# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

**FACULTAD DE INGENIERÍA**

**IMPLEMENTACIÓN DE UNA INTERFACE WEB PARA LA GENERACIÓN DE REPORTES DE LAS VARIABLES OCEANOGRÁFICAS NIVEL DEL MAR, TEMPERATURA Y SALINIDAD**

## TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE

**INGENIERO EN COMPUTACIÓN**

P R E S E N T A N :

**LILIANA TORRES NIETO**

**DAN SCHMIDT VALLE**


DIRECTOR DE TESIS: DR. OSVALDO SÁNCHEZ ZAMORA
CO DIRECTOR DE TESIS: ING. ALEJANDRO VELÁZQUEZ MENA

CIUDAD UNIVERSITARIA 2005

**Dedicatorias**

A mi mamá, Elizabeth

Gracias por brindarme siempre tu apoyo incondicional. Por no dejarme sola, por tus consejos, tolerancia y paciencia. Pero, ante todo, por tu cariño.

A mi papá, Rodolfo

Por demostrarme lo importante que es tener fuerza de voluntad, por ser mi ejemplo a seguir y por todas tus enseñanzas.

A mi hijo, Rodolfo,

Por brindarme tu más bella sonrisa en los momentos más difíciles. Gracias por tu apoyo, pero sobre todo por el amor que me das. Gracias por ser la razón para salir adelante.

A mi hermana, Lénika,

Gracias por ser ante todo mi mejor amiga. Por darme ánimo y apoyo en los momentos difíciles.

A Dan,

Por tu amor, compañía y comprensión.

A mis amigos,

David Flores Tun, Ericka "Bombón" Garcés Rodríguez y Natalia "Burbuja" Coronado Palacio.

Por su amistad, complicidad y cariño.

A mi abuelita Carmen Barrera y a toda mi familia.

A la memoria de mis abuelos Raúl Nieto, Consuelo Calleja y Pablo Torres.

Liliana Torres Nieto

**Dedicatorias**

A mis padres, Martha y Enrique, por ocuparse de mi formación a pesar de mis necedades e inconsistencias.

A Liliana Torres Nieto por acompañarme ayer, hoy y siempre en estos tortuosos caminos.

A Alejandro Orozco y Villa, no hay palabras.

A Rodolfo Torres Nieto, que es la inspiración para seguir adelante.

A José Alberto "Mambo" Cisneros, José Andrés "Pepe" Vázquez Romero y a Jorge Arturo "Polaco" Hinojosa Andrjaszkiewicz, por su amistad, que los hace permanecer cuando los demás se han ido.

A Julio César "Chaynitos" Saynez, Alejandro "Mena" Velázquez, Edgar Eduardo "El Niño" García Cano, Eduardo "Stimpy" Ríos y César "Botarga Tardiñas" Pardiñas, que en distintos momentos y varias formas contribuyeron, con la mejor de las intenciones, a la finalización de este proyecto.

A los integrantes de la banda Modernatto, quienes me han convertido, a base de pequeños estímulos, en la persona que soy hoy.

Dan Schmidt Valle

**Agradecimientos**

A la Universidad Nacional Autónoma de México.

A la Facultad de Ingeniería.

Al Dr. Osvaldo Sánchez Zamora, por su paciencia y esfuerzo.

Al Ing. Alejandro Velázquez Mena, por asumir la responsabilidad de ser el Abogado del Diablo para este proyecto

A nuestros sinodales, M.I. Aurelio Adolfo Millán Nájera, M.I. Jorge Valeriano Assem, Ing. Carlos Alberto Román Zamitiz, por los valiosos comentarios realizados.

Liliana Torres Nieto

Dan Schmidt Valle

I think computer viruses should count as life. I think it says something about human nature that the only form of life we have created so far is purely destructive. We've created life in our own image.

Stephen Hawking

Part of the inhumanity of the computer is that, once it is competently programmed and working smoothly, it is completely honest.

Isaac Asimov

It would appear that we have reached the limits of what it is possible to achieve with computer technology, although one should be careful with such statements, as they tend to sound pretty silly in 5 years.

John von Neumann

# Temario

# Tablas

# Figuras

# Introducción

Este proyecto llegó a nuestra atención a través del Ing. Alejandro Velázquez Mena. El proyecto, parcialmente financiado por CONACYT, involucraba el desarrollo de un sitio Web para el Servicio Mareográfico del Instituto de Geofísica. De los detalles sobre ese sitio conocíamos poco, excepto que involucraba bases de datos y graficación. Para saber más al respecto, fuimos, en compañía del Ing. Velázquez Mena, al Instituto de Geofísica para entrevistarnos con el Dr. Osvaldo Sánchez Zamora.

En la entrevista pudimos conocer al Dr. Sánchez y establecer las condiciones de trabajo y los alcances de proyecto. Discutimos acerca de los procesos que se llevan en el Servicio Mareográfico. Nos fue presentado el resto del personal que labora en el departamento y los responsables de la red del Instituto. El Doctor nos informó que además el CONACYT ofrecía una beca durante nueve meses para una persona.

Los puntos básicos obtenidos de esta reunión pueden tomarse como lista de requerimientos y fueron los siguientes:

- El Servicio Mareográfico del Instituto de Geofísica necesita un sitio Web que represente al organismo.
- El Servicio Mareográfico del Instituto de Geofísica realiza procesos algorítmicos que necesitan automatizarse.
- El Servicio Mareográfico del Instituto de Geofísica recibe gran cantidad de datos de sus estaciones mareográficas que necesitan tener un acceso más eficiente.

Para afinar detalles se sostuvieron reuniones posteriores de las que se obtuvieron los lineamientos para desarrollar el sitio Web:

1. Deberá realizar gráficas de los datos almacenados en la base, claras rotuladas, con marcadores, con etiquetas de valor, respetando los rangos de tiempo sin datos y de tamaño adecuado. Los tipos de gráficas requeridos son:

    - El promedio de todos los días de cada año.
    - El promedio de todos los días de cada mes de un año.
    - El promedio de cada día de un mes y un año.
    - El valor horario de todos lo días de un mes.

2. Deberá obtener datos procesados de los almacenados en la base de datos, tratados para entregar los siguientes resultados:

    - El valor máximo de un mes.
    - El valor mínimo de un mes.
    - Los máximos de un mes definidos como el punto de cambio de pendiente según el criterio de la primera y segunda derivadas.
    - Los mínimos de un mes definidos como el punto de cambio de pendiente según el criterio de la primera y segunda derivadas.

3. Deberá mostrar datos de las estaciones como son:

    - Bancos de nivel
    - Foto
    - Coordenadas geográficas
    - Nombre del lugar
    - Planos de referencia
    - Definiciones de conceptos
    - Croquis de los bancos de nivel

4. Deberá poderse seleccionar la estación desde un mapa sensible o una lista. El mapa sensible deberá tener marcadas las estaciones con un círculo que al accionarse con el mouse llevará a los datos de la estación escogida.
5. Deberá poderse obtener todos los archivos de texto con los datos en el formato típico. Los archivos estarán disponibles en el formato estandarizado, similar al de la Universidad de Hawaii. Habrá un archivo comprimido con todos los archivos de una estación.
6. Deberá validarse el acceso al sitio mediante un nombre de usuario y una contraseña. Este nombre de usuario será para tener un registro de quien accede al sitio. Asimismo, interesará almacenar datos del usuario como son: nombre, apellidos, institución de procedencia, ubicación de la institución y correo electrónico.
7. Deberá existir una zona donde escribir comentarios para los administradores del sitio. Los comentarios los leerán solamente los administradores. En ellos podrá especificarse cualquier queja sobre el sitio, podrán hacerse las solicitudes de ser añadido a los usuarios del sitio o comentar sobre el funcionamiento del Sistema.
8. Deberá haber una zona de filtros donde se escogerá el tipo de gráfica o dato procesado que se desee. Las opciones de datos y gráficas estará dentro de las opciones para cada estación específica.
9. Deberá existir una zona de administración donde pueda manejarse los registros de:

    - Usuarios
    - Estaciones
    - Comentarios
    - Cuentas de Administrador
    - Datos de estaciones

    El manejo deberá entenderse como altas, bajas y modificaciones de todos rubros.

10. El sitio deberá desarrollarse con software gratuito que pueda obtenerse de la red. Dado que el Servicio Mareográfico del Instituto de Geofísica ya cuenta con equipo de hardware para el sitio, únicamente será necesario determinar el software a utilizarse. El sistema operativo será Solaris 8, puesto que está provisto junto con el servidor.

11. La instalación del servidor de aplicaciones correrá por cuenta de los tesistas. Una vez seleccionado el software necesario para realizar el Sistema correrá por cuenta de los tesistas obtenerlo, instalarlo y configurarlo para poder continuar en las etapas de desarrollo y pruebas.
12. La administración del servidor de aplicaciones será responsabilidad de los tesistas. En caso de ser necesario, el mantenimiento del sistema operativo y software instalado deberá ser realizado por los tesistas.

Con la anterior lista de requerimientos fue posible proceder al análisis, diseño, desarrollo, implantación y prueba del Sistema. Los capítulos siguientes se dedican a documentar estos procesos. Los capítulos dos al siete son los que detallan el proceso de creación del Sistema en todas sus etapas. Asimismo, existe un apartado para conclusiones, uno para los anexos del proyecto, uno para el glosario de términos y conceptos y uno para las referencias utilizadas en el documento.

El contenido de los capítulos es el siguiente:

Capítulo 1. Este capítulo está dedicado primeramente al comentario sobre el Servicio Mareográfico del Instituto de Geofísica, sus procesos e historia. Después, se dedica una sección a poner en claro los conceptos necesarios de Ingeniería de Software relacionados con el proyecto. También se realiza un análisis de las alternativas de servidores, lenguajes y clientes disponibles para el desarrollo e implantación. Finalmente, se lista las opciones de servidores y lenguaje a utilizar y las razones para hacerlo.

Capítulo 2. Este capítulo está dedicado enteramente al análisis y diseño de la aplicación. En su mayoría está ocupado por diagramas de UML: casos de uso, diagrama de clases, diagramas de secuencia, diagramas de actividades y diagramas de flujo. La parte restante del capítulo está ocupada por el diseño de base de datos: el modelo conceptual, el modelo físico, el diccionario de datos y la relación entre tablas y datos.

Capítulo 3. Este capítulo está dedicado a comprender los procesos automatizables que se realizan en el Servicio Mareográfico del Instituto de Geofísica y a preparar algoritmos para implementar algunos de ellos. Asimismo, la secuencia de pantallas, en base a los algoritmos obtenidos, se propone en este capítulo.

Capítulo 4. El quinto capítulo tiene como objetivo tratar acerca del desarrollo del Sistema. Para ello, se discuten los estándares de toda la programación del sitio Web, a saber: estándares generales y particulares al proyecto de la programación en Java; estándares ANSI para la comunicación con la base de datos y funciones propias del manejador PostgreSQL; descripción del funcionamiento del API para generar las gráficas y la evolución entre versiones, además de las decisiones de particularización de las mismas; estándares del lenguaje Javascript, la relación con los navegadores más comunes y la forma de utilizarse en el sitio; por último, estándares, funcionamiento en los navegadores más comunes y uso en el sitio de las hojas de estilo, así como la discusión del diseño gráfico para la vista.

Capítulo 5. El tema de este capítulo es tratar con la etapa de pruebas del Sistema, etapa necesaria con el fin de alcanzar la liberación. Las pruebas debieron realizarse sobre los componentes individuales al generarse cada uno. Posteriormente, al agruparse para formar un módulo, debieron volver a probarse los componentes para asegurar que el funcionamiento del conjunto. Esto se hizo con las distintas funciones del Sistema, completándose las pruebas unitarias y las integrales. Asimismo, las sentencias SQL se probaron por separado dentro del cliente PostgreSQL para asegurar que realizaban lo deseado. Finalmente, una de las preocupaciones que teníamos era el posible desenlace en caso de que varios usuarios concurrieran en un momento dado con las peticiones. La última prueba fue realizada mediante un software especializado para medir cuánto stress puede soportar un servidor Web.

Capítulo 6. Este capítulo es el de liberación. Es, en sí, una lista de todo aquello dentro del proyecto que ha sido aprobado para ser liberado. La lista incluye las vistas en HTML, el código de Java, las sentencias SQL, las hojas de estilo, los scripts de validación y las imágenes empleadas.

# 1. Conceptos Básicos

## 1.1 Servicio Mareográfico del Instituto de Geofísica de la UNAM

### 1.1.1 Antecedentes

México cuenta con 11,122 Km. de litorales y no es nada nuevo que la riqueza que se tiene en los mares no se ha explotado como debiera. Sin embargo, se hacen esfuerzos para que los recursos y oportunidades que ofrece nuestro país en esa materia sean aprovechados. Por otro lado, no se debe olvidar que por su ubicación geográfica y características tectónicas, es vulnerable al riesgo de huracanes, sismos, tsunamis, etc., con los consecuentes daños a la población y a la infraestructura.

En particular, en materia de investigación oceanográfica, se trabaja para eliminar el rezago de tantos años, y si se quiere avanzar en esta materia, es importante contar con un inventario de esos recursos. Ya desde hace algunos años se han iniciado estas tareas y un ejemplo de ello es el Servicio Mareográfico.

Desde 1952 el Servicio Mareográfico del Instituto de Geofísica de la UNAM opera la red nacional de estaciones mareográficas, que actualmente consta de quince estaciones, ubicadas en las costas mexicanas del Océano Pacífico, Golfo de México y Mar Caribe. Este servicio se encarga de la instalación, operación y mantenimiento de las estaciones mareográficas, en las que se mide en forma continua las variaciones del nivel del mar y diariamente, temperatura y densidad del agua de mar y temperatura ambiente.

Dentro de los objetivos del Servicio Mareográfico están:

1. Operar la red de estaciones mareográficas y con los datos que se obtienen del nivel del mar, calcular las Tablas de Predicción de Mareas, Planos de Referencia, incluyendo el Nivel Medio del Mar y establecer cotas de los Bancos de Nivel referidos al Nivel Medio del Mar.
2. Proporcionar información del Nivel del Mar para: investigación oceanográfica, geofísica y biológica, estudios geodésicos, delimitación de la zona federal marítimo terrestre, planeación y construcción de obras marítimas, obras hidráulicas, levantamientos topohidrográficos, cartas náuticas y portulanos.
3. En el contexto internacional, proporcionar información al "Permanent Service for Mean Sea Level" (PSMSL) y al "World Data Center-A for Oceanography" (WDCA), a través del "University of Hawaii Sea Level Center" (UHSLC).

Uno de los principales problemas del Servicio Mareográfico es que la base de datos no está actualizada y hay una gran preocupación por parte de la comunidad científica para que se actualice y se ponga a disponibilidad tanto de la propia comunidad como de la sociedad en general.

Otro problema es la antigüedad de los instrumentos, por lo que se hace necesario hacerles algunas adecuaciones con el objeto de mejorar la calidad de los datos, y simplificar la operación. De los 15 mareógrafos con que cuenta el Servicio Mareográfico, 10 son analógicos y 5 son analógico-digitales. La información es enviada mensualmente por correo al Instituto de Geofísica, en donde se digitaliza y procesa. La digitalización requiere de mucho tiempo y el personal con que cuenta el Servicio Mareográfico actualmente, es insuficiente para atender todas las estaciones. Por este motivo hay un gran rezago en lo que a digitalización se refiere.

El estado actual de la información existente corre el grave peligro de volverse inaccesible al paso del tiempo, por lo que es indispensable que a la brevedad se catalogue y se organice con una metodología moderna y de fácil acceso, para asegurar su existencia y disponibilidad de manera segura y confiable por muchos años.

## Utilidad de la base de datos

Los datos que se generan en el Servicio Mareográfico, particularmente los datos de variaciones del nivel del mar tienen una gama muy amplia de aplicaciones. Aquí se exponen algunas de ellas. Los datos de nivel del mar son esenciales para la investigación en Oceanografía Física, con

repercusiones en estudios sobre variación del clima, corrientes y su relación con pesquerías, etc.

Uno de los fenómenos de mayor impacto tanto a nivel global como a nivel nacional es el de El Niño. Una de las primeras manifestaciones cuando este fenómeno se presenta es el incremento en el nivel del mar en el Océano Pacífico Este, que incluye las costas mexicanas. Por este motivo resulta indispensable que para entender mejor este fenómeno se cuente con información de la más alta calidad sobre como está variando el nivel del mar en esta parte del mundo. De la misma manera resulta de gran importancia que dicha información esté disponible lo más rápidamente posible, idealmente, en tiempo real, con el objeto de poder detectar la presencia del fenómeno con cierta anticipación.

Las variaciones del nivel del mar a lo largo de la costa pacífico de México han mostrado gran correlación con señales de propagación hacia el norte. Estas señales son en parte debidas a la perturbación por huracanes que existen en el Sur del país, por lo cual los registros del nivel del mar 'guardan' memoria de eventos que han ocurrido a lo largo de varias décadas, siendo un registro de la climatología mundial. En escalas locales los registros también son importantes pues presentan cambios de temperatura y cambios en los sitios de referencia (ejemplos: terremotos y tsunamis).

Para la modelación numérica de la marea, un mecanismo de forzamiento es a través del nivel del mar y otro ingrediente importante en la calibración de los modelos es a través del mismo nivel del mar a lo largo del dominio modelado. También el análisis de propagación de señales es una cuestión muy interesante y este banco de datos es imprescindible para tal fin. Aunque se cuenta con datos de altimetría, siempre es bueno tener los datos 'reales' para calibración y/o comparación.

Una de las zonas sísmicas que mayores daños causan en nuestro país se ubica en las costas del Océano Pacífico. Estos sismos son producidos por la subducción de las placas tectónicas Rivera y Cocos por debajo de la placa de Norte América. Entender el fenómeno sísmico en esta región adquiere una relevancia sin discusión. Un aspecto muy importante en estos estudios consiste en observar la deformación de la corteza terrestre antes, durante y después de un sismo.

Las observaciones de las variaciones del nivel del mar permiten de manera indirecta obtener información acerca de la deformación de la corteza. Específicamente, los sismos del 9 de octubre de 1995 en Jalisco-Colima (M=8.0), 19 de septiembre de 1985 en Michoacán (M = 8.1), 18 de mayo de 1962 en Acapulco (M = 6.2) produjeron deformaciones de la corteza que fueron detectadas en los datos de los mareógrafos ubicados en esa región

y a partir de los cuales se ha cuantificado la magnitud de esas deformaciones. Esta información es utilizada para modelar las características de la falla que rompió y originó dichos sismos.

Los únicos datos medidos mediante instrumentos sobre variación de nivel del mar al arribo de tsunamis a México, sean de origen local o lejano, son los registros mareográficos. Por ende, estos datos son imprescindibles e insustituibles para:

- Validación de los modelos numéricos con que se simula la extensión de inundaciones costeras y las alturas de olas esperables, con fines de prevención.
- En tiempo real, operar sistemas locales y regionales de alerta.
- Mediante inversión, determinar los mecanismos tectónicos o de deslizamiento en su generación, reconstruir sus parámetros, y estimar su eventual recurrencia.
- Estudiar resonancias en bahías y golfos, etc.

Dentro de otras aplicaciones en donde es importante conocer las variaciones del nivel del mar, se encuentra lo relativo a la delimitación de la Zona Económica Exclusiva (ZEE), así como la delimitación de la Zona Federal Marítimo Terrestre (ZFMT).

La ZFMT se delimita a partir de la pleamar máxima registrada, de tal manera que se hace indispensable conocer este nivel de referencia con la mayor precisión en toda la costa de nuestro país para regular los límites de la propiedad federal.

Este nivel de referencia se obtiene a partir de las mediciones de las variaciones del nivel del mar que se registran en las estaciones del Servicio Mareográfico. De manera similar, a partir de estas mediciones se obtiene el nivel de referencia para la delimitación de la ZEE, la cual separa las aguas territoriales de las aguas internacionales o de las ZEEs de los países vecinos.

Es importante conocer las variaciones del nivel del mar para la construcción de puertos y toda la obra civil que se ubica cerca de la costa incluyendo la infraestructura turística. En acuicultura, conocer el régimen de mareas permite planear la entrada y salida del agua de mar a los estanques de cultivo.

Las aplicaciones en las que se utilizan los datos del nivel del mar son innumerables. Esta información se aprovecha en: navegación marítima, pesca, buceo, muestreo de especies marinas, huracanes, etc.

La Predicción de Mareas conocida como las Tablas y Calendarios de Predicción de Mareas es una de las aplicaciones que es quizá la más conocida por los usuarios y, por lo tanto, muy necesaria. La información del nivel del mar en una localidad particular, junto con la información de la posición del sol y de la luna para una fecha y hora determinados permiten realizar el cálculo del comportamiento de la marea para esa localidad fecha y hora. El cálculo del comportamiento de la marea puede realizarse para fechas futuras y de allí el nombre de predicción de la marea. Las Tablas y Calendarios de Predicción de Mareas es el producto que se genera en el Servicio Mareográfico más buscado por la mayoría de la gente que realiza actividades relacionadas con el mar.

**Futuro del Servicio Mareográfico.**

Se propone la modernización de los equipos de adquisición de datos del nivel del mar, por lo que este tipo de trabajo (la digitalización) ya no será necesario; pues se van a incorporar equipos que registren desde el principio los datos en forma digital. En lo que respecta a la base de datos, será necesario actualizarla continuamente, lo cual será una operación rutinaria; ya que el sistema que se implemente deberá tener esta capacidad.

## 1.1.2 Objetivos y metas

**Objetivos.**

- Desarrollar una base de datos del nivel del mar para los principales puertos de México, que contenga la información que se ha registrado desde 1952.
- Desarrollar el sistema que permita tener acceso a la base de datos a través de la Internet, a toda la comunidad científica y usuarios en general.

**Metas.**

- Compilar todos los datos existentes del nivel del mar.
- Crear la base de datos del nivel del mar.
- Poner a disposición de los usuarios, la información del nivel del mar.

### 1.1.3 Infraestructura del SM

La red del Servicio Mareográfico consta de las siguientes 15 estaciones mareográficas:

1. Ensenada, Baja California
2. La Paz, Baja California Sur
3. Guaymas, Sonora
4. Mazatlán, Sinaloa
5. Puerto Vallarta, Jalisco
6. Lázaro Cárdenas, Michoacán
7. Acapulco, Guerrero
8. Salina Cruz, Oaxaca
9. Puerto Madero, Chiapas
10. Tuxpan, Veracruz
11. Veracruz, Veracruz
12. Coatzacoalcos, Veracruz
13. Ciudad del Carmen, Campeche
14. Puerto Progreso, Yucatán
15. Puerto Morelos, Quintana Roo

En cada estación mareográfica se cuenta con una caseta en donde se encuentra instalado el tubo del flotador, un mareógrafo y los instrumentos para medir temperatura y densidad.

En el Servicio Mareográfico del Instituto de Geofísica se cuenta con 2 estaciones de trabajo SUN SunBlade 1000, 2 computadoras personales, 2 mareógrafos para reposición y 3 mareógrafos para pruebas de laboratorio.

A través de esta red y de otras estaciones que ya no existen, se ha generado una base de datos que contiene información del nivel del mar para cerca de 35 localidades a lo largo de las costas mexicanas; con serie de datos con duraciones variables, desde unos cuantos meses hasta series con más de cuarenta años

## 1.2 Ingeniería de Software

La Ingeniería de Software es una disciplina o área de la informática o ciencias de la computación, que ofrece métodos y técnicas para desarrollar y mantener software de calidad.

**Cualidades:**

| | |
|---|---|
| Corrección: | Habilidad del software de realizar sus tareas como fueron definidas en la especificación de requerimientos. |
| Fortaleza: | Confiabilidad, habilidad del software de funcionar a pesar de la presencia de condiciones anormales. |
| Extensión: | Facilidad con la que se adapta a los cambios de especificación. Dos principios son esenciales para mejorar este aspecto: simplicidad en el diseño y descentralización. |
| Reusabilidad: | Es la habilidad de los productos de software de ser reusados, en general o en partes, en nuevas aplicaciones. |
| Compatibilidad: | Es la facilidad de los productos de software para ser combinados con otros. |
| Eficiencia: | Es el buen uso de los recursos de hardware. |
| Portabilidad: | Posibilidad de usarlo en diferentes ambientes de software o hardware. |
| Verificabilidad: | Facilidad del software para preparar test de datos, procesos para detectar fallas. |
| Integridad: | La habilidad de los sistemas de proteger sus componentes de accesos y codificaciones no autorizadas. |
| Facilidad de uso: | Facilidad de aprender a usar un sistema, preparar los datos de entrada, interpretar sus resultados. |

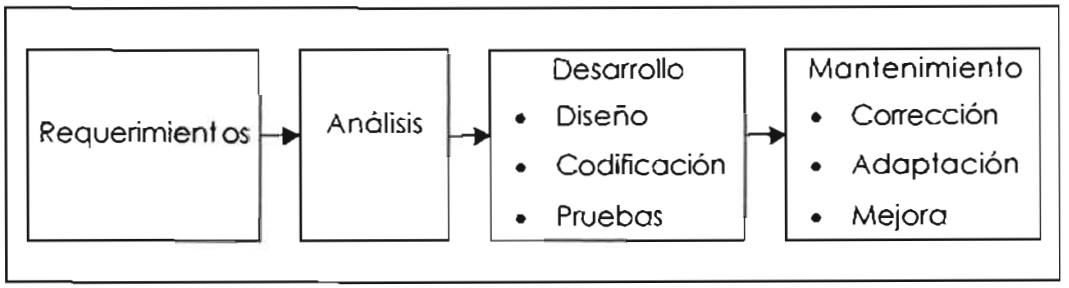



**Figura 1.1 Generación del sistema**

### 1.2.1 Análisis

El Análisis tiene que ser independiente del ambiente de implementación. Los cambios en el ambiente de implementación no afectan el resultado del análisis.

El modelo de análisis debe ser orientado a la aplicación; se realiza en un ambiente ideal. Debe describir los elementos de la aplicación en conceptos relativos a la aplicación. La estructura de la implementación refleja la estructura del problema. El modelo de análisis no tiene que tener una alto nivel de formalización, dado que debe ser adaptado al ambiente de implementación. El propósito es identificar las cualidades requeridas de la aplicación (QUÉ HACE) en términos de funcionalidad, desempeño, facilidad de uso, portabilidad, etc. Se puede completar con una lista de atributos de calidad.

El resultado de esta fase es un documento de especificación de requerimientos. El propósito de este documento es doble. Primero, debe ser analizado y confirmado por el diente en orden de verificar si este captura todas las expectativas del cliente. Segundo, es usado por el ingeniero de software para desarrollar una solución que cumpla los requerimientos. El documento debe ser entendible, preciso, completo, consistente y no ambiguo. También debe ser fácil de modificar, puesto que debe acomodarse a la evolución natural de grandes sistemas.

Se puede complementar el documento de especificación de requerimientos con una versión preliminar del manual de usuario. El manual describe de forma precisa cómo interactúa eventualmente con el sistema. Tal descripción puede ser muy usada en la estimación de resultados del análisis de requerimientos.

Otra entrega de la fase de análisis de requerimientos es la definición de un "plan de pruebas del sistema". El sistema debe ser probado contra sus requerimientos, por lo tanto, debe estar hecho de acuerdo con el cliente y documentado a lo largo de la especificación de requerimientos.

La aplicación debe ser entendida y descrita en diferentes niveles de abstracción, deber ser fraccionada en partes analizables. Debe ser posible ver, entender, y describir la aplicación desde diferentes puntos de vista.

La aplicación usual de modularidad es vertical, en la cual cada módulo oculta las decisiones del diseño del nivel bajo. La modularidad horizontal estructura el sistema como una colección de vistas al mismo nivel de abstracción. Un camino típico para usar la modularidad horizontal es separar los requerimientos funcionales en tres vistas:

- Un modelo de datos que es operado por la aplicación.
- Un modelo de funciones ejecutables.
- Un modelo de control de como gobernar tales funciones.

La manera en que actualmente los requerimientos son especificados son sujetos a procedimientos de estandarización. Los estándares deben prescribir la forma y estructura del documento de especificación de requerimientos; el uso de métodos específicos de análisis y notación y las formas de revisión y aprobación del documento.

Una lista tentativa del contenido del documento de especificación de requerimientos es la siguiente:

1. Requerimientos funcionales: Describe qué debe hacer el producto usando notaciones formales, seminformales, o informales, o una combinación adecuada de éstas.
2. Requerimientos no funcionales: Estos se pueden clasificar en las siguientes categorías:
    - Confiabilidad (disponibilidad, integridad, seguridad).
    - Exactitud de los resultados.
    - Desempeño.
    - Salidas de la interfase computadora a humano.
    - Restricciones operativas.
    - Restricciones físicas.
    - Resultados de la portabilidad.
    - Otros.

3. Requerimientos del proceso de desarrollo y mantenimiento: Esto incluye procedimientos de control de calidad; en particular, los procedimientos de prueba del sistema, prioridades de funciones requeridas, así como los cambios a los procedimientos de mantenimiento y otros requerimientos.

## 1.2.2 Desarrollo

La fase de desarrollo se centra en el cómo. Durante esta fase, el que desarrolla el software intenta descubrir cómo han de diseñarse las estructuras de datos y la arquitectura del software, cómo han de implementarse los detalles procedurales, cómo ha de traducirse el diseño a un lenguaje de programación y cómo ha de realizarse la prueba. Los métodos aplicados durante la fase de desarrollo varían, pero de alguna forma se aplican tres pasos concretos:

- **Diseño de software:**

  El diseño traduce los requisitos de software a un conjunto de representaciones (algunas gráficas y otras tabulares o basadas en lenguajes) que describen las estructuras de bases de datos, la arquitectura, el procedimiento algorítmico y las características de la interfaz.

- **Codificación:**

  Las representaciones del diseño deberán ser traducidas a un lenguaje artificial (un lenguaje de programación convencional o un lenguaje no procedural), dando como resultado unas instrucciones ejecutables en la computadora.

- **Prueba del software:**

  Una vez que el software ha sido implementado en una forma ejecutable por la máquina, debe ser probado para descubrir los defectos que puedan existir, en la función, en la lógica y en la implementación.

### 1.2.3 Mantenimiento

La fase de mantenimiento se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas por la evolución del entorno del software y a las modificaciones debidas a los cambios de requisitos del usuario dirigidos a reforzar o ampliar el sistema. La fase de mantenimiento vuelve a aplicar las fases de análisis y de desarrollo, pero en el contexto del software ya existente. Durante la fase de mantenimiento se encuentran tres tipos de cambios:

- **Corrección:**

  Incluso llevando a cabo las mejores actividades de garantía de calidad, es muy probable que el cliente descubra defectos en el software. El mantenimiento correctivo cambia el software para eliminar los defectos.

- **Adaptación:**

  Con el paso del tiempo es probable que cambie el entorno original (sistemas operativos, equipos periféricos, etc.) para los que se desarrollo el software. El mantenimiento adaptivo consiste en modificar el software para acomodarlo a los cambios de su entorno externo.

- **Mejora:**

  Conforme utilice el software, el usuario puede descubrir funciones adicionales que podría interesar que estuvieran incorporadas en el software. El mantenimiento perfectivo amplía el software mas allá de sus requisitos funcionales originales.

# 1.3 Arquitecturas

## 1.3.1 Arquitectura de 1 capa Stand-Alone

Sistema que realiza su función requiriendo poca o nada de asistencia de sistemas de interfaz. Esta constituido por una computadora que funciona en forma aislada.

En la tabla 1.1 se pueden ver las ventajas y las desventajas de la arquitectura de 1 capa o *Stand-Alone*.

| Ventajas: | Desventajas: |
|---|---|
| • Este tipo de sistemas posee un alto grado de control sobre las aplicaciones.<br>• Se puede adaptar fácilmente a las necesidades específicas de cada sector.<br>• Sus costos son módicos.<br>• Posee un alto potencial para integrarse a redes en forma gradual.<br>• Fácil manejo. | • Limitado alcance.<br>• Puede ser operado simultáneamente por un solo usuario. |

**Tabla 1. 1 Ventajas y desventajas de la arquitectura de 1 capa**

## 1.3.2 Arquitectura de 2 capas Cliente/Servidor

Conjunto de servidores *stand-alone*, los cuales proporcionan servicios específicos como impresión, manejo de datos, etc.

