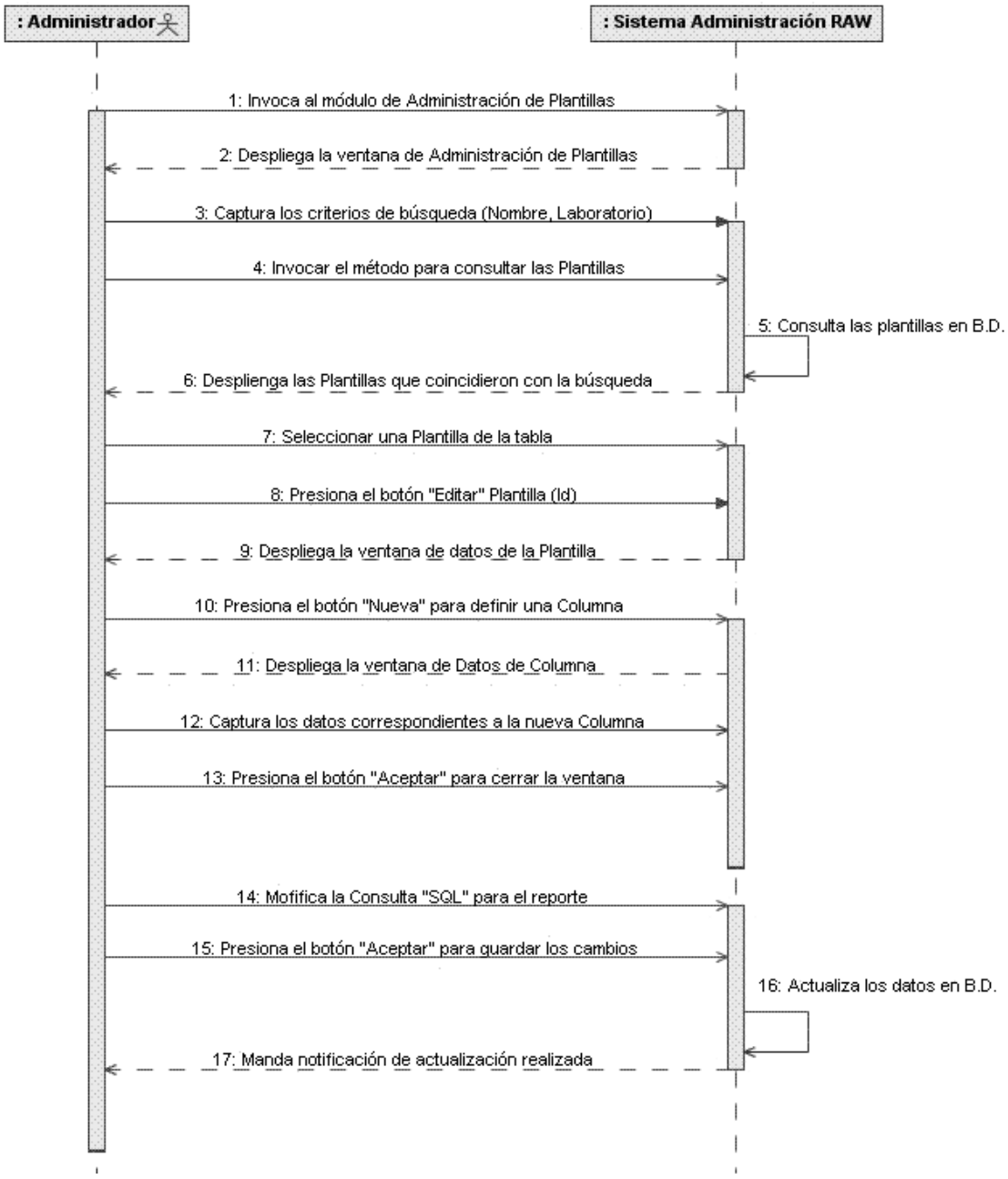


Diagrama de Secuencia del Caso de Uso “Asignación de Plantillas a Clientes”

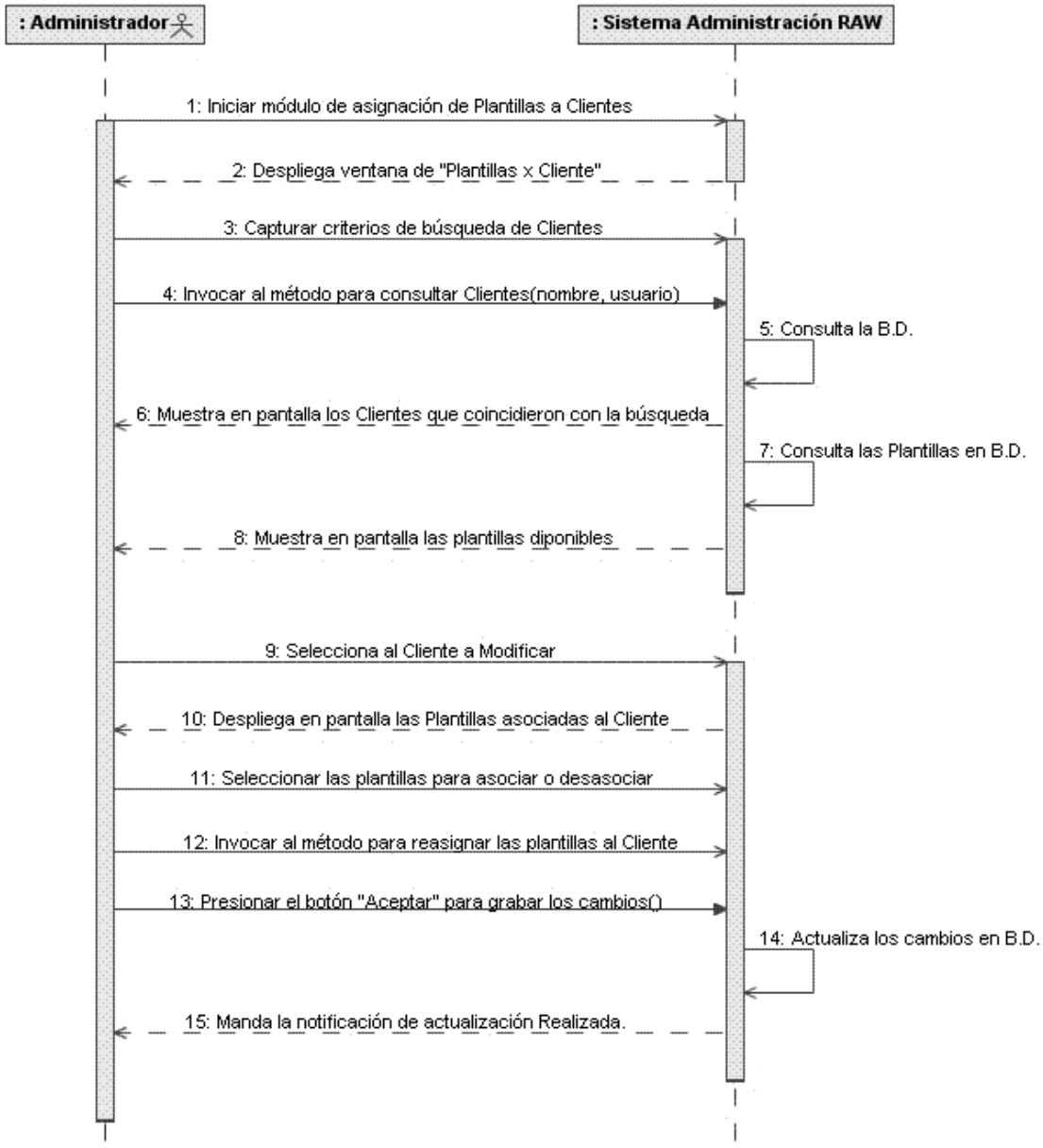


Diagrama de Secuencia para el Caso de Uso “Firmarse al Sistema Vía Web”

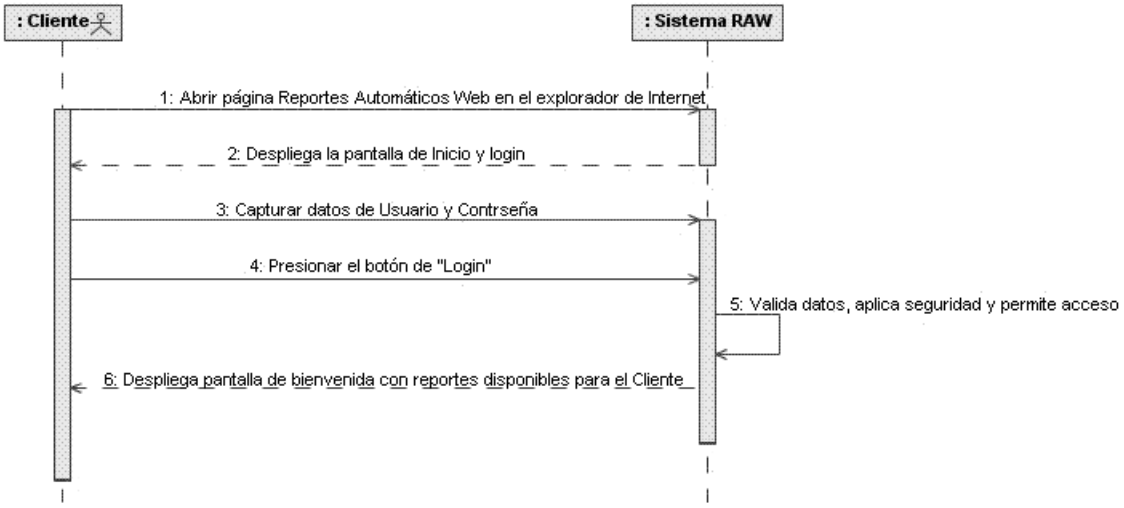


Diagrama de Secuencia para el Caso de Uso “Generación de Reporte”

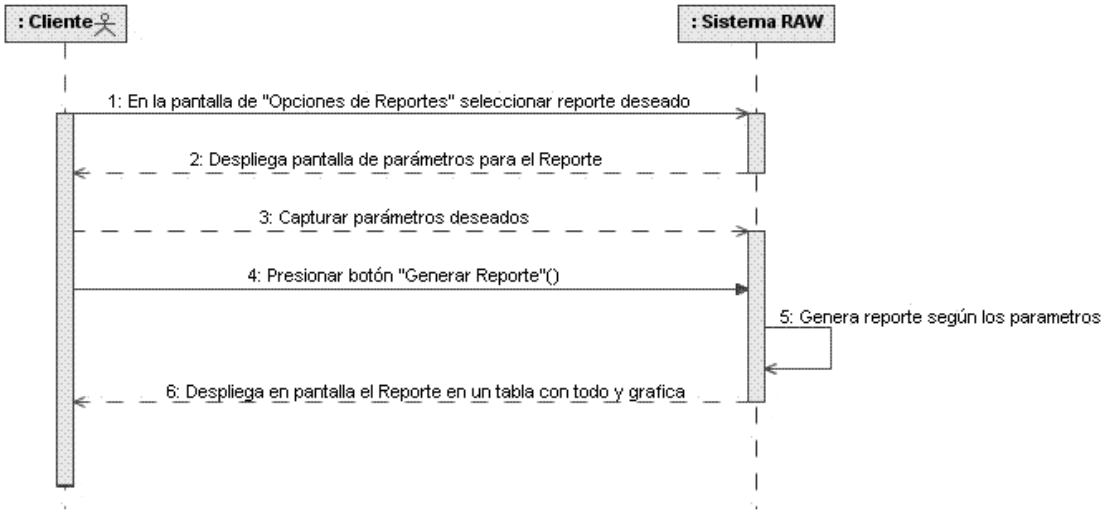
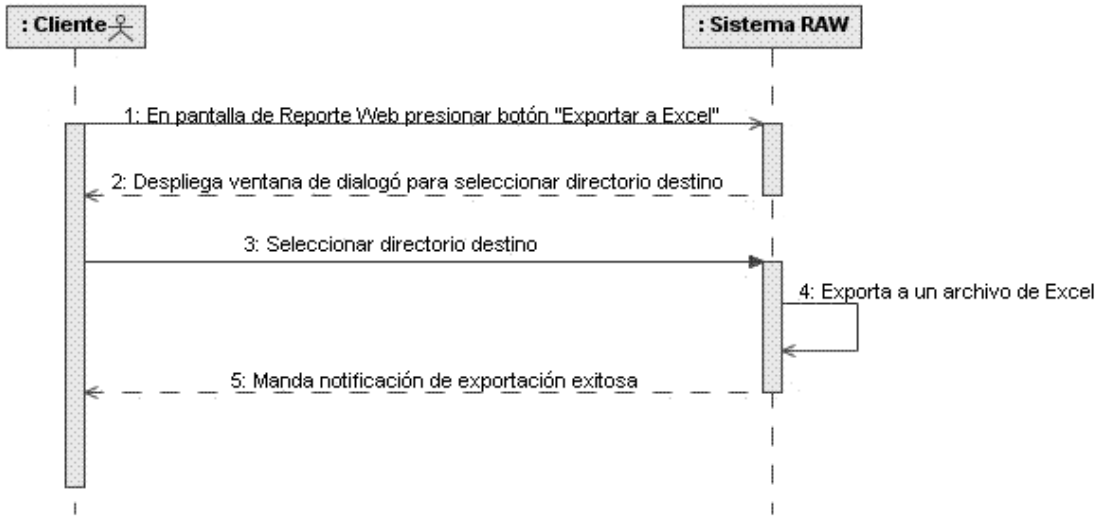


Diagrama de Secuencia para el Caso de Uso “Exportar Reportes Web a Excel”



2.3.3 Diagrama de Actividades

El diagrama de actividades de UML es muy parecido a un diagrama de flujo. Muestra los pasos, puntos de decisión y bifurcaciones. Este tipo de diagrama es útil para representar las operaciones de un objeto y los procesos de negocios.

El diagrama de Actividades es una extensión del diagrama de estados. Los diagramas de estados destacan los estados y representan actividades como flechas entre los estados. Los de Actividad se enfocan a las actividades. Cada actividad se representa como un rectángulo con esquinas redondeadas, más redondeados en apariencia que la representación de un estado. El diagrama de Actividades utiliza los mismos símbolos que el de estados para los puntos de inicio y final.

A continuación se muestran los diagramas de actividades para el sistema “RAW”.

Diagrama de Actividades para el Caso de Uso "Iniciar Sistema Administración RAW"

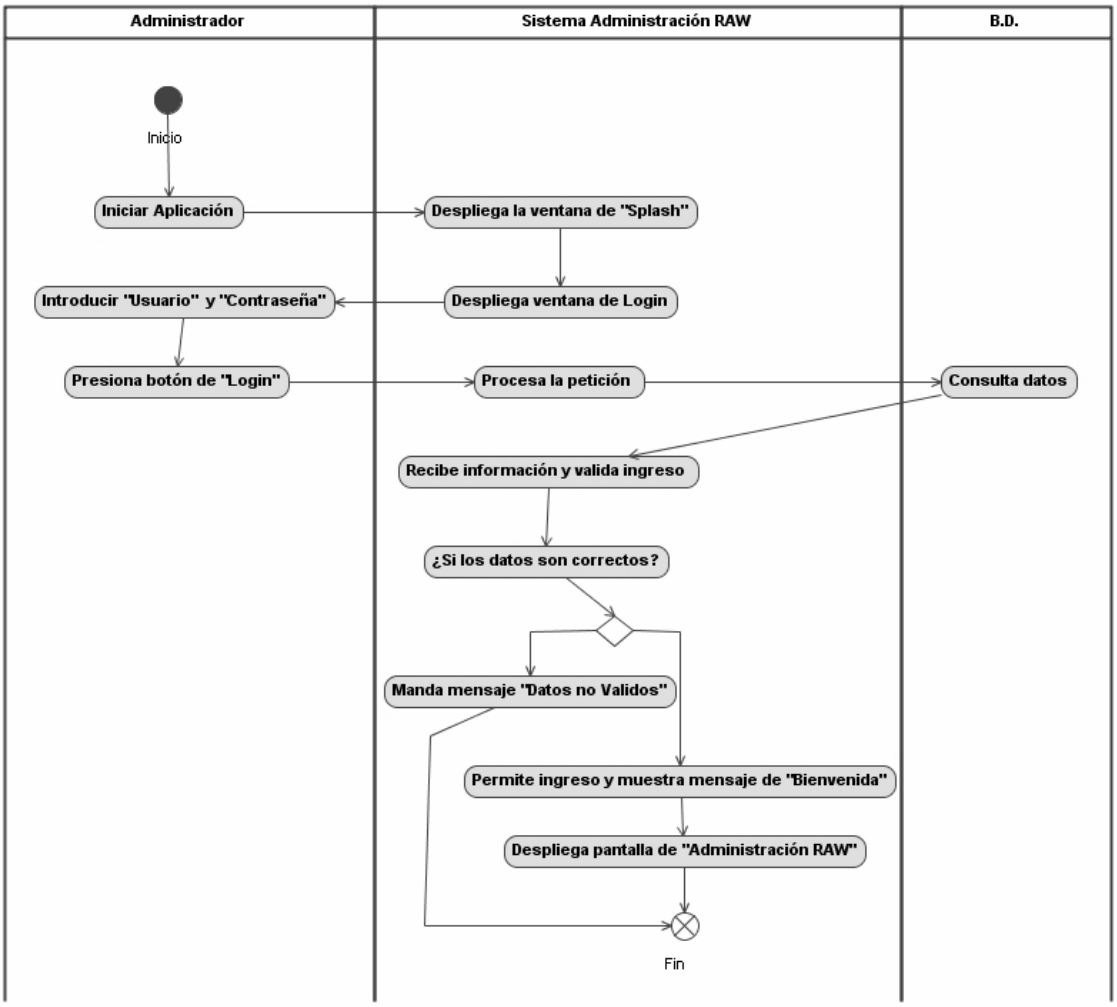


Diagrama de Actividades para el Caso de Uso "Administración Usuarios"

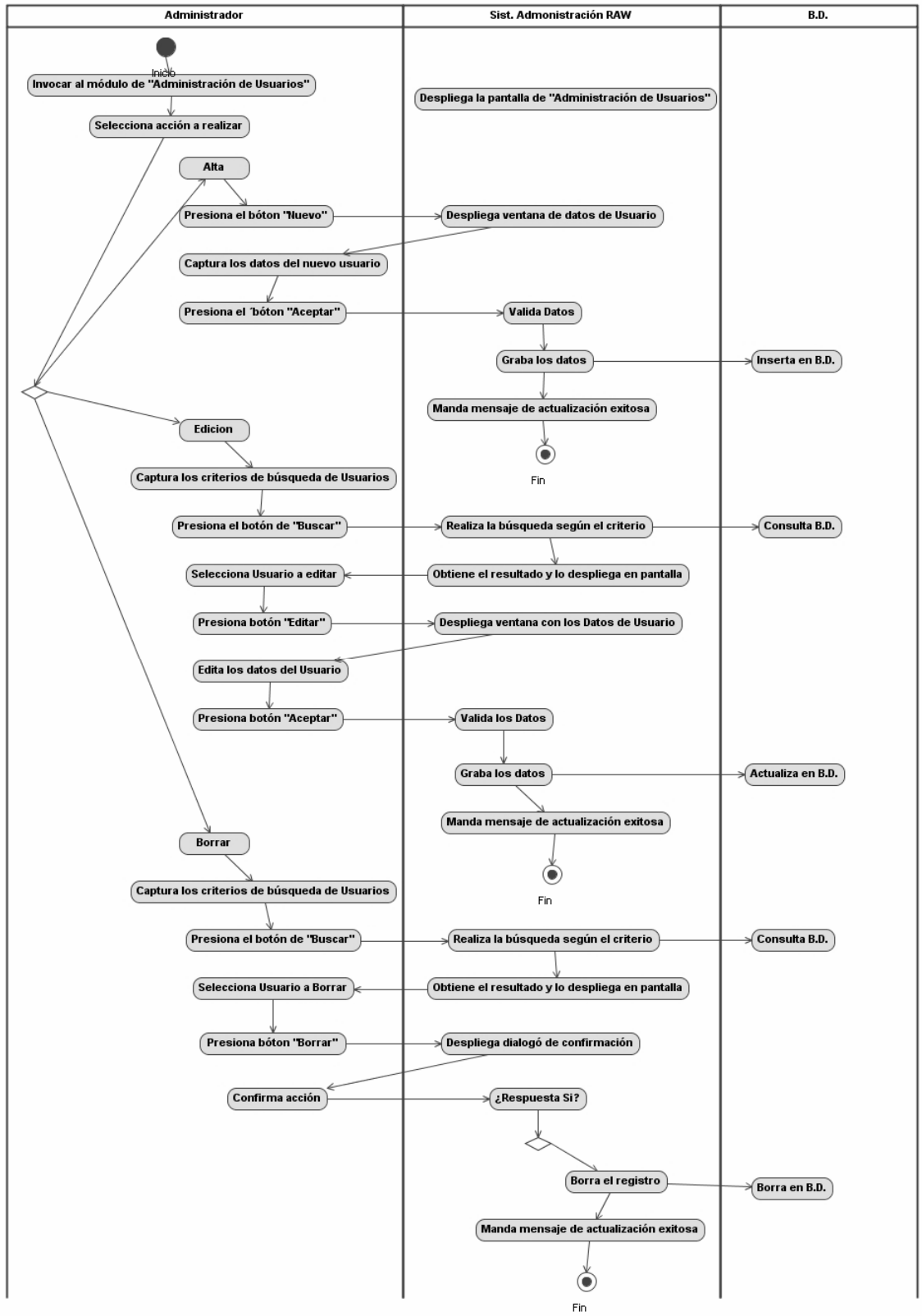


Diagrama de Actividades para el Caso de Uso "Administración Clientes"

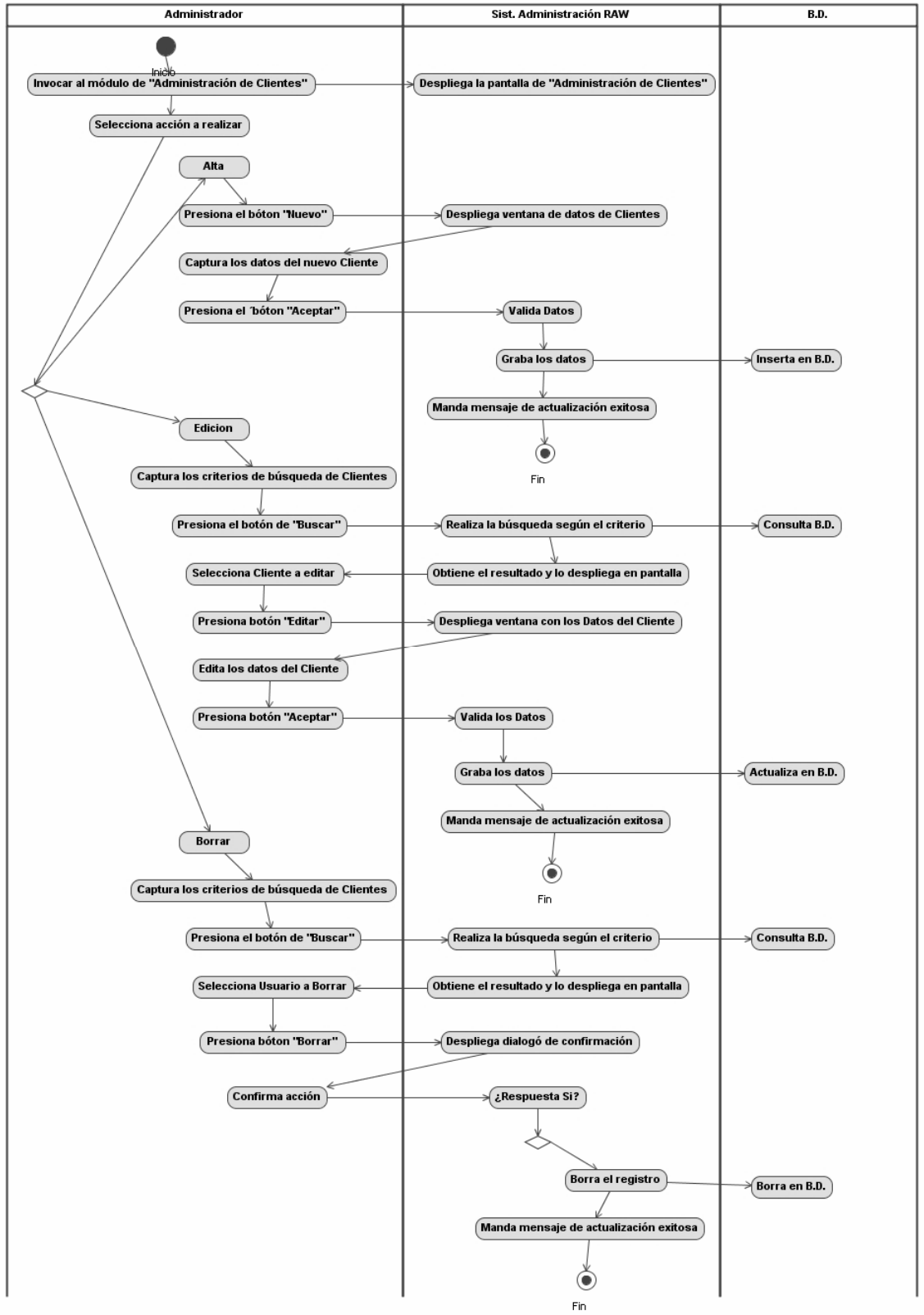


Diagrama de Actividad para el Caso de Uso "Administración Plantillas de Reportes"

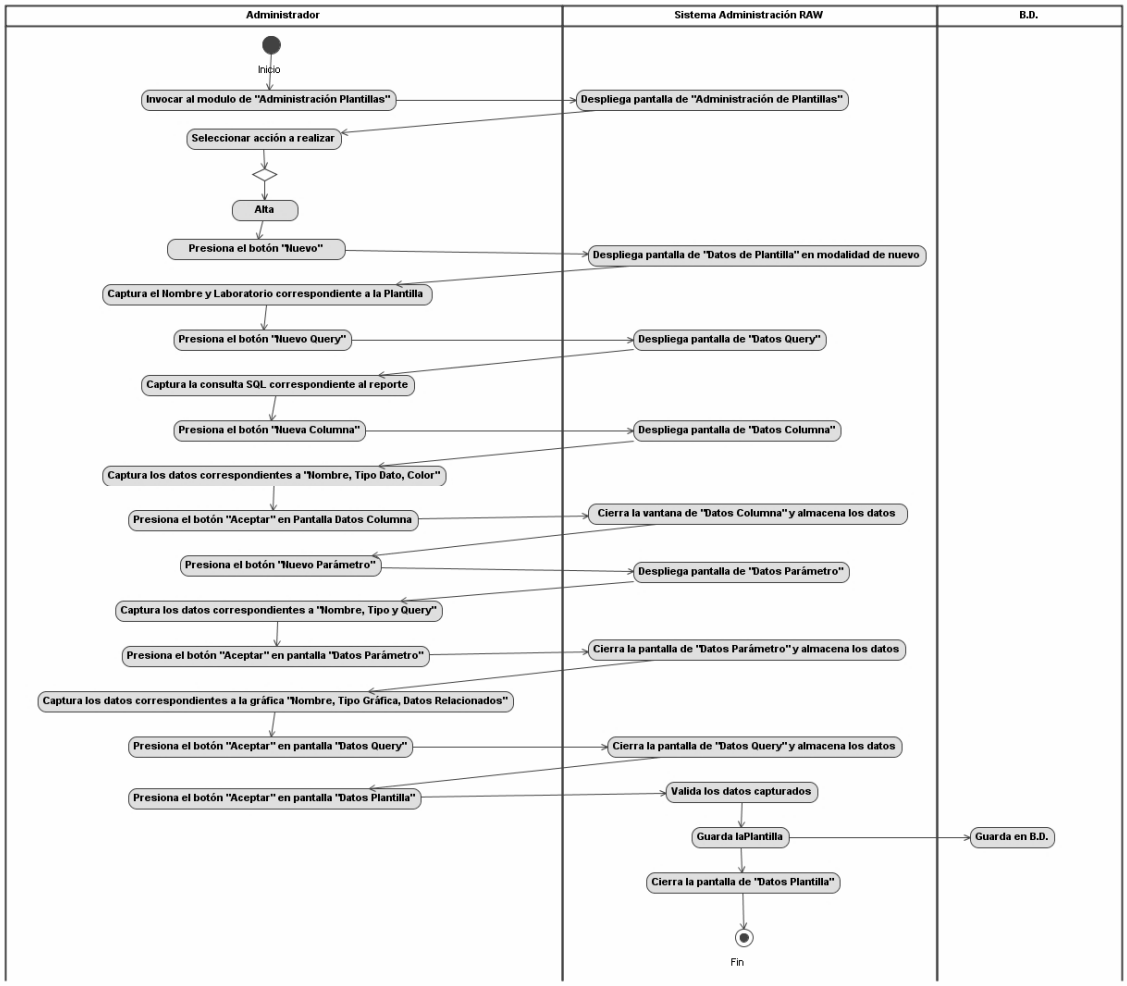


Diagrama de Actividades correspondiente al Caso de Uso "Asignar Plantilla a Cliente"

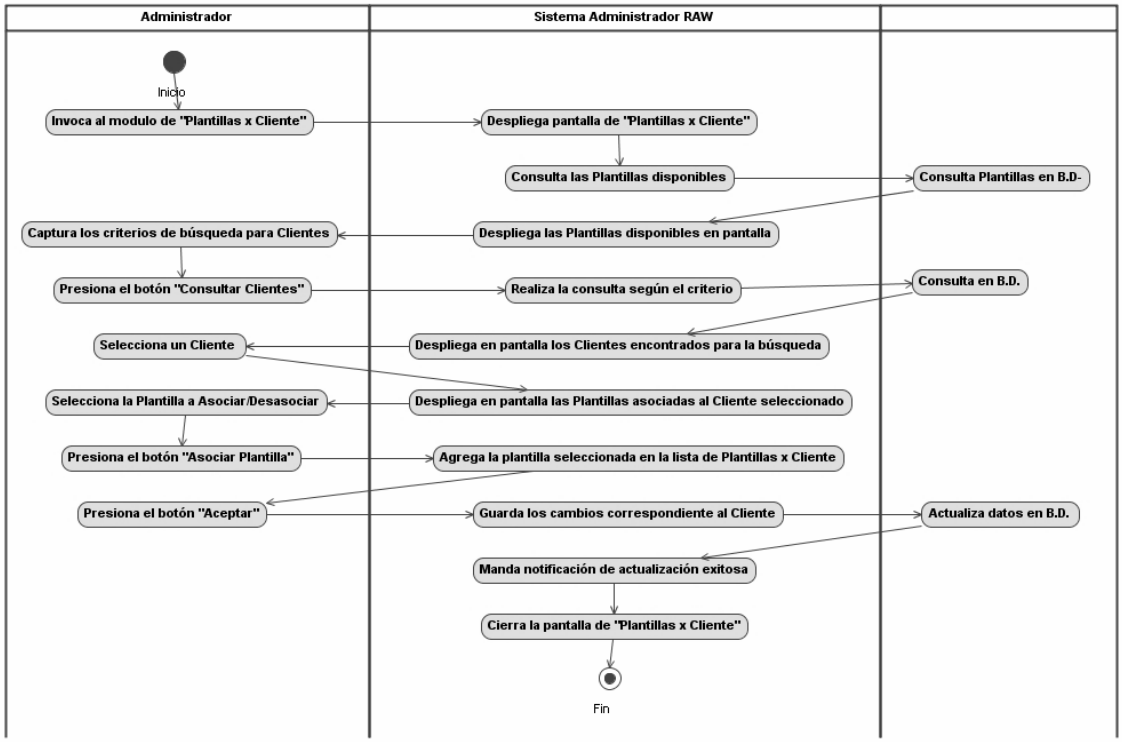


Diagrama Actividades correspondiente al Caso de Uso "Firmarse Vía Web"

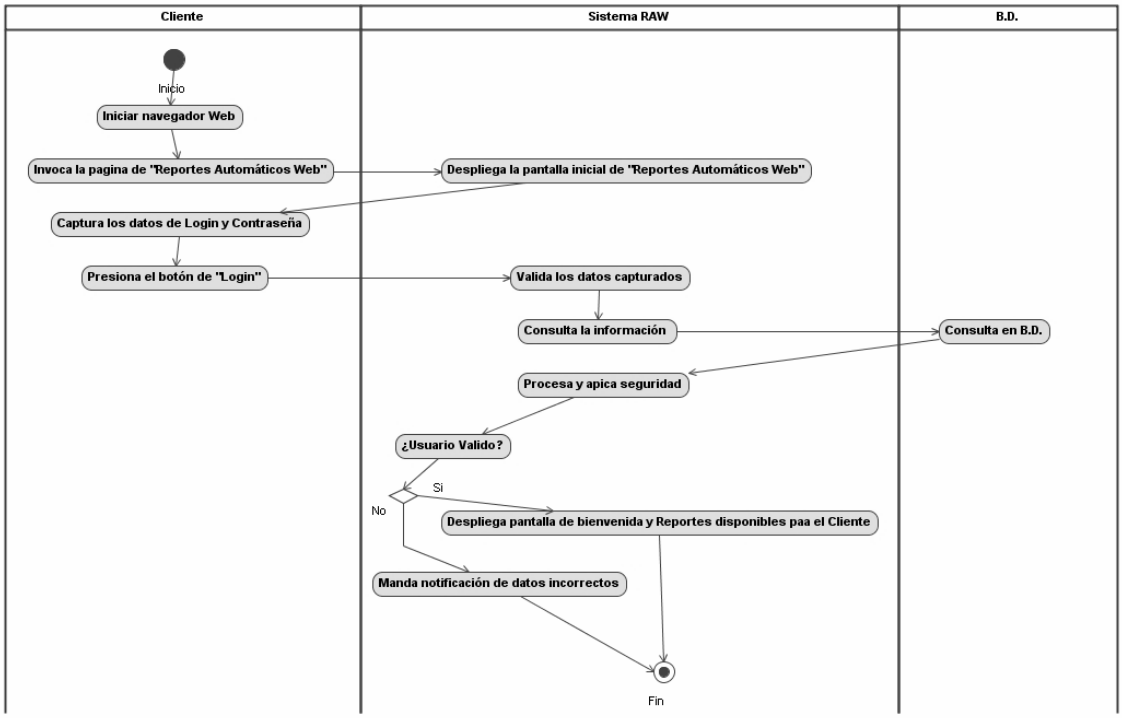


Diagrama Actividad correspondiente al Caso de Uso "Generación Reporte Vía Web"

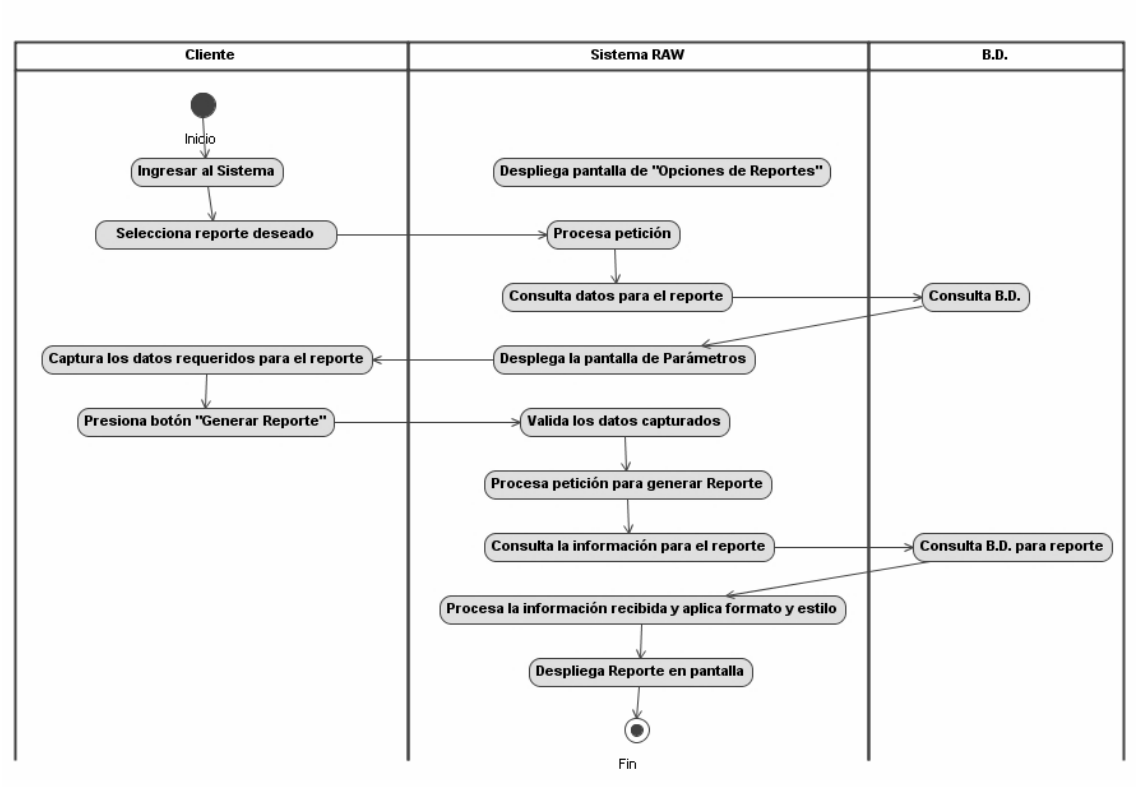
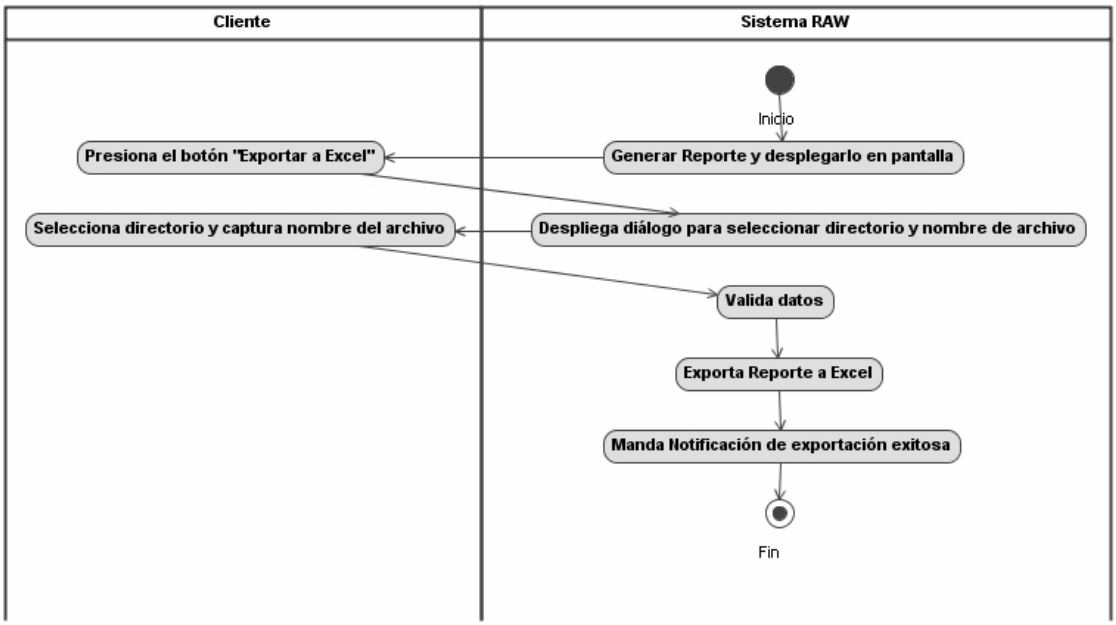


Diagrama Actividad correspondiente al Caso de Uso "Exportar Reporte a Excel"



2.3.4 Diagrama de Clases

Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas informáticos, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

Propiedades también llamados **atributos** o **características**, son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. Suponiendo que el objeto es una puerta, sus propiedades serían: la marca, tamaño, color y peso.
Operaciones son aquellas actividades o verbos que se pueden realizar con/para este objeto, como por ejemplo abrir, cerrar, buscar, cancelar, acreditar, cargar. De la misma manera que el nombre de un atributo, el nombre de una operación se escribe con minúsculas si consta de una sola palabra. Si el nombre contiene más de una palabra, cada palabra será unida a la anterior y comenzará con una letra mayúscula, a excepción de la primera palabra que comenzará en minúscula. Por ejemplo: abrir Puerta, cerrarPuerta, buscarPuerta, etc.

Interfaz es un conjunto de operaciones y/o propiedades que permiten a un objeto comportarse de cierta manera, por lo que define los requerimientos mínimos del objeto.

Herencia se define como la reutilización de un objeto padre ya definido para poder extender la funcionalidad en un objeto hijo. Los objetos hijos heredan todas las operaciones y/o propiedades de un objeto padre. Por ejemplo: Una persona puede subdividirse en Proveedores, Acreedores, Clientes, Accionistas, Empleados; todos comparten datos básicos como una persona, pero además tendrá información adicional que depende del tipo de persona, como saldo del cliente, total de inversión del accionista, salario del empleado, etc.

A continuación se muestran los diagramas de clases del sistema “RAW”.

Diagrama de Clases del paquete “Aplications”.

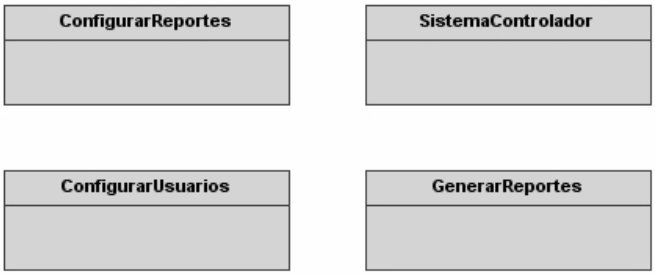


Diagrama de Clases del paquete “Entity”.

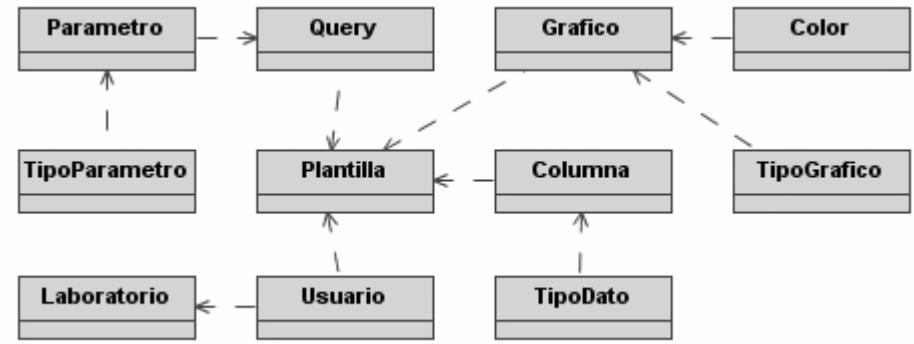


Diagrama de clases del paquete “BO”.