La arquitectura Cliente/Servidor describe la relación entre dos programas de computadoras en los cuales, uno, el Cliente, hace una petición de servicio al otro programa, el Servidor. En una red, el modelo Cliente/Servidor ofrece una forma conveniente de interconectar programas distribuidos en diferentes localidades. Las transacciones entre computadoras utilizando el modelo Cliente/Servidor son muy comunes.

El modelo Cliente/Servidor se ha convertido en una de las ideas centrales en las redes computacionales. La mayoría de las aplicaciones empresariales hoy en día usan el modelo Cliente/Servidor.

En el modelo básico Cliente/Servidor, un servidor, también llamado *daemon*, se activa y espera las peticiones de los clientes. Usualmente, varios programas clientes comparten los servicios de un servidor común. Ambos programas, cliente y servidor, comúnmente son parte de un programa o una aplicación más grande.

En la tabla 1.2, se listan las ventajas y desventajas de la arquitectura de 2 capas.

| Ventajas: | Desventajas: |
|---|---|
| • La distribución de datos es directa.<br>• Permite el uso efectivo de sistemas de red.<br>• Puede requerir hardware barato.<br>• Es fácil añadir nuevos servidores o actualizar los existentes. | • El modelo no comparte datos con los diferentes subsistemas empleados en la organización. El intercambio de datos puede ser ineficiente.<br>• Administración redundante en cada servidor.<br>• No existen registros centrales de nombres y servicios, esto hace difícil encontrar los servidores y servicios disponibles. |

**Tabla 1. 2 Ventajas y desventajas de la arquitectura de 2 capas**

### 1.3.2.1 Servidor Web

Un servidor Web es un programa que se ejecuta continuamente y que espera conexiones de clientes Web solicitando información, normalmente archivos. Los servidores y los navegadores se comunican mediante el Protocolo de Transferencia de HiperTexto (*HTTP*), un lenguaje creado para transferir documentos de hipertexto a través de la Web. Los servidores Web normalmente se llaman servidores *HTTPD* (La "D" de *HTTPD* es la inicial de *daemon*).

Un servidor Web es un programa de aplicación que satisface las solicitudes *HTTP* realizadas por los navegadores. Para ello, la computadora que la soporta debe estar conectada a la Internet y, por lo tanto, ha de tener asignada una dirección *IP*.

Un servidor Web debe soportar los protocolos estándar en Internet, por ejemplo *HTTP* que facilita el intercambio de datos entre el servidor Web y el navegador. Además, para publicar una página se suele utilizar un protocolo más antiguo, el *FTP*.

Adicionalmente, deben ofrecer soporte a *scripts* y aplicaciones en los lenguajes más comunes utilizados en aplicaciones de Internet, como Java, PHP y otros. Finalmente, deben contener algunos elementos de seguridad.

Los navegadores, por su parte, pueden recibir archivos mediante *HTTP* y *FTP* y poseer capacidad para interpretar *scripts* en lenguajes con Java y *JavaScript*.

#### 1.3.2.2 Servidor FTP

*FTP* (Protocolo de Transferencia de Archivos) es, como su nombre lo indica, un protocolo especializado para enviar y recibir información a través de Internet. Esto, en pocas y simples palabras significa una optimización cuando se trata de trabajar con datos en general (mayor velocidad y seguridad en el envío y recepción). Un servidor *FTP* es una máquina (computadora) que es la encargada de ofrecer información a través de este mencionado protocolo.

### 1.3.3 Arquitectura de 3 capas

Una arquitectura de 3 capas se define también como el modelo de servicios. Las bases de datos, las herramientas de desarrollo y los corporativos se están moviendo hacia esta arquitectura dadas las limitaciones de la arquitectura de dos capas.

La capa adicional provee de una capa explícita para las reglas de los negocios que se sitúa entre lo que se ha llamado *front end* y *back end*. Esta capa intermedia encapsula el modelo de negocios (o "reglas de negocios") asociado con el sistema y le separa de la presentación y el código de bases de datos.

La arquitectura de tres capas depende del proceso interno de comunicación. Para que dos servicios se comuniquen, deberán hacerlo en el mismo lenguaje o protocolo.

En la tabla 1.3 se pueden ver las ventajas y desventajas de la arquitectura de 3 capas.

| Ventajas: | Desventajas: |
|---|---|
| • Mejor manipulación del sistema.<br>• Servidores de negocios que pueden compartirse.<br>• Los objetos y procesos son seguros.<br>• Los usuarios pueden crear sus propias aplicaciones cliente.<br>• Se pueden lograr sistemas distribuidos, que son físicamente localizados en varias máquinas dentro de varias locaciones.<br>• El grado en que pueda distribuirse una aplicación es directamente proporcional al grado en que lógicamente se ha particionado. | • Hay más servidores que administrar.<br>• El modelo debe ser aislado de la lógica de negocios y de la presentación.<br>• Mayor complejidad de documentación.<br>• Requiere conocimientos de redes y telecomunicaciones.<br>• El monitoreo del sistema debe hacerse en tres niveles.<br>• Exige una modularización absoluta. |

**Tabla 1. 3 Ventajas y desventajas de la arquitectura de 3 capas**

La comunicación deberá ser local (en las mismas máquinas) o remota (en máquinas distantes). Deberá ser transparente al proceso que envuelve sin importar si es local o remota.

La arquitectura de 3 capas es una guía, no un requerimiento. Este es un modelo lógico, no físico, que describe cómo se diseña la aplicación no cómo se despliega.


**Figura 1.2 Arquitectura de tres capas**

### 2.3.3.1 Front End

Los sistemas de *front end* son la colección de servidores que proveen el núcleo de los servicios Web, tales como *HTTP, FTP*, a las aplicaciones cliente de *workflow*. Habitualmente se agrupan estos sistemas *front end* en conjuntos de sistemas idénticos llamados clones.

Todos corren el mismo software y tienen acceso mediante replicación u otros medios al mismo contenido, archivos *HTML*, archivos *ASP, scripts*, etc. Mediante la utilización de sistemas de balanceo de carga y de detección de fallas, se puede aumentar considerablemente la disponibilidad y escalabilidad.

Un *front end* acepta un programa fuente, entrega un programa intermedio y genera una tabla de símbolos que contiene información de variables y funciones. Las partes típicas de un *front end* se pueden ver en la Figura 1.3:

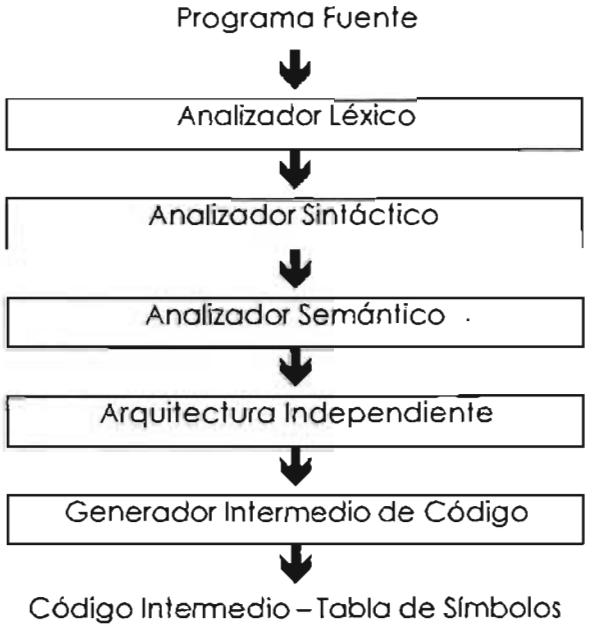



**Figura 1.3 Partes típicas de un Front End**

El analizador léxico lee caracteres en el programa fuente y los agrupa en *tokens* como palabras clave, identificadores, etc. El analizador sintáctico agrupa los *tokens* del programa fuente en frases gramaticales. El análisis semántico revisa errores semánticos en el programa fuente y reúne

información para la generación de código. Después del análisis, el *front end* genera un código intermedio y una tabla de símbolos.

Los análisis léxico, sintáctico y semántico son dependientes del lenguaje de programación, pero son independientes de la arquitectura del procesador. Si el lenguaje intermedio es independiente del procesador, el código intermedio generado es también común en cada arquitectura de procesador.

### 1.3.3.1 Middleware

Software de comunicaciones que reside físicamente en el cliente remoto y en un servidor de comunicaciones, localizado entre el cliente y el servidor de aplicaciones. Es el software que actúa como un traductor universal entre distintas tecnologías de radiofrecuencia y protocolos.

*Middleware* es un término general para cualquier programación que sirve para "unir" o mediar entre dos programas separados y existentes. Una aplicación común del *middleware* es permitir que programas escritos en una base de datos particular tengan acceso a otra base de datos sin la necesidad de codificación.

*Middleware* es comúnmente conocido como el mensajero de un sistema de información ya que lleva datos e información de una forma transparente entre diferentes *back ends* y aplicaciones finales de usuario.

### 1.3.3.2 Back End

Los sistemas de *Back End* se encargan del almacenamiento de los datos de las aplicaciones y de ejecutar los procesos necesarios para responder a las solicitudes de los clientes. Una aplicación o programa *back end* sirve indirectamente como soporte a los servicios del *front end*, usualmente al estar más cerca de la fuente requerida o al tener la capacidad de comunicarse con la fuente requerida. La aplicación *back end* puede interactuar directamente con la aplicación *front end* o bien, más comúnmente, es un programa llamado por el programa intermedio (*middleware*) que sirve como mediador entre las actividades del *front end* y las del *back end*.

### 1.3.4 Navegadores

Un navegador es un programa que actúa como una interfase entre el usuario y los contenidos de Internet, específicamente la Web. Los navegadores también se conocen como clientes Web, o clientes universales, debido a que en el modelo cliente/servidor, el navegador funciona como un programa cliente.

El navegador actúa interpretando al usuario. El navegador:

- Contacta al servidor Web y envía un pedido de información.
- Recibe la información y la muestra en la pantalla del usuario.

Un navegador puede estar basado en texto o en gráficos y puede hacer el uso de la Internet más sencillo y más intuitivo. Un navegador gráfico permite al usuario ver imágenes en su computadora, "apuntar" y accionar el ratón para seleccionar vínculos de hipertexto, usar menús en cascada y emplear botones en una barra de herramientas para navegar por la Internet.

La *WWW* incorpora hipertexto, fotografías, sonido, video, etc. lo que puede experimentarse completamente con un navegador gráfico. Los navegadores a menudo incluyen aplicaciones de ayuda los cuales son realmente programas que son necesarios para mostrar imágenes, escuchar sonidos o correr secuencias de animación. Esas aplicaciones de ayuda son invocadas automáticamente por el navegador cuando el usuario selecciona un vínculo o un recurso que los requiera.

Un navegador basado en textos permite al usuario ver solamente textos. Los elementos gráficos no se muestran. Los vínculos de hipertexto se acceden usando el teclado.

Existen muchos tipos diferentes de navegadores. Todos realizan básicamente las mismas funciones (transfiriendo hipertexto) pero muchos tienen características especiales únicas.

Ejemplos de algunos navegadores comunes son:

- Internet Explorer.
- Netscape.
- Mozilla.
- Opera.

- **Internet Explorer**

  Es una suite abierta e integrada de software para Internet o Intranet. Incluye las soluciones de clientes y la colaboración básica para usuarios finales, administradores de tecnología de información.

  Con Internet Explorer es muy fácil sacar el máximo partido de WWW. Además, puede ahorrar tiempo al realizar tareas rutinarias de Web, como rellenar automáticamente direcciones y formularios Web, y detectar automáticamente el estado de la red y de la conexión.

  Internet Explorer permite conectarse a Internet para tener acceso a gran cantidad de información. Con los canales, los mejores proveedores del mundo envían automáticamente el contenido de Web más actual.

  Características:

  - Excelente exploración.
  - Comunicación y colaboración completa.
  - Mejoras para personalizar la información.
  - Verdadera integración con Web.

  La barra de menús de Internet Explorer contiene todas las órdenes en los menús.

  - Archivos: Este menú permite realizar operaciones con archivos e inclusive abrir varias páginas Web al mismo tiempo.
  - Edición
  - Ver.
  - Ir a.
  - Favoritos: Permite administrar las paginas que se seleccionen como favoritas. Trabaja de la misma forma que un separador de libros.
  - Ayuda: Esta opción resuelve dudas de cómo usar o para qué sirve cada una de las opciones.

  La barra de direcciones de Internet Explorer muestra la dirección con la cual se encuentra conectado. La característica autocompletar sugiere una coincidencia de las direcciones basándose en los sitios Web visitados anteriormente.

- **Netscape**

  Netscape fue desarrollado en 1993 por Marc Andreessen y un equipo de estudiantes en el National Center for Supercomputing Applications (NCSA). El navegador Netscape es un navegador Web gráfico que permite al usuario utilizar hipertexto, gráficos, sonido, video, etc. disponibles en la *WWW*.

  Netscape permite al usuario seleccionar los vínculos de hipertexto usando el ratón y menús desplegables para navegar y acceder a recursos en Internet.

- **Mozilla**

  En 1998 Microsoft Netscape decidió hacer público el código de Netscape Communicator con tres fines: gratuidad del navegador, permitir a desarrolladores de todo el mundo participar en la creación de un nuevo navegador competitivo y respetuoso con estándares y utilizable en los principales sistemas operativos (Linux, Solaris, MS Windows, etc.). A este proyecto nuevo se le dio el nombre de Mozilla.

  Después de un costoso arranque, la primera versión de Mozilla vió la luz el 5 de junio de 2002. Hasta ahora más o menos cada tres meses Mozilla publica una versión nueva. Cada versión contiene funciones y características nuevas y lógicamente arreglos de errores en versiones anteriores.

  Características:

  - Desarrollo para muchos Sistemas Operativos (MS Windows, Solaris, FreeBSD, etc.). Su utilización en MS Windows es igual que en Linux.
  - Disponible en muchos idiomas.
  - Selección del lenguaje del interfaz independiente del lenguaje del Sistema Operativo.
  - Se puede cambiar el aspecto de las ventanas, barras e iconos.
  - Programa basado en Código de Fuente Abierto (*Open Source*) lo cual permite aportaciones e innovación de participantes de todo el mundo.
  - Totalmente gratuito
  - Se pueden utilizar varias ventanas y más de una página dentro de cada ventana utilizando pestañas.

- Ventanas emergentes o no solicitadas (*popup windows*) pueden ser bloqueadas totalmente o de forma selectiva.
- Se puede guardar un conjunto de pestañas en un solo marcador.
- Un simple Gestor de Descargas lista todos los archivos que se han bajado de Internet en el pasado.
- En una sola ventana se pueden manejar múltiples cuentas de correo de forma separada e independiente.
- Los virus informáticos incluidos en mensajes no pueden infectar o destrozar el contenido del disco duro.

- **Opera**

  Opera fue el primer navegador comercial que incluyó una herramienta que permitía el acceso a la Red desde un teléfono móvil, la conocida tecnología *WAP*. Ahora también existen versiones para agendas *PDA*, televisión y automóviles

  Opera nace en Noruega como un proyecto de la compañía de telecomunicaciones de ese país en 1992. Dos años después sus desarrolladores, conscientes del potencial del nuevo software, se independizan de la compañía y crean Opera Software con el objetivo de mejorar y comercializar el navegador.

  Desde entonces, Opera ha crecido hasta su versión 7.01 (Windows) con más de un millón y medio de copias bajadas de su servidor en 2002.

  Además, este navegador es válido para cualquier sistema operativo: Windows, Solaris, QNX, OS/2, Macintosh, Linux o BeOS.

### 1.3.5 Servidores Web y Aplicaciones

Los servidores Web son aquellos que permiten a los clientes compartir datos, documentos y multimedia en formato Web. Aunque parte de la tecnología Cliente/Servidor, el servidor Web aporta ventajas adicionales en aspectos muy importantes como son:

- El Web se crea normalmente como un sistema abierto al que cualquiera puede contribuir y acceder desde cualquier punto de la red; no requieren usuario ni contraseña como los sistemas tradicionales Cliente/Servidor.

- La información solicitada puede ser de cualquier tipo (datos, documentos, multimedia, etc.), gracias a la utilización de los estándares de Internet. La información es de sólo lectura pues, a diferencia de los sistemas normales Cliente/Servidor, el usuario no puede hacer cambios en el dispositivo original de los datos.
- Dado que el servidor Web es de fácil acceso, hace posible publicar información de forma instantánea, mediante un simple almacenamiento de la misma en el servidor.
- Un servidor Web en la Intranet puede servir la misma copia de un archivo de la misma forma que un único servidor Web en Internet puede servir al mundo entero. Así, sólo hay una única copia del archivo a actualizar, y cuando se actualiza, la nueva versión es distribuida instantáneamente.
- La amplitud de la red suele ser mayor que otros sistemas Cliente/Servidor, pues la mayor parte de Intranet se construyen sobre Redes de Área Extensa (*WAN*), más que sobre redes *LAN*.

**Servidores HTTP**

- **Apache**

  El servidor *HTTP* Apache es un servidor de código abierto para plataformas Unix (BSD, GNU/Linux y otros), Windows y otras plataformas desarrollado por Apache Software Foundation. Nació en 1995 como una evolución del servidor NCSA *HTTPD* 1.3 y es el más popular.

  El servidor *HTTP* Apache es una de las aplicaciones estrella del mundo del software libre, ya que es el servidor Web de mayor implantación.

- **Tomcat**

  El servidor Tomcat es una aplicación Web basada en Java creada para ejecutar *servlets* y páginas *JSP*, siendo la implementación oficial de referencia de las especificaciones *Servlet* 2.3 y *Java Server Pages* 1.2.

  El servidor Tomcat es uno de los proyectos más interesantes liderado por la organización Apache. En el proyecto Jakarta se ha desarrollado la aplicación Tomcat, la cual no es más que un servidor de aplicaciones con las características de servir como motor de *servlets* y *JSP*. Es la solución gratuita para el soporte de aplicaciones Web basadas en Java.

- **JBOSS**

JBoss es una implementación *open source* de un Contenedor *EJB*. Es mediante este tipo de productos que es posible llevar acabo un desarrollo con *EJB's (Enterprise Java Bean's)*. Este tipo de producto generalmente no es distribuido como producto individual y por esta razón se le pudiera considerar a JBoss como un producto diferente mas no único.

La gran gama de productos en este ramo de Java (*J2EE*) han sido comercializados como *Java Application Servers*. Casi todos los *Application Servers* en el mercado hoy en día son conocidos como *Fully J2EE Compliant*, este término implica que cumplan todas las especificaciones *J2EE* definidas por Sun Microsystem y es aquí donde es notable la diferencia con JBoss.

Cuando se utiliza un Servidor de Aplicación es posible ejecutar tanto *JSP/Servlets* así como *EJB's*. Sin embargo, el ambiente se encuentra altamente integrado para que la comunicación entre *JSP/Servlets* y *EJB's* sea transparente, al menos para el programador final. El producto JBoss es únicamente un Contenedor *EJB* y es por esto que generalmente se utiliza en conjunción con un *Web-Container*, el *Web-Container* puede ser cualquiera disponible en el mercado.

**Servidores de Aplicación**

- **WebSphere**

WebSphere es el software de la infraestructura de la Internet conocido como *middleware*. Permite desarrollar, desplegar e integrar los usos de próxima generación del comercio electrónico, tales como para el negocio-a-negocio, y apoya usos de comercio en la red.

El servidor de aplicación WebSphere trabaja junto con el servidor Web para ofrecer funciones dinámicas en un sitio Web. WebSphere proporciona interfaces de ejecución y de programación de aplicaciones (*API*) para componentes Java del lado del servidor, basadas en las especificaciones de Sun Microsystems.

También, incluye *API* para manipular documentos *XML*. Estas *API* pueden añadirse a los componentes Java del lado del servidor para generar, validar, analizar y presentar documentos *XML*.

WebSphere incluye utilidades para administrar el producto. Entre estas utilidades se cuenta una consola administrativa basada en Java, una consola basada en Web, una interfaz de línea de mandatos y una

función que permite importar y exportar los datos administrativos como documentos *XML*.

El servidor de aplicación WebSphere no proporciona herramientas para el desarrollo de componentes Java del lado del servidor.

- **WebLogic**

  Como una plataforma líder en la industria del comercio electrónico, el servidor WebLogic permite desarrollar y desplegar rápidamente, aplicaciones fiables, seguras, escalables y manejables. WebLogic utiliza tecnologías de la plataforma *Java 2 Enterprise Edition (J2EE)*. *J2EE* es la plataforma estándar para desarrollar aplicaciones multicapa basadas en el lenguaje de programación Java.

  Las aplicaciones *J2EE* están basadas en componentes estandarizados y modulares. El servidor WebLogic proporciona un conjunto completo de servicios para esos componentes y maneja automáticamente muchos detalles del comportamiento de la aplicación, sin requerir programación. WebLogic también proporciona características esenciales para desarrollar y desplegar aplicaciones críticas de comercio electrónico a través de entornos de computación distribuidos y heterogéneos.

- **iPlanet**

  El servidor de aplicación iPlanet proporciona un soporte robusto para los estándares *J2EE*. Para casi virtualmente cualquier nivel de desarrollo y despliegue, se tienen las interfaces características y las funcionalidades de iPlanet que cumplen con la especificación *J2EE*. Además, iPlanet proporciona capacidades claves adicionales a través de servicios como el balance de carga, el control de fallos, y una alta disponibilidad aplicada a estos mismos estándares *J2EE*.

  iPlanet proporciona mejoras significantes frente a versiones anteriores del producto. Esto incluye una total compatibilidad con los *APIs* y especificaciones de la plataforma *Java 2, Enterprise Edition*, mejoras de rendimiento, escalabilidad y eficiencia, y mejoras en la manejabilidad y administración. La plataforma iPlanet proporciona un nuevo marco para la integración que simplifica y acelera el desarrollo de aplicaciones que integran múltiples sistemas de información.

  El servidor de aplicación iPlanet proporciona una arquitectura de quinta generación con un rendimiento probado, una escalabilidad y eficiencia

líderes en sitios de comercio electrónicos. Para clientes que requieren una alta integridad de transacciones y continuos tiempos de marcha, iPlanet elimina los puntos de fallo a través del control de fallos en cada nivel de entorno de despliegue *J2EE*, incluyendo *JavaServer Pages, Java Servlets,* y *Java Beans Enterprise*. El servidor de aplicación iPlanet también asegura que la información del usuario y los datos de la aplicación no se perderán durante el fallo distribuyendo la información de estado y de sesión de una transacción en varios servidores.

La arquitectura de iPlanet permite que aplicaciones divididas se ejecuten incluso si fallan uno o más servidores. En una configuración de balance de carga de servidor, la lógica de la aplicación puede estar replicada en varias servidores. Si un servidor falla, el módulo de balance de carga dirige automáticamente la solicitud a otros servidores disponibles, así se evita el fallo de la aplicación.

## 1.3.6 Bases de Datos y Manejadores de Bases de Datos

**Base de datos jerárquica**

Una base de datos de tipo jerárquico utiliza jerarquías o árboles para la representación lógica de los datos. Los archivos son organizados en jerarquías, y normalmente cada uno de ellos se corresponde con una de las entidades de la base de datos. Los árboles jerárquicos se representan de forma invertida, con la raíz hacia arriba y las hojas hacia abajo.

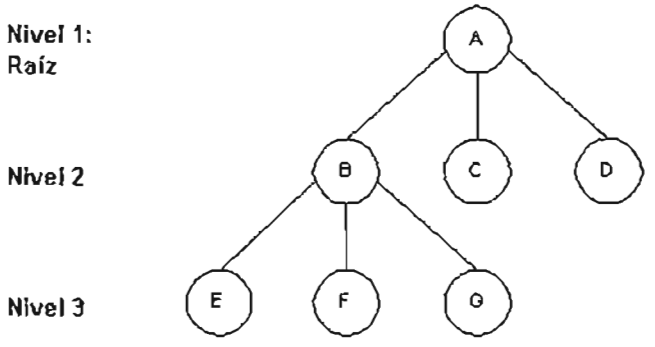



**Figura 1.4 Estructura de un árbol jerárquico.**

Una base de datos de tipo jerárquico recorre los distintos nodos de un árbol en un preorden que requiere tres pasos:

- Visitar la raíz.
- Visitar el hijo más a la izquierda, si lo hubiera, que no haya sido visitado.
- Si todos los descendientes del segmento considerado se han visitado, volver a su padre y repetir el procedimiento.

Cada nodo del árbol representa un tipo de registro conceptual, es decir, una entidad. A su vez, cada registro o segmento está constituido por un número de campos que los describen, las propiedades o atributos de las entidades. Las relaciones entre entidades están representadas por las ramas. En la figura 1.5, cada departamento es una entidad que mantiene una relación de uno a muchos con los profesores, que a su vez mantienen una relación de uno a muchos con los cursos que imparten.

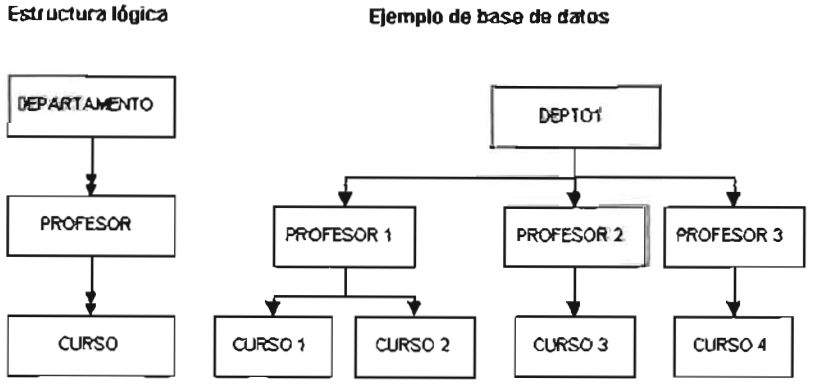



**Figura 1.5 Base de datos jerárquica. Estructura lógica y ejemplo.**

A modo de resumen, se enumeran las siguientes características de las bases de datos jerárquicas:

- Los segmentos de un archivo jerárquico están dispuestos en forma de árbol.
- Los segmentos están enlazados mediante relaciones uno a muchos.
- Cada nodo consta de uno o más campos.
- Cada ocurrencia de un registro padre puede tener distinto número de ocurrencias de registros hijos.

- Cuando se elimina un registro padre se deben eliminar todos los registros hijos (integridad de los datos).
- Todo registro hijo debe tener un único registro padre excepto la raíz.

Las reglas de integridad en el modelo jerárquico prácticamente se reducen a la ya mencionada de eliminación en cadena de arriba a abajo. Las relaciones muchos a muchos no pueden ser implementados de forma directa. Este modelo no es más que una extensión del modelo de archivos.

Como ejemplos de base de datos basados en este enfoque podemos citar el IMS de IBM Corporation y el SYSTEM 2000 de Intel Corporation.

**Base de datos de red**

Una base de datos de red, como su nombre lo indica, está formada por una colección de registros, los cuales están conectados entre sí por medio de enlaces. El registro es similar a una entidad como las empleadas en el modelo entidad-relación.

Un registro es una colección de campos (atributos), cada uno de los cuales contiene almacenado un sólo valor. El enlace es la asociación entre dos registros exclusivamente, así que puede verse como una relación estrictamente binaria.

Una estructura de datos de red, llamada algunas veces estructura plex, abarca más que la estructura de árbol porque un nodo hijo en la estructura de red puede tener más de un padre. En otras palabras, la restricción de que, en un árbol jerárquico, cada hijo puede tener un solo padre, se hace menos severa. Así, la estructura de árbol se puede considerar como un caso especial de la estructura de red.

**Base de datos orientada a objetos**

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los *objetos* completos (estado y comportamiento).

**Base de datos relacional**

Una base de datos relacional es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores. Todas las operaciones de la base de datos operan sobre estas tablas.

Estas bases de datos son percibidas por los usuarios como una colección de relaciones normalizadas de diversos grados que varían con el tiempo. En términos tradicionales una relación se asemeja a un archivo, una tupla a un registro, y un atributo a un campo. Pero estas correspondencias son aproximadas, en el mejor de los casos. Una relación no debe considerarse como solo un archivo, sino más bien como un archivo disciplinado, siendo el resultado de esta disciplina una simplificación considerable de las estructuras de datos con las cuales debe interactuar el usuario, lo cual, a su vez, simplifica los operadores requeridos para manejar esas estructuras.

Características principales de los archivos relacionales:

- Cada archivo contiene sólo un tipo de registros.
- Los campos no tienen un orden específico, de izquierda a derecha.
- Los registros no tienen un orden específico, de arriba hacia abajo.
- Cada campo tiene un sólo valor.
- Los registros poseen un campo identificador único (o combinación de campos) llamado clave primaria.

Así, todos los datos en una base de datos relacional se representan de una y sólo una manera, a saber, por su valor explícito (esta se denomina en ocasiones principio básico del modelo relacional). En particular, las conexiones lógicas dentro de una relación y entre las relaciones se representan mediante esos valores; no existen ligas o apuntadores visibles para el usuario, ni ordenamientos visibles para el usuario, ni grupos repetitivos visibles para el usuario.

**Tipos de Base de Datos Relacional**

- **Oracle**

  El servidor Oracle es un *DBMS* objeto-relacional que provee una aproximación integral al manejo de información. Un servidor Oracle consiste en una base de datos y una instancia.

  Cada vez que parte de una base de datos se inicia, el servidor crea un área global de sistema (*SGA*) donde los procesos de Oracle son almacenados. Las *SGAs* son áreas de memoria usadas por la información compartida por la base de datos y por los usuarios. Esta combinación de procesos y buffers de memoria es una instancia. Una instancia consta de dos tipos de procesos, los del usuario que ejecutan el código de una aplicación o herramienta y los de Oracle que realizan los procesos del usuario y la manutención del servidor.

  La estructura lógica de una base de datos Oracle consta de:

  - Espacios de tabla:

    Es la unidad de almacenamiento lógica de una base de datos. Agrupa las estructuras relacionadas juntas.

  - Esquemas y objetos de esquema:

    Son colecciones de objetos de bases de datos. Un objeto de esquema es la estructura lógica que se refiere directamente al dato de la base de datos. Entre estos objetos se encuentran:

    - Tablas: Donde se almacenan los esquemas de los datos mediante lenguaje *SQL*
    - Vistas: Consultas estáticas o dinámicas de los datos en las tablas.
    - Secuencias: Variables que almacenan valores en orden creciente o decreciente.
    - Procedimientos almacenados: Funciones sobre los datos.
    - Otros datos como sinónimos, índices, clusters y enlaces a otras bases de datos.

- **PostgresSQL**

  PostgresSQL es un proyecto *open source*, o de código abierto. Código abierto significa que es libre de modificar PostgresSQL para adaptarlo a sus necesidades particulares. PostgresSQL es un administrador de bases

de datos relacionales que soporta las instrucciones de SQL, este manejador es uno de los más populares y funcionales dentro de lo que es el software libre.

El modelo relacional sustituyó modelos previos en parte por su simplicidad. Sin embargo, esta simplicidad también hace muy difícil la implementación de ciertas aplicaciones. Postgres ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos: clases, herencia, tipos y funciones. Otras características aportan potencia y flexibilidad adicional: Restricciones, Disparadores, Reglas e Integridad Transaccional.

Postgres tiene algunas características que son propias del mundo de las bases de datos orientadas a objetos. De hecho, algunas bases de datos comerciales han incorporado recientemente características en las que Postgres fue pionera.

Características:

- Se puede instalar un número ilimitado de veces sin temor de sobrepasar la cantidad de licencias.
- Velocidad y rendimiento excepcionales.
- Flexibilidad para extenderse según se requiera.
- Diseño escaloble.
- Mínimos requerimientos de administración.
- Sigue los estándares ANSI.

- **Sybase**

Sybase es el cuarto productor de bases de datos tras la compra de Informix por IBM. Sybase es una base de datos relacional, esto es, almacena datos en tablas relacionadas entre sí a través de índices o claves. Estas tablas están compuestas de filas de datos que equivalen a registros, que se intersecan con columnas que equivalen a campos.

Adicionalmente Sybase es líder en las siguientes tecnologías:

1. Computación Móvil e Inalámbrica a través de su base de datos SQL Anywhere.
2. *Business Intelligence*, con soluciones verticales desarrolladas para *Data warehousing* en banca, seguros y telecomunicaciones.
3. Base de datos y alta disponibilidad
4. Soluciones en Internet y portales.

- **Teradata**

  El Sistema de Gestión de Base de Datos Relacional (*RDBMS*) Teradata de NCR, es una base de datos relacional poralela de soporte de decisiones muy poderosa para *data warehousing*. Teradata ofrece soporte a las empresas para la escalabilidad desde gigabytes a terabytes, petabytes y más.

  Teradata también proporciona conectividad open client a virtualmente todos los sistemas operativos, incluyendo Microsoft DOS, Windows, Windows 95, Windows NT, Windows 2000, Windows XP, IBM OS/2 y Apple Macintosh. Además, soporta conectividad a cliente desde las aplicaciones basadas en el *host* que corre sobre sistemas UNIX y en *mainframes*. De hecho, ofrece una conectividad de *mainframes* muy resistente con canal paralelo bidireccional de alta velocidad.

  El producto inicial de la Corporación Teradata NCR se diseñó específicamente para descargar el procesamiento reiterativo a gran escala de las *mainframes* IBM, permitiéndoles enfocar hacia sus tareas operacionales principalmente.

  En 1991, la Corporación NCR compró la Corporación Teradata. NCR adquirió su avanzada y única tecnología de procesamiento en paralelo comercial. La arquitectura de hardware del DBC 1012 original evolucionó en la actual familia WorldMark Server y el software de base de datos se ha convertido en NCR Teradata *RDBMS*.

  Hoy, el NCR Teradata *RDBMS* corre sobre plataformas Intel basadas en UNIX y Windows NT. Es aún el único producto de base de datos capaz de soportar *data warehouses* por sobre los 500 Gigabytes de datos de usuario real.

## 1.3.7 Lenguajes de Programación

- **.NET**

  El corazón de la plataforma .NET es el *CLR (Common Language Runtime)*, que es una aplicación similar a una máquina virtual que se encarga de gestionar la ejecución de las aplicaciones para ella escritas. A estas aplicaciones, les ofrece numerosos servicios que facilitan su desarrollo y mantenimiento y favorecen su fiabilidad y seguridad.

.Net es básicamente:

- La infraestructura .NET comprendida por: el Framework .NET, Microsoft Visual Studio .NET, .NET Enterprises Servers y Microsoft Windows .NET.
- Servicios de Internet, o la posibilidad de acceso por programa a ciertos servicios como obtener la temperatura actual en cualquier lugar del mundo, el valor de cambio actualizado de cualquier moneda, almacenamiento de archivos, comprobación de identidad, etc.
- Programas .NET que acceden a estos servicios.

Lo más visible de .NET es la infraestructura, a esto es a lo que uno hace referencia cuando dice .NET, y se trata de todas las tecnologías que conforman el nuevo entorno que permite a los desarrolladores crear y ejecutar aplicaciones.

- **Java**

Java es un lenguaje de programación orientado a objetos con el que se puede 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 con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas de punta.

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Esto quiere decir que un programa hecho en Java podrá funcionar en cualquier computadora del mercado. 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.

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 la computadora como móviles, agendas y en general para cualquier cosa que se le ocurra a la industria.

Java es 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 Jova y muchas veces con grandes ventajas. Con Java se pueden 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 se desee hacer con acceso a través Web se puede hacer utilizando Java.

Características:

- Java corre en cualquier plataforma que tenga intérprete para su *byte-code*.
- Java es dinámico ya que puede cargar un nuevo código de cualquier parte de la red y compilarlo al vuelo. Lo cual implica que los cambios y nuevas versiones de *applets*, al ser compilados en forma independiente, son automáticamente integrados a la nueva versión de la aplicación.
- Java es seguro, ya que fue diseñado para proveer el máximo nivel de seguridad en la Red, tanto para virus como para intrusos que deseen modificar el código.
- El intérprete de Java es bastante rápido, además que ya se ha liberado lo que se llama el Compilador al momento, que permite compilar los *applets* "al vuelo" conforme se van recibiendo, guardándolos en caché de tal forma que si se requieren volver a utilizar ya no sea necesario recompilarlos.
- Java es parecido al C++, pero sin sus complicaciones.
- Java es robusto, compacto y multihilos.

- **PHP**

PHP es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación.

PHP se escribe dentro del código *HTML*, lo que lo hace realmente fácil de utilizar pero con algunas ventajas como su gratuidad, independencia de plataforma, rapidez y seguridad. Es independiente de plataforma, puesto que existe un módulo de PHP para casi cualquier

servidor Web. Esto hace que cualquier sistema pueda ser compatible con el lenguaje y significa una ventaja importante, ya que permite portar el sitio desarrollado en PHP de un sistema a otro prácticamente sin trabajo.

PHP permite configurar el servidor de modo que se permitan o rechacen diferentes usos, lo que puede hacer al lenguaje más o menos seguro dependiendo de las necesidades de cada cual. Fue creado originalmente en 1994 por Rasmus Lerdorf, pero como PHP está desarrollado en política de código abierto, a lo largo de su historia ha tenido muchas contribuciones de otros desarrolladores.

Algunas de las más importantes capacidades de PHP son:

- Compatibilidad con las bases de datos más comunes.
- Incluye funciones para el envío de correo electrónico y carga de archivos.

Ventajas:

- Muy sencillo de aprender.
- Similar en sintaxis a C.
- Soporta en cierta medida la orientación a objeto. Clases y herencia.
- El análisis léxico para recoger las variables que se pasan en la dirección lo hace PHP de forma automática. Librándose el usuario de tener que separar las variables y sus valores.
- Se puede incrustar código PHP con etiquetas *HTML*.
- Excelente soporte de acceso a base de datos.
- La comprobación de que los parámetros son válidos se hace en el servidor y no en el cliente (como se hace con ***javascript***) de forma que se puede evitar revisar que no se reciban solicitudes adulteradas. Además PHP viene equipado con un conjunto de funciones de seguridad que previenen la inserción de órdenes dentro de una solicitud de datos.

Desventajas:

- Todo el trabajo lo realiza el servidor y no delega al cliente. Por lo tanto, puede ser más ineficiente a medida que las solicitudes aumenten de número.

- La legibilidad del código puede verse afectada al mezclar sentencias *HTML* y PHP.
- La orientación a objetos es aún muy deficiente para aplicaciones grandes.

### 1.3.8 Decisión

La decisión incluye el utilizar JAVA en el servidor Web Tomcat, la base de datos Postgres, el lenguaje de validación del lado del cliente *JavaScript* y hojas de estilo o *CSS*. Estas decisiones se discuten a continuación.

El servidor Web Tomcat permite la utilización de *JSPs*, si bien no es el único que lo permite, de manera sencilla, eficaz y eficiente. No obstante, la razón de peso para utilizar tanto el lenguaje como este servidor, es que la herramienta para generar gráficas es primariamente funcional en las versiones superiores a 4.0 de Jakarta Tomcat, y está escrita enteramente en JAVA. De esta manera, se justifica el uso tanto del lenguaje, que es robusto y orientado a objetos, como del servidor.

La base de datos Postgres es el único manejador gratuito del mundo que soporta bases de datos objeto y de datos. Sigue los estándares ANSI del *SQL* y además incorpora funciones útiles para nuestros propósitos como pueden ser: los distintos formatos para las fechas, funciones definibles para promedios, agrupación de datos, extracción de subdatos por columna, etc. El hecho de que sea gratuita es otro factor a favor de esta base de datos. La administración de la misma es bastante sencilla y existe documentación bastamente disponible en Internet, principalmente en el sitio www.postgresql.com, donde además, existe un foro y un chat en línea, lo cual ofrece una mayor cobertura a los posibles problemas de los usuarios.

El lenguaje *JavaScript* hemos decidido incorporarlo por la amplia variedad de procesos que permite realizar del lado del cliente. En un sistema que pretende dar servicio de análisis de datos a un gran número de usuarios, es importante intentar minimizar la cantidad de operaciones realizadas por el servidor. Esto involucra la necesidad de evitar que cada vez que se envíen datos para ser procesados sea el servidor quien deba validar su consistencia y corrección. El *JavaScript* permite hacer este análisis de datos en el lado del cliente asegurando, de esta manera, que los datos enviados al servidor serán siempre válidos.

Por último, hemos optado por la utilización de las hojas de estilo para poder facilitar el diseño de imagen del sitio. Desde su invención, las *CSS* han

provisto a los desarrolladores una forma de evitar las distintas vistas mediante la definición de plantillas de estilo, que incluso ayudan a quien viene detrás para generar adiciones o modificaciones a un módulo existente.

## 1.4 Patrón Modelo – Vista – Controlador

Este patrón se usa en aplicaciones interactivas que requieren una interfaz de usuario flexible. Es adecuado para situaciones donde:

- Es necesario presentar los mismos datos de diferentes formas o utilizando diferentes interfaces de usuario.
- Si se cambia un dato, los datos se deben de actualizar en todas las representaciones.
- La interfaz de usuario debe de ser fácil de modificar.

El patrón Modelo-Vista-Controlador (MVC) descompone una aplicación interactiva en tres grandes bloques: modelo, vistas y controlador. El modelo contiene los datos y la funcionalidad de la aplicación. Es independiente de la representación de los datos. Las vistas muestran la información al usuario de una cierta forma. Existen todas las que se necesite definir. Cada vista tiene un controlador asociado. Los controladores reciben entradas en forma de eventos que responden a mandos realizados por el usuario a través del ratón o del teclado. El control traduce estos eventos a peticiones a la vista o al modelo.

Si se ordenan estos tres grupos por probabilidad de ser reutilizable, se tiene un resultado como el siguiente:

- Lo más reutilizable y que es menos susceptible de cambio, es el modelo.
- En un punto intermedio está el controlador.
- Finalmente, lo que más cambia es la vista.

El modelo debe ser independiente. Las clases del modelo no deben ver a ninguna clase de los otros grupos. Siguiendo con el orden de posibilidad de reutilización, el controlador podría ver clases del modelo, pero no de la vista.

La vista es lo más cambiante, así que se puede hacer que vea clases del modelo y del controlador. Si se cambia algo del controlador o del modelo, es bastante seguro que se tendrá como mínimo que recompilar la interfaz gráfica.

En la tabla 1.4 se puede ver la estructura del Patrón Modelo-Vista-Controlador (MVC). Y en la tabla 1.5, se pueden ver las ventajas y desventajas del mismo.

| Clase | Colaboradores | Responsabilidad |
|---|---|---|
| Modelo | • Vista<br>• Controlador | • Contiene la funcionalidad de la aplicación.<br>• Lleva un registro de las vistas y controladores del sistema.<br>• Notifica los cambios en los datos a los componentes. |
| Controlador | • Vista<br>• Modelo | • Acepta los eventos de entrada.<br>• Traduce los eventos de entrada a peticiones al modelo o a las vistas.<br>• Implementa el procedimiento actualizar si es necesario. |
| Vista | • Modelo<br>• Controlador | • Crea e inicializa su controlador asociado<br>• Muestra información al usuario.<br>• Actualiza la información.<br>• Recoge datos del modelo. |

**Tabla 1. 4 Estructura del Patrón Modelo-Vista-Controlador**

| Ventajas: | Desventajas: |
|---|---|
| • Múltiples vistas del mismo modelo.<br>• Vistas sincronizadas.<br>• Flexibilidad para cambiar las vistas y los controladores.<br>• La aplicación puede soportar distintos tipos de interfaz de usuario. | • Complejidad creciente.<br>• Cambios innecesarios. Puede ser que no todas las vistas estén interesadas en todos los cambios.<br>• Conexión entre la vista y el controlador. Hay que usar los dos a la vez.<br>• Si cambia la interfaz del modelo, hay que cambiar todas las vistas y todos los controladores.<br>• Acceso ineficiente a los datos en la vista. Puede necesitar varias llamadas al modelo para actualizar todos sus datos.<br>• Tanto la vista como el controlador son específicos de una plataforma.<br>• Algunas herramientas de diseño de interfaces de usuario incorporan parte del procesamiento de eventos entrada. El controlador deja de ser necesario. |

**Tabla 1. 5 Ventajas y desventajas del Patrón MVC**

# 2. Análisis del Sistema

## 2.1 UML

Es el lenguaje unificado de modelo sucesor de la oleado de métodos de análisis y diseño orientados a objetos que surgió a finales de la década de 1980. El UML unifica, sobre todo, los métodos de Booch, Rumbaugh (OMT) y Jacobson. Se puede definir UML como una notación, principalmente gráfica de que se valen los métodos de modelado para expresar los diseños.

### 2.1.1. Diagrama de Casos de Usos

Técnica para capturar información de cómo un sistema o negocio trabaja, o de cómo se desea que trabaje. No pertenece estrictamente al enfoque orientado a objetos. Es una técnica para captura de requisitos.

- Describen bajo la forma de acciones y reacciones el comportamiento de un sistema desde el punto de vista del usuario.
- Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno.
- Son descripciones de la funcionalidad del sistema independientes de la implementación.
- Basados en el lenguaje natural, es decir, son accesibles por los usuarios.

### 2.1.1.1 Actores

Usuario: Persona que recurre al sistema para consultar datos y/o gráficas

Administrador: Persona que administra el sistema.

### 2.1.1.2 Diagrama

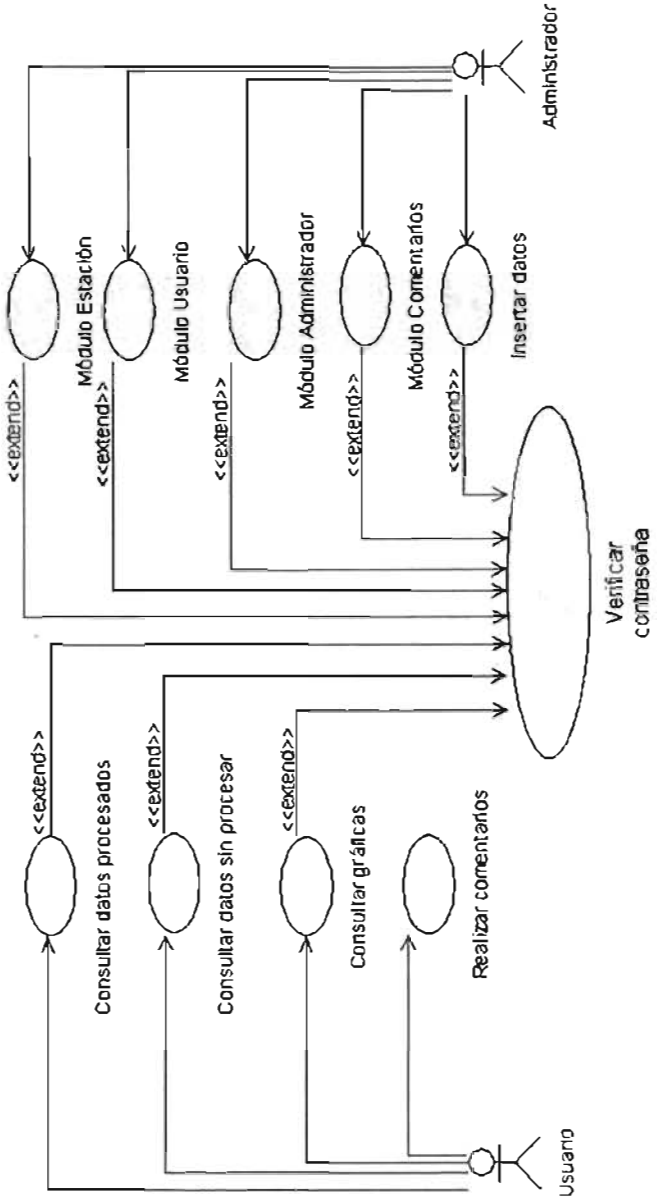



**Figura 2.1 Diagrama de casos de uso**

### 2.1.1.3 Casos de Uso

Nombre: Consultar datos procesados

Resumen: Muestra datos procesados de una estación.

Descripción: El caso de uso comienza cuando un usuario desea consultar los datos procesados de una estación particular. Los datos procesados pueden ser:

- Máximo de un mes de un año determinado
- Mínimo de un mes de un año determinado
- Máximos de un mes de un año determinado
- Mínimos de un mes de un año determinado

El usuario debe seleccionar la estación de la cual desea realizar la consulta. Posteriormente, el usuario debe seleccionar el tipo de dato procesado que desea y proporcionar año y/o mes.

El caso de uso termina cuando se muestran los datos solicitados.

Nombre: Consultar datos sin procesar

Resumen: Muestra datos sin procesar de una estación.

Descripción: El caso de uso comienza cuando un usuario desea consultar los datos sin procesar de una estación particular.

El usuario debe seleccionar la estación de la cual desea realizar la consulta.

El caso de uso termina cuando se muestran los datos solicitados.

Nombre: Consultar gráficas

Resumen: Muestra gráficas de una estación.

Descripción: El caso de uso comienza cuando un usuario desea consultar gráficas de una estación particular. Las gráficas pueden ser:

- Promedios anuales
- Promedios mensuales por año
- Promedios diarios por mes
- Promedio por hora
- Alturas horarias por mes

El usuario debe seleccionar la estación de la cual desea realizar la consulta. Posteriormente el usuario debe seleccionar el tipo de gráfica que desea y proporcionar año y/o mes. Finalmente el usuario debe seleccionar el tamaño de la gráfica y si la desea con ordenada al origen.

El caso de uso termina cuando se muestra la gráfica solicitada.

Nombre: Realizar comentarios

Resumen: El usuario realiza comentarios.

Descripción: El caso de uso comienza cuando un usuario desea realizar comentarios. El usuario debe proporcionar:

- Nombre
- Correo electrónico
- Organización
- Mensaje

El caso de uso termina cuando se recibe un mensaje de confirmación o de fallo.

Nombre: Verificar contraseña

Resumen: Verifica la contraseña del usuario o administrador.

Descripción: El caso de uso comienza cuando un usuario o un administrador desea utilizar el sistema

El usuario o administrador debe proporcionar su contraseña para poder utilizar el sistema.

El caso de uso termina cuando la operación solicitada es realizada.

Nombre: Módulo Estación

Resumen: Permite al administrador insertar, modificar o borrar una estación.

Descripción: El caso de uso comienza cuando un administrador desea:

- Insertar una estación.

  El administrador debe proporcionar el nombre de la estación, coordenadas, latitud y longitud, croquis, dirección, bancos de nivel y planos de referencia de la estación que desea insertar.

- Modificar los datos de una estación.

  El administrador debe seleccionar en la lista de estaciones la estación que desea modificar. Posteriormente modificar los datos necesarios.

- Borrar una estación.

  El administrador debe seleccionar en la lista de estaciones la estación que desea borrar.

El caso de uso termina cuando se recibe un mensaje de confirmación o de fallo.

Nombre: Módulo Usuario

Resumen: Permite al administrador insertar, modificar o borror un usuario.

Descripción: El caso de uso comienza cuando un administrador desea:

- Insertar un usuario.

  El administrador debe proporcionar el nombre, apellidos, lugar, organización, teléfono y correo electrónico del usuario que desea insertar. Posteriormente debe proporcionar el login de usuario, su password y confirmar el password.

- Modificar los datos de un usuario.

  El administrador debe seleccionar en la lista de usuarios el usuario que desea modificar. Posteriormente modificar los datos necesarios.

- Borrar un usuario.

  El administrador debe seleccionar en la lista de usuarios el usuario que desea borrar.

El caso de uso termina cuando se recibe un mensaje de confirmación o de fallo.

Nombre: Módulo Administrador

Resumen: Permite al administrador insertar, modificar o borrar un administrador.

Descripción: El caso de uso comienza cuando un administrador desea:

- Insertar un administrador.

  El administrador debe proporcionar el nombre, apellidos, login, organización, password y confirmar el password.

- Modificar los datos de un administrador.

  El administrador debe seleccionar en la lista de administradores el administrador que desea modificar. Posteriormente modificar los datos necesarios.

- Borrar un administrador.

  El administrador debe seleccionar en la lista de administradores el administrador que desea borrar.

El caso de uso termina cuando se recibe un mensaje de confirmación o de fallo.

Nombre: Insertar datos

Resumen: Permite al administrador insertar datos de una estación.

Descripción: El caso de uso comienza cuando un administrador desea insertar datos de una estación. Es necesario que la estación haya sido dada de alta anteriormente.

El administrador debe indicar la localización del archivo que contiene los datos que desea subir e indicar la estación a la cual pertenecen.

El caso de uso termina cuando se recibe un mensaje de confirmación o de fallo.

Nombre: Módulo Comentarios

Resumen: Permite al administrador ver o borrar los comentarios que se han realizado.

Descripción: El caso de uso comienza cuando un administrador desea:

- Ver un comentario.

  El administrador debe seleccionar en la lista de comentarios el comentario que desea ver.

- Borrar un comentario.

  El administrador debe seleccionar en la lista de comentarios el comentario que desea borrar.

El caso de uso termina cuando se recibe un mensaje de confirmación o de fallo.

## 2.1.2 Diagrama de Clases

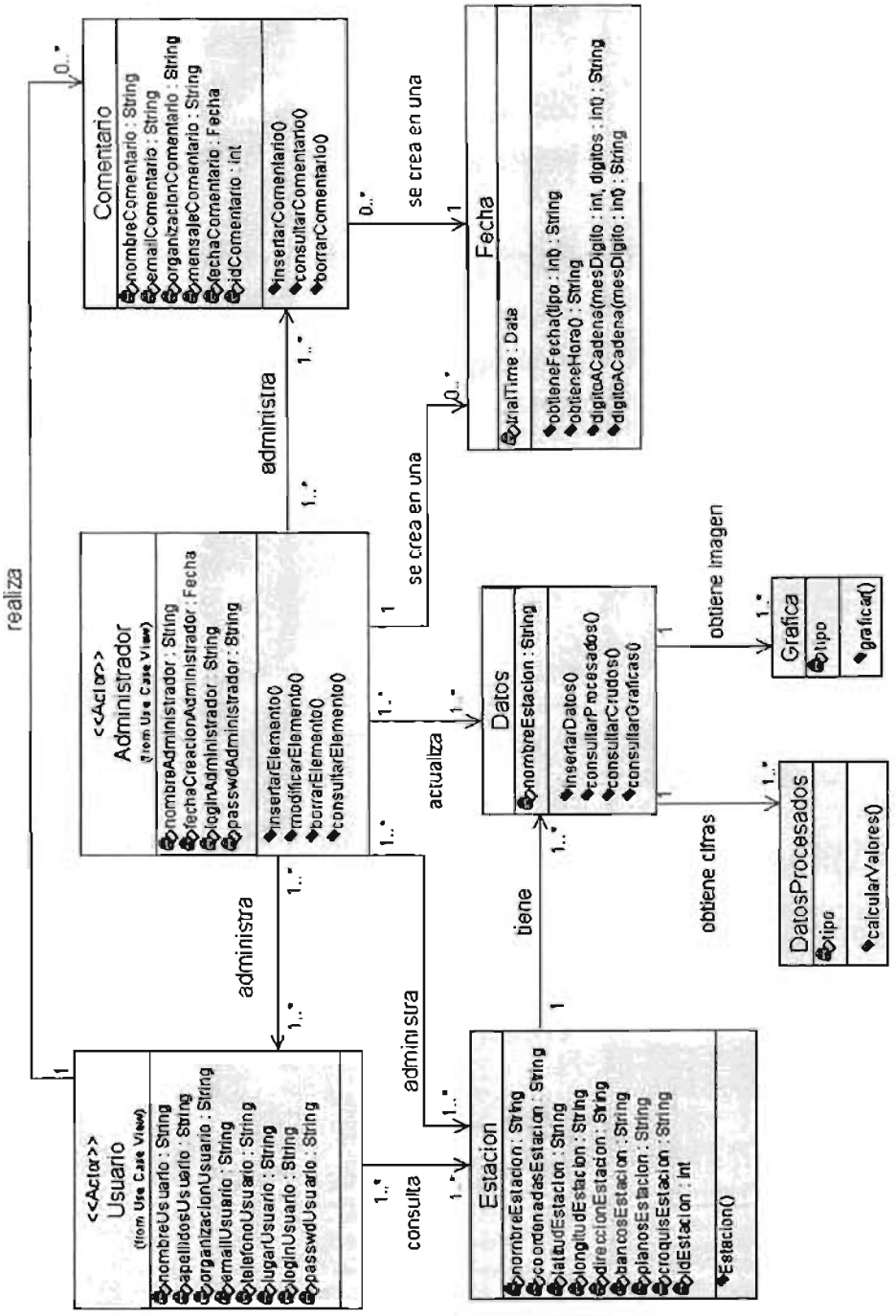
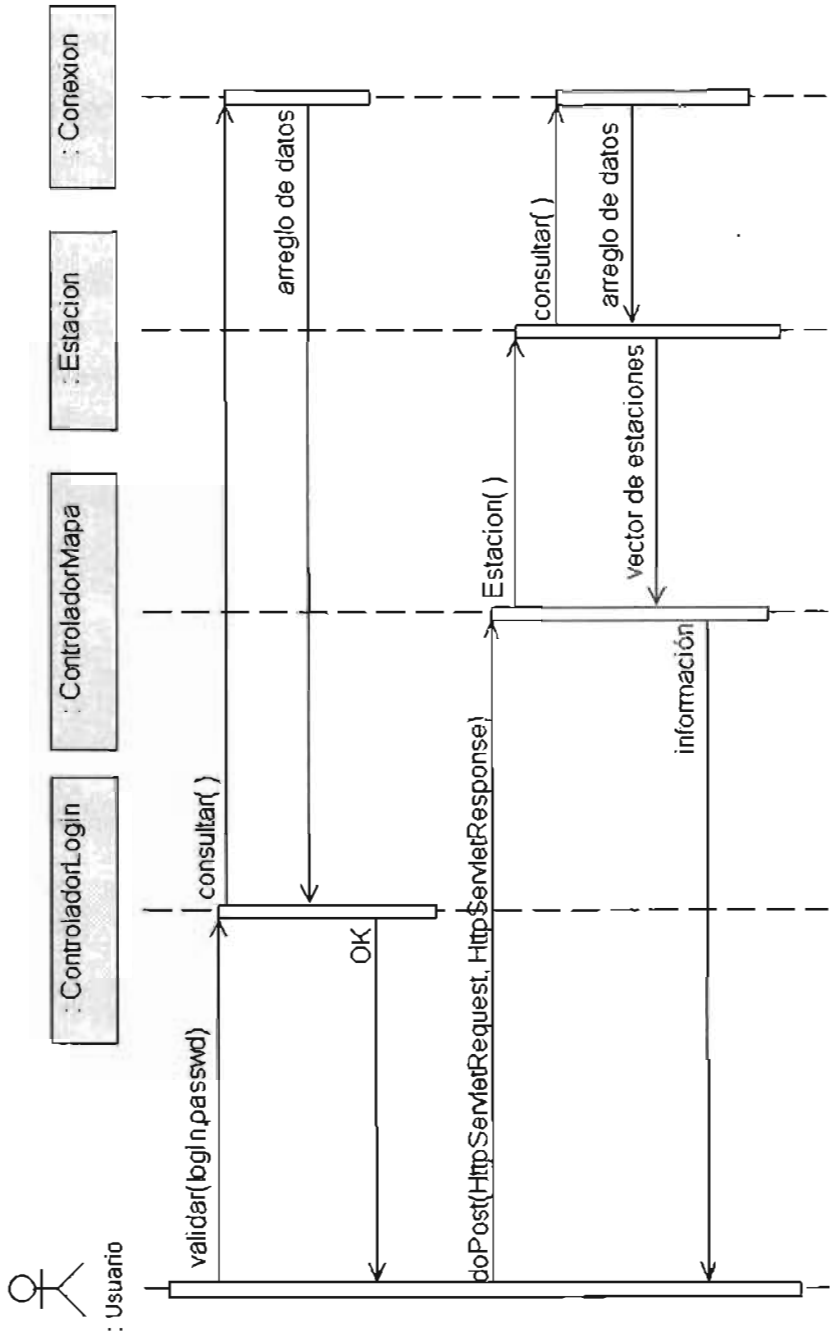