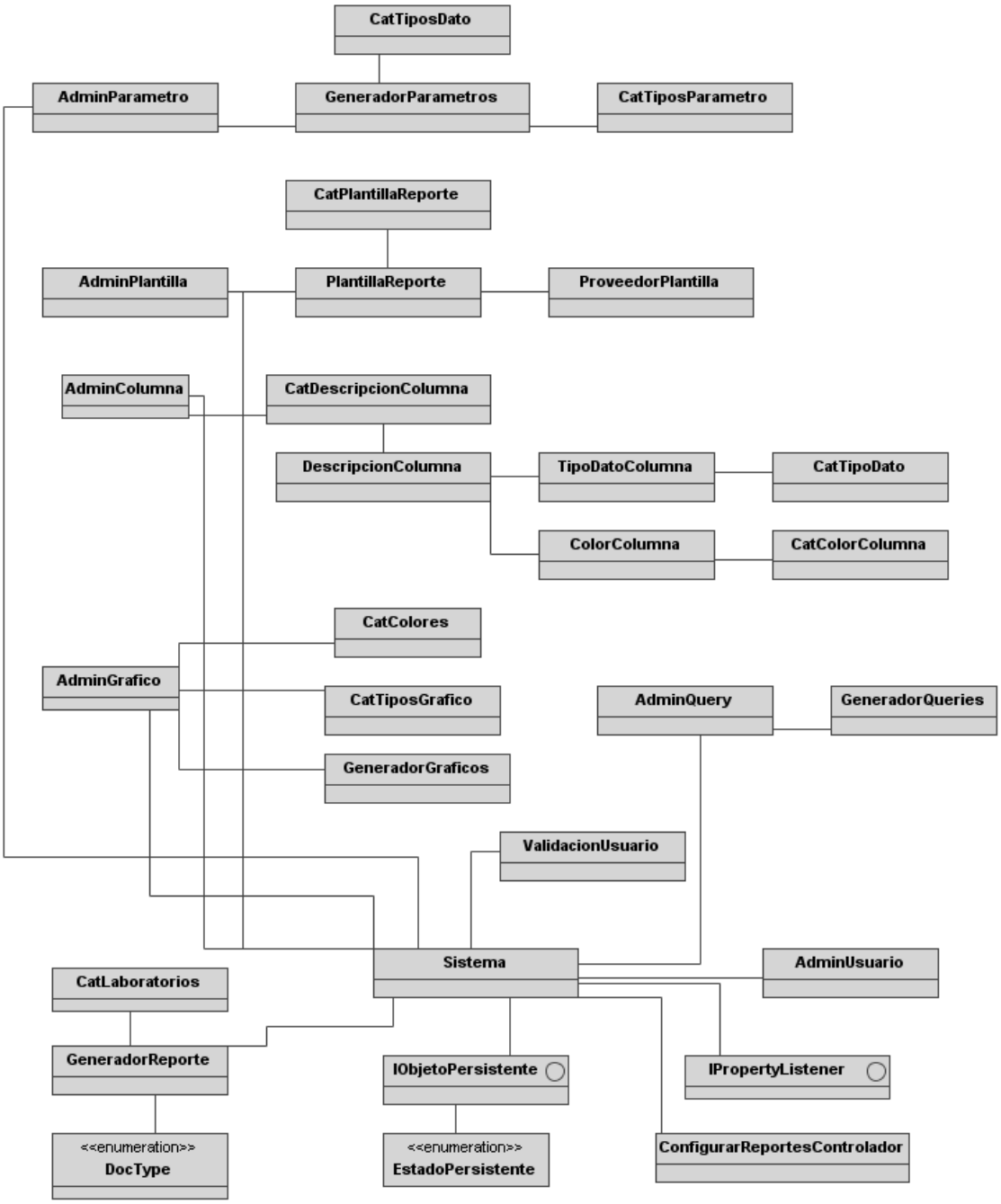


Diagrama de Clases del paquete “Servicios.Notificacion”

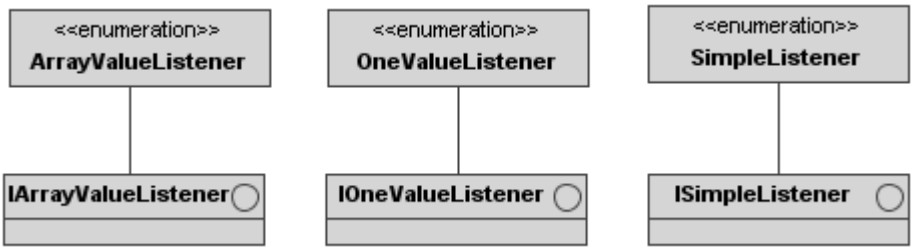


Diagrama de Clases del paquete “Servicios.Persistencia”

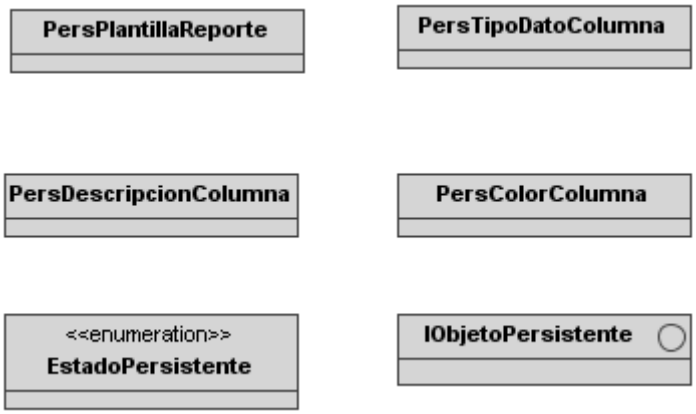


Diagrama de Clases del paquete “Data”

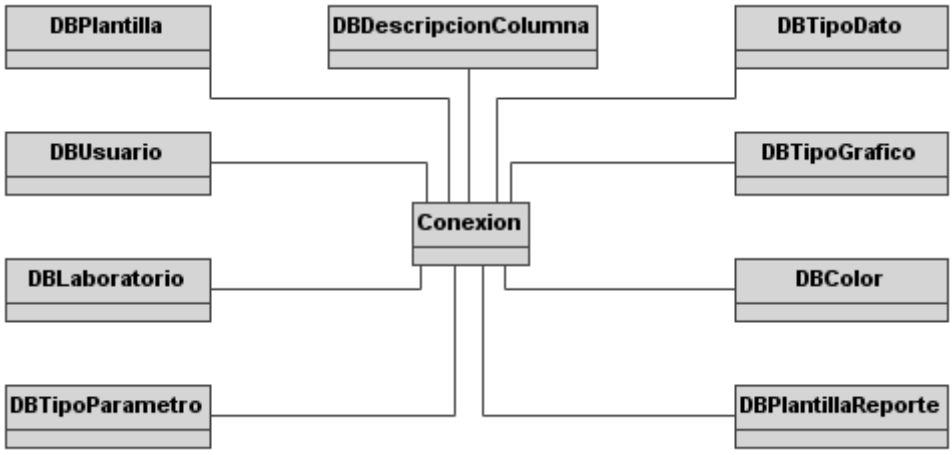


Diagrama de Clases del paquete “UI”

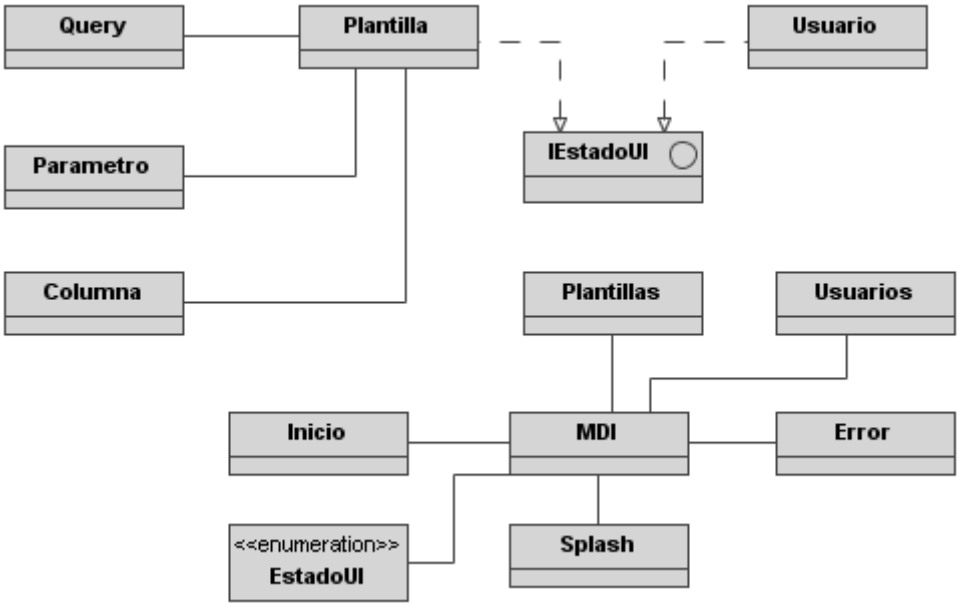


Diagrama de Clases del paquete “UIRAW”

2.3.5 Diagrama de Componentes

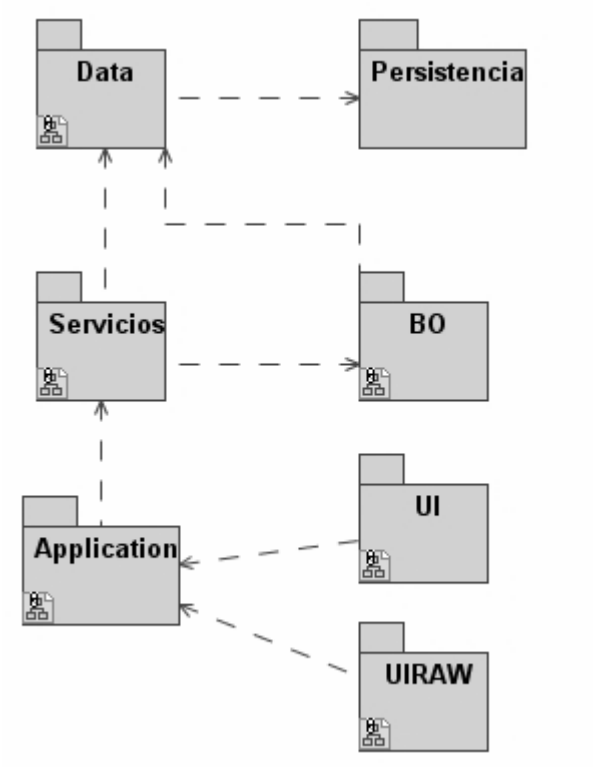
Un diagrama de componentes UML es un conglomerado de figuras de los diagramas de UML. En lugar de representar una entidad conceptual como una clase o estado, un diagrama de componentes representa a un elemento real: un componente de software. Estos componentes se encuentran a la computadora, no en la mente del analista.

Un componente puede accederse a través de su interfaz, una colección de operaciones. La relación entre componentes y su interfaz se llama “realización”. Un componente puede acceder los servicios de otro. Cuando se hace, utiliza una interfaz de “importación”. El componente que realiza la interfaz con tales servicios proporciona una interfaz de “exportación”.

La representación de un componente es un rectángulo con otros dos rectángulos pequeños sobrepuestos en su lado izquierdo. Puede representar una interfaz de dos formas: La primera es un rectángulo que contiene información de la interfaz y se conecta con el componente mediante una línea discontinua con una punta de flecha representada por un triángulo sin

relleno. La otra es un pequeño circulo conectado al componente con una línea continua. Ambos tipos de conexión pretenden mostrar una relación de realización.

Diagrama de componentes del sistema RAW.



2.3.6 Diagrama de Paquetes

Cualquier sistema grande se debe dividir en unidades más pequeñas, de modo que las personas puedan trabajar con una cantidad de información limitada, a la vez y de modo que los equipos de trabajo no interfieran con el trabajo de los otros.

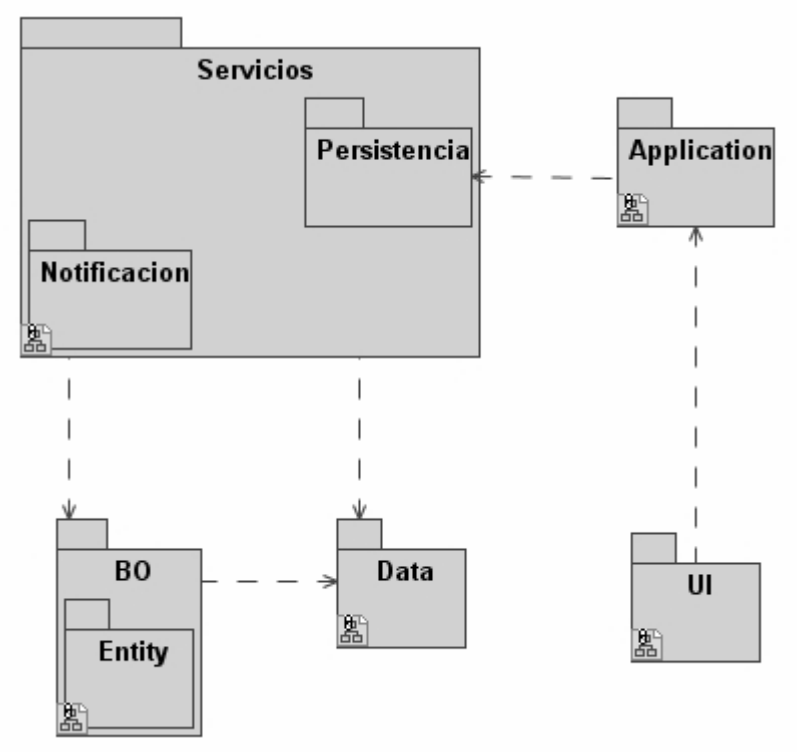
Un paquete es una parte de un modelo. Cada parte del modelo debe pertenecer a un paquete. Pero para ser funcional, la asignación debe seguir un cierto principio racional, tal como funcionalidad común, implementación relacionada y punto de vista común. UML no impone una regla para componer los paquetes.

Los paquetes ofrecen un mecanismo general para la organización de los modelos/subsistemas agrupando elementos de modelado. Cada paquete corresponde a un submodelo (subsistema) del modelo (sistema). Los paquetes son unidades de organización jerárquica de uso general de los modelos de UML. Pueden ser utilizados para el almacenamiento, el control de acceso, la gestión de la configuración y la construcción de bibliotecas que contengan fragmentos reutilizables del modelo.

Un paquete puede contener otros paquetes, sin límite de anidamiento pero cada elemento pertenece a (está definido en) sólo un paquete.

Los paquetes contienen elementos del modelo al más alto nivel, tales como clases y sus relaciones, máquinas de estado, diagramas de casos de uso, interacciones y colaboraciones; atributos, operaciones, estados, líneas de vida y mensajes están contenidos en otros elementos y no aparecen como contenido directo de los paquetes.

Diagrama de paquetes del sistema “RAW”.



2.4 Diseño de Base de Datos

2.4.1 Base de Datos Relacionales

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

2.4.2 Normalización de Base de Datos

La normalización de la base de datos es el proceso de organizar datos en sistemas distintos y únicos.

Los propósitos de la normalización son:

- Reducir o eliminar el almacenaje de datos duplicados
- Organizar los datos en una estructura eficiente y lógica
- Evitar la redundancia de los datos.
- Evitar problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos.

El proceso de la normalización implica el determinarse de qué datos se deben almacenar en cada tabla de la base de datos.

Por la tradición, el proceso de la normalización implica el trabajar con los pasos bien definidos, llamados las formas normales.

En la primera forma normal (1NF) que eliminas columnas duplicadas de la misma tabla, que creas las tablas separadas para cada grupo de datos relacionados, y que identificar cada fila con una columna única o el sistema de las columnas (las llaves primarias).

En la segunda forma normal (2NF) quitas subconjuntos de los datos que se aplican a las filas múltiples de una tabla, las ponen en tablas separadas, y crean relaciones entre estas tablas nuevas y las tablas de la original con el uso de llaves extranjeras.

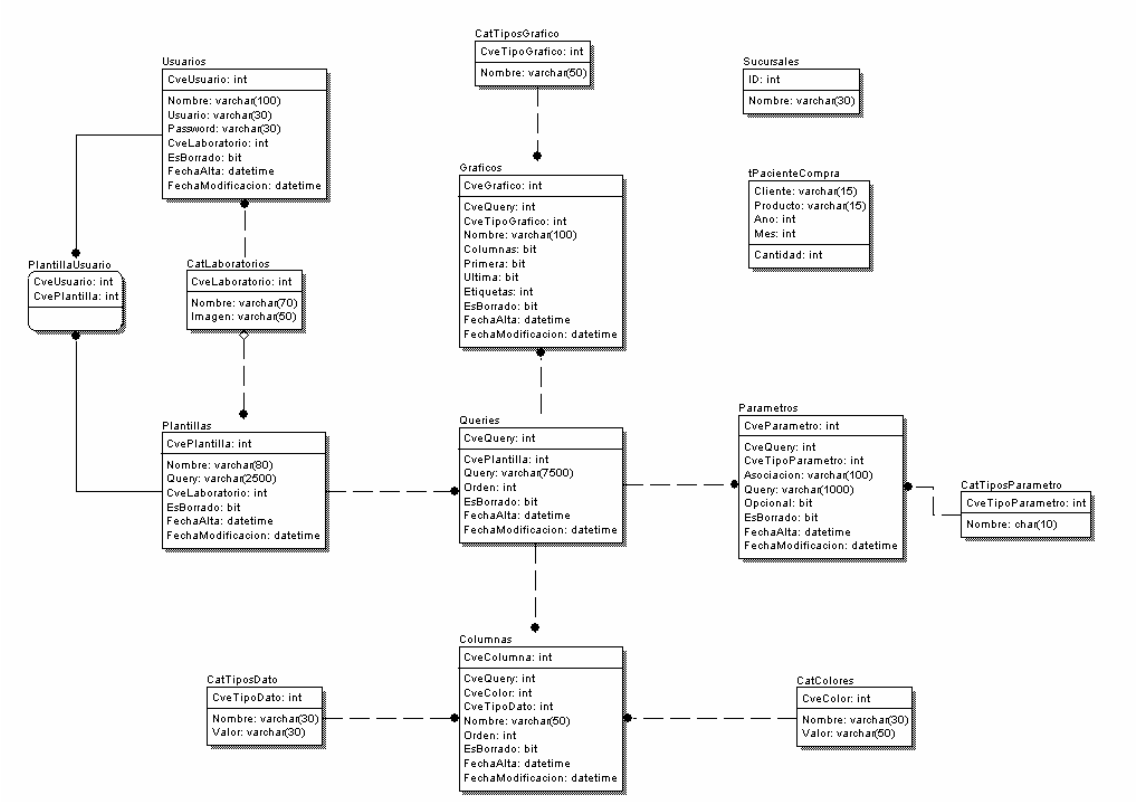
En la tercera forma normal (3NF) quitas las columnas que no son dependientes sobre la llave primaria.

Se han definido, pero se utilizan menos comúnmente las formas normales adicionales. Estas formas normales avanzadas incluyen la cuarta forma normal (4NF), la quinto forma normal (5NF), la forma normal de Boyce Codd (BCNF), y la forma normal de la Dominio-Llave (DK/NF).

2.4.3 Diagrama Físico de Base de Datos para el Sistema RAW

A continuación se muestra el diagrama físico de la base de datos del sistema "RAW". En donde se puede apreciar la estructura general, a continuación se describen las tablas más importantes:

- Usuarios: Tabla en donde se guardan los datos generales de los Usuarios y Clientes.
- CatLaboratorios: Catálogo de Laboratorio/Proveedor.
- Plantilla: Tabla para almacenar la estructura general de los reportes a generar.
- Queries: En esta tabla se almacenan las consultas (Query en ingles) en SQL para la extracción de la información de la base de datos.
- Parámetros: Lista de parámetros que van a recibir las consultas y que serán capturadas desde la página Web.
- Columna: En esta tabla se guarda la información que describe cada Columna en el reporte.
- Gráfico: Tabla que se encarga de guardar toda la información que describen los gráficos asociados a las Consultas y Plantillas.
- PantillaUsuario: Se encarga de relacionar una Pantilla con un Usuario/Cliente.
- CatTipoGrafico: Se guardan los tipos de Gráficos soportador por el sistema RAW.
- CatTipoDato: Se guardan los tipos de Datos (entero, texto, fecha) que soporta el sistema RAW.
- CatColores: Se guardan los Colores que pueden ser asociados a las Columnas de los reportes generados.
- CatTipoParametro: Se guardan los tipos de Parámetros que se pueden asignar a los Parámetros que se utilizan para la generación de los reportes.



Capitulo 3 – Desarrollo de la Aplicación

3.1 Lenguaje de Programación

Java es un lenguaje de programación orientado a objetos con el que podemos realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía Sun Microsystems a principios de los años 1990 con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras.

El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel como punteros. JavaScript, un lenguaje interpretado, comparte un nombre similar y una sintaxis similar, pero no está directamente relacionado con Java.

Sun Microsystems proporciona una implementación GNU General Public License de un compilador Java y una máquina virtual Java, conforme a las especificaciones del Java Community Process, aunque la biblioteca de clases que se requiere para ejecutar los programas Java no es software libre.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre.

La independencia de plataforma es una de las razones por las que Java es interesante para Internet, ya que muchas personas deben tener acceso con computadoras distintas. Pero no se queda ahí, Java está desarrollándose incluso para distintos tipos de dispositivos además de las PC's, como teléfonos celulares, agendas (PDA's) y en general para cualquier cosa que se le ocurra a la industria.

La tecnología Java está compuesta básicamente por 2 elementos: el lenguaje Java y su plataforma. Con plataforma nos referimos a la máquina virtual de Java (Java Virtual Machine).

Orientado a Objetos
Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

Pasado y presente
Java fue pensado originalmente para utilizarse en cualquier tipo de electrodoméstico pero la idea fracasó. Uno de los fundadores de Sun rescató la idea para utilizarla en el ámbito de Internet y convirtieron a Java en un lenguaje potente, seguro y universal gracias a que lo puede utilizar todo el mundo y es gratuito. Una de los primeros triunfos de Java fue que se integró en el navegador Netscape y permitía ejecutar programas dentro de una página web, hasta entonces impensable con el HTML.

Actualmente Java se utiliza en un amplio abanico de posibilidades y casi cualquier cosa que se puede hacer en cualquier lenguaje se puede hacer también en Java y muchas veces con grandes ventajas. Para lo que nos interesa a nosotros, con Java podemos programar páginas web dinámicas, con accesos a bases de datos, utilizando XML, con cualquier tipo de conexión de red entre cualquier sistema. En general, cualquier aplicación que deseemos hacer con acceso a través web se puede hacer utilizando Java.

Distribuido
Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

Interpretado y compilado a la vez
Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador. Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).

Robusto
Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

Seguro
Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

Indiferente a la arquitectura
Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Eso quiere decir que si hacemos un programa en Java podrá funcionar en cualquier computadora del mercado. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple, etc. Esto lo consigue porque se ha creado una Máquina de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente.

Para acomodar requisitos de ejecución tan variados, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.

Portable
La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas. Estas dos últimas características se conocen como la *Máquina Virtual Java* (JVM).
Alto rendimiento

Multihilo
Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (*multithreading*) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

Dinámico
El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red.

Produce applets
Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets.
Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de Web HotJava, escrito íntegramente en Java.
Por su parte, las applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones, etc.

3.1.1 Convenciones de Código Java

Las convenciones de código de Java o “Code Conventions” son una serie de reglas preestablecidas que sirven para homologar la codificación en el lenguaje Java.

Las convenciones de codificación son importantes para los programadores por varias razones:

* El 80% del coste del tiempo de vida de una pieza de software se va en mantenimiento.
* Casi nunca ningún software es mantenido durante toda su vida por su autor original.
* Las convenciones de nombrado mejoran la lectura del software, permitiendo a los ingenieros entender el nuevo código más rápidamente y mejor.
* Si lanzamos nuestro código fuente como un producto, necesitamos asegurarnos de que está tan bien empaquetado y limpio como cualquier otro producto que creemos.

Para que las convenciones cumplan su objetivo es necesario que los programadores de software se adhieran a ellas lo más cerca posible. .
.
Para verlas más a detalle referirse al Anexo.