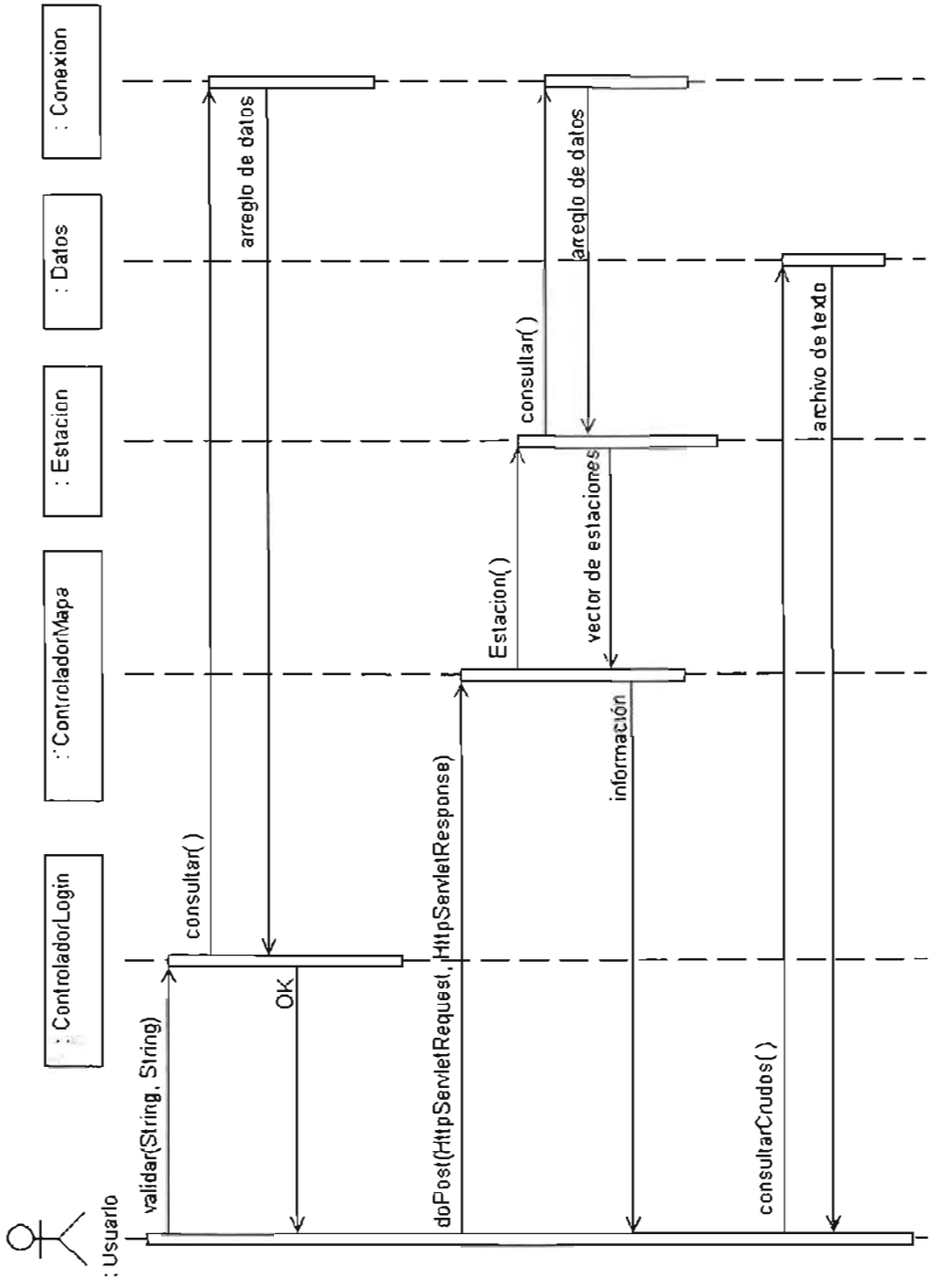

**Figura 2.2 Diagrama de Clases**

### 2.1.3 Diagramas de Secuencia

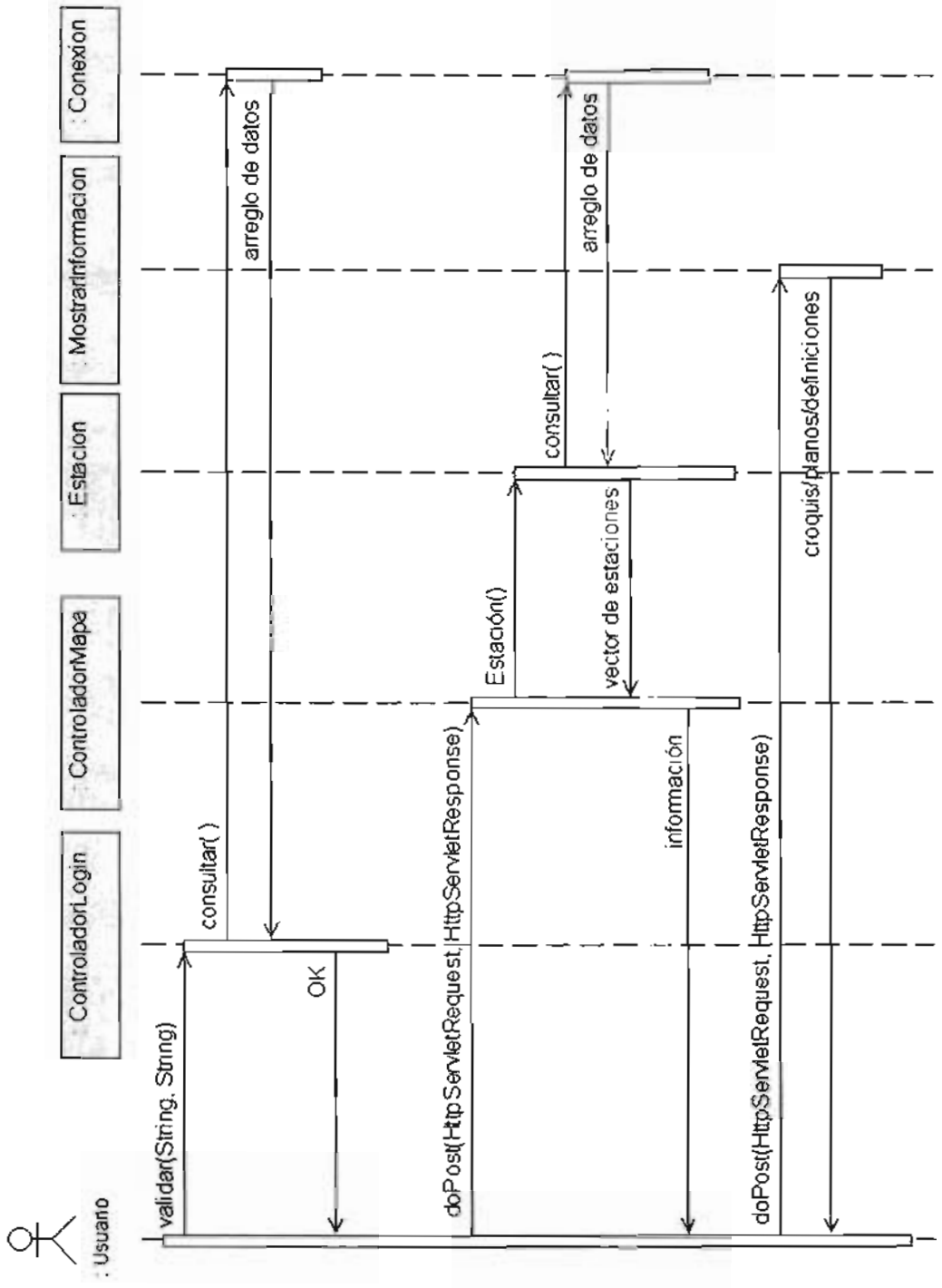
Ver información de estación



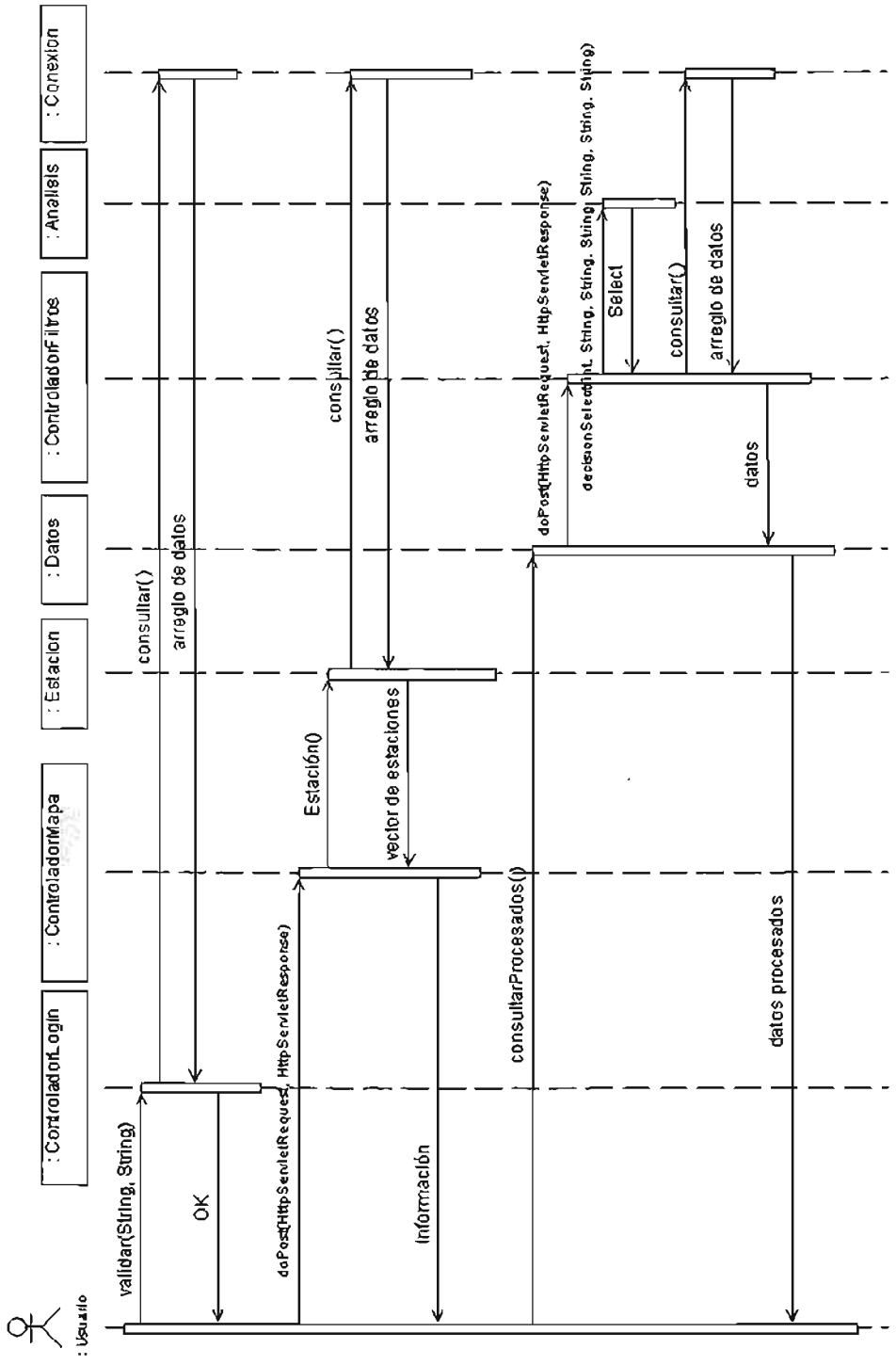
Ver datos Crudos



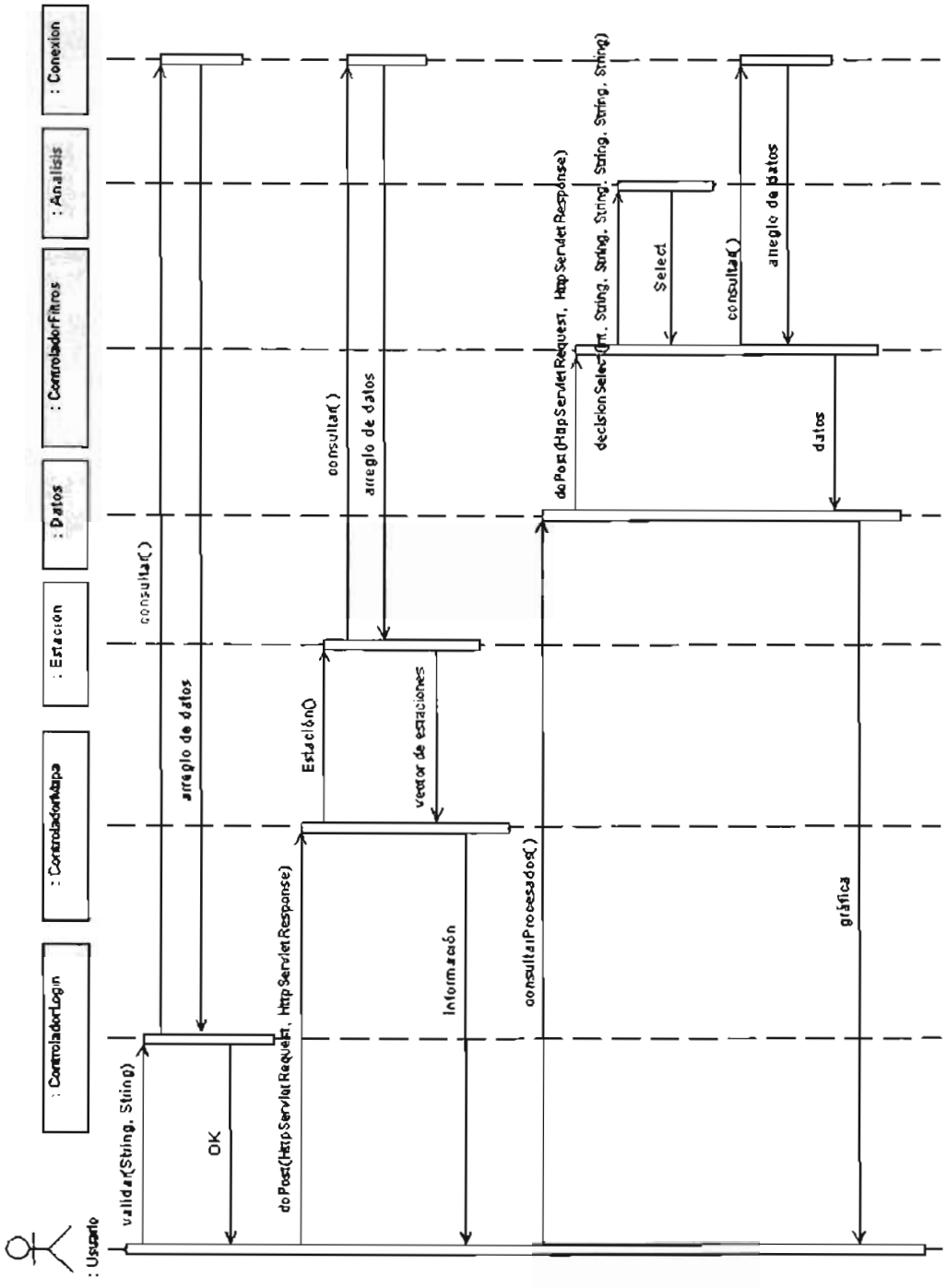
Consultar Croquis/ Planos/ Definiciones



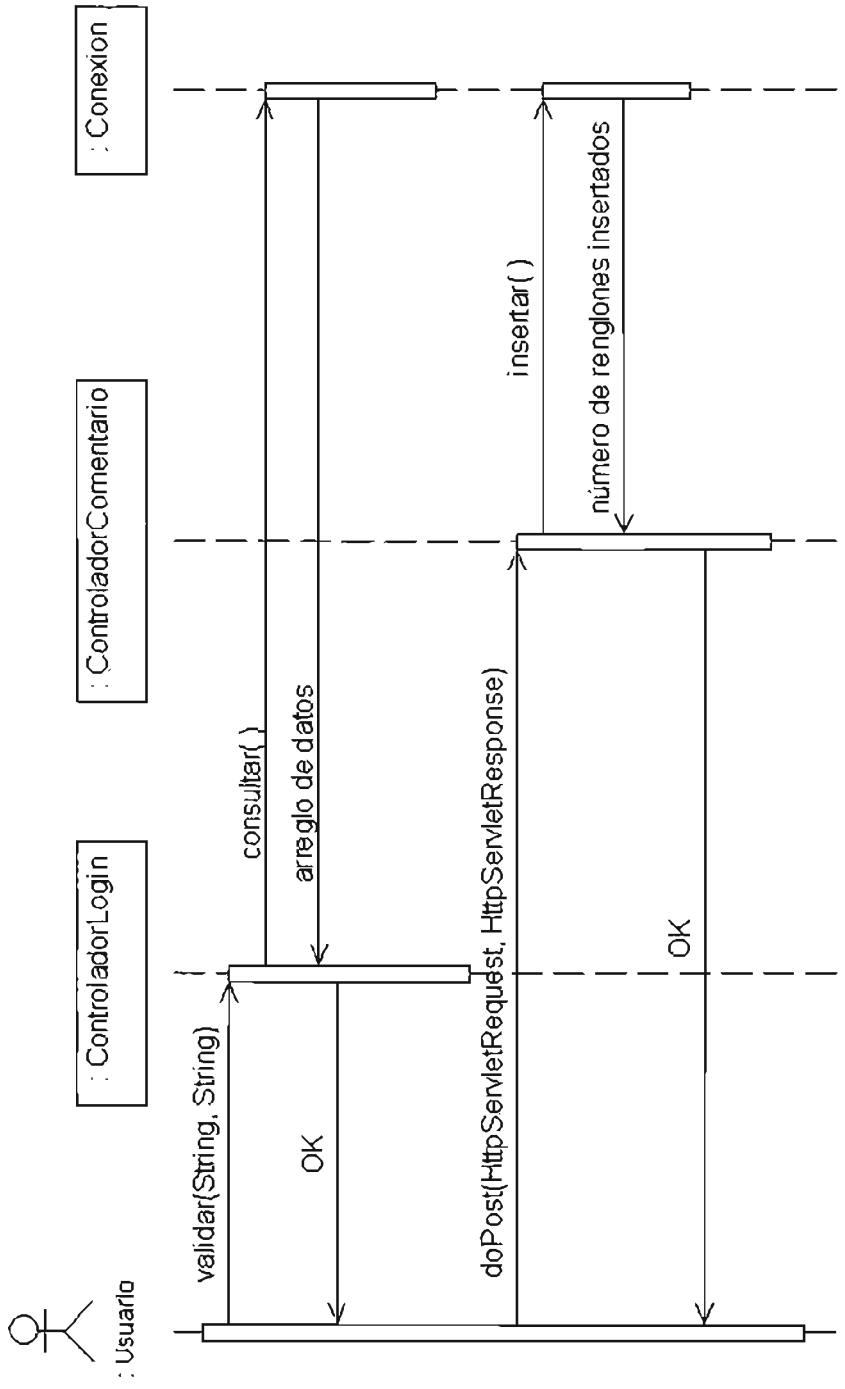
Consultar Datos Procesados



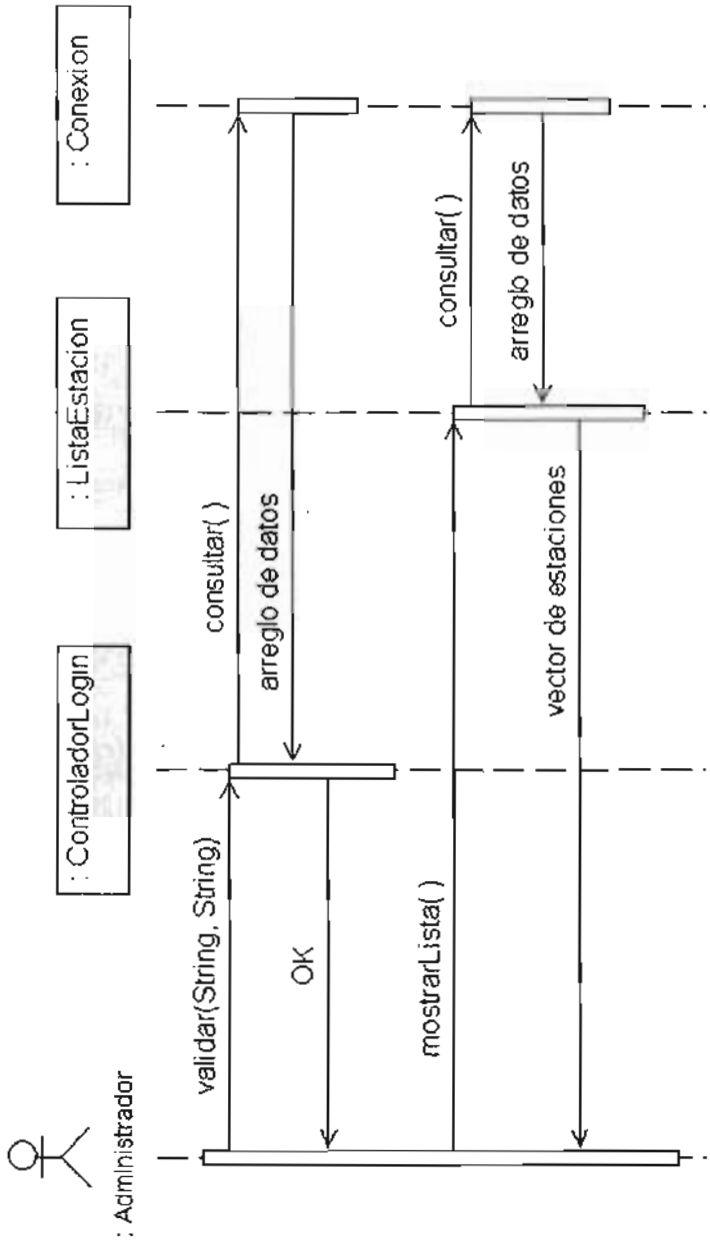
Consultar Gráficas



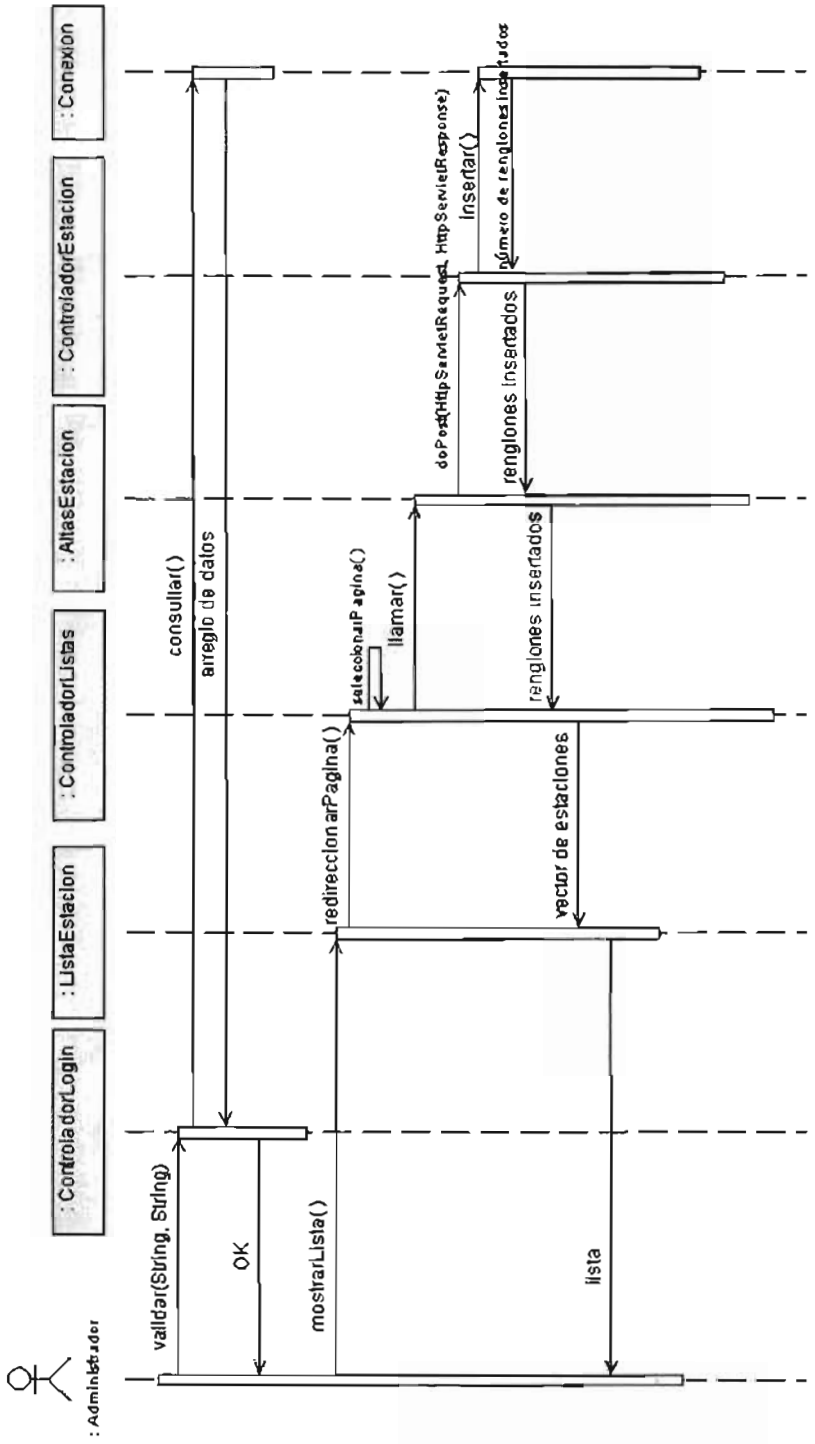
Insertar Comentario



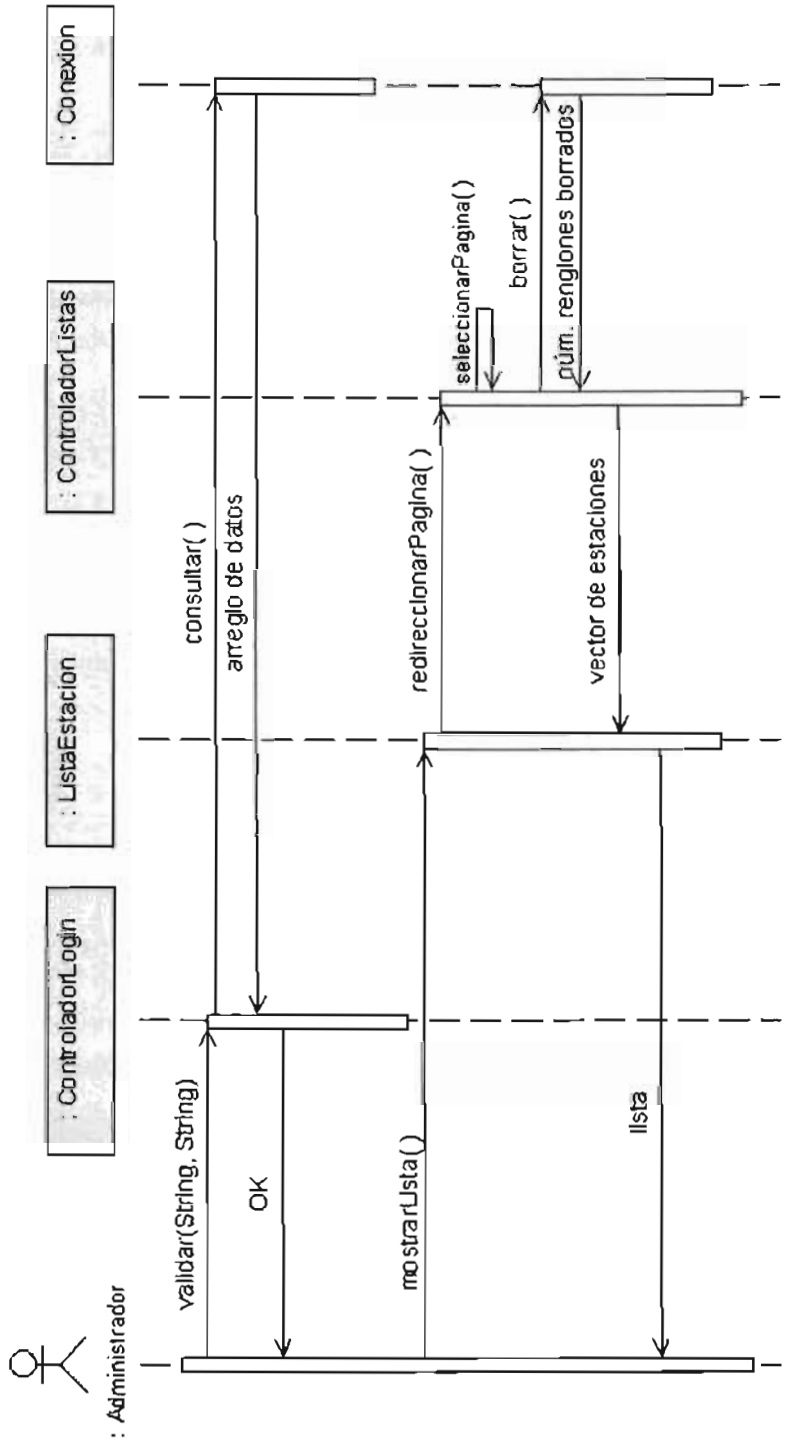
Ver Lista Estación

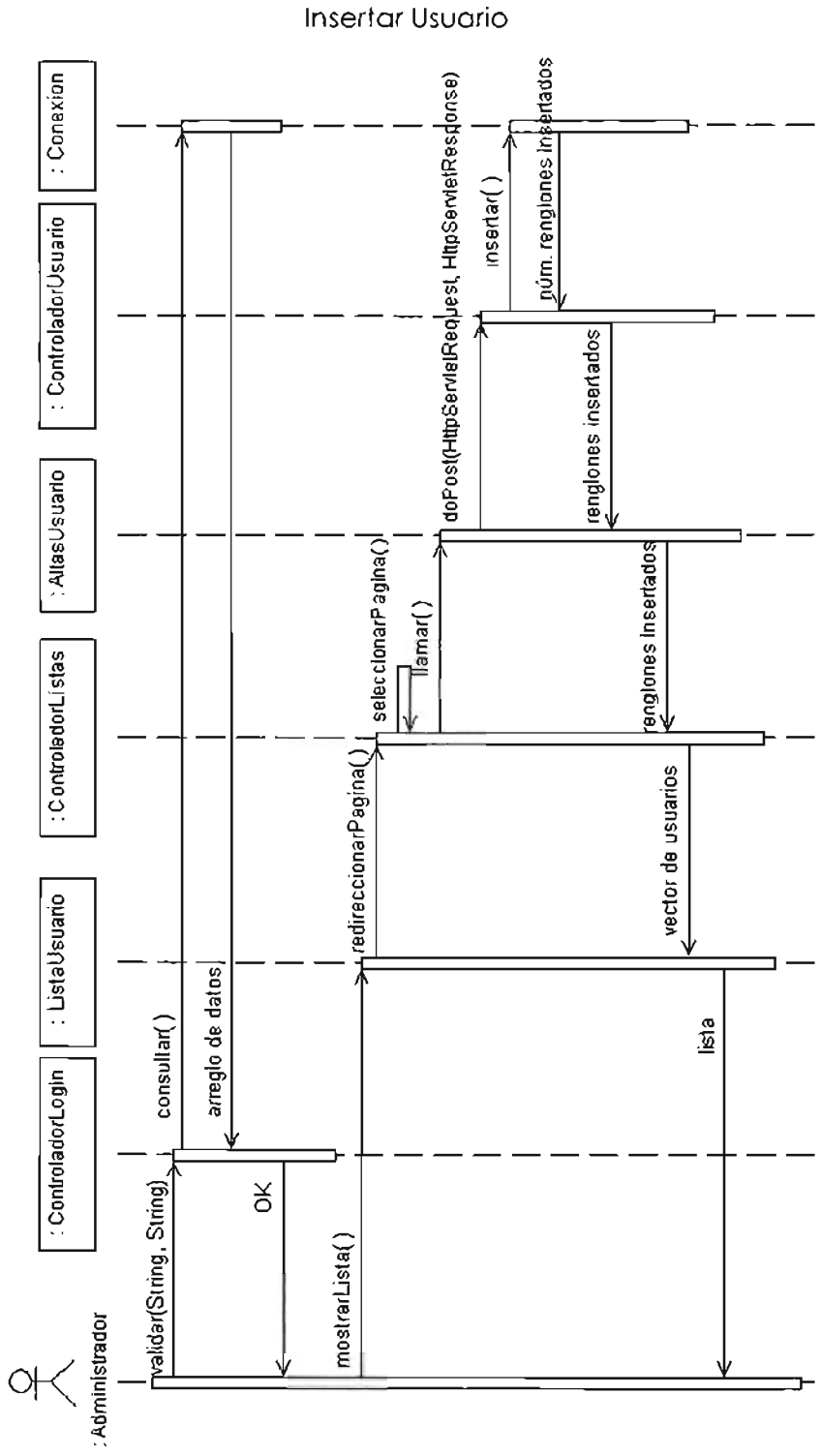


Insertar Estación

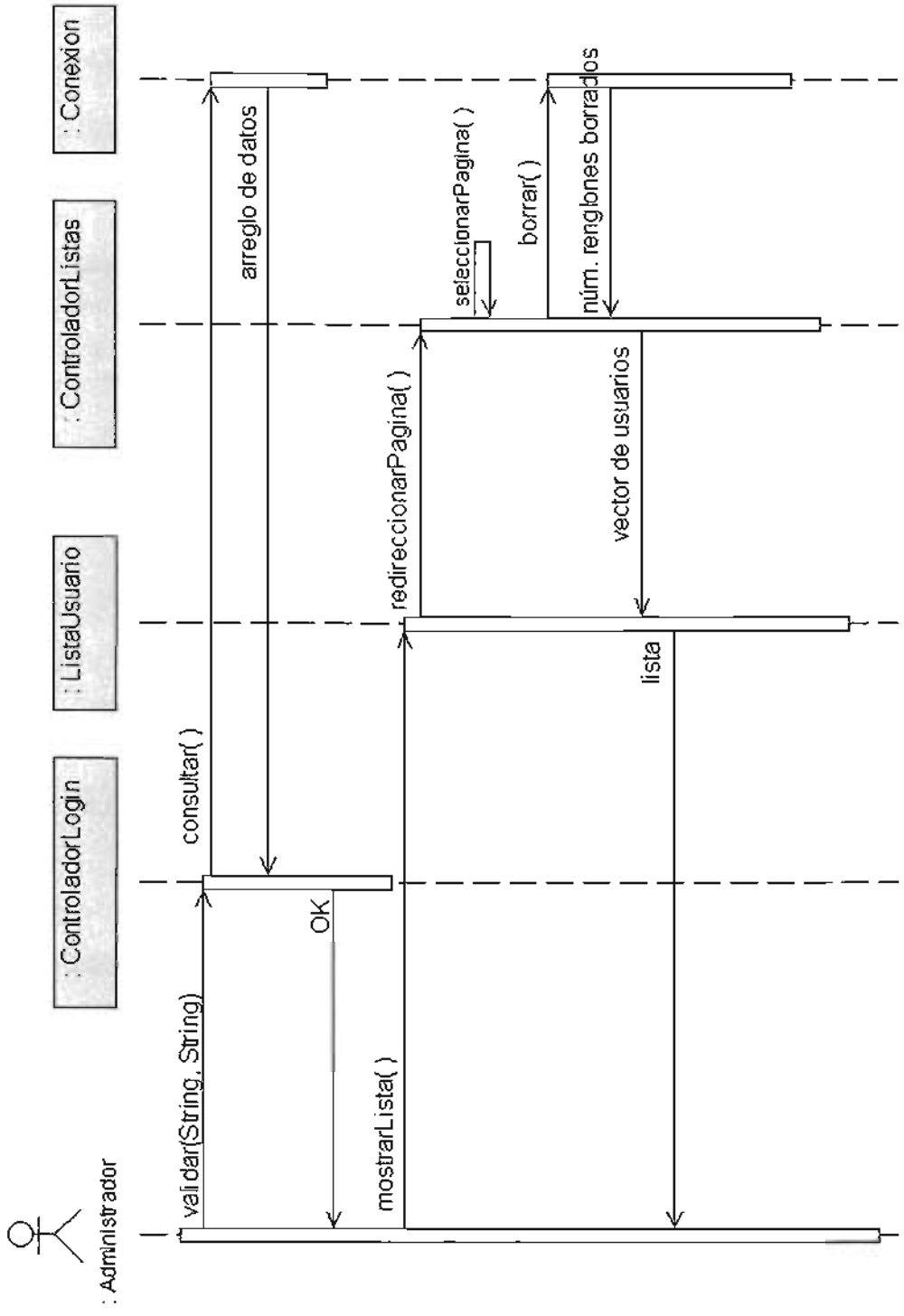


Eliminar Estación

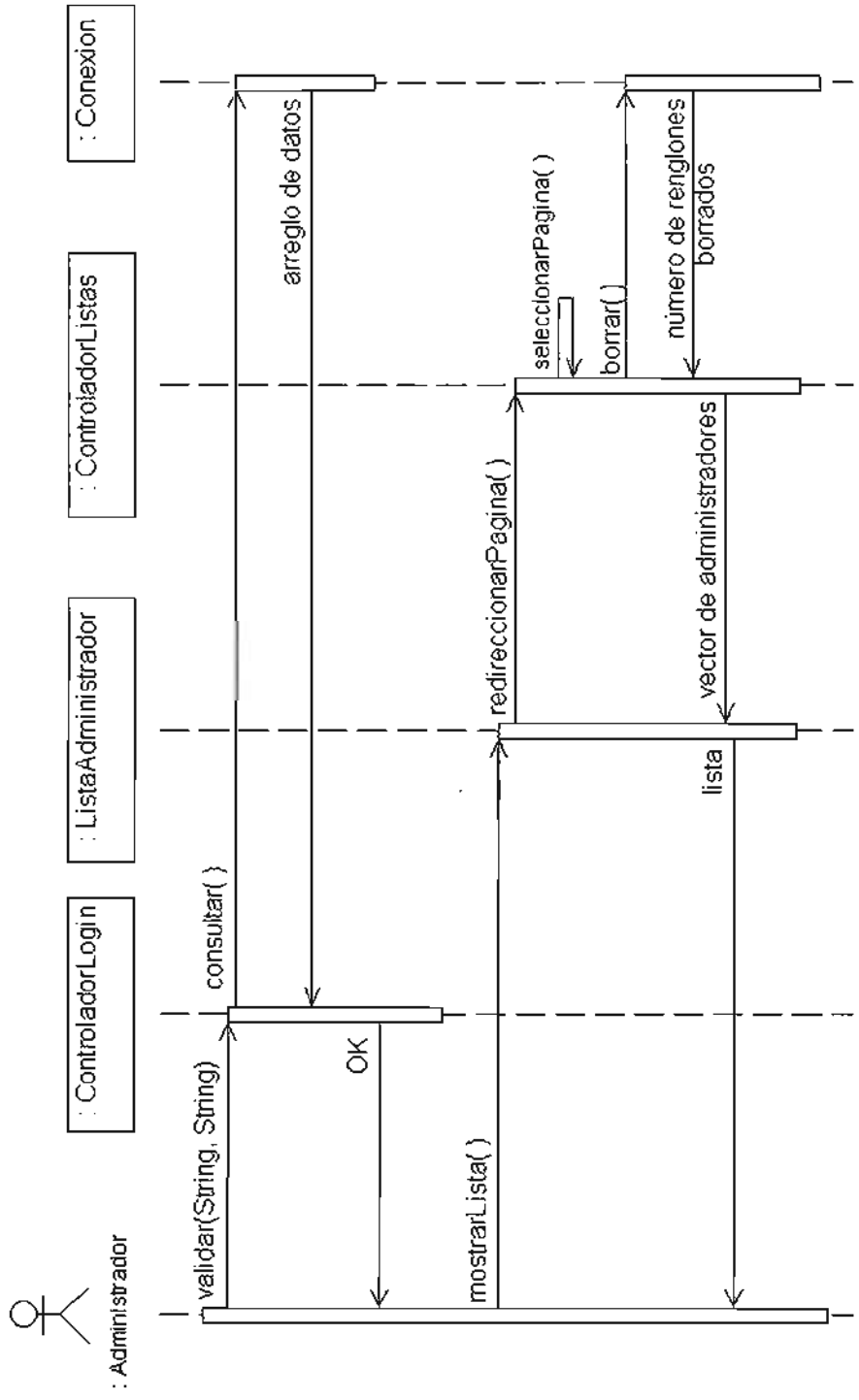



Insertar Usuario
: Administrador
: ControladorLogin
: ListaUsuario
: ControladorListas
: AltasUsuario
: ControladorUsuario
: Conexion
validar(String, String)
OK
consultar( )
arreglo de datos
mostrarLista( )
redireccionarPagina( )
seleccionarPagina( )
llamar( )
doPost(HttpServletRequest, HttpServletResponse)
insertar( )
núm. renglones insertados
renglones insertados
renglones insertados
vector de usuarios
lista

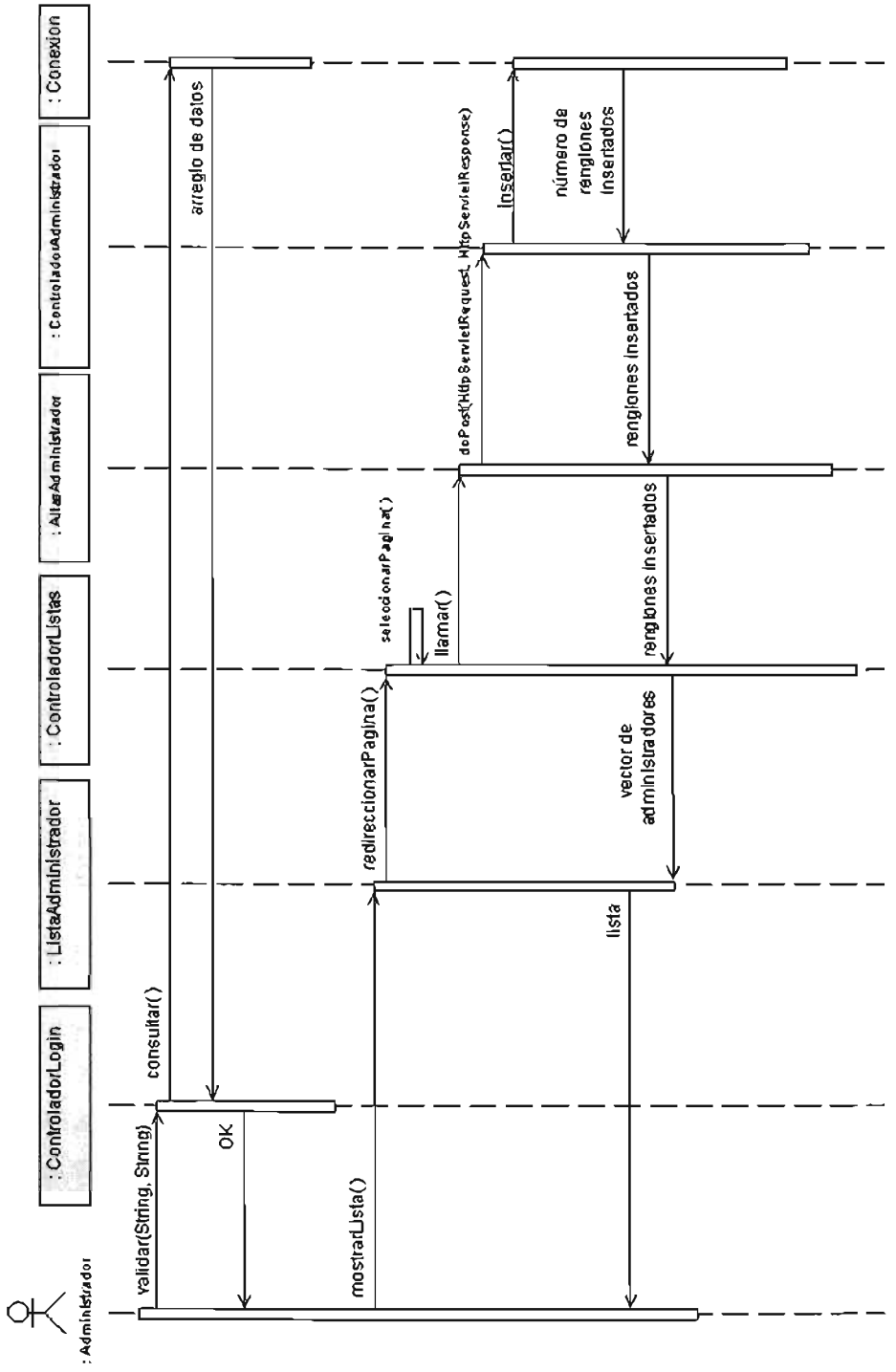
Eliminar Usuario



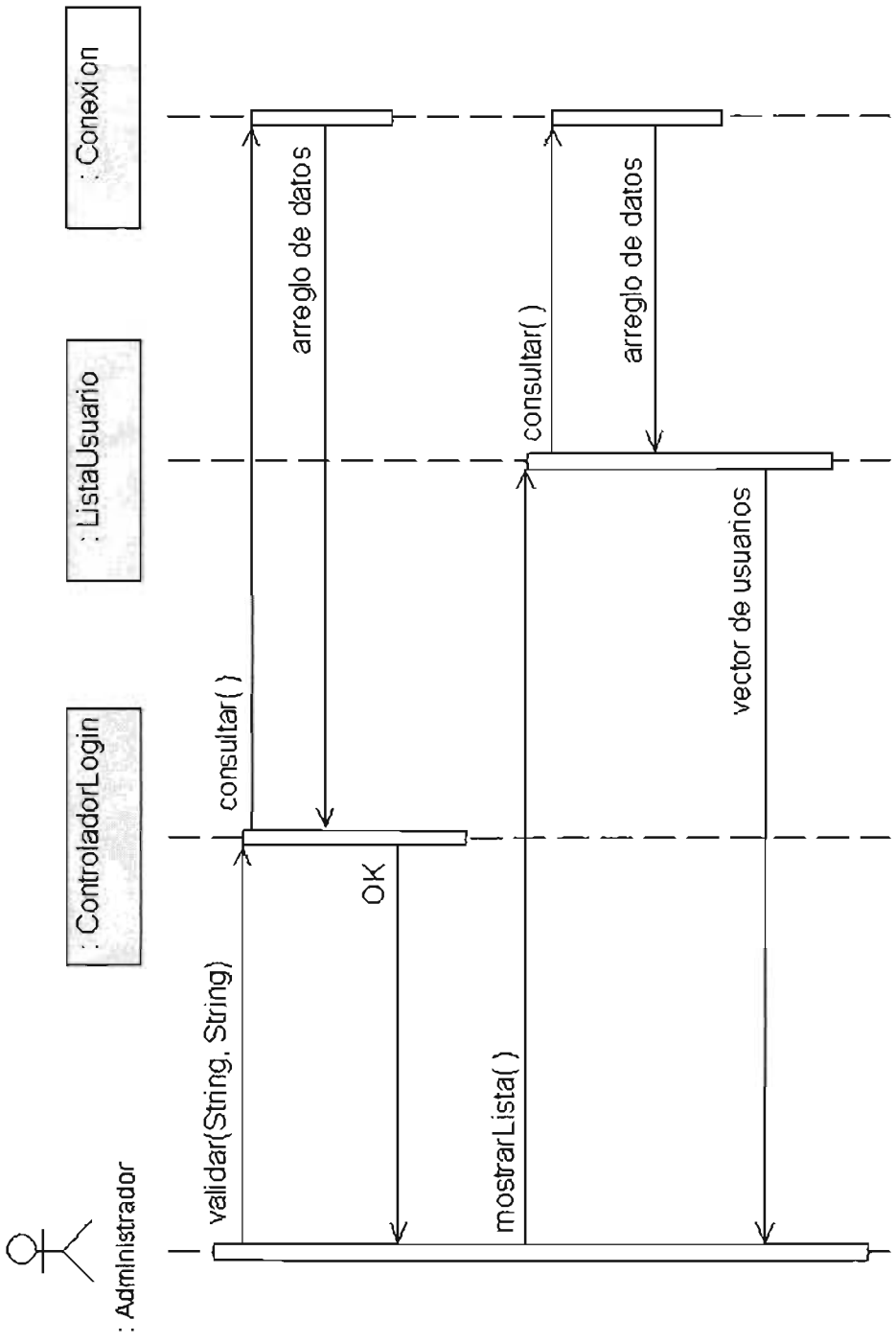
Eliminar Administrador



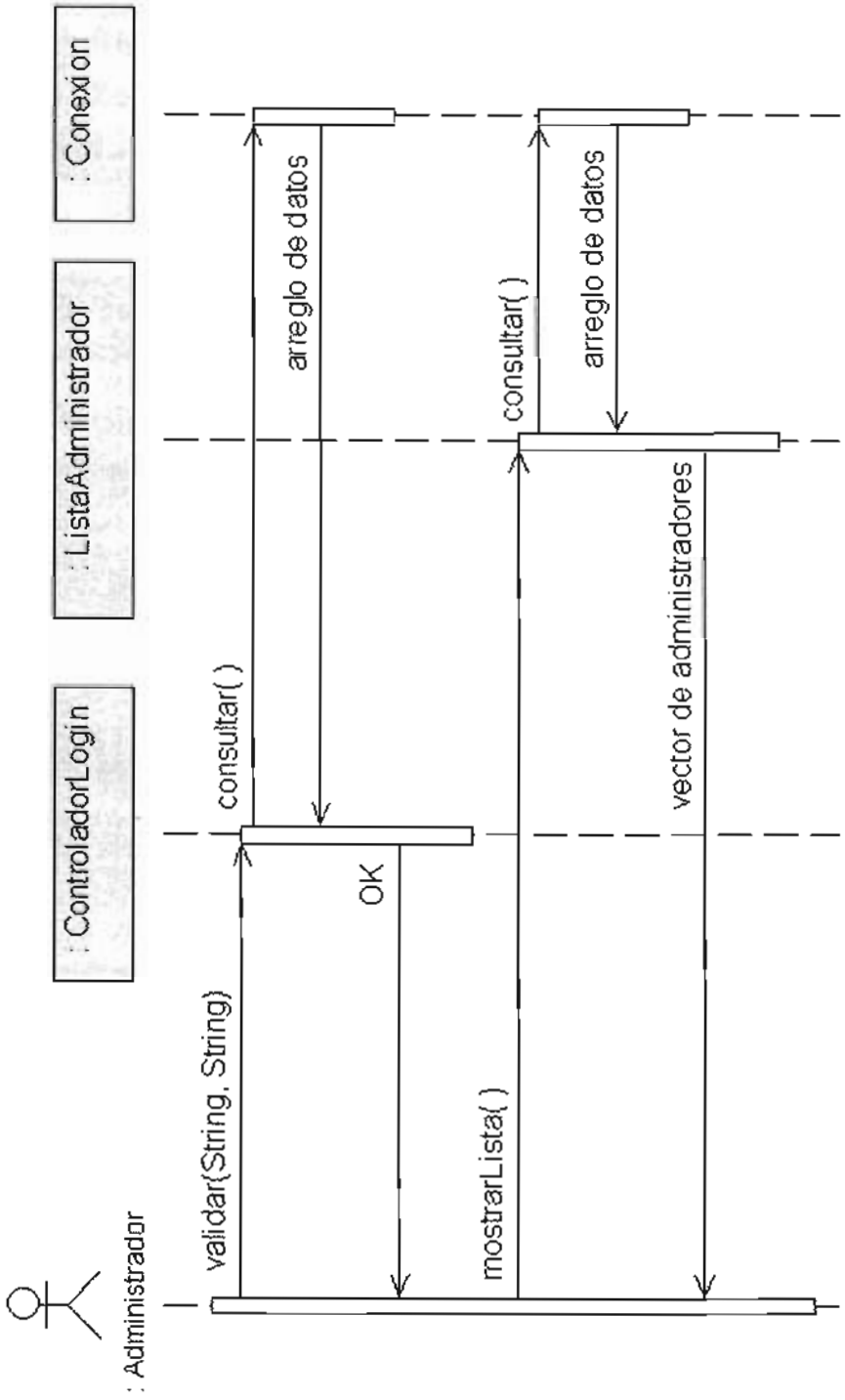
Insertar Administrador



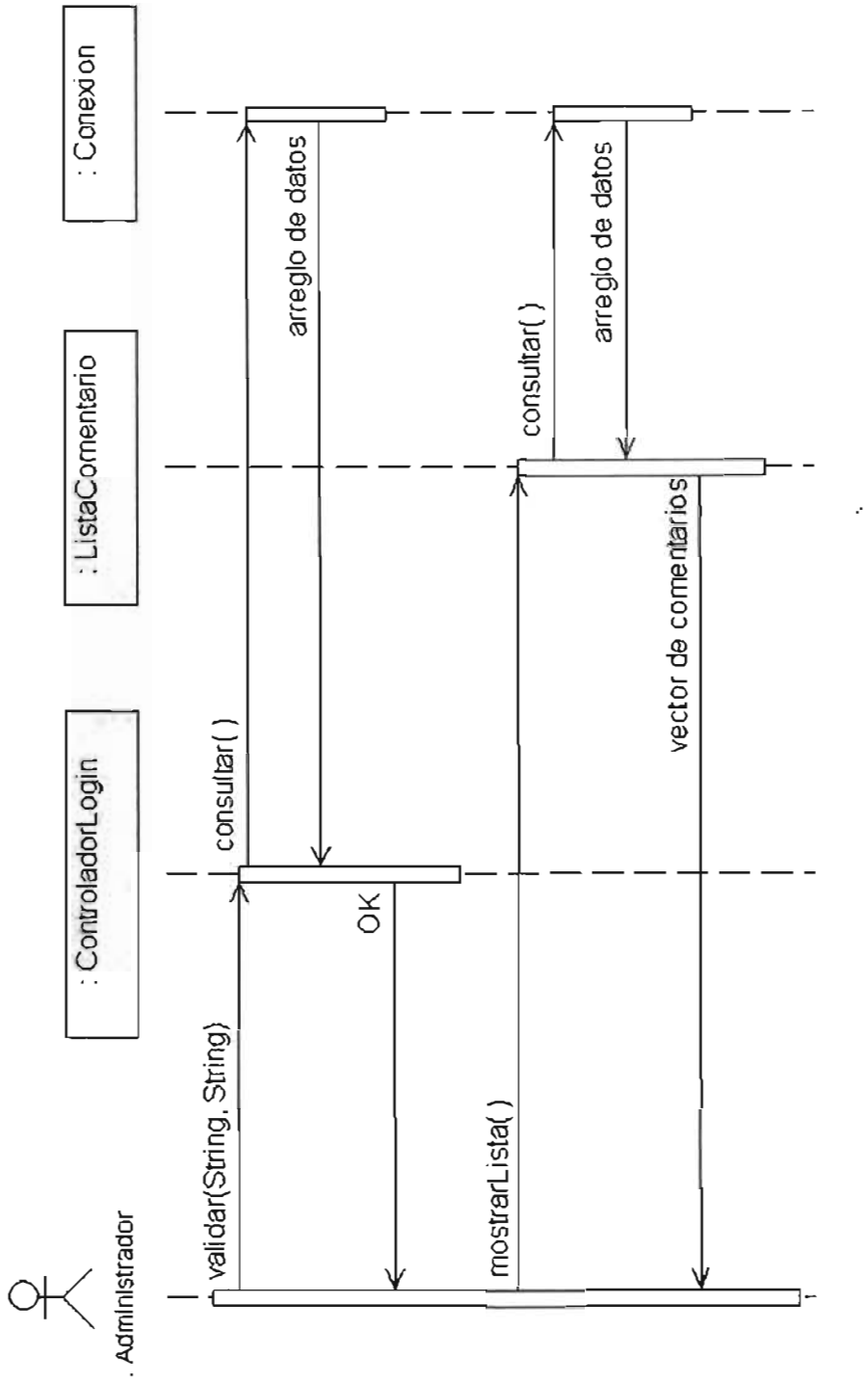
Ver Lista Usuario



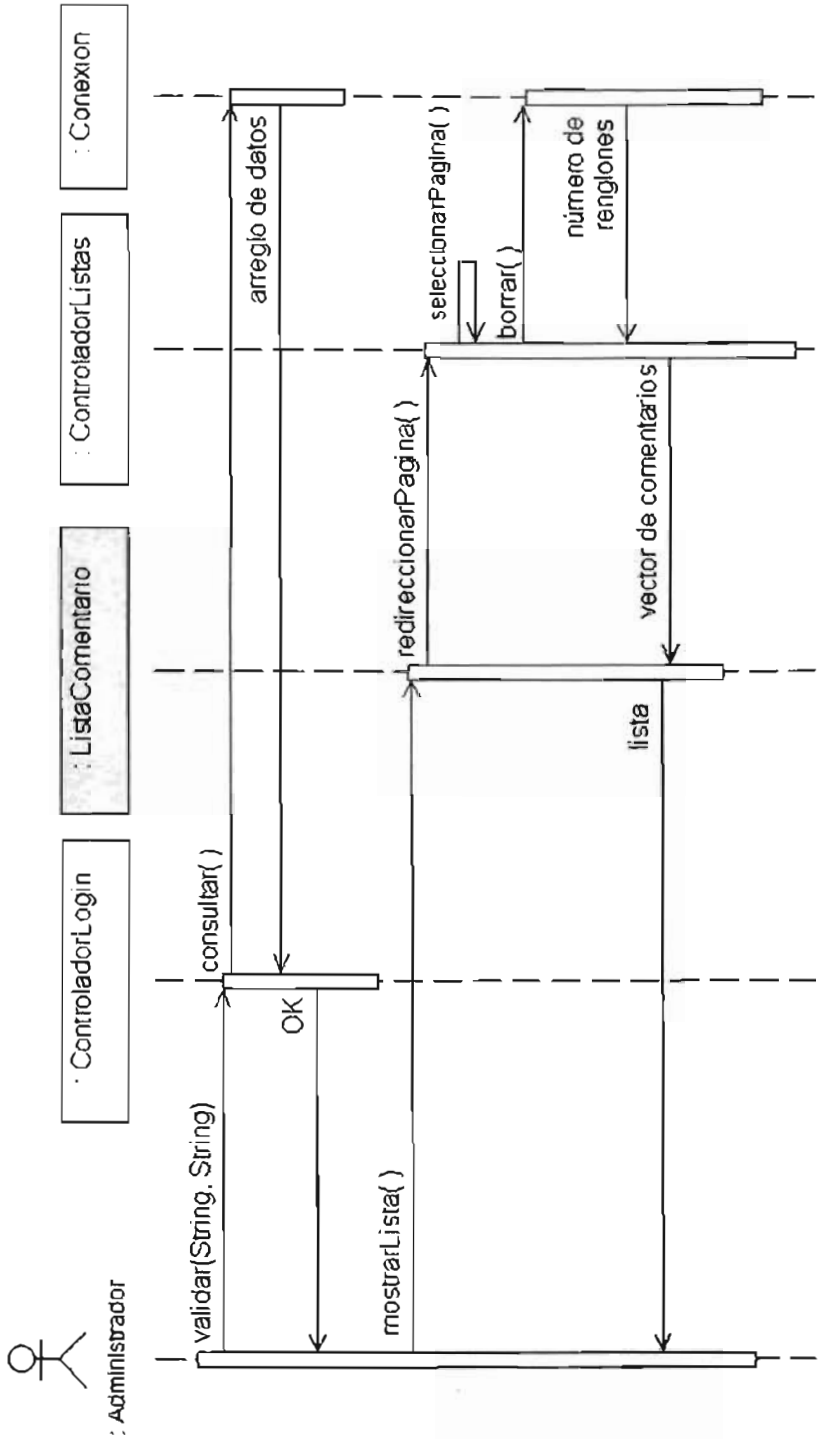
Ver Lista Administrador



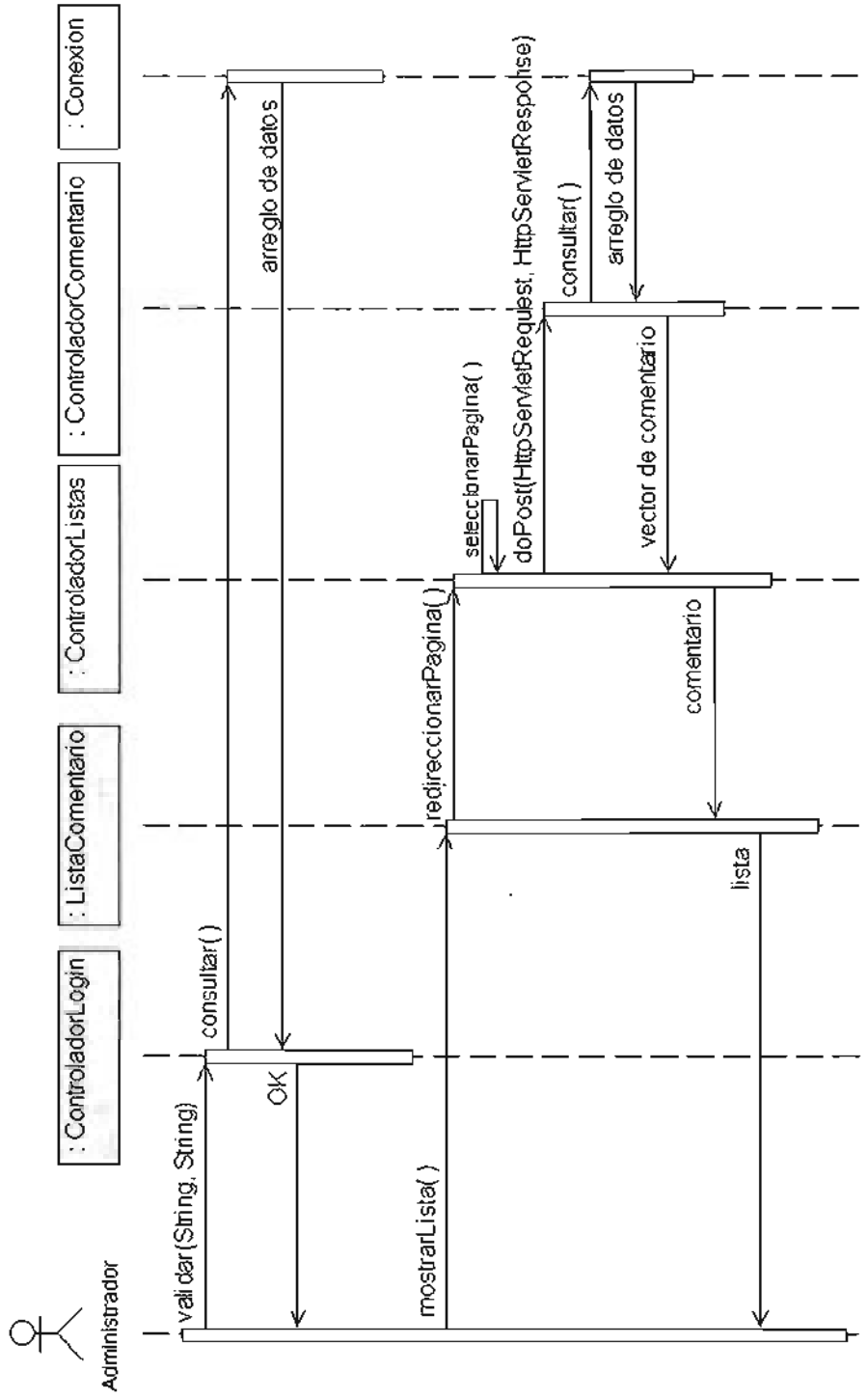
Ver Lista Comentarios



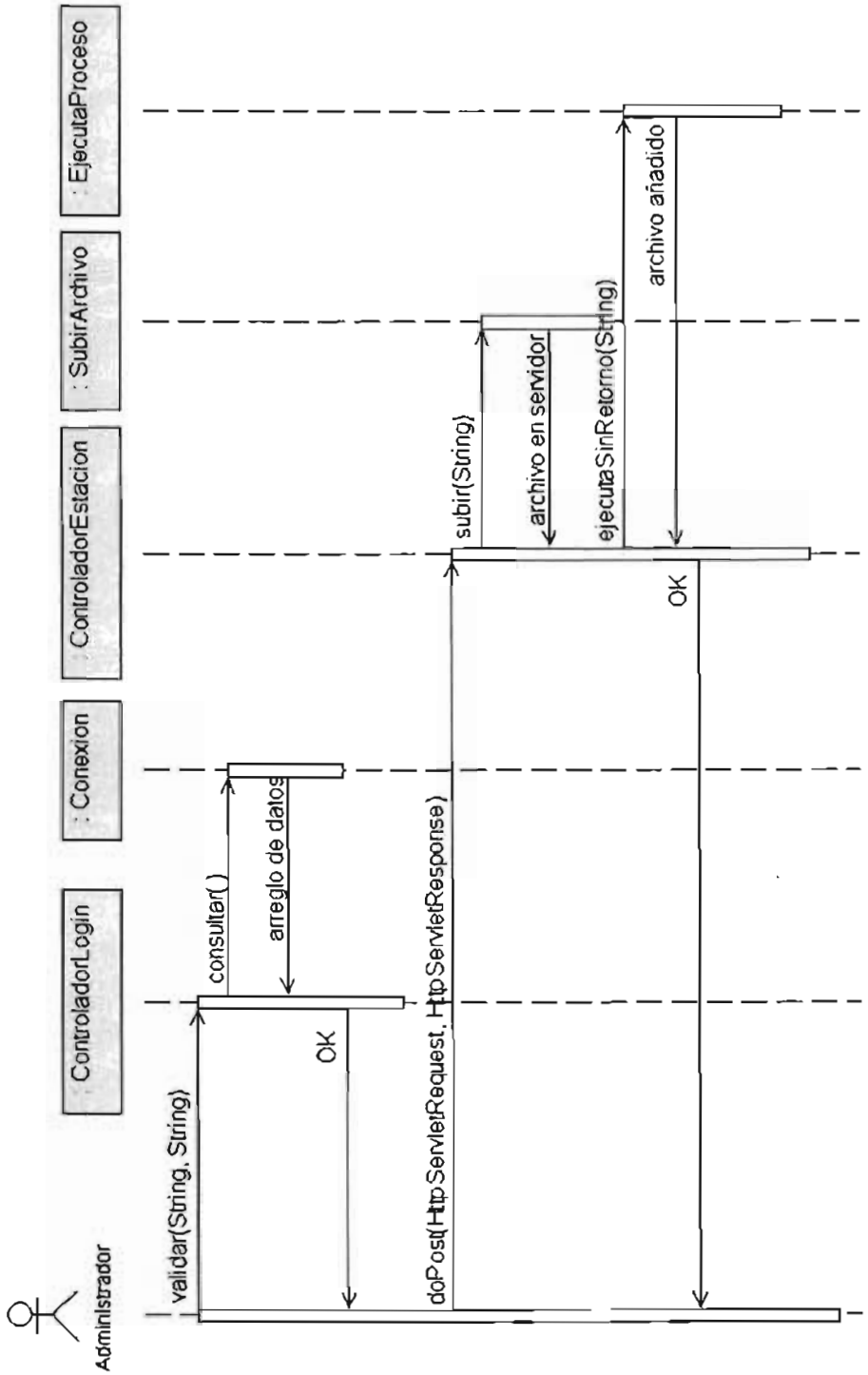
Eliminar Comentario



Revisar Contenido Comentario

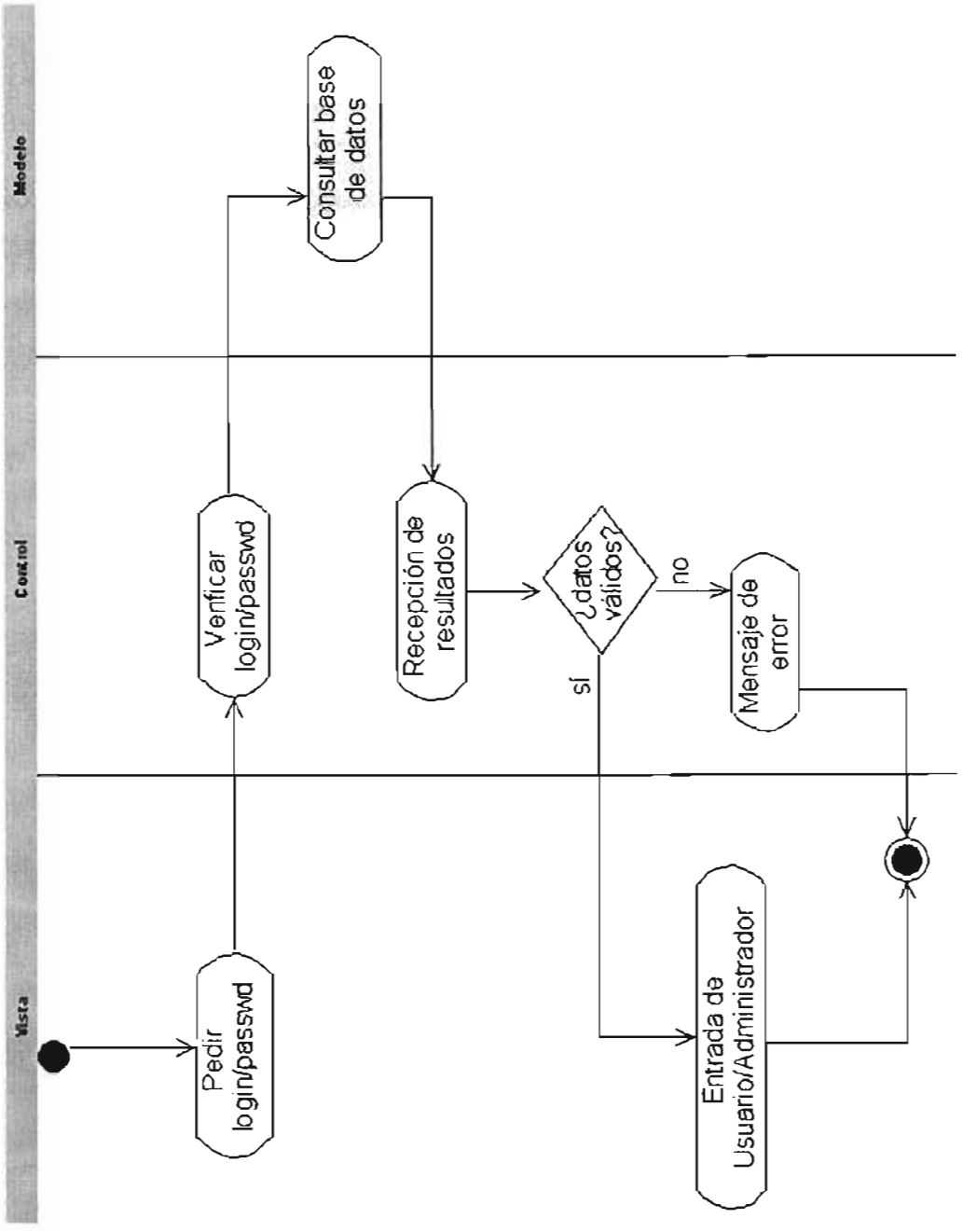


Subir Datos

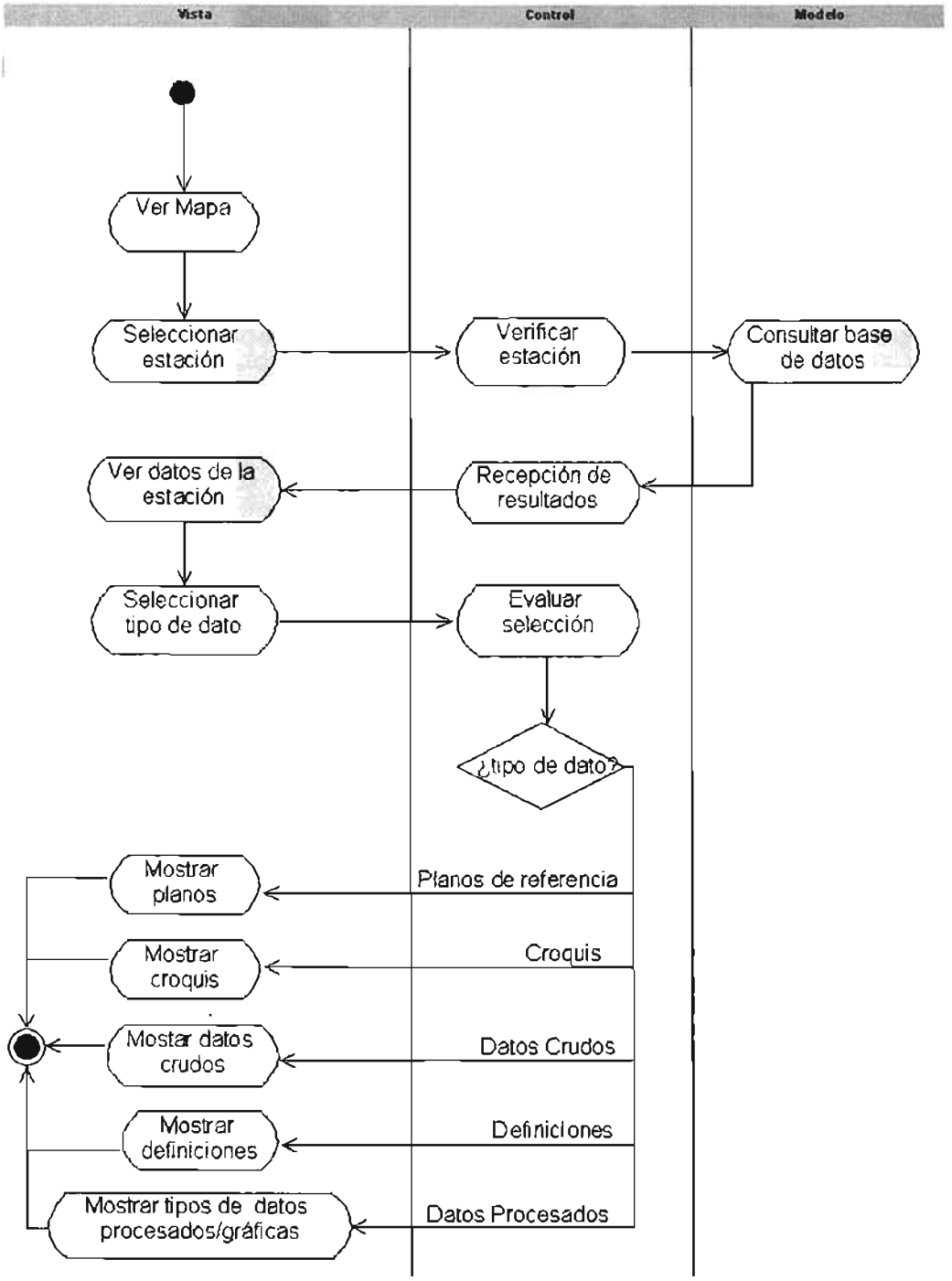

Administrador
: ControladorLogin
: Conexion
: ControladorEstacion
: SubirArchivo
: EjecutaProceso
validar(String, String)
consultar( )
arreglo de datos
OK
doPost(HttpServletRequest, HttpServletResponse)
subir(String)
archivo en servidor
ejecutaSinRetorno(String)