3.2 Tipo de Aplicación

Para el sistema de Reportes Automáticos Web “RAW” en su módulo administrativo se considera como una aplicación con Arquitectura Cliente Servidor, por lo que más adelante se dará una breve descripción de cómo funciona una aplicación de este tipo. Por otra parte, el módulo de generación de reportes es una Aplicación Web por lo que también se verá una breve descripción y características que componen una Aplicación Web.

3.2.1 Cliente Servidor

Esta arquitectura consiste básicamente en que un programa “Cliente” realiza peticiones a otro programa, el “Servidor”, que le da respuesta.

Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa.

Una disposición muy común son los sistemas multicapas en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

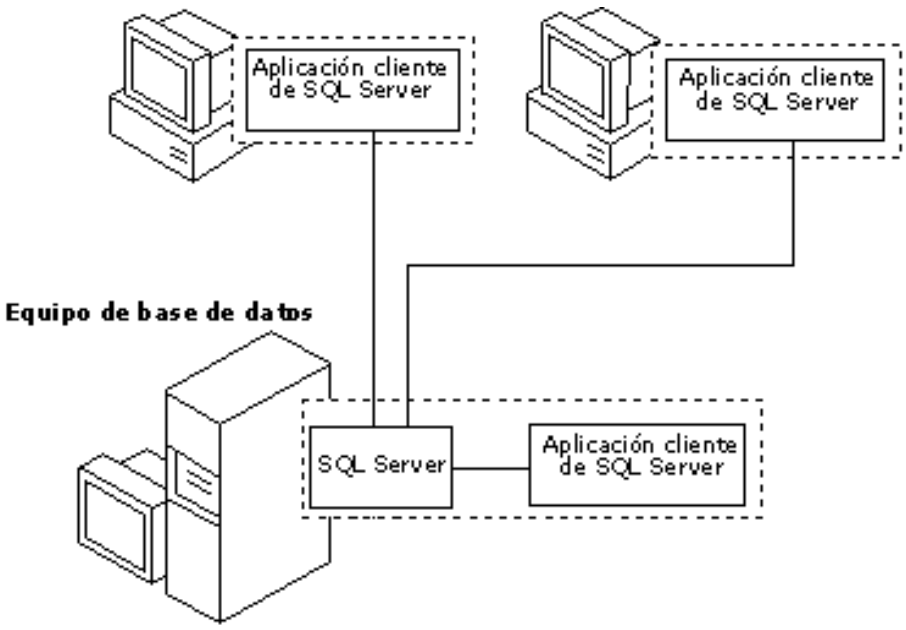
Ventajas de la arquitectura cliente-servidor:

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado.

El servidor de cliente es la arquitectura de red que separa al cliente (a menudo un uso que utiliza un interfaz utilizador gráfico) de un servidor. Cada caso del software del cliente puede enviar peticiones a un servidor. Los tipos específicos de servidores incluyen los servidores web, los servidores del uso, los servidores de archivo, los servidores terminales, y los servidores del correo. Mientras que sus propósitos varían algo, la arquitectura básica sigue siendo igual.

Aunque esta idea se aplica en una variedad de maneras, en muchas diversas clases de usos, el ejemplo más fácil de visualizar es el uso actual de páginas Web en el Internet. Por ejemplo, si se está consultando una información en la página de “www.unam.mx”, la computadora y el navegador Web serían considerados como un cliente, y las computadoras, las bases de datos, y las aplicaciones que componen la página de la UNAM serían consideradas como el servidor. Cuando el navegador Web solicita una información en particular de la página, el servidor de la UNAM encuentra toda la información requerida en la base de datos del servidor de la UNAM, la monta en la página Web y la envía de nuevo al navegador de Internet.

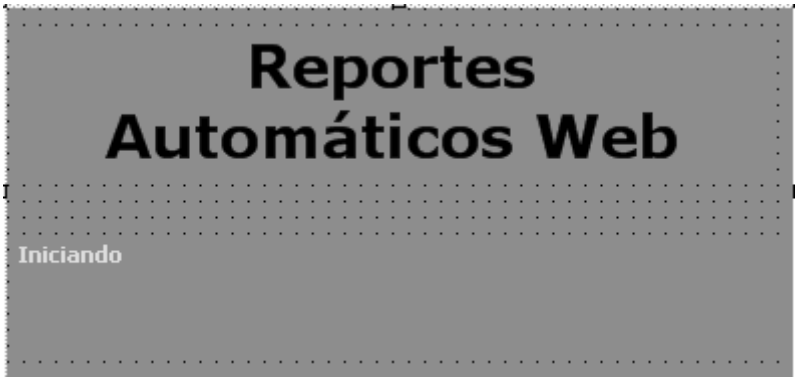
Diagrama que muestra la distribución de una aplicación Cliente Servidor.



3.2.1.1 Interfaz Gráfica Cliente Servidor del Sistema RAW

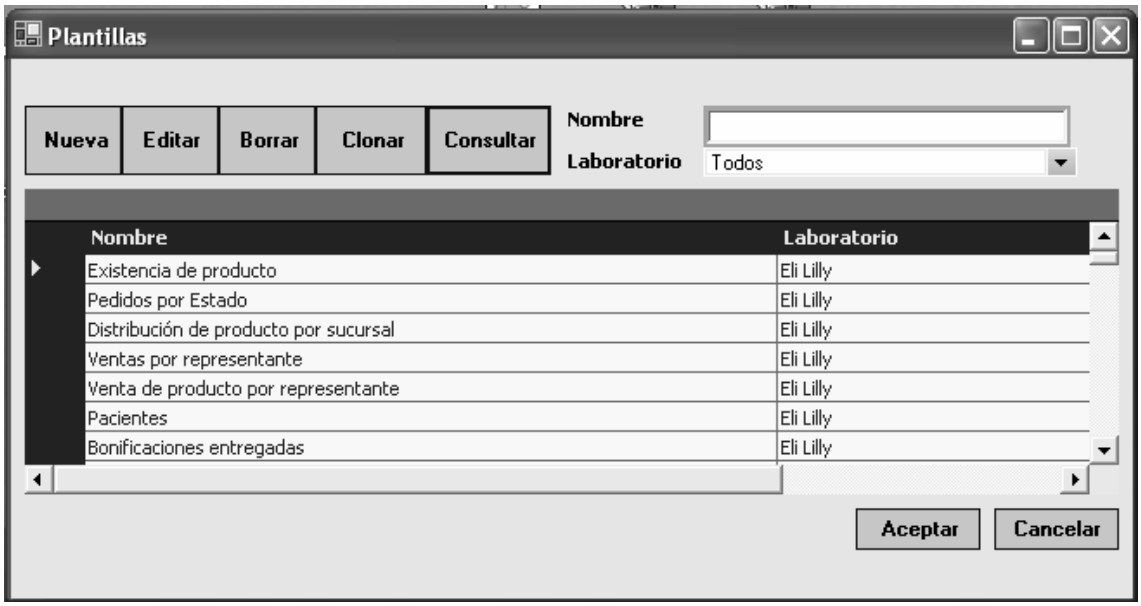
A continuación se muestra el diseño final de las interfaces gráficas del sistema de Reportes Automáticos Web correspondientes al módulo Cliente Servidor. En las cual se puede apreciar un diseño sencillo pero funcional que consta de nueva pantallas

Pantalla de presentación “Splash” del módulo administrativo del Sistema RAW.



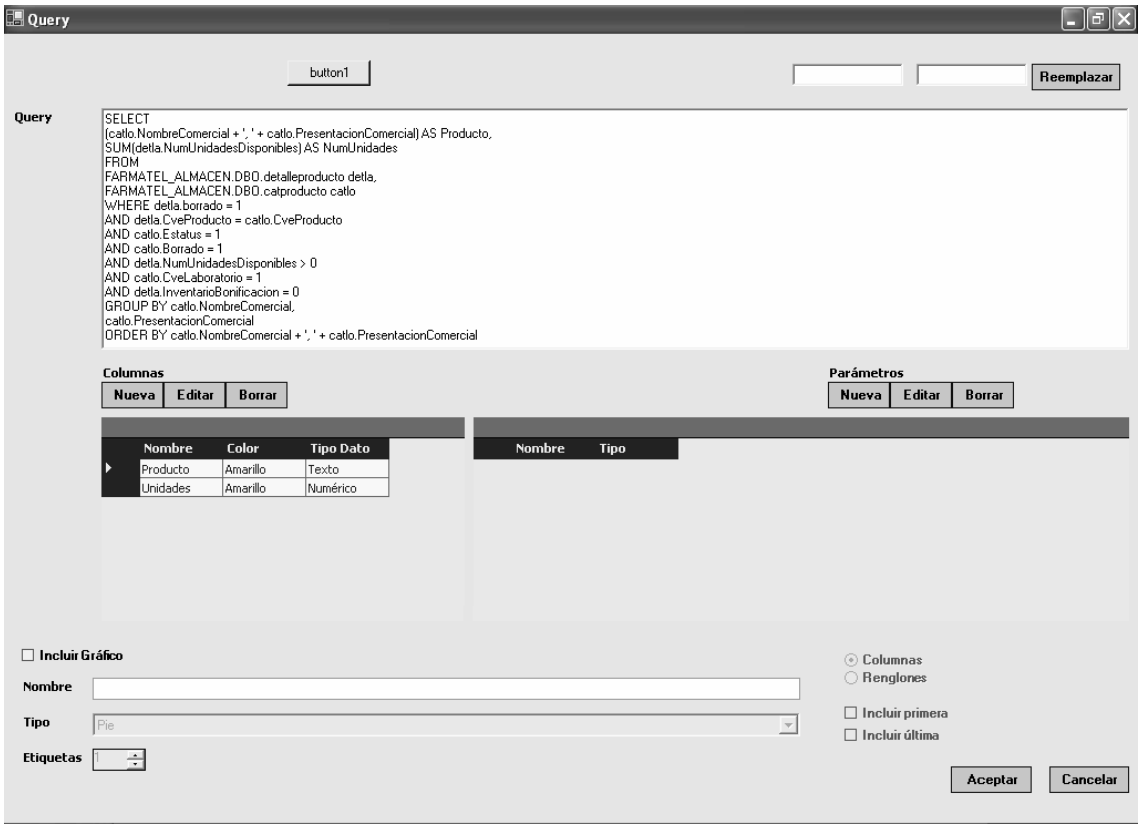
Pantalla inicial del módulo administrativo del sistema RAW.

Pantalla de “Administración de Plantillas” del módulo administrativo del sistema RAW.

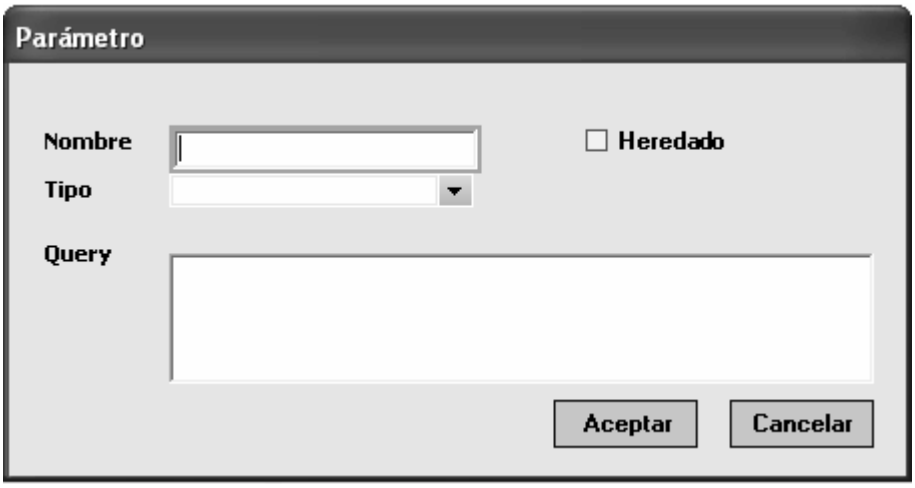


Pantalla de “Datos de Plantilla” en el módulo administrativo del sistema RAW.

Pantalla de “Datos de Query” (Consultas) en el módulo administrativo del sistema RAW.

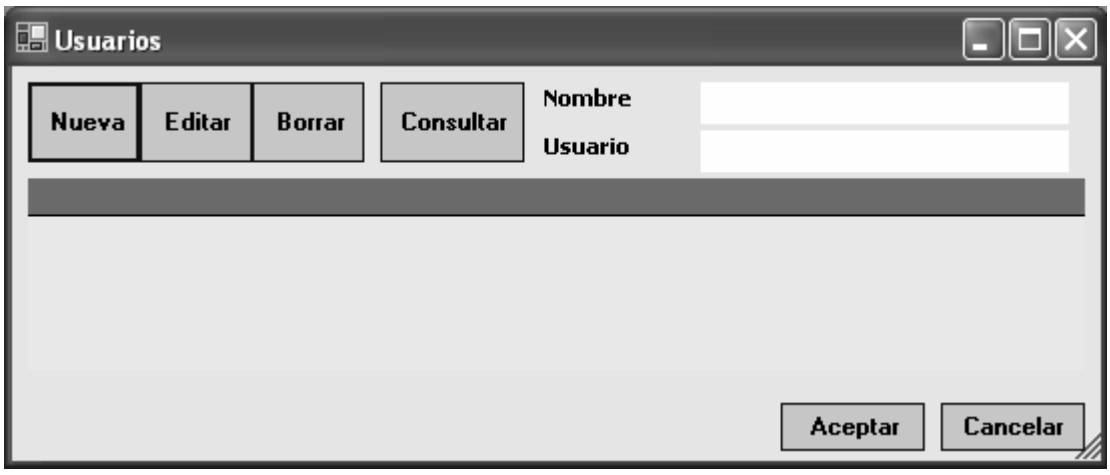


Pantalla de “Datos de Parámetro” en el módulo administrativo del sistema RAW.



Pantalla de “Datos de Columna” en el módulo administrativo del sistema RAW.

Pantalla de “Administración de usuarios” en el módulo administrativo del sistema RAW.



Pantalla de “Datos de Usuario” en el módulo administrativo del sistema RAW.

3.2.2 Aplicación Web

Una aplicación Web es un sistema informático que los usuarios utilizan accediendo a un servidor Web a través de Internet o de una intranet. Las aplicaciones Web son populares debido a la practicidad del navegador Web como cliente ligero. La facilidad para actualizar y mantener aplicaciones Web sin distribuir e instalar software en miles de potenciales clientes.

Una ventaja significativa en la construcción de aplicaciones Web que soporten las características de los Navegadores Web estándar es que deberían funcionar igual independientemente de la versión del sistema operativo instalado en el cliente. En vez de crear clientes para Windows, Mac OS X, GNU/Linux, u otros sistemas operativos, la aplicación es escrita una vez y es mostrada casi en todos lados.

Sin embargo, aplicaciones inconsistentes de HTML, CSS, DOM y otras especificaciones de Navegadores Web pueden causar problemas en el desarrollo y soporte de aplicaciones Web. Adicionalmente, la habilidad de los usuarios a personalizar muchas de las características de la interfaz (como tamaño y color de fuentes, tipos de fuentes, inhabilitar Javascript) puede interferir con la consistencia de la aplicación Web.

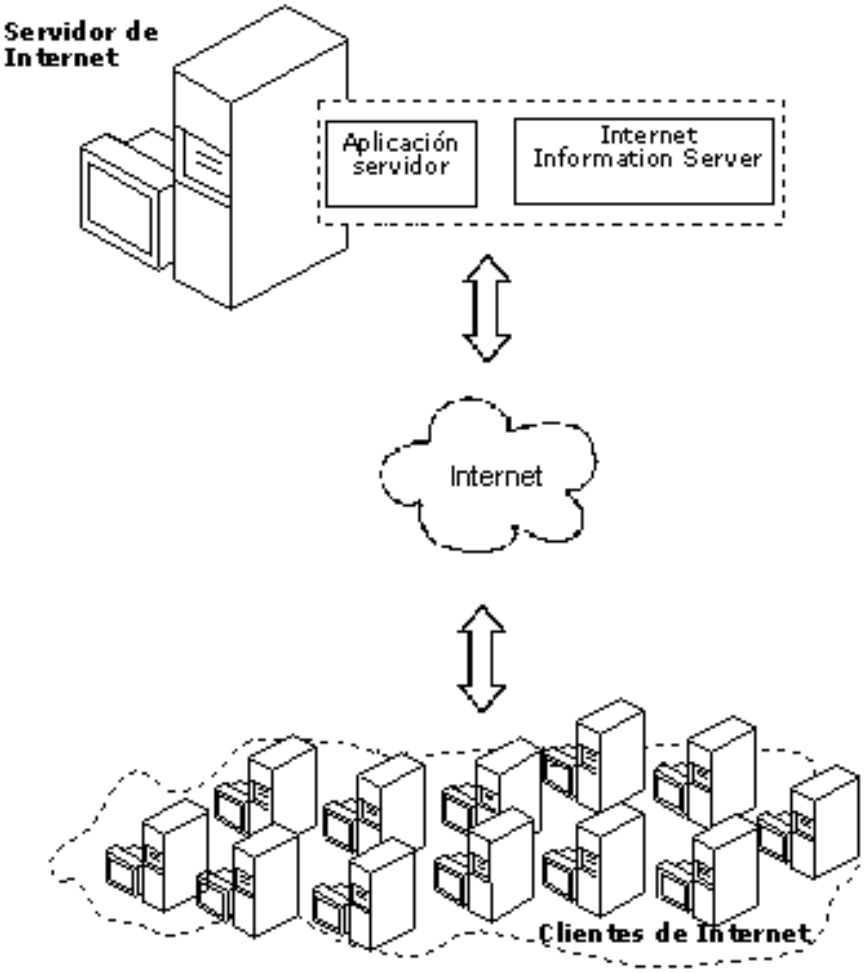
Las interfaces Web tienen ciertas limitaciones en la funcionalidad del cliente. Métodos comunes en las aplicaciones de escritorio como dibujar en la pantalla o arrastrar-y-soltar no están soportadas por las tecnologías Web estándar. Los desarrolladores Web comúnmente utilizan lenguajes interpretados del lado del cliente para añadir más funcionalidad, especialmente para crear una experiencia interactiva que no requiera recargar la página cada vez (cosa que suele molestar a los usuarios). Recientemente se han desarrollado tecnologías para coordinar estos lenguajes con tecnologías del lado del servidor, como por ejemplo PHP. AJAX, es una técnica de desarrollo Web que usa una combinación de varias tecnologías.

Otra variedad de desarrollo Web es utilizar Macromedia Flash o Java applets para producir parte o toda la interfaz de usuario. Como casi todos los Navegadores de Internet incluyen soporte para estas tecnologías (usualmente por medio de plug-ins), aplicaciones basadas en Flash o Java pueden ser implementadas con aproximadamente la misma facilidad. Como hacen caso omiso de las configuraciones de los Navegadores estas tecnologías permiten más control sobre la interfaz, aunque incompatibilidad entre implementaciones de Flash o Java puedan traer nuevas complicaciones. Por las similitudes con una arquitectura cliente-servidor, con un cliente un poco “especializado”, hay disputas sobre si llamar a estos sistemas “Aplicaciones Web”; un término alternativo es “Aplicación Enriquecida de Internet”.

Aunque muchas variaciones son posibles, una aplicación Web está comúnmente estructurada como una aplicación de tres-capas. En su forma más común, el navegador Web es la primera

capa, un motor usando alguna tecnología Web dinámica (ejemplo: CGI, PHP, Java Servlets, JSP o ASP) es la capa de en medio, y una base de datos como última capa. El navegador Web manda peticiones a la capa media, que la entrega valiéndose de consultas y actualizaciones a la base de datos generando una interfaz de usuario.