archivo añadido
OK

## 2.1.4 Diagramas de Actividades

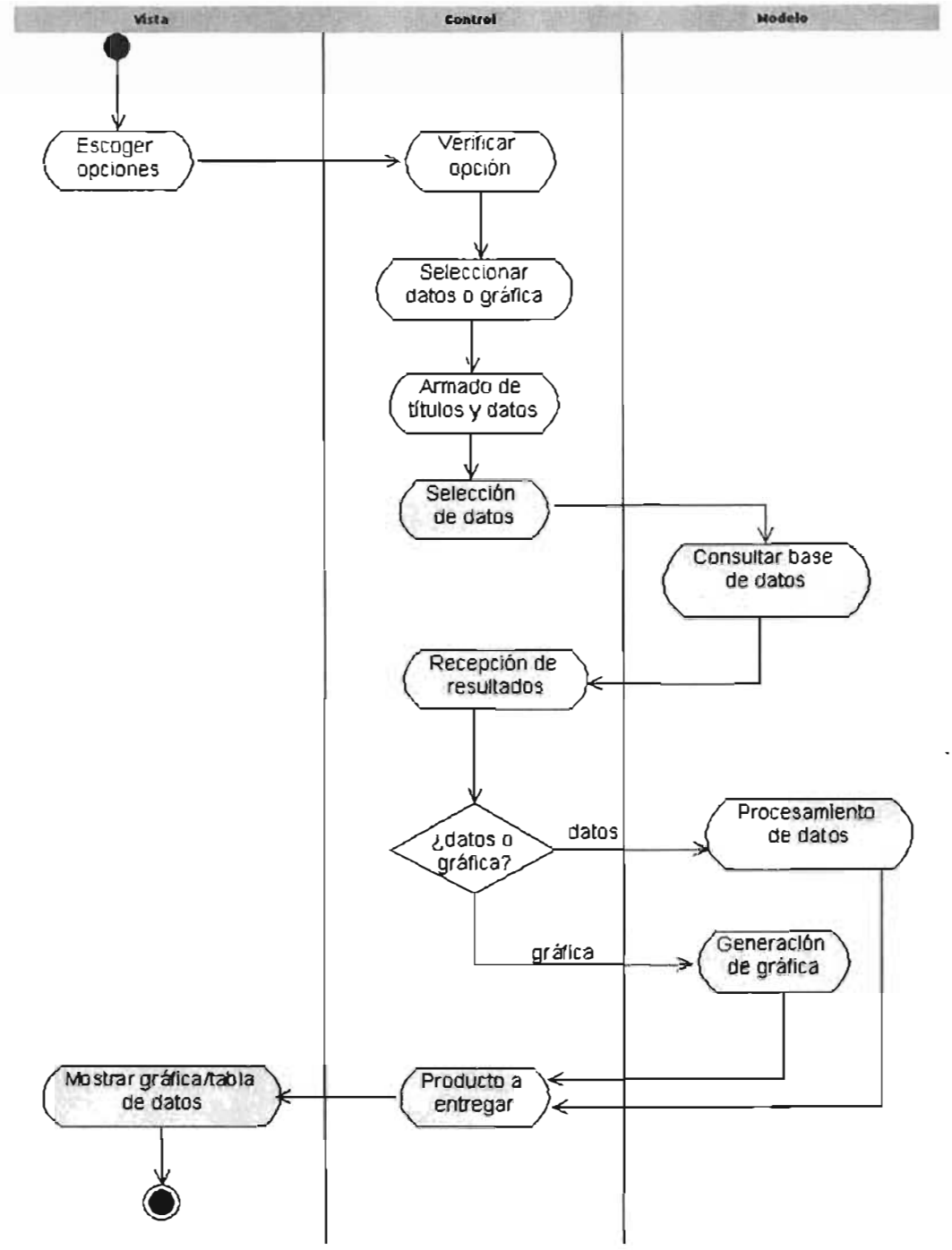
Validación Login



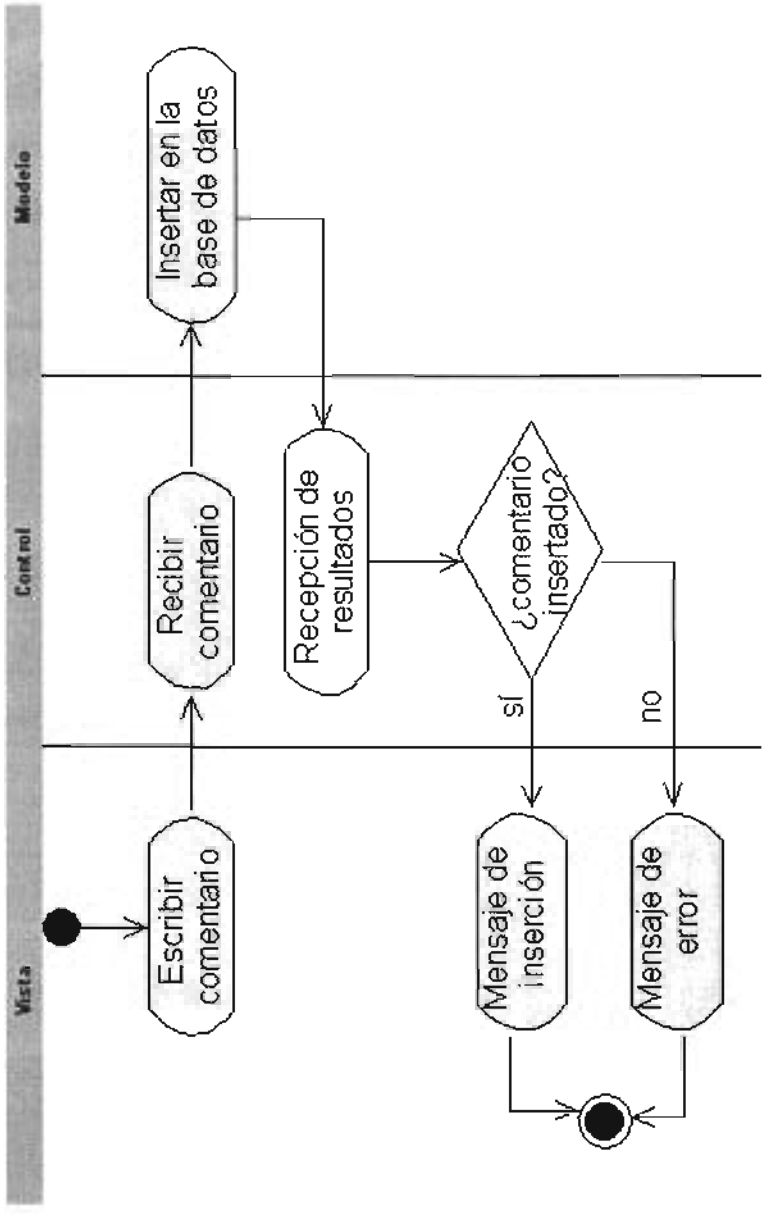
Tipo de Datos



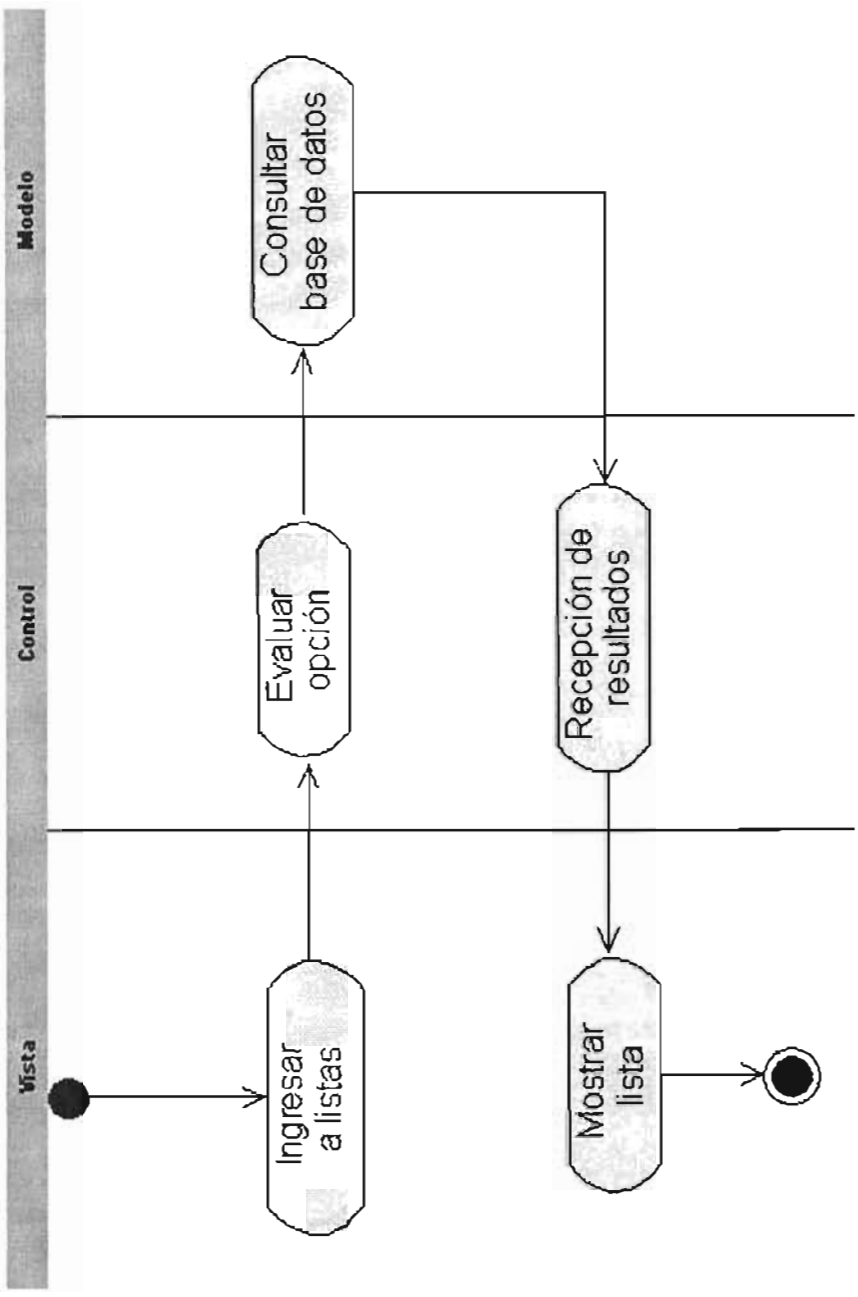
Datos Procesados / Gráficas



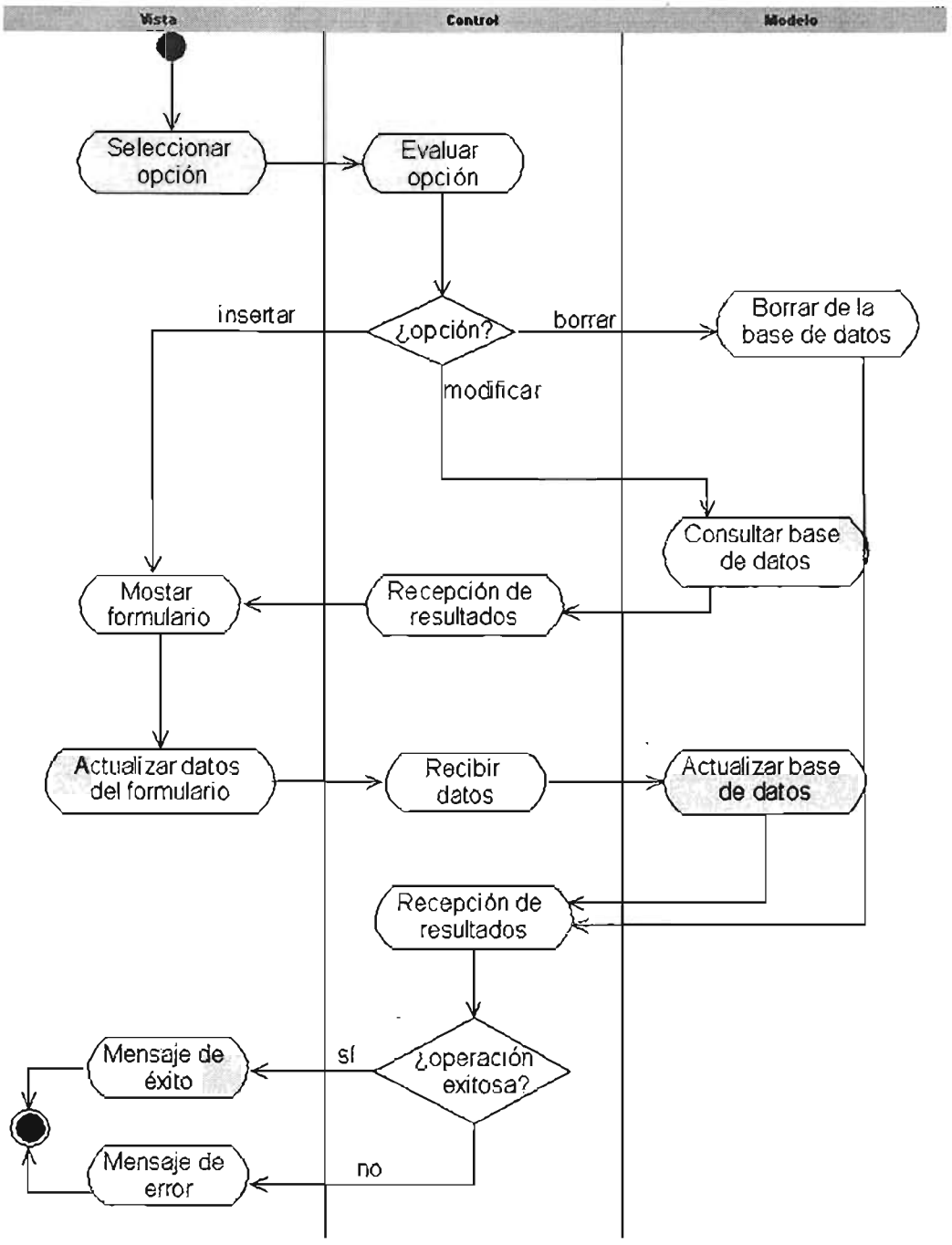
Comentarios Usuario



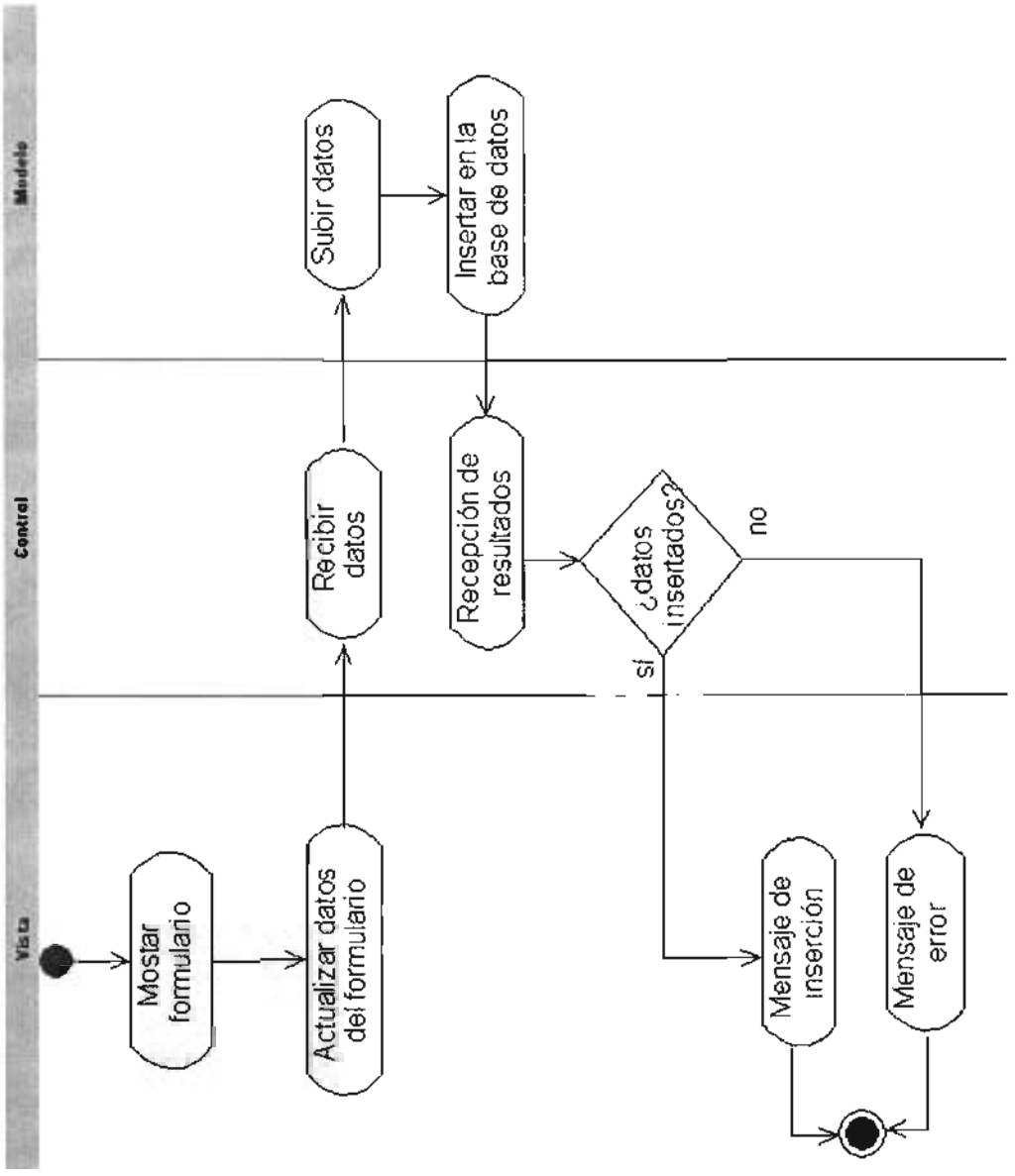
Lista


Modelo
Control
Vista
Consultar base de datos
Evaluar opción
Recepción de resultados
Ingresar a listas
Mostrar lista

Insertar / Eliminar / Modificar

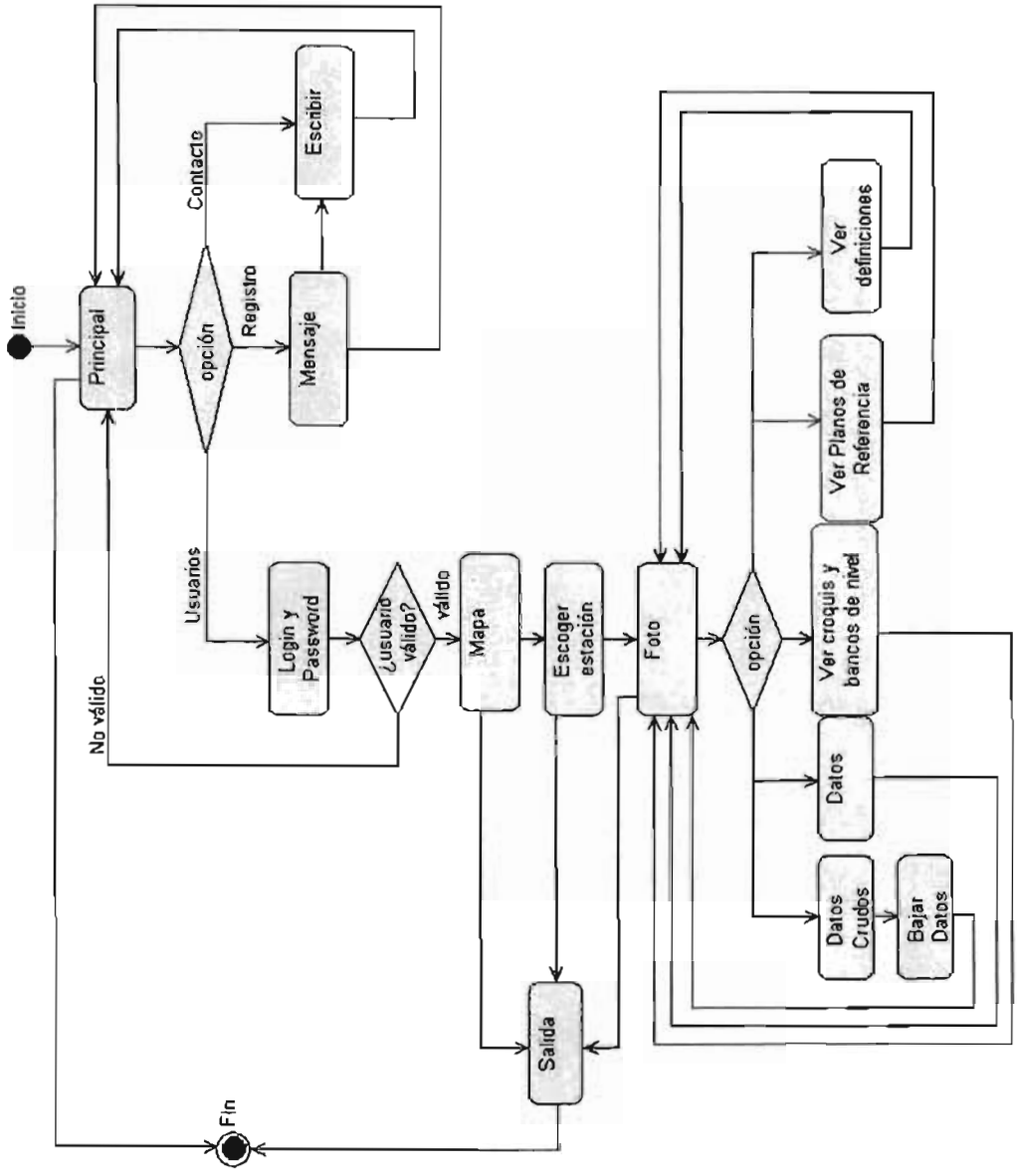


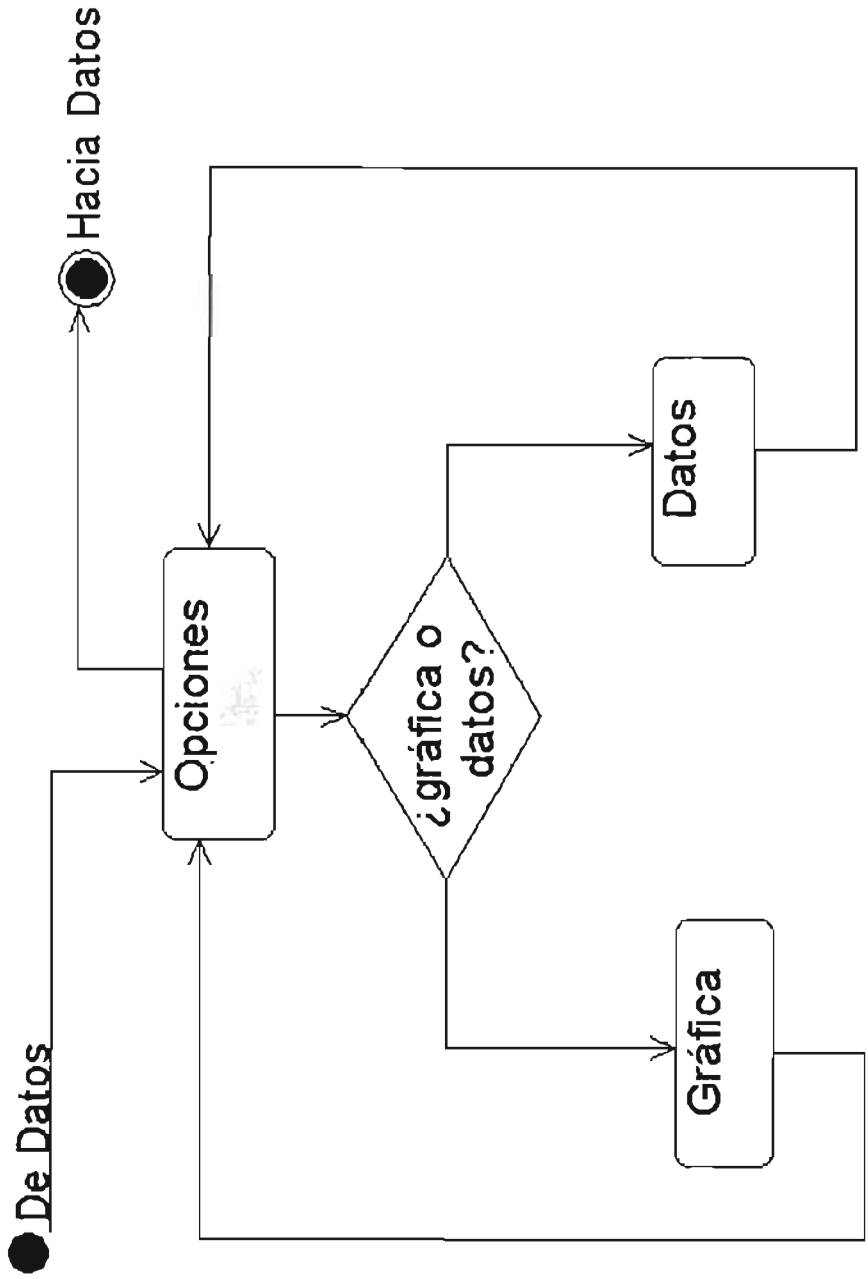
Subir Datos


Vista
Control
Modelo
Mostar formulario
Actualizar datos del formulario
Recibir datos
Subir datos
Insertar en la base de datos
Recepción de resultados
¿datos insertados?
sí
no
Mensaje de inserción
Mensaje de error