Diagrama que muestra una visión general de una aplicación Web

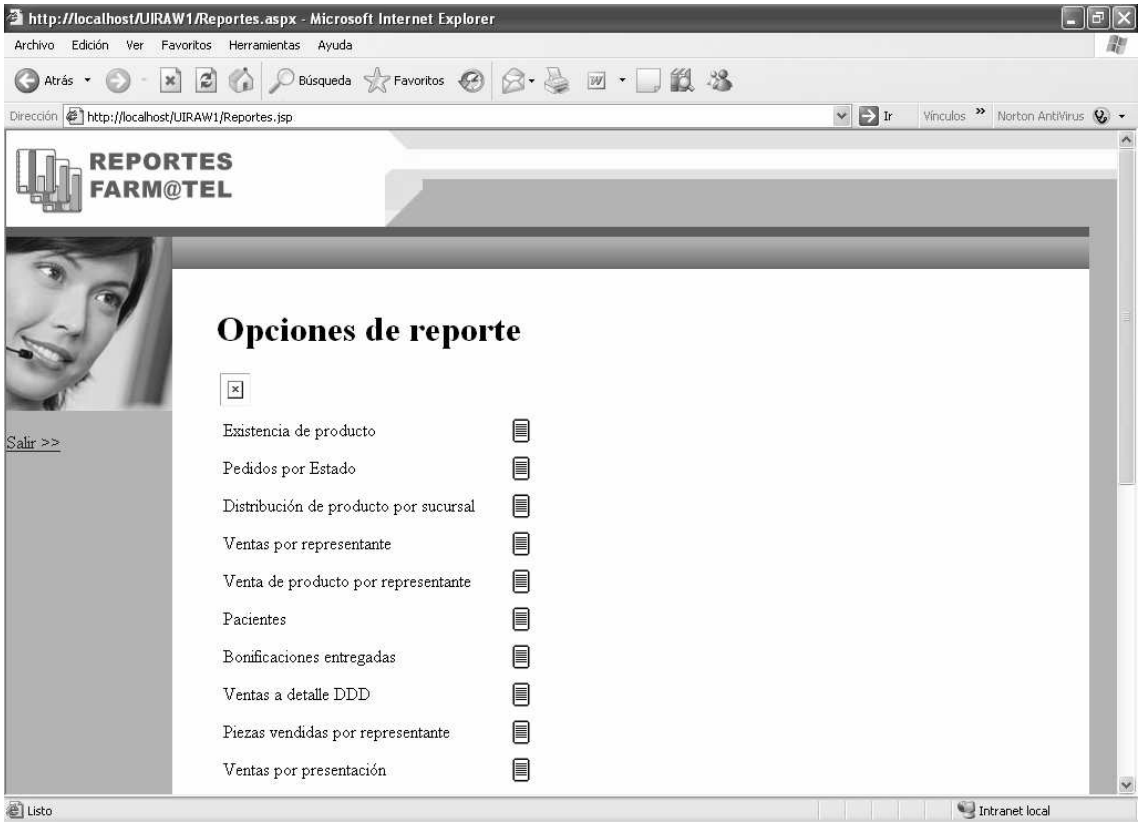


3.2.2.1 Interfaz Grafica Web del sistema RAW

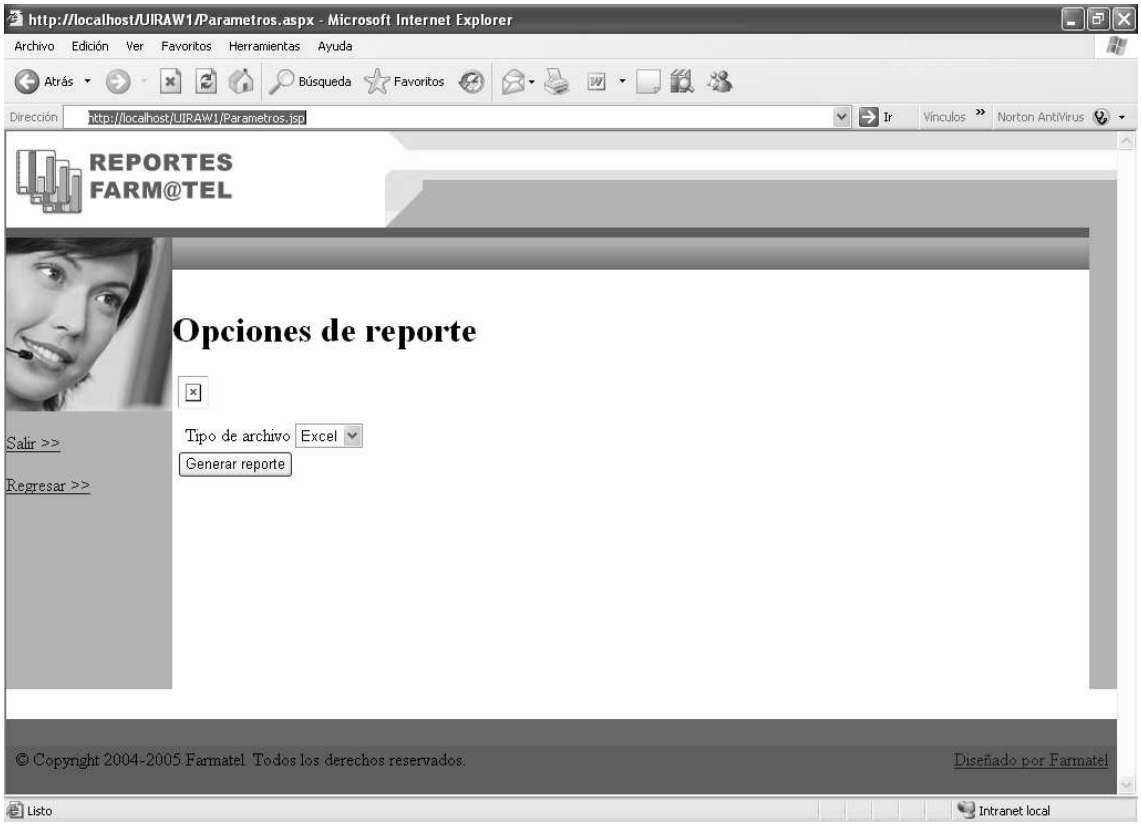
A continuación se muestra el diseño final de las interfaces gráficas del sistema de Reportes Automáticos Web correspondientes al módulo WEB. En las cual se puede apreciar un diseño uniforme con tonalidades en color azul y el logo de la empresa en la esquina superior izquierda.

Pantalla de “Bienvenida y Login” en el Sistema RAW.

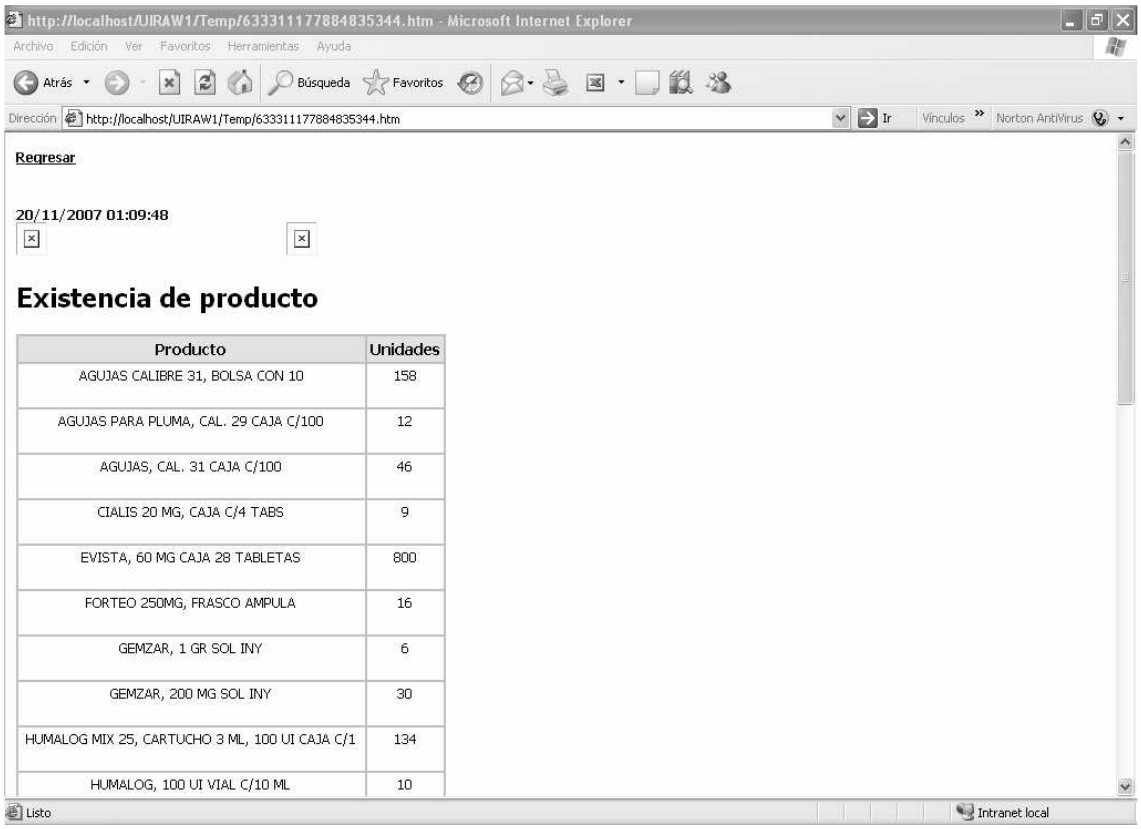
Pantalla de “Opciones de Reportes” en el Sistema RAW.



Pantalla de “Generación de Reporte” en el Sistema RAW



Pantalla de “Reporte Generado” en el Sistema RAW.



3.3 Herramientas de desarrollo

3.3.1 Herramienta de Desarrollo Java

Para el desarrollo del proyecto RAW, se utilizo la herramienta NetBeans 5.0. la cual se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un Entorno integrado de desarrollo (IDE).

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios (¡y subiendo!) en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

Historia
NetBeans comenzó como un proyecto estudiantil en Republica Checa (originalmente llamado Xelfi), en 1996 bajo la tutoría de la Facultad de Matemáticas y Física en la Universidad de Charles en Praga. La meta era escribir un entorno integrado de desarrollo (IDE) para Java parecida a la de Delphi. Xelfi fue el primer entorno integrado de desarrollo escrito en Java, con su primer pre-release en 1997.

Xelfi fue un proyecto divertido para trabajar, ya que las IDEs escritas en Java eran un territorio desconocido en esa época. El proyecto atrajo suficiente interés, por lo que los estudiantes, después de graduarse, decidieron que lo podían convertir en un proyecto comercial. Prestando espacios web de amigos y familiares, formaron una compañía alrededor de esto. Casi todos ellos siguen trabajando en NetBeans.

Tiempo después, ellos fueron contactados por Roman Stanek, un empresario que ya había estado relacionado con varias iniciativas en la Republica Checa. Él estaba buscando una buena idea en que invertir, y encontró en Xelfi una buena oportunidad. Ellos se reunieron, y el negocio surgió.

El plan original era desarrollar unos componentes JavaBeans para redes. Jarda Tulach, quien diseñó la arquitectura básica de la IDE, surgió con la idea de llamarlo NetBeans, con el fin de describir lo que ellos harían. Cuando las especificaciones de los Enterprise JavaBeans salieron, ellos decidieron trabajar con este estándar, ya que no tenía sentido competir con él, sin embargo el nombre de NetBeans se quedó.

En la primavera de 1999, DeveloperX2 fue lanzado, soportando Swing. Las mejoras de rendimiento que llegaron con el JDK 1.3, lanzado en otoño de 1999, hicieron a NetBeans una alternativa realmente viable para el desarrollo de herramientas. En el verano de 1999, el equipo trabajó duro para rediseñar a DeveloperX2 en un NetBeans más modular, lo que lo convirtió en la base de NetBeans hoy en día.

Algo más paso en el verano de 1999. Sun Microsystems quería una mejor herramienta de desarrollo de Java, y comenzó a estar interesado en NetBeans. En otoño de 1999, con la nueva generación de NetBeans en Beta, el acuerdo fue realizado.

Sun adquirió otra compañía de herramientas al mismo tiempo, Forté, y decidió renombrar NetBeans a Forté for Java. El nombre de NetBeans desapareció de vista por un tiempo.

Seis meses después, se tomó la decisión de hacer a NetBeans open source. Mientras que Sun había contribuido considerablemente con líneas de código en varios proyectos de código

abierto a través de los años, NetBeans se convirtió en el primer proyecto de código abierto patrocinado por ellos. En Junio del 2000 NetBeans.org fue lanzado.

NetBeans Hoy

Un proyecto de código abierto no es nada más ni nada menos que un proceso. Toma tiempo encontrar el balance perfecto. Alrededor del primer año, (a través de NetBeans 3.2) fue crucial como inicio. Los dos años siguientes, se aprendió que sirve en términos de procesos de código abierto. En los primeros dos años, el proceso de desarrollo era tan abierto, que había más debate que implementación.

Con NetBeans 3.5 se mejoró enormemente en desempeño, y con la llegada de NetBeans 3.6, se reimplementó el sistema de ventanas y la hoja de propiedades, y se limpió enormemente la interfaz. NetBeans 4.0 fue un gran cambio en cuanto a la forma de funcionar del IDE, con nuevos sistemas de proyectos, con el cambio no solo de la experiencia de usuario, sino del reemplazo de muchas piezas de la infraestructura que había tenido NetBeans anteriormente. NetBeans IDE 5.0 introdujo un soporte mucho mejor para el desarrollo de nuevos módulos, el nuevo constructor intuitivo de interfaces Matisse, un nuevo y rediseñado soporte de CVS, soporte a Sun ApplicationServer 8.2, Weblogic9 y JBoss 4. Netbeans recomendable

NetBeans IDE

El IDE NetBeans es un IDE - una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

El NetBeans IDE es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.

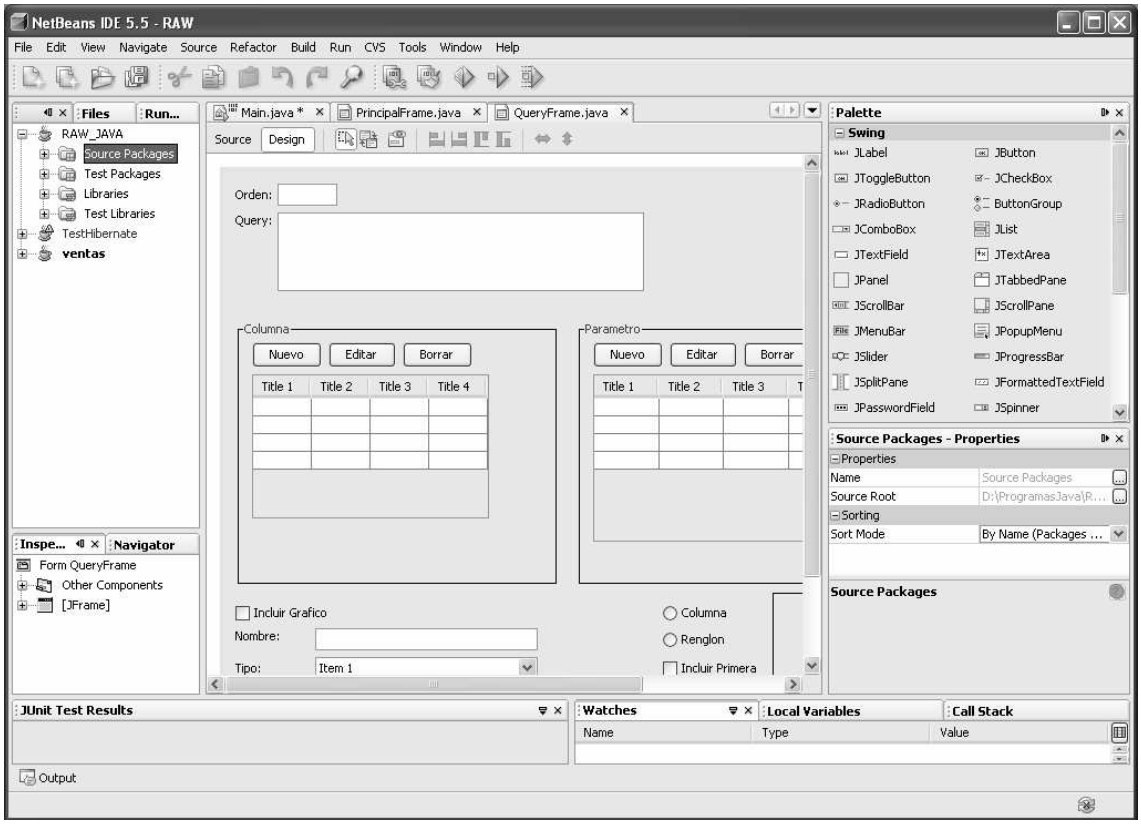
La versión actual es NetBeans IDE 5.5, la cual fue lanzada en Octubre de 2006. NetBeans IDE 5.5 extiende las características existentes del Java EE (incluyendo Soporte a Persistencia, EJB 3 y JAX-WS). Adicionalmente, el NetBeans Enterprise Pack soporta el desarrollo de Aplicaciones empresariales con Java EE 5, incluyendo herramientas de desarrollo visuales de SOA, herramientas de esquemas XML, orientación a web servicies (for BPEL), y modelado UML. El NetBeans C/C++ Pack soporta proyectos de C/C++.

Modularidad. Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente.

Sun Studio, Sun Java Studio Enterprise, y Sun Java Studio Creator de Sun Microsystems han sido todos basados en el IDE NetBeans.

Desde Julio de 2006, NetBeans IDE es licenciado bajo la Common Development and Distribution License (CDDL), una licencia basada en la Mozilla Public License (MPL).

Pantalla que muestra la aplicación NetBeans IDE 5.5.

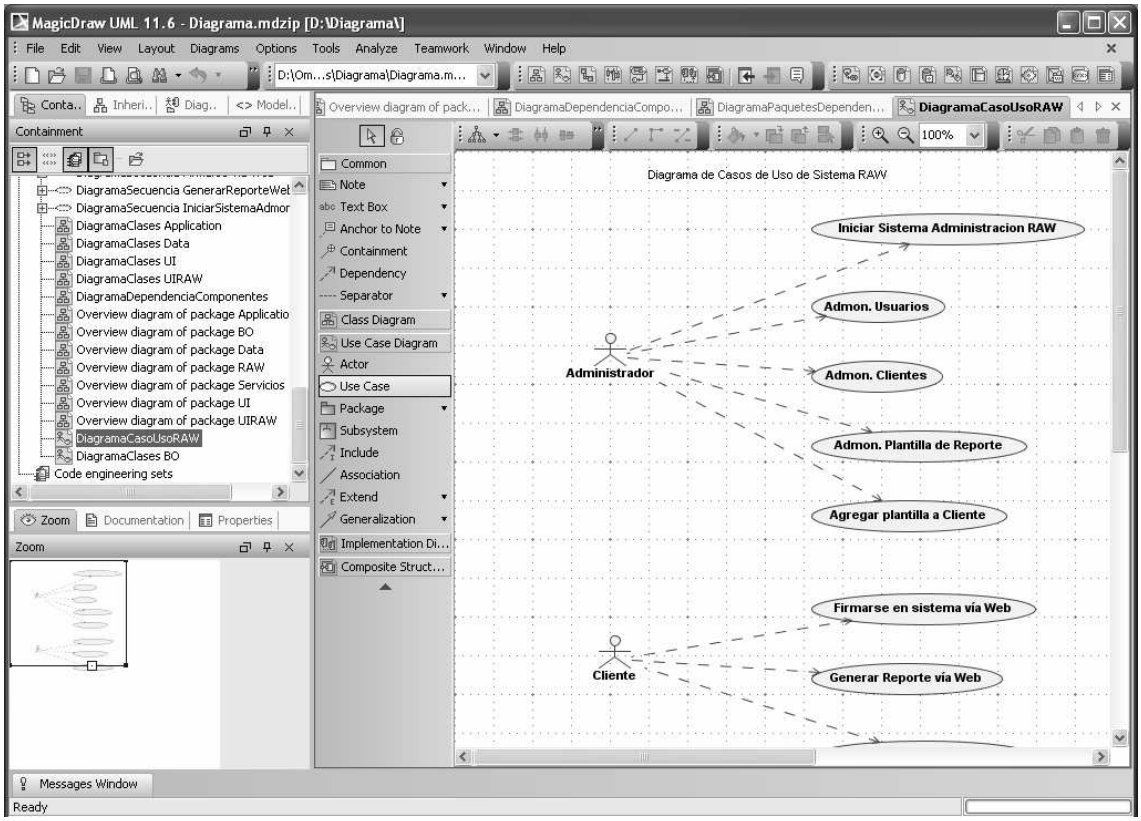


3.3.2 Herramienta de Diseño de Diagramas UML

Para el diseño y análisis de los diagramas en UML para el sistema RAW se utilizo el software llamado MagicDraw Community Edition, la cual es una versión gratuita para desarrollos personales.

MagicDraw 10.5 es una herramienta de modelaje con completas características UML. Ha sido implementada al 100% en JAVA y se ejecuta en la mayor parte de las plataformas.
Diseñado para los analistas del negocio, los analistas del software, los programadores, los ingenieros del QA, y los escritores de la documentación, esta herramienta de desarrollo dinámica y versátil facilita análisis y el diseño de los sistemas (OO) y de las bases de datos orientados objeto.

Pantalla que muestra la aplicación MagicDraw UML.



3.3.3 Herramienta de Diseño de Base de Datos

ERwin es una herramienta de base de datos que ayuda a diseñar, modelar, generar y mantener aplicaciones de base de datos de calidad y alto rendimiento. Desde un modelo lógico de sus requerimientos de información y reglas del negocio que definen su base de datos, hasta un modelo físico, optimizado por las características específicas de su base de datos de destino, ERwin le permite visualizar la estructura adecuada, los elementos clave y un diseño optimizado de su base de datos.

ERwin genera tablas automáticamente y miles de líneas de stored procedures y código trigger para las principales bases de datos. Su tecnología "complete-compare" permite el desarrollo interactivo, de manera que su modelo está siempre sincronizado con su base de datos. A través de la integración con los ambientes de desarrollo líderes en la industria, ERwin también acelera la creación de aplicaciones data-centric.

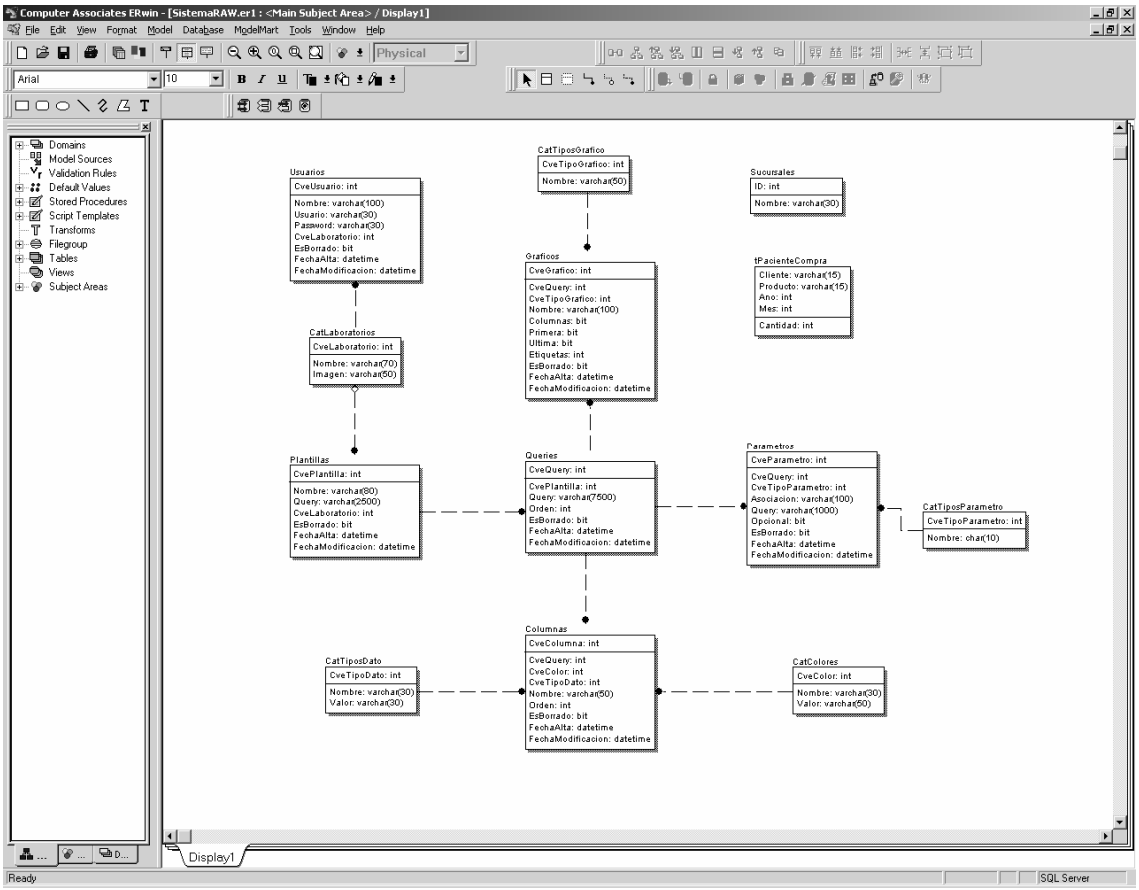
Beneficios de ERwin

- Asegura consistencia, reuso, e integración de los datos del proyecto al proporcionar el bosquejo que las IT necesitan para entender, analizar y comunicar la estructura de la base de datos.
- Mejora la productividad entre los desarrolladores cuando los diseños de la base de datos son divididos, compartidos, y reutilizados.
- El ambiente gráfico facilita la visualización de la estructura completa, los elementos claves y el diseño optimizado de la base de datos.
- Le ahorra tiempo al acelerar la creación de bases de datos de alta calidad, transaccionales de alto rendimiento y para data warehouse.
- Mantiene los recursos y mejora la precisión al sincronizar el modelo y la base de datos.

AllFusion ERwin Data Modeler es una herramienta de diseño de base de datos que ayuda a los usuarios a diseñar, generar y mantener alta calidad de las aplicaciones de base de datos de alta performance. AllFusion ERwin Data Modeler permite al usuario visualizar la estructura correcta, elementos claves y el diseño optimizado de su base de datos, desde los requerimientos de un modelo lógico de información y reglas de negocio que definen la base de datos, a un modelo físico optimizado para las características específicas de la base de datos seleccionada.

AllFusion ERwin Data Modeler automáticamente genera tablas y miles de líneas de procedimientos almacenados y códigos disparadores para las base da datos líderes. Su tecnología de "comparación completa" permite el desarrollo iterativo, de forma tal que los modelos están siempre sincronizados con la base de datos del usuario. Al integrarse con entornos de desarrollo líderes, AllFusion ERwin Data Modeler también acelera la creación de aplicaciones centralizadas en datos.

Pantalla que muestra la aplicación ERwin.



3.4 Entorno de Ejecución de la Aplicación

3.4.1 Sistema Operativo

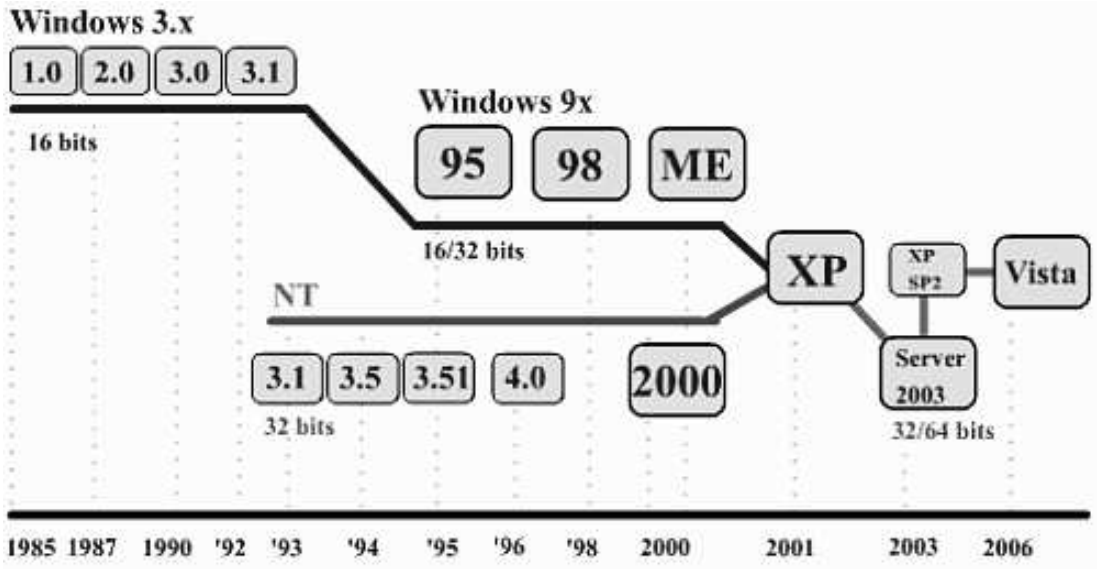
El sistema de Reportes Automáticos Web “RAW” en su módulo administrativo y Web al ser programado en el lenguaje Java lo hacen multiplataforma, es decir podrá ser ejecutado en plataformas Windows, Unix, Linux, etc. Pero se tiene contemplado que sea ejecutado en un ambiente Microsoft Windows (2000, XP o Vista).

Microsoft Windows
(Conocido simplemente como Windows) es un sistema operativo con interfaz gráfica para computadoras personales cuyo propietario es la empresa Microsoft. Las distintas versiones de Windows, las cuales ofrecen un entorno gráfico amigable y sencillo, principalmente desde la versión Windows 95, ha convertido en Windows en el sistema operativo más utilizado en el mundo. Debido a ello la mayoría de las empresas fabricantes de hardware y software en el mundo tienden a desarrollar sus aplicaciones basadas en dicho sistema.

Windows ha incorporado a través de sus diferentes versiones múltiples herramientas que se han convertido en estándares en la mayoría de los usuarios en el mundo. Así, Windows incorpora, entre otro software, herramientas como Internet Explorer y el Reproductor de Windows Media, los cuales se han convertido en el navegador de internet y reproductor multimedia, respectivamente, más populares en el mundo.

Windows es utilizado principalmente en computadoras personales existiendo también diferentes versiones para servidores y dispositivos móviles

Diagrama que muestra la evolución que ha tenido Windows.



3.4.2 Manejador de Base de Datos

Debido a que el sistema RAW va a trabajar con información que es generada por otros sistemas en producción que utilizan como manejador de base de datos a Microsoft SQL Server 2000, entonces lo más eficiente en cuanto características, compatibilidad y licenciamiento fue utilizar la misma aplicación para gestionar los reportes.

Microsoft SQL Server 2000
Es un sistema de gestión de bases de datos relacionales (SGBD) basada en el lenguaje Transact-SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Así de tener unas ventajas que a continuación se pueden describir.
Entre sus características figuran:

- Manejo de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.

- Permite trabajar en modo cliente-servidor donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos

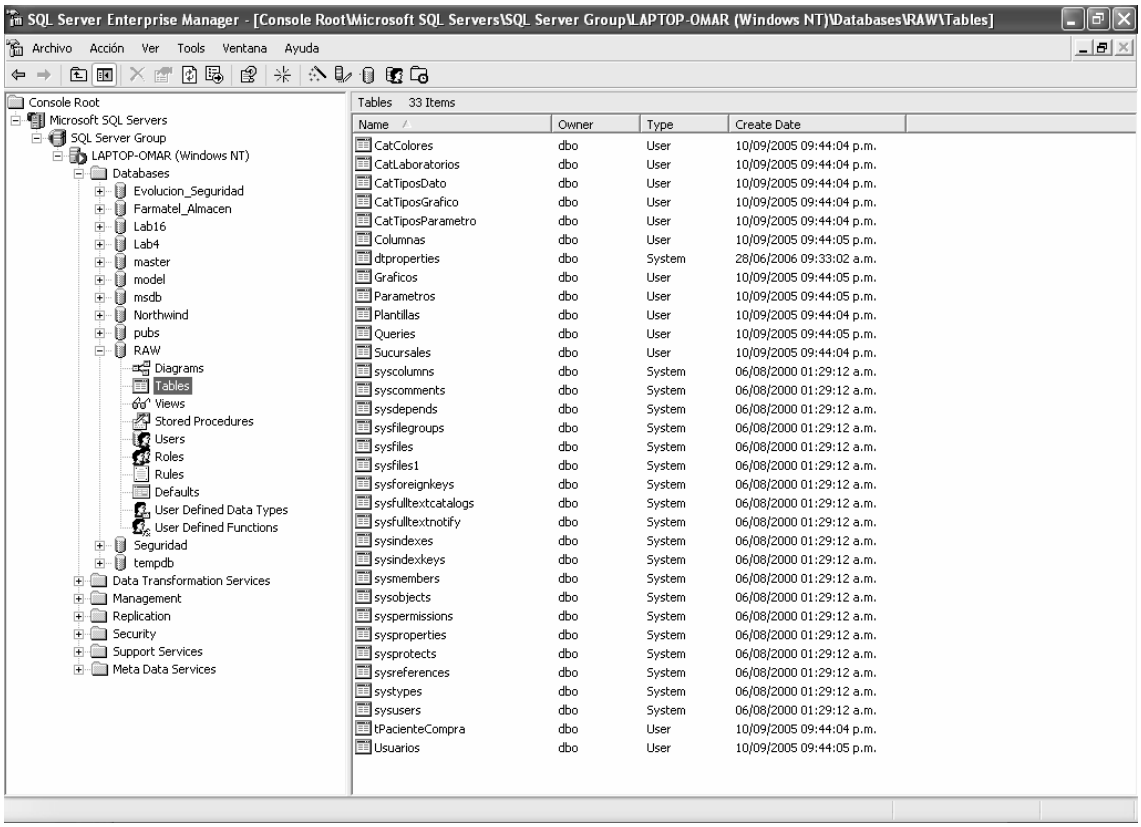
Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños, que en su versión 2005 pasa a ser el SQL Express Edition.

Microsoft SQL Server constituye la alternativa de Microsoft contra otros sistemas gestores de bases de datos como son Oracle, Sybase, MySQL (libre), Postgree (libre), etc.

Es común desarrollar proyectos enteros complementados por Microsoft SQL Server a través de los llamados ADP (Access Data Project). De esta forma se complementa una potente base de datos (Microsoft SQL Server) con un entorno de desarrollo cómodo y de alto rendimiento (VBA Access) a través de la utilización de aplicaciones de tres capas mediante el uso de formularios Windows.

Microsoft SQL Server, al contrario de su más cercana competencia, no es multiplataforma, ya que sólo está disponible en Sistemas Operativos de Microsoft.

Pantalla en donde se muestra la aplicación SQL Server.



3.4.3 Servidor Web

Apache TomCat

Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

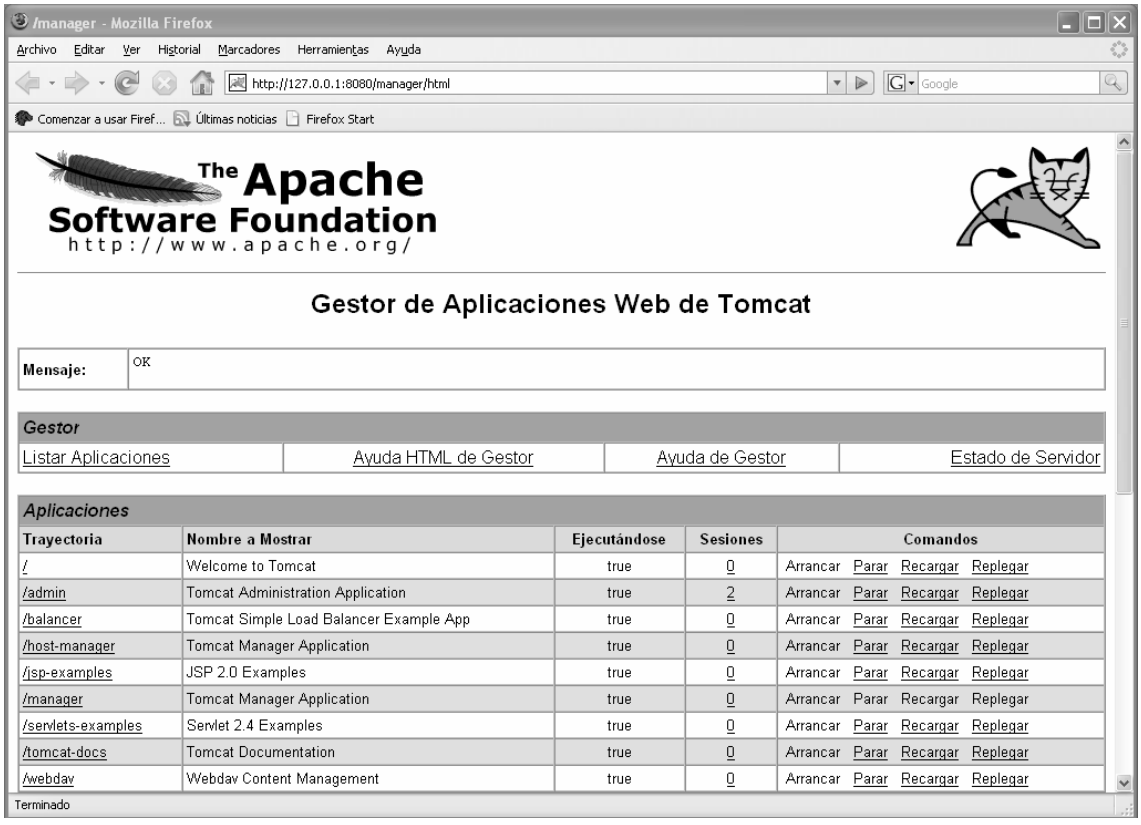
Tomcat es un servidor Web con soporte de servlets y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor Web Apache. Aunque también puede funcionar como servidor Web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor Web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

Tomcat empezó siendo una implementación de la especificación de los servlets comenzada por James Duncan Davidson, que trabajaba como arquitecto de software en Sun Microsystems y que posteriormente ayudó a hacer el proyecto open source y en su donación a la Apache Software Foundation.

Duncan Davidson inicialmente esperaba que el proyecto se convirtiese en open source y dado que la mayoría de los proyectos open source tienen libros de O'Reilly asociados con un animal en la portada, quiso ponerle al proyecto nombre de animal. Eligió Tomcat (gato), pretendiendo representar la capacidad de cuidarse por sí mismo, de ser independiente.

Pantalla del Manager Web de Apache TomCat



3.4.4 Interfaz Grafica de Usuario

3.4.4.1 Microsoft Internet Explorer

Microsoft Windows Internet Explorer (también conocido antes como Internet Explorer, IE o MSIE) es un navegador de Internet producido por Microsoft para su plataforma Windows y más tarde para Apple Macintosh y Solaris Unix. Las versiones para estos dos últimos sistemas fueron descontinuadas en el 2006 y 2002 respectivamente.

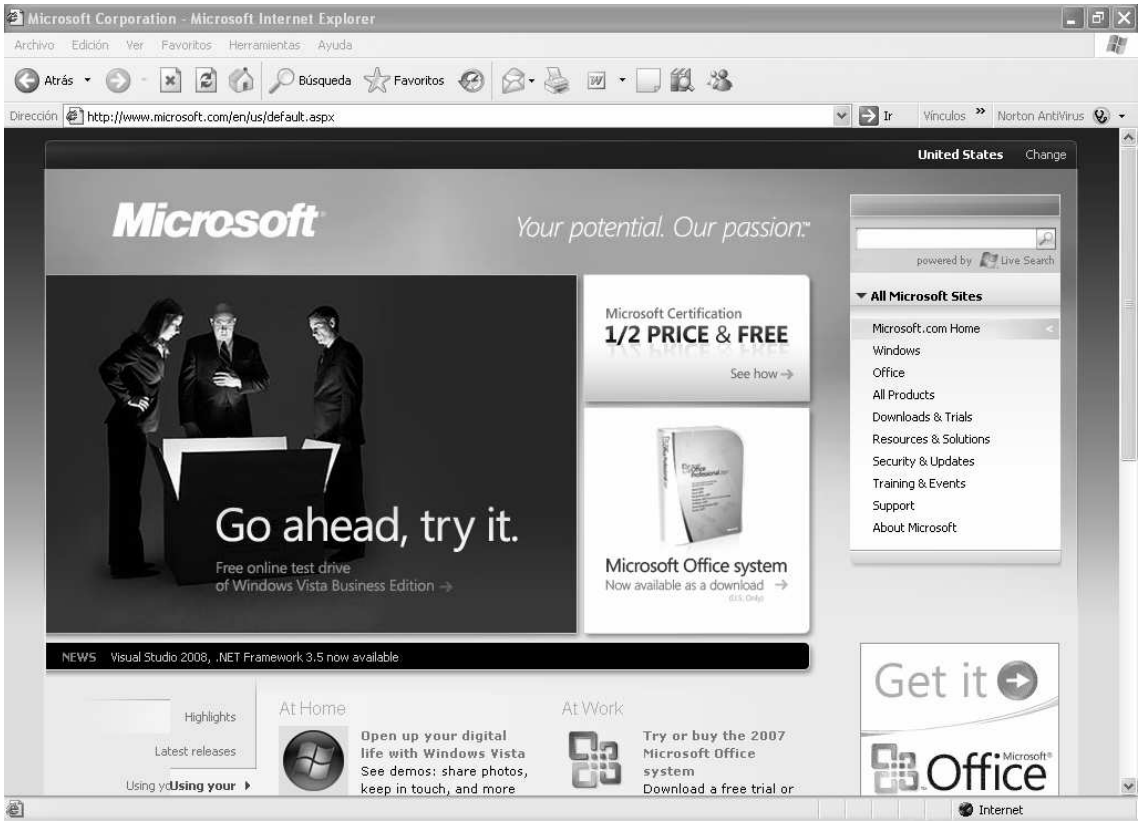
Fue creado en 1995 tras la adquisición por parte de Microsoft del código fuente de Mosaic, un navegador desarrollado por Spyglass, siendo rebautizado entonces como Internet Explorer. Actualmente es el navegador de Internet más popular y más utilizado en el mundo, rebasando en gran medida a las competencias existentes, aún cuando algunas de éstas han incrementado su popularidad en los últimos años. Las primeras versiones, basadas en Mosaic, no supusieron ninguna amenaza para el entonces dominante Netscape Navigator, ya que eran bastante simples y no eran compatibles con algunas de las extensiones más populares de Netscape que dominaban la web de la época (como los marcos o JavaScript).

Tras llegar a controlar un aplastante 94% del mercado de los navegadores, a partir de junio de 2004 su cuota de mercado empezó a disminuir ligeramente en beneficio de la familia de navegadores basados en Mozilla (basados en una versión libre del antiguo Netscape), especialmente el navegador Mozilla Firefox, que pasaron de 1% a 3% del mercado en ese mismo período. En la actualidad (2007), están rebasando el 15%, sin embargo, Internet Explorer sigue estando a la cabeza de popularidad por gran medida. Otro de los competidores es Opera, que destaca por su innovación, además de tener la mayoría del mercado de los navegadores en móviles con su versión Opera Mini.

Uno de los motivos de este progresivo declive es la seguridad. La débil seguridad del navegador en conjunto con su profunda integración en el sistema operativo han comprometido en numerosas ocasiones la seguridad integral de la plataforma Windows. Actualmente se tiene constancia de que ciertos agujeros conocidos de seguridad del navegador siguen aún sin enmendar. Además, debido a la falta de actualizaciones, el nivel de conformidad con los estándares actuales de la web establecidos por el W3C es muy inferior al de los navegadores modernos.

El 31 de enero de 2006, ante los progresivos avances de su competencia, Microsoft sacó la versión 7 beta de su navegador, con un soporte mucho mayor de los estándares del W3C, aunque sin alcanzar aún el nivel de sus competidores. Se ha anunciado, sin embargo, que la nueva versión se actualizará con regularidad, de forma similar a sus competidores. Como ya ocurrió con la versión 6, esta nueva versión causará problemas en páginas desarrolladas pensando únicamente en Internet Explorer.

Pantalla que muestra la aplicación Microsoft Explorer.



3.4.4.2 Mozilla Firefox

Mozilla Firefox es un navegador de Internet, con interfaz gráfica de usuario desarrollado por la Corporación Mozilla y un gran número de voluntarios externos. Firefox, oficialmente abreviado como Fx, y comúnmente como FF, comenzó como un derivado del Mozilla Application Suite, que terminó por reemplazarlo como el producto bandera del proyecto Mozilla, bajo la dirección de la Fundación Mozilla.

Mozilla Firefox es un navegador web multiplataforma, que está disponible en versiones para Microsoft Windows, Mac OS X y GNU/Linux. Sin embargo el código ha sido portado por terceros a otros sistemas operativos como FreeBSD, OS/2, Solaris, SkyOS, BeOS y más recientemente, Windows XP Professional x64 Edition.

El código fuente de Firefox está disponible libremente bajo la triple licencia de Mozilla como un programa libre y de código abierto. La versión estable actual es la 2.0.0.7, que fue publicada el 18 de septiembre de 2007.

Firefox se basa en el motor XULRunner, desarrollado en su mayor parte utilizando el lenguaje XUL. Comenzó como un fork del navegador de la Mozilla Application Suite, y se ha convertido en el principal foco de desarrollo de la Fundación Mozilla junto con el cliente de correo electrónico y lector de noticias, (Thunderbird), reemplazando a Mozilla Suite como producto estrella de la fundación. Otros proyectos que surgieron de este esfuerzo son Nvu (editor web) y Mozilla Sunbird (agenda electrónica).

Antes de la liberación de la versión 1.0 el 9 de noviembre de 2004, Firefox ya había concentrado las miradas de ciertos medios de comunicación, incluyendo a Forbes y el Wall Street Journal.

Con más de 25 millones de descargas en los 99 días siguientes a la liberación de la v1.0, Firefox se convirtió en una de las aplicaciones libres más descargadas, especialmente entre los usuarios domésticos. El 19 de octubre de 2005, Firefox había alcanzado la cifra de 100 millones de descargas en menos de un año (344 días). La versión 1.5 llegó el 29 de noviembre de 2005, superándose la cota de los 2 millones de descargas en las primeras 36 horas. Para agosto de 2006 se habían superado los 200 millones.

Firefox incorpora bloqueo de ventanas emergentes, navegación por pestañas, marcadores dinámicos, soporte para estándares abiertos, y un mecanismo para añadir funcionalidades mediante extensiones. Aunque otros navegadores también incluyen estas características, Firefox fue el primero en incluir algunas de ellas y conseguir una amplia difusión.

Firefox ha atraído la atención de otros navegadores como Internet Explorer de Microsoft o Safari de Apple, que son incluidos por defecto en sus respectivos sistemas operativos, Windows y Mac OS X. Éste es el principal obstáculo por el que todavía muchos usuarios nóveles desconocen otras alternativas.

Para julio de 2006, se estima que el 12% de consultas a páginas web se realizan con Mozilla Firefox, encabezando Finlandia la lista de países por uso con un 37%.

Porcentaje de uso

Algunos navegantes han adoptado Firefox rápidamente, a pesar del dominio de Internet Explorer en el mercado de los navegadores. Según varias fuentes (citadas en la referencia de estadísticas), en noviembre de 2005, Firefox tenía alrededor de un 9,4% del mercado global, y un 10% en Norteamérica.

Pantalla que muestra la Aplicación Mozilla Firefox.