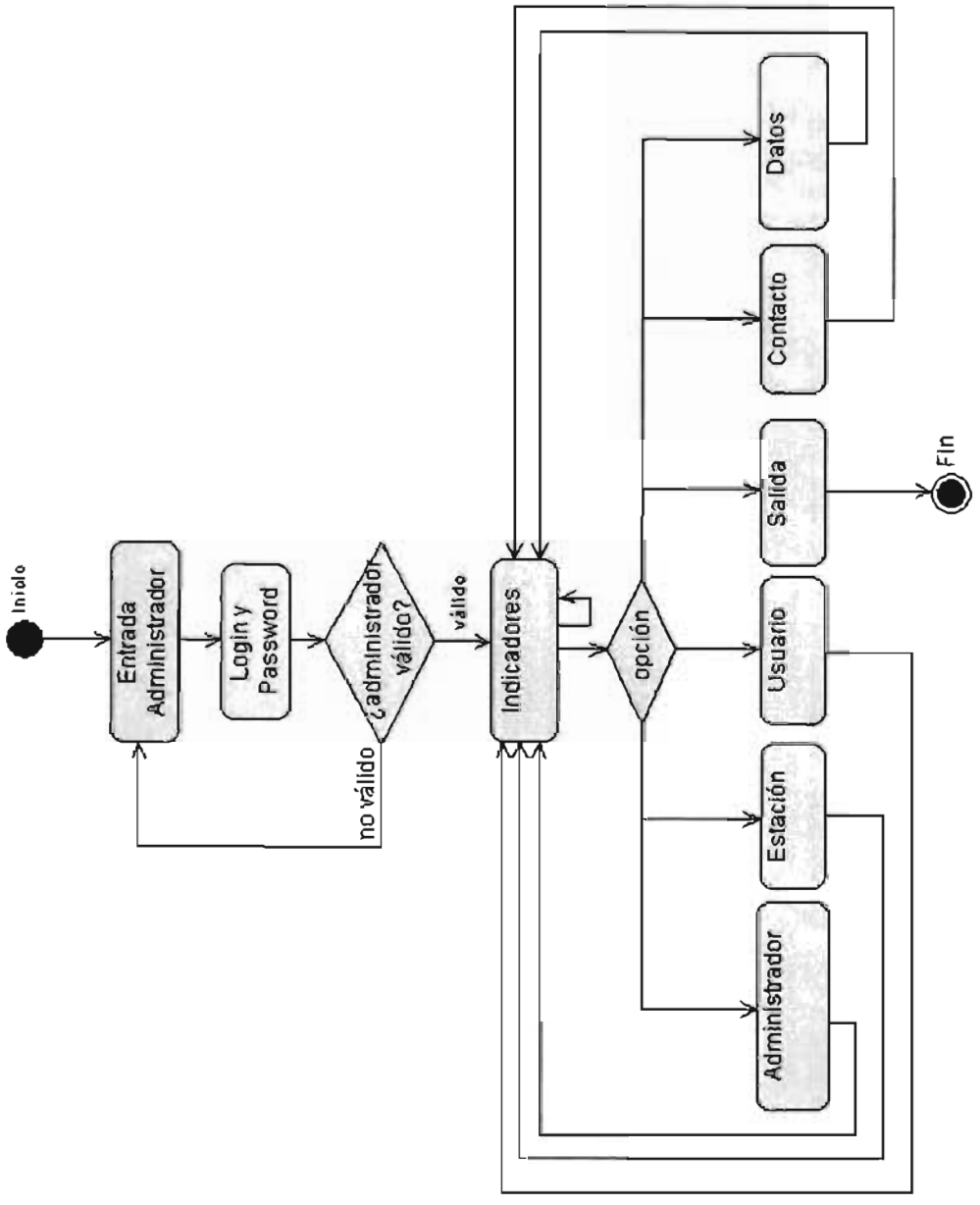
## 2.1.5 Diagramas de Flujo

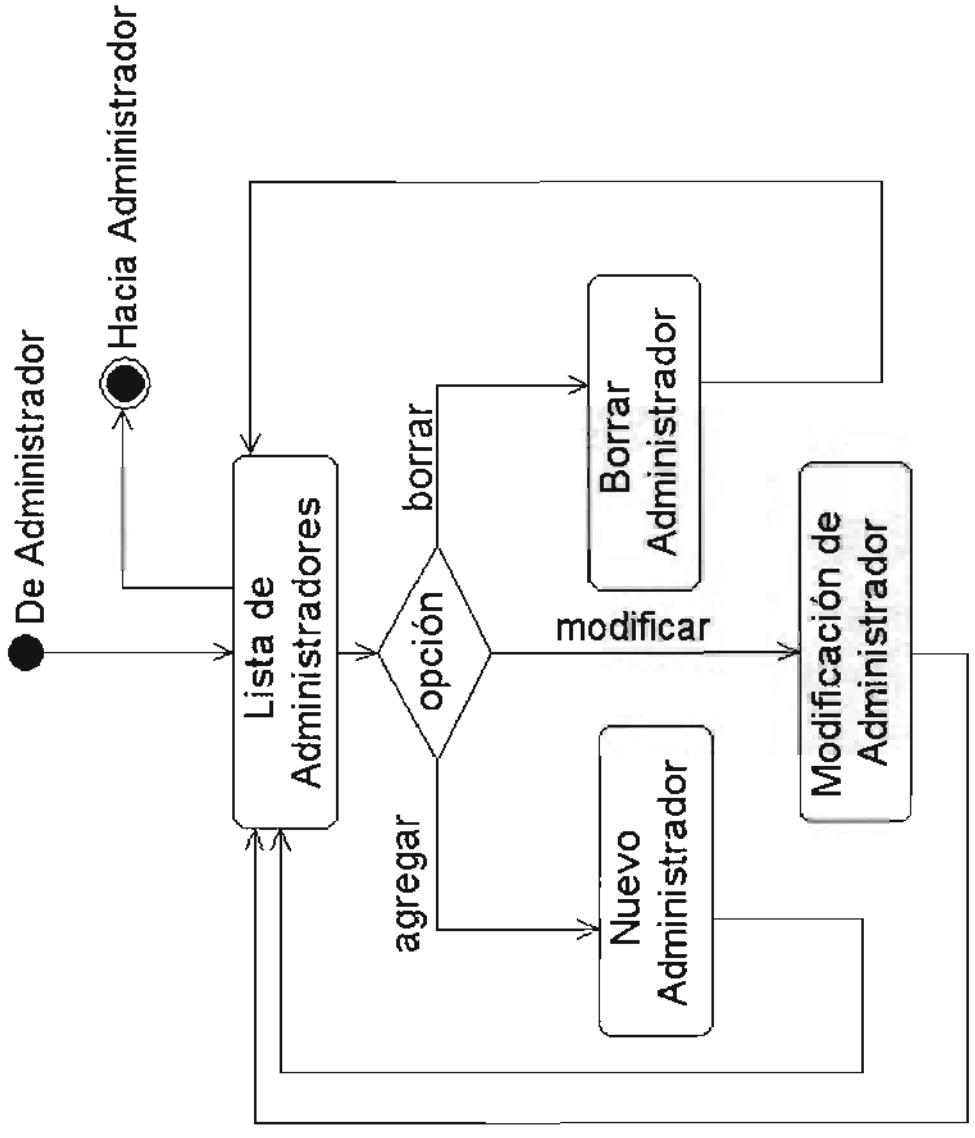
Usuario

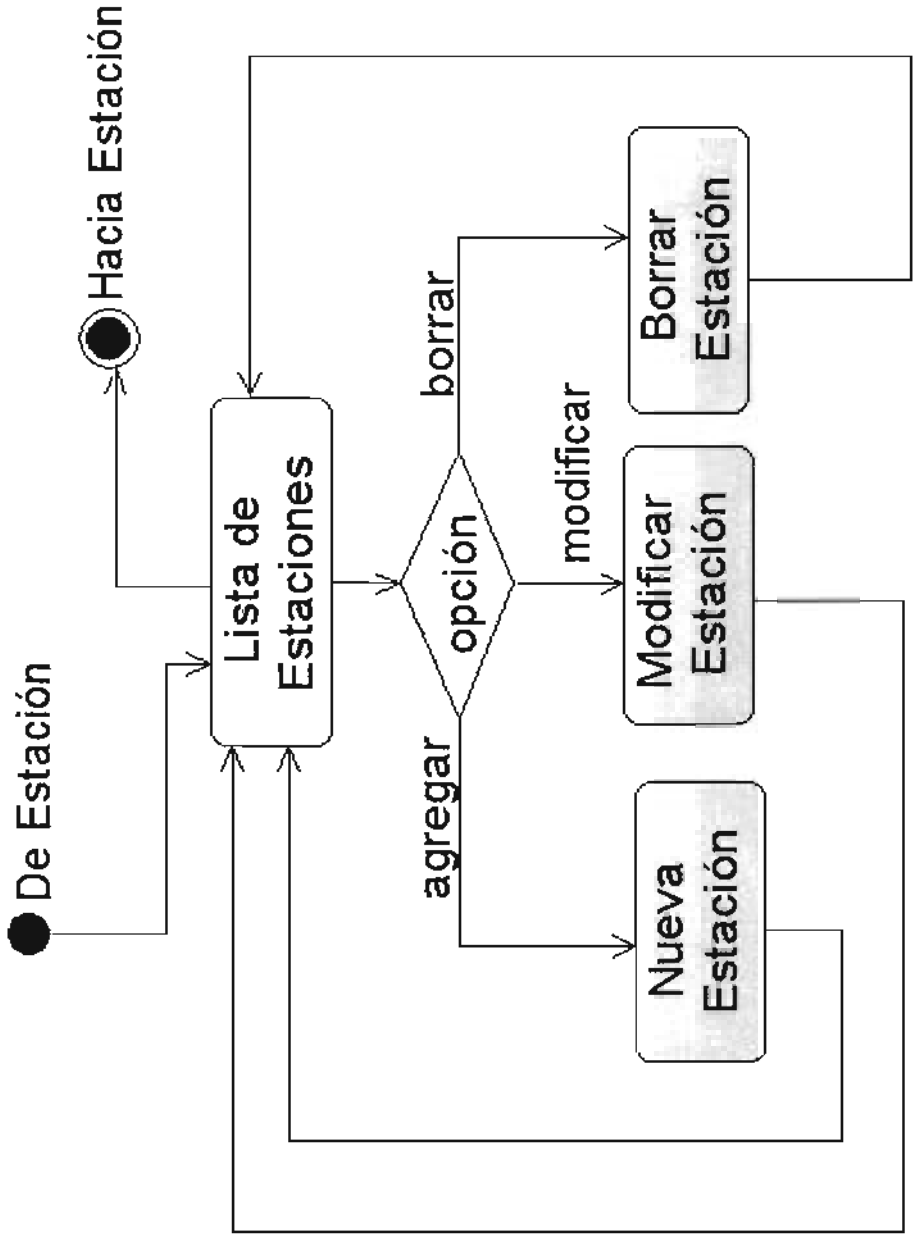


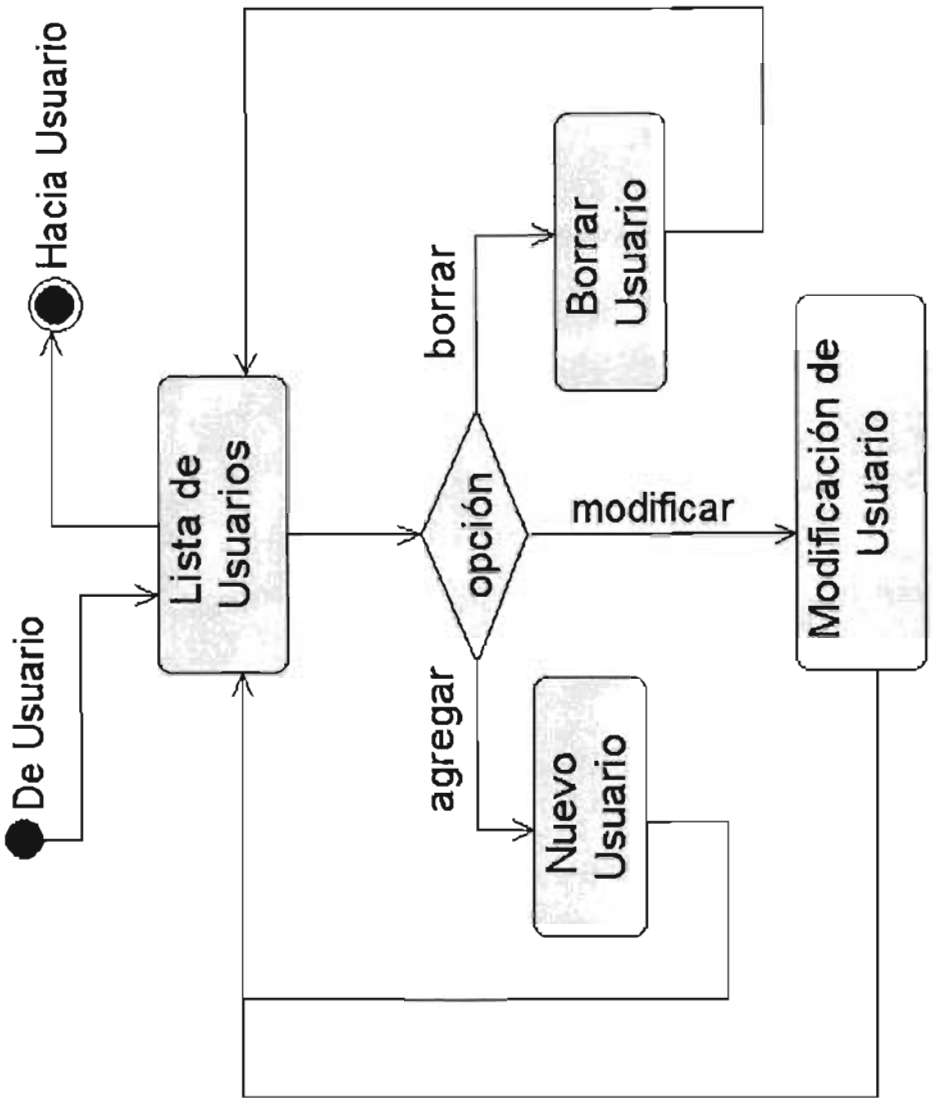

Hacia Datos
Datos
Opciones
¿gráfica o datos?
Gráfica
De Datos

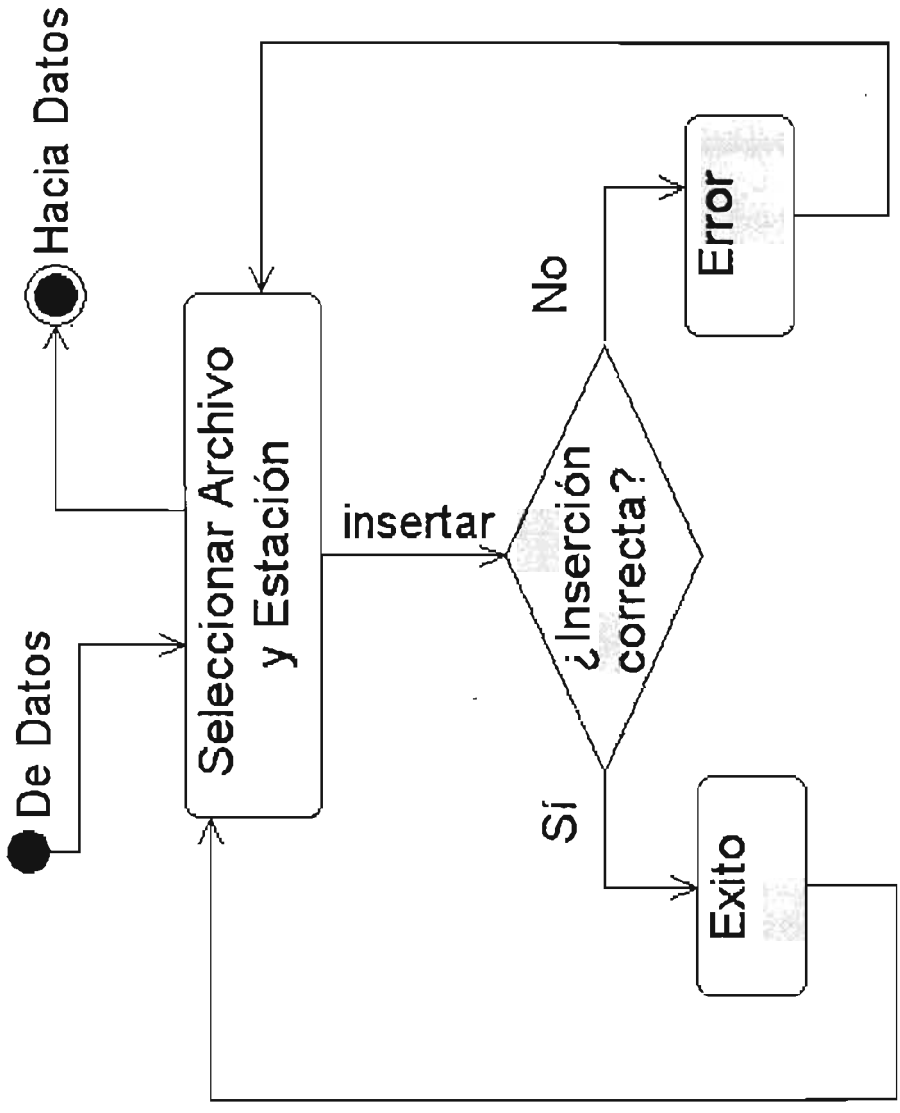
Administrador

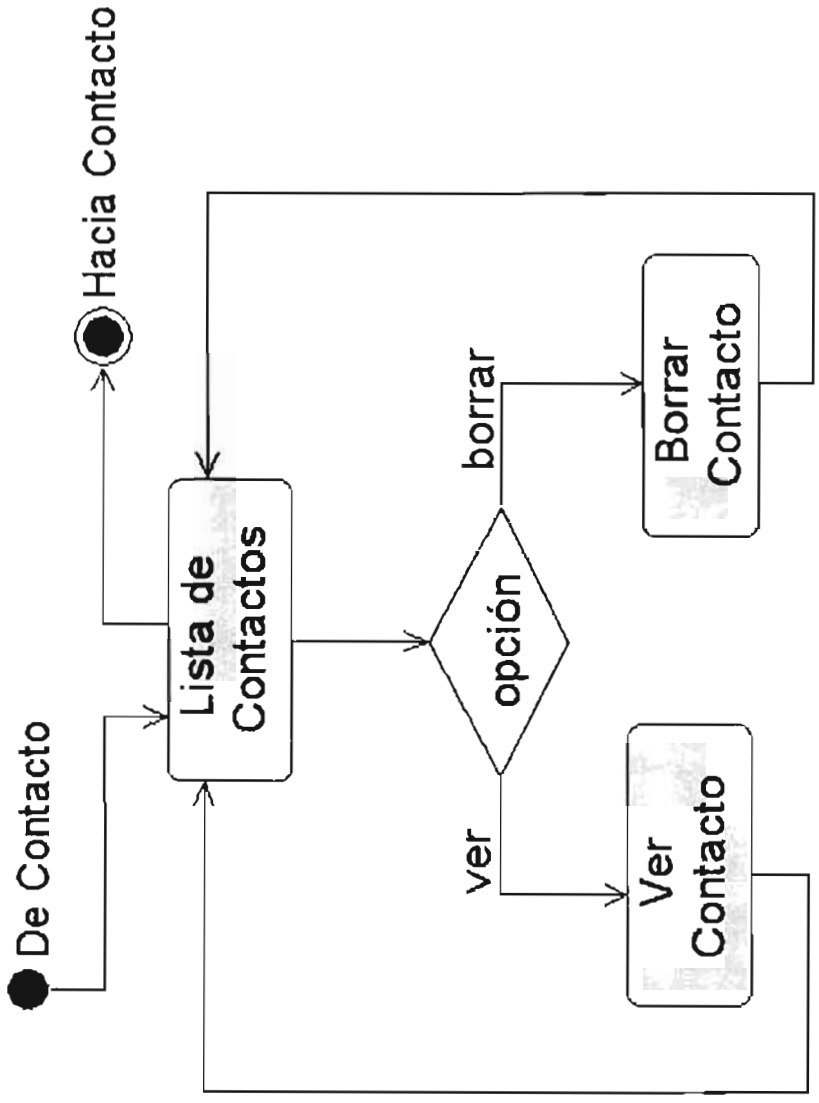



De Administrador
Hacia Administrador
Lista de Administradores
opción
borrar
agregar
modificar
Borrar Administrador
Nuevo Administrador
Modificación de Administrador


Hacia Estación
De Estación
Lista de Estaciones
opción
borrar
modificar
agregar
Borrar Estación
Modificar Estación
Nueva Estación


Hacia Usuario
De Usuario
Lista de Usuarios
opción
borrar
Borrar Usuario
modificar
Modificación de Usuario
agregar
Nuevo Usuario


Hacia Datos
De Datos
Seleccionar Archivo y Estación
insertar
¿Inserción correcta?
No
Sí
Error
Exito


Hacia Contacto
De Contacto
Lista de Contactos
opción
borrar
ver
Borrar Contacto
Ver Contacto

# 2.2 Datos (SQL)

## 2.2.1 Modelo Conceptual

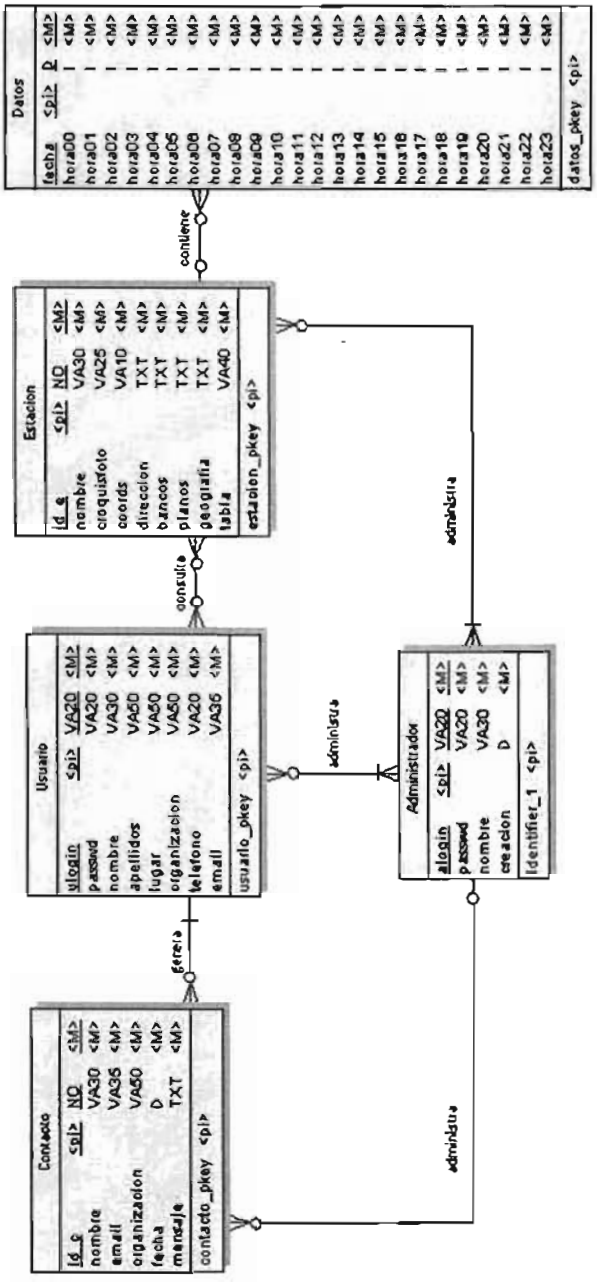



**Figura 2.3 Modelo Conceptual**

## 2.2.2 Modelo Físico

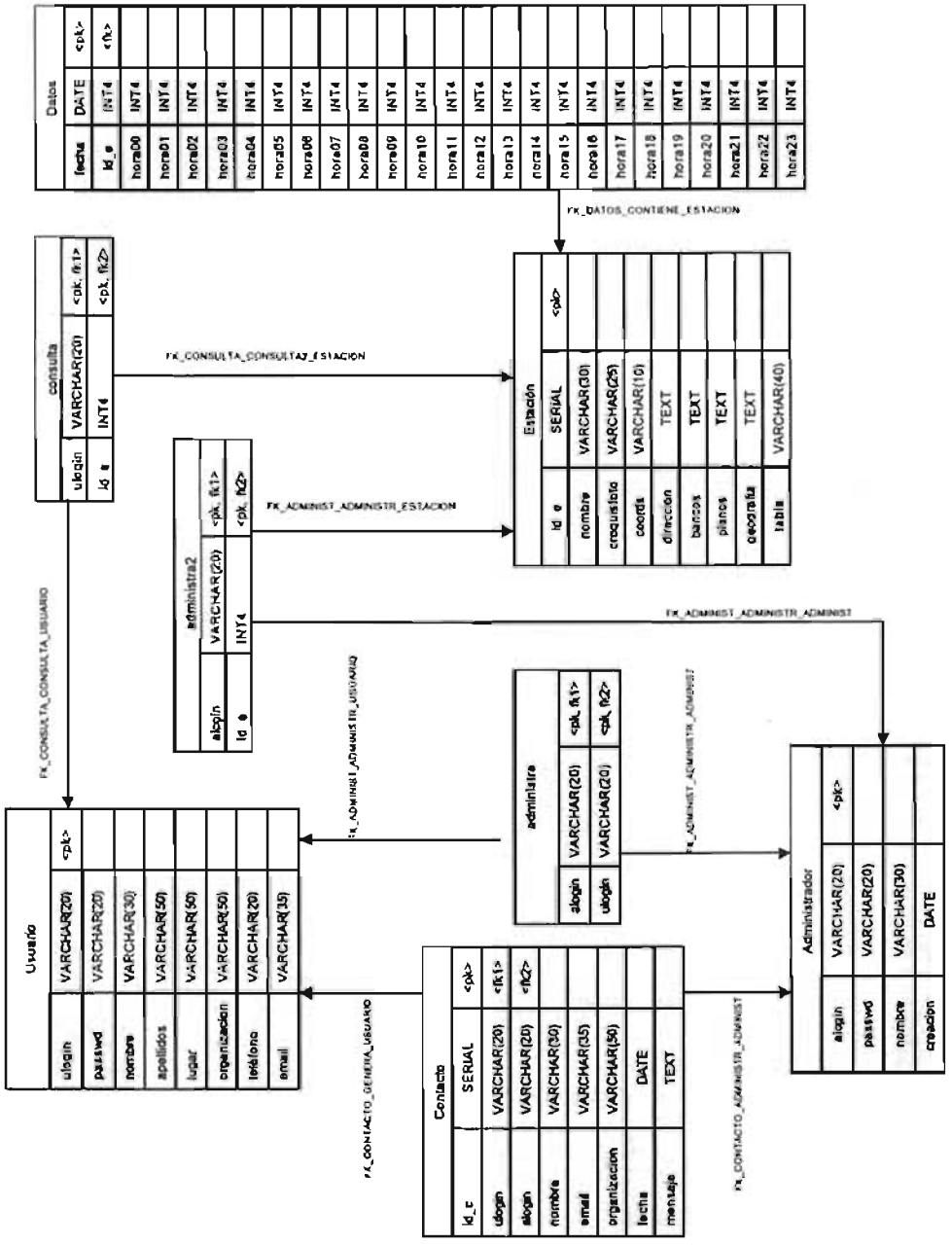



**Figura 2.4 Modelo Físico**

## 2.3 Diccionario de Datos

### 2.3.1 Modelo Conceptual

| Nombre | Tipo de Dato | Longitud |
|---|---|---|
| ulogin | VARCHAR(20) | 20 |
| passwd | VARCHAR(20) | 20 |
| nombre | VARCHAR(30) | 30 |
| apellidos | VARCHAR(50) | 50 |
| lugar | VARCHAR(50) | 50 |
| organizacion | VARCHAR(50) | 50 |
| telefono | VARCHAR(20) | 20 |
| email | VARCHAR(35) | 35 |
| alogin | VARCHAR(20) | 20 |
| creacion | DATE | |
| id_e | NO | |
| croquisfoto | VARCHAR(25) | 25 |
| coords | VA10 | 10 |
| direccion | TXT | |
| bancos | TXT | |
| planos | TXT | |
| geografia | TXT | |
| tabla | VARCHAR(40) | 40 |
| id_c | NO | |
| fecha | DATE | |
| mensaje | TXT | |
| hora00 | INTEGER | |
| hora01 | INTEGER | |
| hora02 | INTEGER | |
| hora03 | INTEGER | |
| hora04 | INTEGER | |
| hora05 | INTEGER | |
| hora06 | INTEGER | |
| hora07 | INTEGER | |
| hora08 | INTEGER | |
| hora09 | INTEGER | |

| | | |
|---|---|---|
| hora10 | INTEGER | |
| hora11 | INTEGER | |
| hora12 | INTEGER | |
| hora13 | INTEGER | |
| hora14 | INTEGER | |
| hora15 | INTEGER | |
| hora16 | INTEGER | |
| hora17 | INTEGER | |
| hora18 | INTEGER | |
| hora19 | INTEGER | |
| hora20 | INTEGER | |
| hora21 | INTEGER | |
| hora22 | INTEGER | |
| hora23 | INTEGER | |

**Tabla 2. 1 Diccionario de datos del Modelo Conceptual**

| Nombre | Entidad |
|---|---|
| usuario_pkey | Usuario |
| administrador_pkey | Administrador |
| estacion_pkey | Estacion |
| contacto_pkey | Contacto |
| datos_pkey | Datos |

**Tabla 2. 2 Llaves Primarias de cada entidad**

## 2.4 Relación entre datos y tablas

Esta relación está expresada como la existente entre todos los datos que se tratan dentro del sistema y los que se almacenan en la base de datos, mediante tablas. En general, puede afirmarse que la mayoría de los datos que se almacenan en la base de datos son necesarios para el funcionamiento del sistema. Este hecho es el que promueve tal almacenamiento.

No obstante, hay gran cantidad de datos en las distintas entidades que están involucradas en el sistema que son exclusivamente de uso informativo para el usuario o administrador. Todos los datos solicitados en los formularios HTML están reflejados en la base de datos. De esta manera puede generarse la relación siguiente:

**Formularios:**

Cambio de Contraseña

loginUsuario → login en la tabla usuario

passwdUsuario → passwd en la tabla usuario

nuevoPasswd → passwd en la tabla usuario

Contacto

nombreContacto → nombre en la tabla contacto

emailContacto → email en la tabla contacto

organizacionContacto → organización en la tabla contacto

mensajeContacto → mensaje en la tabla contacto

Entrada de Usuarios

loginUsuario → login en tabla usuario

passwdUsuario → passwd en tabla usuario

Mapa

estacion → id en tabla estacion

Filtros

lanio → año del atributo fecha en la tabla de la estación seleccionada.

lmes → mes del atributo fecha en la tabla de la estación seleccionada.