Capitulo 4 – Implantación del Sistema

4.1 Descripción del Proceso de Implantación

La implantación del Sistema de Reportes Automáticos Web “RAW” fue la parte menos compleja en el ciclo de vida de desarrollo del sistema, pero en cuanto al tempo que se llevo la liberación fue quizás la más larga.

Por otro lado en este capitulo no nos enfocaremos en explicar como se preparan los servidores de Aplicaciones, Web y de Base de Datos. Ya que este tema es demasiado largo y en ocasiones complejo como para querer resumirlo. Por lo tanto sólo nos enfocaremos en la instalación del sistema.

4.1.1 Pruebas Finales

Las pruebas finales del sistema corrieron por cuenta de los usuarios que anteriormente tenían que elaborar los reportes de forma manual, por lo que ellos eran las personas mas indicadas para validar la información generada por la aplicación.

Estas pruebas consistieron básicamente en hacer una comparación a detalle de la información generada de forma manual contra la obtenida a través del sistema de Reporte Automáticos Web. Obviamente se trataba de muchos reportes que revisar, por lo que se optó por tomar una muestra de diferentes tipos de reportes y sobre ellos hacer las revisiones a detalle.

El resultado de estás pruebas fue reportado al departamento de sistema para así poder discriminar el tipo de error y de ser necesario hacer una corrección sobre la aplicación.

En el caso de errores sobre el funcionamiento de la aplicación, esto fueron reportado y resueltos inmediatamente y puestos en evaluación para los usuario.

Para el caso de errores o inconsistencias de la información generada, se tuvo que hacer una revisión más detallada para así poder encontrar el problema, después de solucionado, esté era evaluado nuevamente por el usuario.

Es necesario mencionar que la mayoría de los errores e inconsistencias que se presentaron en esta fase de pruebas, fueron consecuencia de una configuración errónea más que de problemas del mismo sistema RAW o de inconsistencias en la información.

Después de que se contó con la aprobación de los usuarios se dio por finalizada la fase de pruebas y se procedió a implementar el sistema en un ambiente productivo, como se describe posteriormente.

4.1.2 Migración de Estructura de Base de Datos a Ambiente Productivo

El primer paso que se hace para migrar la base de datos a un ambiente productivo, es generar el Script de creación desde el modelador de base de datos, el cual tendrá el nombre “RawUnit.sql”, para así asegurarnos que hasta el último cambio en el diseño de la base de datos sea integrado.

Lo siguiente que se hace es el servidor de Base de Datos es crear la base de datos vacía con el nombre “SistemaRAW”. Después en la aplicación “Query Analizer” se deberá ejecutar el Script de creación con el fin de generar todas las Tablas, Índices, Llaves Primarias, Llaves Foráneas, y relaciones entre tablas etc.

Una ves que ya contamos con la base de datos para el Sistema RAW, se deberá dar de alta un usuario con el nombre de “ReportesRAW” a el cual se de asignaran permisos de propietario sobre la base de datos, esto con el fin de que este usuario pueda acceder a todas las tablas para leer y escribir en ellas.

Con estos simples pasos de da por terminado la migración de la base de datos a un ambiente de producción.

4.1.3 Instalación del Módulo Administrativo

Java es el primer lenguaje que tiene la virtud de ser compilado e interpretado de forma simultánea. Cuando un programador realiza una aplicación o un applet en Java y lo compila, en realidad, el compilador no trabaja como un compilador de un lenguaje al uso. El compilador Java únicamente genera el denominado ByteCode. Este código es un código intermedio entre el lenguaje máquina del procesador y Java. Evidentemente este código no es ejecutable por sí mismo en ninguna plataforma hardware, pues no se corresponde con el lenguaje de ninguno de los procesadores que actualmente se conocen (habrá que esperar a ver qué ocurre con los procesadores Java). Por lo tanto, para ejecutar una aplicación Java es necesario disponer de un mecanismo que permita ejecutar el ByteCode. Este mecanismo es la denominada Máquina Virtual Java. En cada plataforma (Unix, Linux, Windows 95/NT, Macintosh, etc.) existe una máquina virtual específica.

- **Instalación de la Maquina Virtual de Java**

La instalación es muy sencilla, una vez que contamos con el archivo de instalación j2re-1_5_0_(versión)-windows-i586-p.exe el cual es un archivo ejecutable. Por lo tanto sólo hay que ejecutarlo desde la línea de comando o desde el Administrador de Archivo, una ves hecho lo anterior sólo se deben seguir los pasos que indica el asistente de instalación.

- **Instalación de la Aplicación Administrativa RAW**

Lo primero que se debe hacer es copiar el archivo “AdminRAW.jar” en el directorio “C:\Farmatel\AdminsitradorRAW” y las librerías necesarias para ejecutar la aplicación en el directorio “C:\Farmatel\AdminsitradorRAW \Lib”.

Como se describió anteriormente las aplicaciones Java se ejecutan mediante un archivo ByteCode que es procesado y ejecutado por la maquina virtual de java, para lo cual se ejecuta desde la línea de comando una instrucción similar a la siguiente:

Java –jar “C:\Directorio Aplicacion\AdminRAW.jar” (Enter). ó

Java –jar “C:\Farmate\AdministradorRAW \ AdminRAW.jar” (Enter).

Para evitar ejecutar este comando cada vez que se deseé utilizar el programa, lo que se hace es guardar este comando en un archivo por lotes de MS-DOS mejor conocido como “Archivo.bat”, el cual se ejecuta como si fuera un “archivo ejecutable” o “Archivo.exe”. Este archivo .bat se guarda en el mismo directorio en donde se guardo la aplicación y desde ahí se crea un “Acceso Directo” el cual se guarda en el escritorio de Windows y desde el cual el Usuario final podrá arrancar la aplicación.

Contenido del archivo de configuración, el cual se guarda en un archivo de texto plano con una estructura de XML y en el cual se puede ver la cadena de conexión encriptada.

```
<? xml version="1.0" encoding="utf-8"?>
<configuration>
        <appSettings>
```

```
        <add key="Conexion" value =
"ad4EMzj9LIqGa7B+J2saKPzFf70tcPX1yG0tlddbu7I7PCjdzfcrW/Tgko1n4kPAlLZ97j7m1X9sjso
D6T5IDlu29PfQGy9BrZixov2bPOe+WfqJDsT+xLCC6RfLi4PZ8yEI1TlIMFc=" />
        <add key="Nombre" value="Sharifft" />
    </appSettings>
</configuration>
```

4.1.4 Instalación del Módulo Web en el Servidor de Aplicaciones

Después de que se ha compilado el proyecto Web, se deben seguir los siguientes pasos:

1.- Se debe copiar la aplicación con todo y su estructura de directorios en la siguiente ruta “C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\webapps”

2.- Posteriormente vamos a tener el archivo de configuración C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\webapps\RAW\WEB-INF\Web.xml

Contenido del archivo Web.xml:

```
 <?xml version="1.0" encoding="ISO-8859-1" ?>
- <!--
 Licensed to the Apache Software Foundation (ASF) under one or more  contributor license agreements.  See the NOTICE file distributed with  this work for additional information regarding copyright ownership.  The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with  the License.  You may obtain a copy of the License at ttp://www.apache.org/licenses/LICENSE-2.0

 Unless required by applicable law or agreed to in writing, software  distributed under the License is distributed on an "AS IS" BASIS,  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  See the License for the specific language governing permissions and  limitations under the License.
 -->
-<web-app
        xmlns="http://java.sun.com/xml/ns/j2ee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">

        <display-name>RAW</display-name>
        <description>Reportes Automáticos Web.</description>

        <listener>
            <listener-class>libreria.ContextListener</listener-class>
        </listener>

        <welcome-file-list>
            <welcome-file>Login.jsp</welcome-file>
        </welcome-file-list>
 </web-app>
>
```

3.- Por ultimo se debe reiniciar el servidor Apache Tomcat para que la aplicación Web funcione.

4.1.5 Configuración de Aplicación

- **Alta de Laboratorios/Proveedores**

El alta de Laboratorios/Proveedores es el primer paso que se debe realizar para configurar el Sistema de Reportes Automáticos Web, y esto consiste básicamente en dar de alta uno por uno los más de 50 Laboratorios/Proveedores con los que se va a trabajar en el catálogo correspondiente.

La carga inicial se realizará sólo una vez, posteriormente y conforme se vayan incluyendo más laboratorios esto se hará de forma ocasional.

- **Alta de Plantillas**

El alta de Plantillas es la tarea más delicada e importante en la configuración de la aplicación, ya que se deben crear una serie de plantillas previamente definidas para cada Laboratorio/Proveedor y en ocasiones estas plantillas están personalizadas para cubrir las necesidades individuales de cada Cliente. Por lo que se deberá prestar mucha atención en como se realiza esta tarea, ya que como se menciono anteriormente la información que se le presenta a nuestros clientes deberá ser confiable y cuidando estrictamente la confidencialidad de la información.

- **Alta de Usuario y Clientes**

Este último pasó a realizar para la configuración del Sistema de Reportes Automáticos Web, ya que consiste básicamente en dar de alta todos los Usuarios finales que van a utilizar la aplicación vía Web.

Una vez que se ha terminado de hacer la carga de todos los Clientes para el Sistema, lo siguiente es asignar las plantillas a los Usuarios. Es esta acción una de las de más delicadas por lo que se debe prestar mucha atención, para evitar errores en la asignación de Plantillas a Clientes/Usuarios los cuales no debería tener acceso a cierta información.

4.1.6 Liberación del sistema a Producción con los Clientes

El último paso que se llevo a cabo para concluir con la implantación del sistema RAW fue la distribución a los usuarios de sus cuentas y contraseñas, para que así cada uno pudiera verificar el correcto acceso al sistema.

Este paso se realizo exclusivamente por correo electrónico, es decir que a cada uno de los usuarios finales se le envió un correo con un mensaje de bienvenida, “Usuario”, “Contraseña” y una breve explicación del funcionamiento de la aplicación Web. Es importante mencionar que este tarea fue realizada en su totalidad por el Gerente del Departamento de Sistemas, ya que es el quien tiene comunicación directa con los Usuarios.

Con este último punto se dio por concluida la implantación del sistema de Reportes Automáticos Web “RAW”.

Conclusiones

Después de haber elaborado este proyecto desde su concepción hasta su implementación en un ambiente productivo, puedo afirmar que lo que empezó como una simple idea para facilitar el trabajo de mis compañeros de empresa, termino como una gran idea que supero por mucho las expectativas que en un principio teníamos todas las persona que intervenimos directa o indirectamente en este sistema.

Esta aplicación ha sido de gran utilidad y los beneficios reales que ha representado para Farmatel son varios, por ejemplo: en las horas de trabajo que mensualmente se dedicaba a esta actividad se ha visto un ahorro significativo, ha sido un beneficio real para los Clientes/Proveedores ya que pueden obtener información real en el momento que ellos lo requieran, etc.

Dentro de Farmatel el Sistema de Reporte Automáticos Web tiene un gran futuro por delante, ya que si mejoramos sus características actuales y le agregamos nuevos módulos (ejemplo un modulo de reportes Estadísticos), podríamos presentar un producto final que represente mayores beneficios comerciales y económicos para la empresa.

Por otra parte puedo decir que este proyecto me dio la oportunidad de trabajar con tecnología Java, de la cual no estaba muy familiarizado, ya que la mayor parte de mi experiencia laboral como programador había sido siempre enfocada al uso de herramientas Microsoft como Visual Basic, C#, .Net, SQL Server, IIS, etc. Por lo que a partir del desarrollo de este sistema mi visión que tenia sobre el uso de tecnologías Open Source (Código Abierto) ha cambiado radicalmente, ya que hemos descubierto todo un mundo de posibilidades de crecimiento, aprendizaje, beneficios tecnológicos, beneficios comerciales y beneficios económicos.

Y como muestra de lo anterior puedo decir que el nuevo Sistema Integral Farmatel (Ventas, Almacén, Compras, Inventarios, Finazas, Sucursales, etc.) en el que se esta trabajando actualmente en el departamento de Sistemas, es desarrollado enteramente con software libre, como MySQL, Linux Fedora, NetBeans, Apache, Tomcat, etc. También cabe mencionar que tenemos pensado una serie de proyectos a futuro en la que vamos a migrar gradualmente toda nuestra infraestructura tecnológica a Software Libre:
- Windows 2000 Server (Small Business) x Linux RedHat.
- Microsoft Exchange x Nopix
- Norton Antivirus x Clam Antivirus (Implementado en algunos equipos del Depto. de Sistemas).
- SQL Server x MySQL.
- Conmutador Telefónico Meridian x Conmutador Telefónico Asterix
- ICQ Corp x Jabber (Ya implementado).
- IIS(Internet Information Server) x Apache Tomcat.

Como comentario final y como programador puedo decir que afortunadamente en un área tan amplia como lo es el desarrollo de Software, día a día cambia la tecnologías, lo que nos da la oportunidad de aprender cosas nuevas y junto con esto el reto de aplicarlas, por otro lado en las empresa, industria y negocios en general siempre hay algo que se pueda automatizar, mejorar o simplemente desarrollar desde cero.

Bibliografía

Java

→Java 2
Ed. Anaya
Autor Steven Hotzner

→Piensa en Java 2ª Edición
Ed. Prentice Hall
Autor Bruce Eckel

→Teach Yourself JE22
Ed. Sams Publishing
Autor Martin Bond y Dan Heywood

UML

→UML y Patrones 2ª Edición
Ed. Prentice Hall
Autor Craig Larman

→Aprendiendo UML
Ed. Prentice Hall
Autor Joseph Schmuller

UML 2 for Dummies
Ed. Wiley Publishing, Inc
Autor Michael Jesse Chonoles

Base de Datos

→SQL Server 2000
Ed. Mc Graw Hill
Autor Coffman

→Oracle Database 10g, Linux Administration
Ed. MC. Graw Hill
Autor Edward Whalen

→Oracle 10g, Administration y Analisis de Base de Datos
Ed. Alfaomega
Autor César Pérez

Referencias (Varias)

→ Como se hace un tesis
Autor Humberto Eco

→Como escribir su Tesis o Disertación
Ed. Michigan State University
Autor Joseph Levin

Referencias de Internet

programacion.com/java/tutorial/convenciones
www.java.sun.com
www.netbeans.com
www.wikipedia.org
www.javahispano.com
www.rup.edu.mx
www.tomcat.apache.org
www.microsoft.com.mx

Anexos

Reglas de Cood para diseño de Base de Datos

En un artículo de 1985 publicado en ComputerWorld, el Dr. Codd presentó doce reglas que una base de datos debe obedecer para que sea considerada relacional. Las doce reglas de Codd se han convertido en la definición teórica de una base de datos relacional. Estas se derivan del trabajo teórico de Codd sobre el modelo relacional y representan realmente más un objetivo ideal que una definición de una base de datos relacional. Dichas doce reglas son:

1. Regla de información. Toda la información de una base de datos relacional está representada explícitamente a nivel lógico y exactamente de un modo: Mediante valores en tablas.
2. **Regla de acceso garantizado**. Todos y cada uno de los datos de una base de datos relacional se garantiza que sean lógicamente accesibles recurriendo a una combinación de nombre de tabla, valor de clave primaria y nombre de columna.
3. Tratamiento sistemático de valores nulo. Los valores nulos (distinto de la cadena de caracteres vacía o de una cadena de caracteres en blanco y distinta del cero o de cualquier otro número) se soportan en los SGBD completamente relaciones para representar la falta de información y la información inaplicable de un modo sistemático e independiente del tipo de datos.
4. Catálogo en línea dinámico basado en el modelo relacional. La descripción de la base de datos se representa a nivel lógico del mismo modo que los datos ordinarios, de modo que los usuarios autorizados puedan aplicar a su interrogación el mismo lenguaje relacional que aplican a los datos regulares.
5. Regla de sublenguaje completo de datos. Un sistema relacional puede soportar varios lenguajes y varios modos de uso terminal (por ejemplo, el modo de rellenar con blancos). Sin embargo, debe haber al menos un lenguaje cuyas sentencias sean expresables mediante alguna sintaxis bien definida, como cadenas de caracteres, y que sea completa en cuanto al soporte de todos los puntos siguientes:

- Definición de datos.
- Definición de vista.
- Manipulación de datos (interactiva y por programa).
- Restricciones de integridad.
- Autorización.
- Fronteras de transacciones (comienzo, cumplimiento y vuelta atrás).

6. Regla de actualización de vista. Todas las vistas que sean teóricamente actualizables son también actualizables por el sistema.
7. Inserción, actualización y supresión de alto nivel. La capacidad de manejar una relación de base de datos o una relación derivada como un único operando se aplica, no solamente a la recuperación de datos, sino también a la inserción, actualización y supresión de los datos.
8. Independencia física de los datos. Los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cualquiera que sean los cambios efectuados, ya sea a las representaciones de almacenamiento o a los métodos de acceso.
9. Independencia lógica de los datos. Los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cuando se efectúen, sobre las tablas de base, cambios preservadores de la información de cualquier tipo que teóricamente permita alteraciones.
10. Independencia de integridad. Las restricciones de integridad específicas para una base de datos relacional particular deben ser definibles en el sublenguaje de datos relacional y almacenables en el catálogo, no en los programas de aplicación.
11. Independencia de distribución. Un SGBD relacional tiene independencia de distribución.
12. Regla de no subversión. Si un sistema relacional tiene un lenguaje de bajo nivel (un solo registro a la vez), ese bajo nivel no puede ser utilizado para subvertir o suprimir las reglas de integridad y las restricciones expresadas en el lenguaje relacional de nivel superior (múltiples registros a la vez).

La regla 1 es la definición informal de una base de datos relacional.
La regla 2 refuerza la importancia de las claves primarias para localizar datos en la base de datos. El nombre de la tabla localiza la tabla correcta, el nombre de la columna encuentra la columna correcta y el valor de la clave primaria encuentra la fila que contiene un dato individual de interés.
La regla 3 requiere soporte para falta de datos mediante el uso de valores NULL.
La regla 4 requiere que una base de datos relacional sea auto descriptiva. En otras palabras, la base de datos debe contener ciertas tablas de sistema cuyas columnas describan la estructura de la propia base de datos.
La regla 5 ordena la utilización de un lenguaje de base de datos relacional. El lenguaje debe ser capaz de soportar todas las funciones básicas de un SGBD (creación de una base de datos, recuperación y entrada de datos, implementación de la seguridad de la base de datos, etc.).
La regla 6 trata de las vistas, que son tablas virtuales utilizadas para dar, a diferentes usuarios de una base de datos, diferentes vistas de su estructura. Es una de las reglas más difíciles de implementar en la práctica.
La regla 7 refuerza la naturaleza orientada a conjuntos de una base de datos relacional. Requiere que las filas sean tratadas como conjuntos en operaciones de inserción, supresión y actualización. La regla esta diseñada para impedir implementaciones que sólo soportan la modificación o recorrido fila a fila de la base de datos.
La regla 8 y la regla 9 aíslan al usuario o al programa de aplicación de la implementación de bajo nivel de la base de datos. Especifican que las técnicas específicas de acceso a almacenamiento utilizadas por el SGBD, e incluso los cambios a la estructura de las tablas en la base de datos, no deberían afectar a la capacidad del usuario de trabajar con los datos.
La regla 10 dice que el lenguaje de base de datos debería soportar las restricciones de integridad que restringen los datos que pueden ser introducidos en la base de datos y las modificaciones que pueden ser efectuadas en ésta.
La regla 11 dice que el lenguaje de base de datos debe ser capaz de manipular datos distribuidos, es decir, localizados en otros sistemas informáticos.
Por último, la regla 12 impide “otros caminos” en la base de datos que pudieran subvertir su estructura relacional y su integridad.
Ningún SGBD relacional actualmente disponible satisface totalmente las doce reglas de Codd. De hecho, se elaboran pruebas para productos SGBD comerciales, que muestran lo bien o mal que éstos satisfacen cada una de las reglas.

Convenciones para programar en Java (Java Code Conventios)

Nombres de Ficheros

Esta página lista los sufijos y nombres de ficheros más utilizados:

- **Extensiones de Ficheros (Sufijos)**

El lenguaje Java usa los siguientes tipo de sufijos:

Tipo de Fichero	Extensión
Fuente Java	.java
bytecodes Java	.class

- **Nombres de Ficheros más Comunes**

Los nombres de ficheros más frecuentemente usados incluyen:

Nombre de Fichero	Uso
GNUmakefile	El nombre preferido para los makefiles. Nosotros usamos gnumake para construir nuestro software.
README	El nombre preferido para el fichero que formará los contenidos de un directorio particular.

Organización de Ficheros

Un fichero consta de secciones que deberían estar separadas por líneas en blanco y un comentario opcional identificando cada sección.

Los ficheros de más de 2000 líneas son aburridos y deberían evitarse.

Puedes ver un ejemplo de un programa Java apropiadamente formateado, en la página Ejemplo de Fichero Fuente Java.

- **Ficheros Fuente Java**

Todo fichero fuente Java contiene una sola clase pública o un interface. Cuando hay clases privadas e interfaces asociados con una clase pública, se pueden poner dentro del mismo fichero fuente que la clase pública. La clase pública debería ser la primera clase o interface en el fichero. Los ficheros fuente Java tienen el siguiente orden:

- Comentarios de inicio
- Sentencias Package e Import
- Declaraciones de clase e interface.

- **Comentarios de Inicio**

Todos los ficheros fuente deberían empezar con un comentario al estilo-C que liste el nombre de la clase, la información de versión, la fecha y las notas de copyright:

```
/*
* Classname
*
* Version information
*
* Date
*
```

```
* Copyright notice
*/
```

- **Sentencias Package e Import**

La primera línea no comentada de la mayoría de los ficheros fuente Java es una sentencia package. Después de esta pueden seguir sentencias import. Por ejemplo:

```
package java.awt;
import java.awt.peer.CanvasPeer;
```

- **Declaraciones de Clase e Interface**

La siguiente tabla describe las partes de una declaración de clase o interface, en el orden en que deberían aparecer. Puedes ver un ejemplo comentado en la página Ejemplo de Fichero Fuente Java:

	Parte de la clase/Interface	Notas
1	Comentario de documentación de Clase/interface (/**...*/)	Ver la sección Comentarios de Documentación
2	Sentencia class o interface	
3	Comentario de implemetnación de Clase/interface (/*...*/), si es necesario	Este comentario debería contener cualquier información sobre la clase o el interface que no fuera apropiada para ponerla en el comentario de documentación.
4	Variables de clase (static)	Primero las variables de clase pública, luego las protegidas, después las de nivel de paquete (sin modificador de acceso), y por último las privadas.
5	Variables de Ejemplar	Primero las variables de clase pública, luego las protegidas, después las de nivel de paquete (sin modificador de acceso), y por último las privadas.
6	Constructores	
7	Métodos	Estos métodos deberían agruparse por funcionalidad en vez de por ámbito o accesibilidad. Por ejemplo, un método de clase privado puede ir entre dos métodos de ejemplar públicos. El objetivo es hacer la lectura y el entendimiento del código más fácil.

Identación

Se deberían usar cuatro espacios como unidad de identación. La construcción exacta de la identación no está especificada (espacios o tabs). Los tabuladores deben ser exactamente cada 8 espacios (no cada 4).

- **Longitud de Línea**

Evitar líneas mayores de 80 caracteres, ya que no son bien manejadas por muchos terminales y herramientas.
Nota:

Los ejemplos para usarlos en documentación deberían tener una longitud de línea más corta, generalmente no más de 70 caracteres.

- **Ruptura de Líneas**

Cuando una expresión no entre en una sola línea, se debe romper de acuerdo a estos principio generales:

- Romper después de una coma.
- Romper antes de un operador
- Preferir las rupturas de alto nivel a las de bajo nivel.
- Alinear la nueva línea con el principio de la expresión al mismo nivel de la línea anterior.

- Si las reglas anteriores conducen a la confusión del código o código que se estrella contra el margen derecho, sólo identamos 8 espacios.

Aquí tenemos algunos ejemplos de rupturas de llamadas a métodos:

```
someMethod(longExpression1, longExpression2, longExpression3,
        longExpression4, longExpression5);

var = someMethod1(longExpression1,
                  someMethod2(longExpression2,
                          longExpression3));
```

Abajo hay dos ejemplos de rupturas de expresiones aritméticas. La primera es la preferida, ya que la ruptura ocurre fuera de la expresión entre paréntesis, que está a un nivel superior:

```
longName1 = longName2 * (longName3 + longName4 - longName5)
                  + 4 * longname6; // PREFER

longName1 = longName2 * (longName3 + longName4
                  - longName5) + 4 * longname6; // AVOID
```

Abajo hay dos ejemplos de identaciones de declaraciones de métodos. La primera es el caso convencional. La segunda desplazaría la segunda y la tercera líneas tan a la derecha si se usara la identación convencional, que por eso se identar sólo 8 espacios:

```
//CONVENTIONAL INDENTATION
someMethod(int anArg, Object anotherArg, String yetAnotherArg,
                    Object andStillAnother) {
...
}

//INDENT 8 SPACES TO AVOID VERY DEEP INDENTS
private static synchronized horkingLongMethodName(int anArg,
            Object anotherArg, String yetAnotherArg,
            Object andStillAnother) {
...
}
```

La ruptura de líneas de sentencias if generalmente usará la regla de 8-espacios, ya que la identación convencional (4 espacios) hace difícil la lectura de código. Por ejemplo:

```
//DON'T USE THIS INDENTATION
if ((condition1 && condition2)
        || (condition3 && condition4)
        ||!(condition5 && condition6)) { //BAD WRAPS
        doSomethingAboutIt(); //MAKE THIS LINE EASY TO MISS
}

//USE THIS INDENTATION INSTEAD
if ((condition1 && condition2)
                    || (condition3 && condition4)
                    ||!(condition5 && condition6)) {
        doSomethingAboutIt();
}

//OR USE THIS
if ((condition1 && condition2) || (condition3 && condition4)
                    ||!(condition5 && condition6)) {
        doSomethingAboutIt();
}
```

Aqí tenemos tres formas aceptas de formatear expresiones terciarias:

```
alpha = (aLongBooleanExpression) ? beta : gamma;

alpha = (aLongBooleanExpression) ? beta
                                 : gamma;

alpha = (aLongBooleanExpression)
        ? beta
        : gamma;
```

Comentarios en Código

Los programas Java pueden tener dos tipos de comentarios: los comentarios de implementación y los comentarios de documentación. Los comentarios de implementación son aquellos encontrados en C++, que están delimitados por /*...*/, y //. Los comentarios de documentación son únicos de Java y están delimitados por /**...*/. Los comentarios de documentación se pueden extraer a ficheros HTML usando la herramienta javadoc.

Los comentarios de implementación se han creado para comentar código o para comentarios sobre la implementación particular. Los comentarios de documentación se han creado para definir la especificación del código, desde un perspectiva libre de implementación para ser leído por desarrolladores que podrían no tener necesariamente a mano el código fuente.

Los comentarios deberían usarse para una introducción del código y proporcionar información adicional que no está disponible en el propio código. Los comentarios sólo deberían tener información que sea relevante para leer y entender el programa. Por ejemplo, información sobre como está construido el paquete correspondiente o en qué directorio reside no debería ser incluida como comentarios.

Las discusiones no triviales o decisiones de diseño no obvias son apropiadas, pero debemos evitar la duplicidad de información que esté presente en el código. Es demasiado fácil que los comentarios redundantes se queden anticuados. En general, debemos evitar cualquier comentario que se pueda quedar anticuado cuando el código evolucione.
Nota:

La frecuencia en los comentarios algunas veces refleja una pobre calidad de código. Cuando nos sintamos obligados a llenarlo de comentarios, debemos considerar la reescritura del código para hacerlo más claro. Los comentarios no deben encerrarse en grandes cajas dibujadas con asteriscos u otros caracteres. Los comentarios nunca deberían incluir caracteres especiales como saltos de página, etc.

- **Formatos de Implementación de Comentarios**

Los programas pueden tener cuatro estilos de implementación de comentarios:

- **Bloque de Comentarios**

Los bloques de comentarios se usan para proporcionar descripciones de ficheros, métodos, estructuras de datos y algoritmos. Los bloques de comentarios podrían usarse al principio de cada fichero y antes de cada método. También pueden usarse en otros lugares, como dentro de los métodos. Los bloques de comentarios dentro de una función o métodos deberían estar identados al mismo nivel que el código que describen. Un bloque de comentario debería ir precedido por una línea en blanco para configurar un apartado del resto del código:

```
/*
* Here is a block comment.
*/
```

Los bloques de comentarios pueden comenzar con /*-, lo que le dice al indent(1) que es el principio de un bloque de comentarios y que no debería ser reformateado, por ejemplo:

```
/*-
* Here is a block comment with some very special
* formatting that I want indent(1) to ignore.
*
* one
*         two
*                 three
*/
```

Nota:

Si no usamos indent(1), no tenemos que usar /*- en nuestro código o hacer cualquier otra concesión a la posibilidad de que alguien pudiera ejecutar indent(1) sobre nuestro código.

- **Comentarios de una línea**

Los comentarios cortos pueden aparecen como una sola línea identada al nivel del código que la sigue. Si un comentario no se puede escribir en una sola línea, debería seguir el formato de los bloques de comentario. Un comentario de una sola línea debería ir precedido de una línea en blanco. Aquí tenemos un ejemplo:

```
if (condition) {

/* Handle the condition. */
...
}
```

- **Comentarios finales**

Los comentarios muy cortos pueden aparecer en la misma línea que el código que describen, pero deberían separarse lo suficiente de las sentencias. Si aparece más de un comentario en el mismo trozo de código, deberían estar identados a la misma altura. Aquí tenemos un ejemplo:

```
if (a == 2) {
return TRUE;                    /* special case */
} else {
return isPrime(a);              /* works only for odd a */
}
```

- **Comentarios de final de línea**

El delimitador de comentario // puede comentar una línea completa o una línea parcial. No debería usarse en líneas consecutivas para comentar texto; sin embargo, si puede usarse en líneas consecutivas para comentar secciones de código. Abajo tenemos ejemplos de los tres estilos:

```
if (foo > 1) {
        // Do a double-flip.
...
}
else{
return false;       // Explain why here.
}

//if (bar > 1) {
//
//          // Do a triple-flip.
// ...
//}
```

```
//else{
// return false;
//}
```

- **Comentarios de Documentación**

Los comentarios de documentación describen clases Java, interfaces, constructores, métodos y campos. Cada comentario de documentación va dentro de los delimitadores de comentario /**...*/, con un comentario por clase, interface o miembro. Este comentario debe aparecer justo antes de la declaración:

```
/**
 * The Example class provides ...
 */
public class Example { ...
```

Observa que las clases e interfaces de alto nivel no están identados, mientras que los miembros si lo están. La primera línea del comentario de documento (/**) para las clases e interfaces no está identada; las subsecuentes líneas del comentario de documentación tendrán un espacio de identación (para alinear verticalmente los asteriscos). Los miembros, incluyendo constructores, tienen cuatro espacios para la primera línea de comentario de documentación y 5 espacios después.

Si necesitamos dar información sobre una clase, un interface, una variable o un método que no es apropiada para documentación, usamos una implementación de bloque de comentario, o una declaración de una línea inmediatamente después de la declaración. Por ejemplo, los detalles sobre la implementación de una clase deberían ir en un bloque de comentario seguido por la sentencia class, no en el comentario de documentación de la clase.

Los comentarios de documentación no deberían colocarse dentro de un bloque de definición de un método o constructor porque Java asocia los comentarios de documetnación con la primera declaración que hay después del comentario...

Declaraciones

- **Número de declaraciones por línea**

Se recomienda una declaración por línea ya que mejora los comentarios. En otras palabras:

```
int level;      // indentation level
int size;       // size of table
```

Se prefiere sobre:

```
int level, size;
```

No debemos poner diferentes tipos en la misma línea. Por ejemplo:

```
int foo, fooarray[]; //WRONG!
```

Nota:

Los ejemplos de arriba usan un espacio entre el tipo y el identificador. Otra alternativa aceptable es usar tabs, por ejemplo:

```
int             level;          // indentation level
int             size;           // size of table
Object          currentEntry;   // currently selected table entry
```

- **Inicialización**

Debemos intentar inicializar las variables locales donde son declaradas. La única razón para no inicializar una variable donde es declarada es si el valor inicial depende de algún cálculo que tiene que ocurrir antes.

- **Situación**

Ponemos las declaraciones sólo al principio de los bloques. (Un bloque es cualquier código encerrado entre corchetes “{” y “}”.) No debemos esperar a declarar variables hasta que son usadas por primer vez; puede confundir la programador despistado y estorbar la portabilidad del código dentro del ámbito.

```
void myMethod() {
        int int1 = 0;               // beginning of method block

        if (condition) {
                int int2 = 0;       // beginning of "if" block
                ...
        }
}
```

La única excepción a esta regla son los indexados para los bucles, que en Java pueden ser declarados en la sentencia for:

```
for (int i = 0; i < maxLoops; i++) { ... }
```

Debemos evitar las declaraciones locales que oculten las declaraciones de nivel superior. Por ejemplo, no debemos declarar el mismo nombre de variable en un bloque interno:

```
int count;
...
        myMethod() {
        if (condition) {
                int count; // AVOID!
        ...
        }
...
}
```

- **Declaraciones de Clases e Interfaces**

Cuando codificamos clases e interfaces Java, se deben seguir las siguientes reglas de formateo:

* No debe haber ningún espacio entre el nombre y el paréntesis “(“ que inicia la lista de parámetros.
* El corchete abierto “{” aparece en la misma línea que la declaración.
* El corchete cerrado “}” empieza una línea por sí mismo, identado a su correspondiente sentencia de apertura, excepto cuando es una sentencia null en la que el “}” debería aparecer inmediatamente después del “{“:

```
    class Sample extends Object {
        int ivar1;
        int ivar2;

        Sample(int i, int j) {
                ivar1 = i;
                ivar2 = j;
        }

        int emptyMethod() {}
        ...
    }
```

* Los métodos están separados por una línea en blanco.

Sentencias

- **Sentencias Simples**

Cada línea debe contener como máximo una sentencia. Por ejemplo:

```
argv++;          // Correct
argc++;          // Correct
argv++; argc--; // AVOID!
```

- **Sentencias Compuestas**

Las sentencias compuestas son sentencias que contienen listas de sentencias encerradas entre corchetes “{ sentencias }”. Puedes ver ejemplos en las siguientes secciones.

* Las sentencias encerradas deben identarse uno o más niveles que la sentencia compuesta.
* El corchete de apertura debe estar al final de la línea que empieza la sentencia compuesta; el corchete de cierre debería empezar una nueva línea y estar identado con el principio de la sentencia compuesta.
* Los corchetes se usan alrededor de todas las sentencias, incluso para sentencias simples, cuando éstas forman parte de una estructura de control como una sentencia if-else o for. Esto hace más fácil la adición de sentencias sin introducir errores debido al olvido de los corchetes.

- **Sentencias de Retorno**

Una sentencia de retorno no debería usar paréntesis a menos que el valor de retorno sea más obvió de esa forma. Por ejemplo:

```
return;
return myDisk.size();
return (size ? size : defaultSize);
```

- **Sentencias if, if-else, if else-if else**

Las sentencias del tipo if-else deberían tener la siguiente forma:

```
if ( condition) {
        statements;
}
if ( condition) {
        statements;
} else {
        statements;
}
if ( condition) {
        statements;
} else if ( condition) {
        statements;
} else {
        statements;
}
```

Nota:

Las sentencias if siempre usan corchetes. Debemos evitar el siguiente error:

```
if ( condition) //AVOID! THIS OMITS THE BRACES {}!
        statement;
```

- **Sentencias for**

Una sentencia for debería tener la siguiente forma:

```
for ( initialization; condition; update) {
        statements;
}
```

Una sentencia for vacía (una en la que todo el trabajo se hace en las cláusulas de inicialización, condición y actualización) debería tener la siguiente forma:

```
for ( initialization; condition; update);
```

Cuando usamos el operador coma en las cláusulas de inicialización o actualización de una sentencia for, debemos evitar la complejidad de usar más de tres variables. Si es necesario, debemos usar sentencias separadas antes del bucle for (para la cláusula de inicialización) o al final del bucle (para la claúsula de actualización).

- **Sentencias while**

Una sentencia while debería tener la siguiente forma:

```
while ( condition) {
        statements;
}
```

Una sentencia while vacía debería tener la siguiente forma:

```
while ( condition );
```

- Sentencias do-while

Una sentencia do-while debería tener la siguiente forma:

```
do {
        statements;
} while ( condition);
```

- Sentencias switch

Una sentencia switch debería tener la siguiente forma:

```
switch ( condition) {
        case ABC:
                statements;
                /* falls through */
        case DEF:
                statements;
                break;
        case XYZ:
                statements;
                break;
        default:
                statements;
                break;
}
```

Cada vez que un case cae (no incluye una sentencia break), debemos añadir un comentario donde normalmente iría la sentencia break. Esto se ve en el ejemplo de código anterior con el comentario /* falls through */.

Toda sentencia switch debería incluir un valor default. El break en el case por defecto es redundante, pero evita un erro de caída si añadimos después otro case.

- **Sentencias try-catch**

Una sentencia try-catch debería tener la siguiente forma:

```
try {
        statements;
} catch (ExceptionClass e) {
        statements;
}
```

Una sentencia try-catch también podría ir seguida de un cloque finally, que se ejecuta sin importar si se ha completado con éxito o no el bloque try.

```
try {
        statements;
} catch (ExceptionClass e) {
        statements;
} finally {
        statements;
}
```

Espacios en Blanco

- **Líneas en Blanco**

Las líneas en blanco mejoran la lectura separando secciones de código que están relacionadas lógicamente.

Siempre se deberían usar dos líneas en blando en las siguientes circunstancias:

* Entre secciones de un fichero fuente
* Entre definiciones de clases e interfaces

Siempre se debería usar una línea en blanco en las siguientes circunstancias:

* Entre métodos
* Entre las variables locales de un método y su primera sentencia
* Antes de un bloque de comentarios o un comentario simple (ver la lección Comentarios
* Entre secciones lógicas dentro de un método para mejorar su lectura

- **Espacios en Blanco**

Los espacios en blanco deberían usarse en las siguientes circunstancias:

* Una palabra clave seguida por un paréntesis deberían estar separados por un espacio en blanco:

```
while (true) {
...
}
```

Observa que no se debería usar un espacio en blanco entre un nombre de método y su paréntesis de apertura. Esto nos ayuda a distinguir las palabras clave de las llamadas a métodos.

* Después de las comas en una lista de argumentos debe aparecer un espacio en blanco.

* Todos los operadores binarios excepto "." deberían estar separados de sus operandos por espacios. Los espacios en blanco nuca deben separar los operadores unitarios como incremento (“++”), y decremento (“--”) de sus operadores. Por ejemplo:

```
a += c + d;
a = (a + b) / (c * d);
while (d++ = s++) {
      n++;
}
prints("size is " + foo + "\n");
```

* Las expresiones de una sentencia deberían estar separadas por espacios. Por ejemplo:

```
for (expr1; expr2; expr3)
```

* Los forzados deberían ir seguidos por un espacio en blanco, por ejemplo:

```
myMethod((byte) aNum, (Object) x);
myMethod((int) (cp + 5), ((int) (i + 3)) + 1);.
```

Convenciones de Nombrado

Las convenciones de nombrado hacen los programas más entendibles haciéndolos más fáciles de leer. También pueden proporcionar información sobre la función del identificador, por ejemplo, si es una constante, un paquete o una clase, lo que puede ayudarnos a entender el código.

Tipo de Identificador	Reglas de Nombrado	Ejemplos
Paquetes	Los prefijos de un nombre de paquete único siempre se escriben en letras ASCII minúsculas y deberían ser uno de los nombres de dominios de alto nivel, actualmente com, edu, gov, mil, net, org, o uno de los códigos de dos letras que identifican los países como especifica el estándar ISO 3166, 1981. Los siguientes componentes del nombre de paquete varían de acuerdo a las convenciones de nombrado internas de una organización. Dichas convenciones podrían especificar que ciertos nombres de componentes directorio serían división, departamento, proyecto, máquina, o nombres de login.	com.sun.eng com.apple.quicktime.v2 edu.cmu.cs.bovik.cheese
Clases	Los nombres de clases deben ser mezclas de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúsculas. Debemos intentar mantener los nombres de clases simples y descriptivos. Debemos usar palabras completas y evitar acrónimos y abreviaturas (a menos que la abreviatura se use muy ampliamente como URL o HTML).	class Raster; class ImageSprite;
Interfaces	Los nombres de interfaces se tratan igual que los nombres de clases	interface RasterDelegate; interface Storing;
Métodos	Los métodos deberían ser verbos, en mayúsculas	run();

	y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas.	runFast(); getBackground();
Variables	Las variables, tanto de ejemplar, de clase, como las constantes de clase se escriben en mayúsculas y minúsculas y con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas. Los nombres de variables no deben empezar con los caracteres guión bajo "_" o moneda "$", incluso aunque estén permitidos. Los nombres de variables deberían ser cortos y llenos de significado. La elección de una variable debería ser mnemónica-es decir, diseñada para indicar al observador casual su utilización. Se deben evitar los nombres de variable de un sólo carácter, excepto para variables temporales. Algunos nombres comunes de este tipo de variables son: i, j, k, m, y n para enteros.	int i; char c; float myWidth;

Constantes	Los nombres de variables constantes de clases y las constantes ANSI deberían escribirse todo en mayúsculas con las palabras separadas por subrayados (“_”). (Se deberían evitar las constantes ANSI para facilitar la depuración.)	static final int MIN_WIDTH = 4;
		static final int MAX_WIDTH = 999;
		static final int GET_THE_CPU = 1;

Prácticas de Programación

- **Proporcionar Acceso a Variables de Ejemplar y de Clase**

No debemos hacer públicas ninguna variable de ejemplar o de clase sin una buena razón. Frecuentemente las variables de ejemplar no necesitan configurarse explícitamente - porque normalmente esto sucede como un efecto lateral de llamadas a métodos.

Un ejemplo de variables de ejemplar públicas apropiadas es el caso donde la clase esencialmente es una estructura de datos, sin comportamiento. En otras palabras, si tuviéramos una estructura en lugar de una clase (si Java soportara estructuras), entonces seria apropiado hacer públicas las variables de ejemplar.

- **Referenciar Variables de Clase y Métodos**

Debemos evitar usar un objeto para acceder a variables de ejemplar (estáticas) o métodos. En su lugar debemos usar la clase. Por ejemplo:

```
classMethod(); //OK
AClass.classMethod(); //OK
anObject.classMethod(); //AVOID!
```

- **Constantes**

Las constantes numéricas (literales) no deberían codificarse directamente, excepto -1, 0, y 1, que pueden aparecer en un bucle for como valores de contador.

- **Asignaciones de Variables**

Debemos evitar asignar varias variables al mismo valor en una sola sentencia. Es difícil de leer, por ejemplo:

```
fooBar.fChar = barFoo.lchar = 'c'; // AVOID!
```

No debemos usar el operador de asignación en lugares donde pueda ser fácilmente confundible con el operador de igualdad, por ejemplo:

```
if (c++ = d++) { // AVOID! (Java disallows)
        ...
}
```

Debería escribirse como:

```
if ((c++ = d++) != 0) {
        ...
}
```

No debemos usar asignaciones embebidas en un intento de mejorar el rendimiento de ejecución. Ese es el trabajo del compilador. Por ejemplo:

```
d = (a = b + c) + r; // AVOID!
```

Debería escribirse como:

```
a = b + c;
d = a + r;
```

- **Prácticas Misceláneas**
- **Paréntesis**

Generalmente es una buena idea usar paréntesis para liberar expresiones que incluyen mézclas de operadores para evitar problemas de precedencia de operadores. Incluso si la precedencia del operador parece clara para nosotros podría no ser así para otros -- no deberíamos asumir que otros programadores conocen la precedencia tan bien como nosotros.

```
if (a == b && c == d) // AVOID!
if ((a == b) && (c == d)) // USE
```

- **Valores de Retorno**

Debemos intentar hacer que la estructura de nuestro programa corresponda con nuestra intención. Por ejemplo:

```
if ( booleanExpression) {
        return true;
} else {
        return false;
}
```

Debería escribirse como:

```
return booleanExpression;
```

De forma similar,

```
if (condition) {
        return x;
}
        return y;
```

Debería escribirse como:

```
return (condition ? x : y);
```

. Expresiones antes del ‘?’ en el Operador Condicional

Si una expresión que contiene un operador binario aparece antes del '?' en el operador terciario ?:, debería ponerse entre paréntesis, por ejemplo:

(x >= 0) ? x : -x;

- **Comentarios Especiales**

Debemos usar XXX en un comentario para marcar algo que es fraudulento pero funciona; y FIXME para marcar algo que es fraudulento y no funciona.