Entrada de Administradores

loginAdministrador → login en la tabla administrador

passwdAdministrador → passwd en la tabla administrador

ListaAdministrador

checkboxN → login en la tabla administrador

AltasAdministrador

nombreAdministrador → nombre en la tabla administrador

loginAdministrador → login en la tabla administrador

passwordAdministrador → passwd en la tabla administrador

ModificacionAdministrador

nombreAdministrador → nombre en la tabla administrador

loginAdministrador → login en la tabla administrador

passwordAdministrador → passwd en la tabla administrador

ListaEstacion

checkboxN → id en la tabla estacion

AltasEstacion

nombreEstacion → nombre en la tabla estacion

coordenadasEstacion → coords en la tabla estacion

croquisFotoEstacion → croquisfoto en la tabla estacion

geografiaEstacion → geografia en la tabla estacion

direccionEstacion → direccion en la tabla estacion

bancosEstacion → bancos en la tabla estacion

planosEstacion → planos en la tabla estacion

ModificacionEstacion
nombreEstacion → nombre en la tabla estacion
coordenadasEstacion → coords en la tabla estacion
croquisFotoEstacion → croquisfoto en la tabla estacion
geografiaEstacion → geografia en la tabla estacion
direccionEstacion → direccion en la tabla estacion
bancosEstacion → bancos en la tabla estacion
planosEstacion → planos en la tabla estacion

ListaUsuario
checkboxN → login en la tabla usuario

AltasUsuario
nombreUsuario → nombre en la tabla usuario
apellidosUsuario → apellidos en la tabla usuario
lugarUsuario → lugar en la tabla usuario
organizacionUsuario → organización en la tabla usuario
telefonoUsuario → telefono en la tabla usuario
emailUsuario → email en la tabla usuario
loginUsuario → login en la tabla usuario
passwordUsuario → passwd en la tabla usuario

ModificacionUsuario
nombreUsuario → nombre en la tabla usuario
apellidosUsuario → apellidos en la tabla usuario
lugarUsuario → lugar en la tabla usuario
organizacionUsuario → organización en la tabla usuario
telefonoUsuario → telefono en la tabla usuario
emailUsuario → email en la tabla usuario
loginUsuario → login en la tabla usuario
passwordUsuario → passwd en la tabla usuario

ListaContacto
checkboxN → id en la tabla contacto

ElegirArchivo
estacion → tabla en la tabla estacion

# 3. Diseño del Sistema

## 3.1 Secuencia de Procesos

Las observaciones mareográficas en las costas de México fueron iniciados por el Servicio Geodésico Interamericano y el Departamento de Fotogrametría de la Secretaría de la Defensa Nacional. Las observaciones se iniciaron en Tampico, Tamps. en el año de 1942 siguiendo en Coatzacoalcos y Progreso en 1946; Acapulco 1949; Guaymas-La Paz, Ensenada en 1950; Veracruz-Salinas Cruz, Mazatlán en 1952.

Los objetivos del Servicio Mareográfico desde su creación han sido mantener en operación la red básica de mareógrafos y registrar las variaciones continuas del nivel del mar a largo plazo.

El inicio de las observaciones mareográficas en las costas de México se derivó de la necesidad que se tuvo para dar cota con relación al nivel medio del mar a las elevaciones del territorio nacional y ciudades, puntos de partida de las triangulaciones geodésicas (tercera coordenada) en todo levantamiento cartográfico, etc. La construcción de Obras marítimas (muelles, escolleras, edificios próximos al mar, termo eléctricas, etc.) tenían la necesidad de contar con información mareográfica confiable para realizar los diferentes estudios y proyectos.

Por medio de las alturas horarias de la marea y mediante programas desarrollados en este Instituto de Geofísica y utilizando la computadora B—7800 de la UNAM año con año se han publicado bs pronósticos de la marea en forma numérica en la revista Datos Geofísicos serie A Océano Pacífico y "Puertos del Golfo" de México y Mar Caribe". Así mismo se han publicado estos pronósticos en forma de calendarios gráficos tanto para los principales puertos del Océano Pacífico como para los del Golfo de México y el Mar Caribe.

Hoy en día, el Servicio Mareográfico cuenta con la adquisición de los siguientes datos:

- Nivel del Mar
- Temperatura del agua
- Salinidad del agua
- Temperatura ambiente

**EQUIPO MAREOGRAFICO**

En la actualidad el Servicio Mareográfico cuenta con varios tipos de mareógrafos, desde los mareógrafos estándar que todavía se mantienen en operación, mareógrafos Fisher and Porter análogo-digitales descontinuados a la fecha y los mareógrafos Stevens modelos 7031 y 7032 también digitales y los modelos más nuevos de gráfica continua tipo A modelo 71.

El mareógrafo automático estándar de gráfica continua, actúa por medio de un flotador que sube y baja dentro de un tubo de hierro 12 excesivo horizontal y vertical del flotador, por medio de orificios laterales. El movimiento de subida y bajada del flotador y su alambre hace girar el tornillo sin fin del lápiz trazador de la curva mareográfica. El papel se mueve de un rollo a otro sobre el tambor principal del aparato por medio del mecanismo del reloj motor a paso uniforme. El movimiento combinado del lápiz y el papel describe la gráfica continua de la subida y bajada de la marea. Cada hora el lápiz marca la hora por conducto de un disparador especial actuado por el otro reloj.

Registro analógico.-Digital (Fisher and Porter), este es el nombre técnico del mareógrafo de cinta perforadora. Este nuevo tipo de registro automático de niveles de la marea funciona con un flotador con perforaciones en clave en cinta de papel. El nivel del agua se perfora en la cinta al

centésimo de pie, a intervalos seleccionados, regulados por un pequeño reloj de control. La cinta perforadora esta en clave binario-decimal que puede interpretarse visualmente o sea puede pasar por una máquina lectora. El sistema de medición consiste en un flotador que actúa en un tubo de metal de 4 pulgadas de diámetro, un cable de flotador o alambre, un extremo del cual está amarrado en el flotador y un tambor en un eje conectado a los discos del registro, al cual está unido con seguridad el otro extremo del alambre. Mientras el flotador sube y baja por la marea, la rueda o tambor del flotador hace girar a los discos del registro unidos por el eje de entrada. El movimiento del eje de entrada se controla por un freno de emergencia que controla el movimiento excesivo en caso de que se reventara el alambre del flotador, un dispositivo de flexión reversible o resorte de contrapeso provee tensión constante y energía entrada y un dispositivo de resortes de doble torsión para contrarrestar a ocular el movimiento perdido del acoplador mientras actúa el mecanismo perforado del papel que marca la lectura del nivel del agua.

**TRABAJO DE CAMPO**

La estación mareográfica de control, es una que se establece para que pueda funcionar por varios años con el objeto de obtener una serie continua de datos mareográficos. Como los registros de tal estación constituyen datos básicos para uso presente y futuro, es importantísimo que tanto la instalación como operación y mantenimiento se organicen teniendo presente el objetivo fundamental de obtener datos de la óptima calidad y precisión posible.

**LOCALIZACIÓN**

Debe tomarse un cuidado especial en la selección de un sitio para una estación de control. Si es posible debe de haber una profundidad no menos de 1.50 m bajo la marea más baja probable. Cuando la determinación del nivel medio del mar es un objetivo importante, la estación debe localizarse en la costa abierta al mar o en bahía de bocana con amplio acceso al mar. Al seleccionar el sitio para una estación de control, es indispensable tomar en cuenta la limpieza del área en general, evitar sitios en áreas altamente congestionadas por barcos, pasajeros y carga, en virtud de que hace que el espacio para la estación sea sumamente restringido e incomodo o inadecuado para una buena estación. Además en tales áreas tanto el aparato registrador como la regla

de mareas están sujetos a daños por barcos que atracan y cargan y descargan constantemente en el muelle cerca de la caseta y hacen que la nivelación entre la regla y los bancos de control, sea especialmente difícil.

**POZO AMORTIGUADOR**

El pozo amortiguador, no es otra cosa que el paga del flotador. Este pozo por lo general consiste de un tubo de hierro, asbesto, fibra de vidrio, o P.V.C., etc. que se extiende desde el piso de la caseta o del muelle hacia el fondo del mar, hasta más debajo de la marea más baja probable en la localidad.

El agua entra al pozo por una abertura en el fondo y sube hasta el nivel más alto que pueda subir el agua por afuera del tubo. El objeto del tubo es proteger el flotador, amortiguar el movimiento del agua causado por el oleaje y así y proporcionar un medio para medir con más precisión, la variación del nivel del agua por efectos de las mareas.

**CASETA DEL MAREÓGRAFO**

Si se dispone de espacio suficiente, la caseta debe ser cuadrada y tener 2 metros por lado y 2.40 metros de altura, una ventana y puerta que abra hacia fuera. En general la caseta se coloca directamente sobre la parte superior del pozo del flotador, de manera que el alambre del flotador pueda conectarse directamente al mareógrafo.

**REGLA DE MAREAS**

Cada estación primaria debe disponer de una regla de mareas para proveer un nivel o cero de referencia, el cual será controlado por bancos de nivel fijos en Tierra firme y por nivelaciones a dichos bancos para que en caso de que la regla sea destruida o deteriorada por las condiciones ambientales se instale otra. El nuevo cero por medio de las nivelaciones diferenciales de los barcos de nivel será controlado y se aplican las correcciones necesarias a las lecturas registradas en los registros mareográficos.

**BANCOS DE NIVEL**

Parte esencial de la estación para el estudio de las mareas es la instalación de una red de marcas de nivelación a las cuales se puedan referir finalmente los registros de las mareas. Una marca de nivelación o banco de nivel es un punto definido en un objeto más o menos permanente. Se deben establecer Bancos de Nivel con cota referida al Nivel Medio del Mar (NMM) en cada estación.

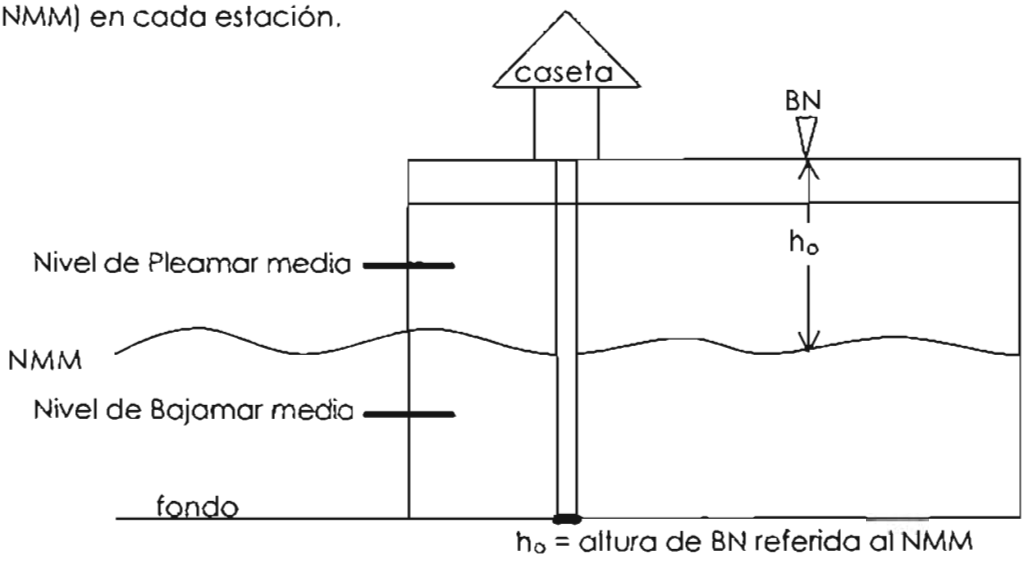



**Figura 3.1 Equipo Mareográfico**

**PREPARACION DE MAREOGRAMAS**

- Trazado de línea de referencia a lo largo del mareograma.
- Subdivisión de horas sobre la línea de referencia en base a las marcas sobre el mareograma de los tiempos de lecturas de regla de mareas realizadas por el observador.
- Proyección sobre la curva de marea de la subdivisión de horas realizada en el paso anterior, para determinar la altura de marea correspondiente.
- Determinación de las pleamares y bajamares sobre la curva de marea.
- Cálculo del valor preliminar del mareograma. (cálculo de la altura de marea correspondiente a la línea de referencia a partir de las lecturas de regla realizadas por el observador y las realizadas con la escala correspondiente del mareograma).

**LECTURA DE ALTURAS HORARIAS ADEMÁS DE TIEMPO Y ALTURA DE PLEAMARES Y BAJAMARES**

- Estimación del valor de las alturas horarias mediante la escala del mareograma correspondiente y vaciado en los formatos correspondientes.
- Estimación de la altura y tiempo de las pleamares y bajamares del mareograma y vaciado en el formato correspondiente.
- Cálculo de los planos de marea del mareograma correspondiente, en base a las alturas horarias y las alturas de las pleamares y bajamares .
- Captura de las alturas horarias y de los valores mensuales del nivel medio del mar y de los planos de mareas.

**PRONÓSTICO DE MAREAS**

- Cálculo de constantes armónicas utilizando el software SLPR2.
- Realización de control de calidad de alturas horarias.
- Cálculo del pronóstico de alturas horarias con el software SLPR2.
- Cálculo del pronóstico de tiempos y alturas de pleamares y bajamares con el software SLPR2.
- Cambio de formato de la salida del pronóstico de pleamares y bajamares al formato utilizado en la publicación.

**DATOS DEL NIVEL DEL MAR**

En cada estación se tienen lecturas de las alturas horarias, esto es, de cada hora. Si se tienen los datos de un año completo, se pueden calcular y obtener pronósticos de otro año. Los pronósticos que se obtienen pueden resultar muy exactos.

Las gráficas que se obtienen son aproximadamente:

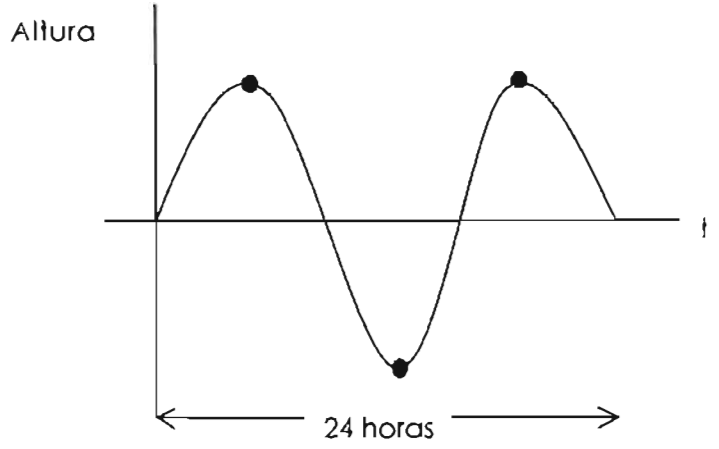



**Figura 3.2 Gráfica aproximada**

Lo más importante son los máximos y los mínimos. De estos se sacan promedios diarios, mensuales y anuales.

**Definiciones**

**Altura Máxima Registrada**: nivel más alto registrado en la estación por efectos de algún tsunomi o ciclón.

**Pleamar Máxima Registrada:** nivel más alto registrado debido a las fuerzas de marea periódica, o también a que tengan influencia sobre las mismas los efectos de condiciones meteorológicos.

**Nivel de Pleamar Media Superior (MHHW)**: promedio de la más alta de las dos pleamares diarias, durante el período considerado en cada estación.

**Nivel de Pleamar Media (MHW)**: promedio de todas las pleamares durante el período considerado en cada estación. Cuando el tipo de marea es diurna, este plano se calcula tomando el promedio de la pleamar más alta diaria, lo que equivale a que la pleamar media en este caso es el mismo que la pleamar madia superior.

**Nivel Medio del Mar**: promedio de las alturas horarias durante el periodo considerado en cada estación.

**Altura Mínima Registrada**: nivel más bajo registrado en la estación por efecto de algún tsunami.

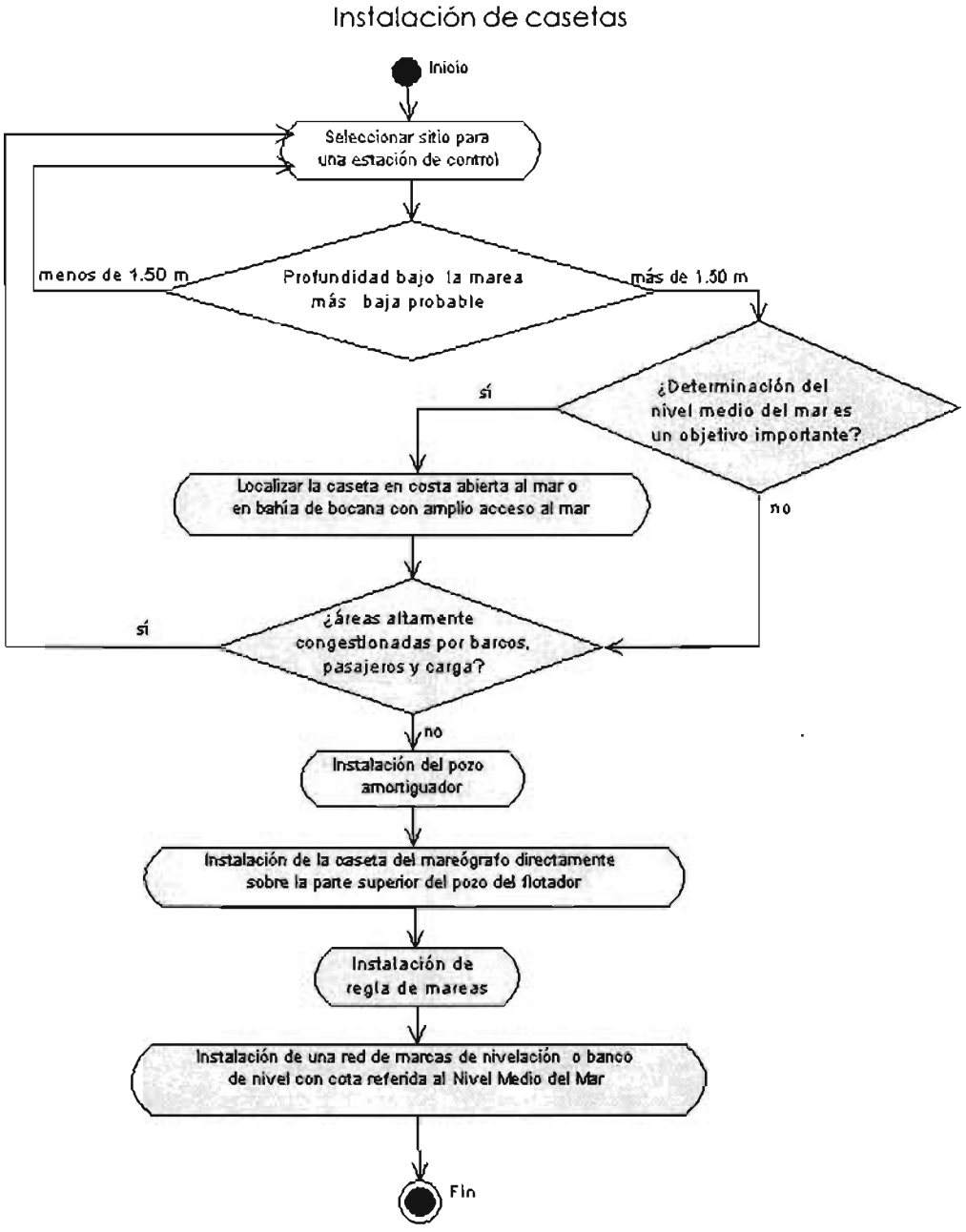
**Bajamar Mínima Registrada**: nivel más bajo registrado debido a las fuerzas de marea periódica, o también que tengan influencia sobre las mismas los efectos de condiciones meteorológicas.

**Nivel de Bajamar Media Inferior (MLLW)**: promedio de las más baja de las dos bajamares diarias, durante el período considerado en cada estación. Este plano es el que se utiliza como plano de referencia para el pronóstico de mareas en la costa del Pacífico y Golfo de California.

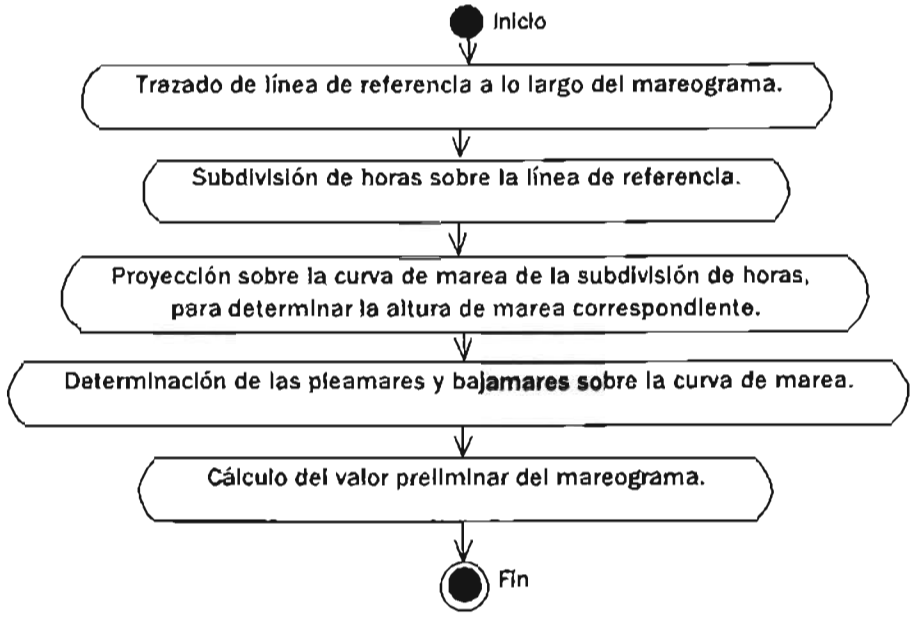
**Nivel de Bajamar Media (MLW)**: promedio de todas las bajamares durante el período considerado en cada estación. Cuando el tipo de marea es diurno este plano se calcula tomando el promedio de la bajamar más baja diaria, lo que equivale en este caso a la bajamar media es lo mismo que la bajamar media inferior.

**Nivel de Media Marea (MTL)**: plano equidistante entre la pleamar media y bajamar media, se obtiene promediando estos dos valores.

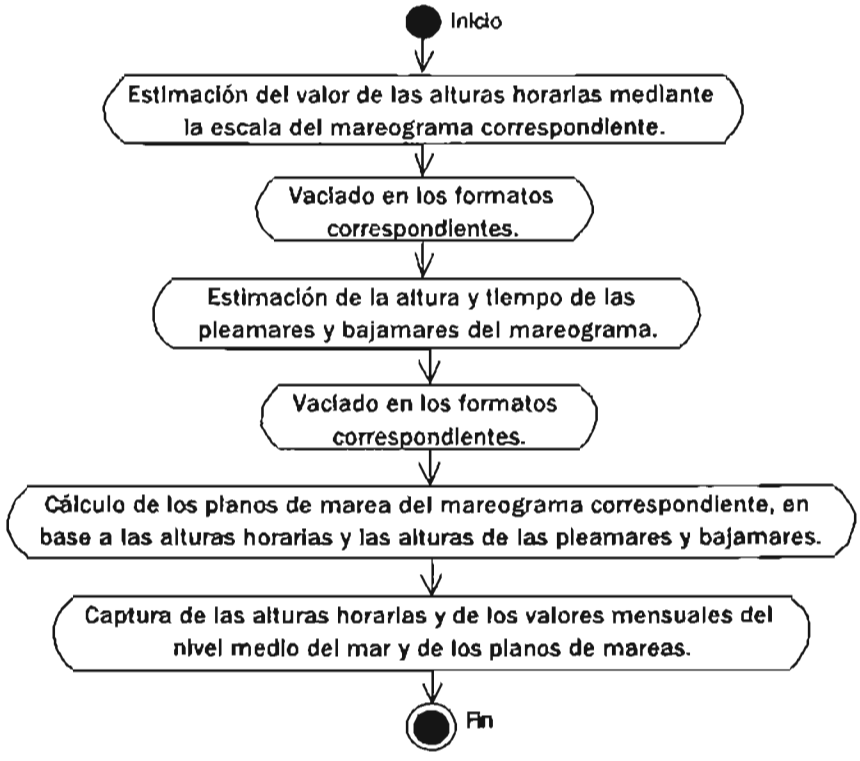
## 3.2 Diagrama de Flujo



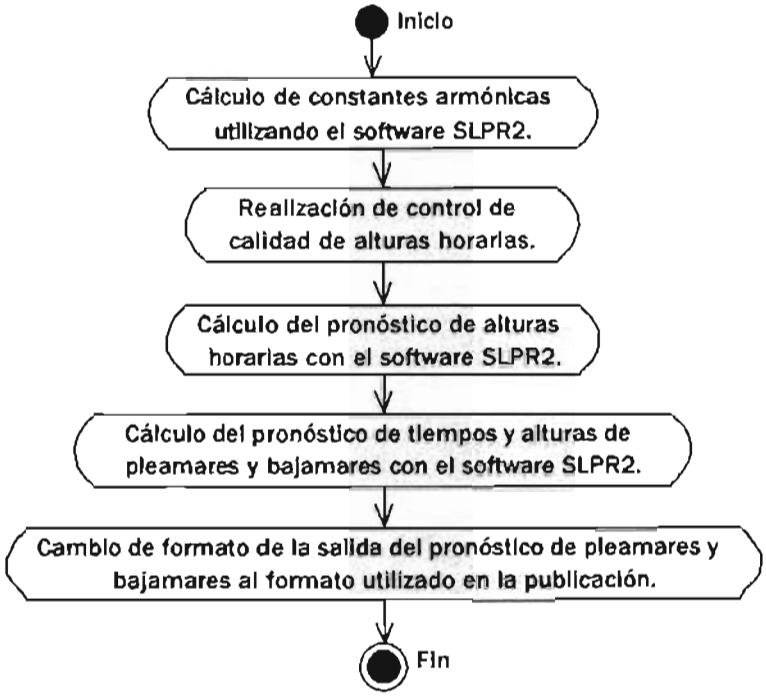
Mareogramas




Lectura de alturas horarias, de tiempo y altura de pleamares y bajamares



Pronóstico de Mareas



# 3.3 Secuencia de Pantallas

## 3.3.1 Zona de Usuarios

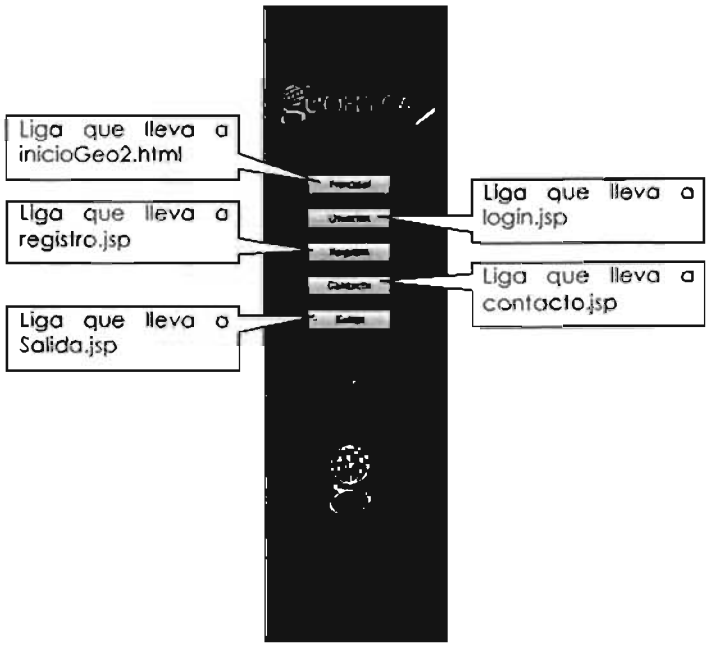
**Nombre:** menu.html

**Página Anterior:** Ninguna

**Página Siguiente:** Ninguna

**Descripción:** Menú de opciones

**Pantalla:**



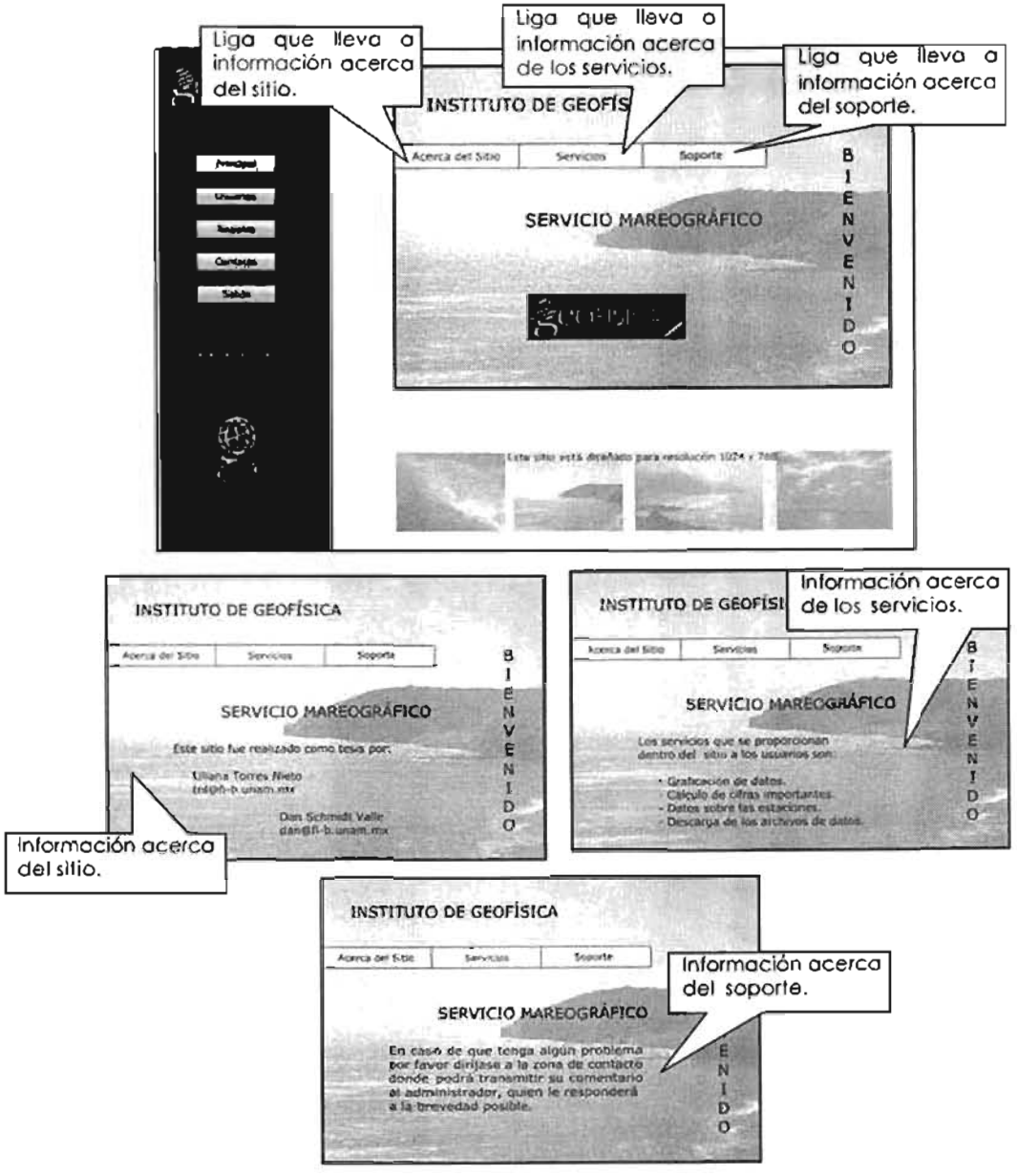

**Nombre:** inicioGeo2.html

**Página Anterior:** Ninguna

**Página Siguiente:** login.jsp
registro.jsp
contacto.jsp
Salida.jsp

**Descripción:** Página principal o presentación del sitio. En esta página, se brinda una breve descripción de los realizadores del sitio, de los servicios prestados y de soporte.

**Pantalla:**


Liga que lleva a información acerca del sitio.
Liga que lleva a información acerca de los servicios.
Liga que lleva a información acerca del soporte.
INSTITUTO DE GEOFÍS
Acerca del Sitio
Servicios
Soporte
SERVICIO MAREOGRÁFICO
BIENVENIDO
INSTITUTO DE GEOFÍSICA
Este sitio fue realizado como tesis por:
Información acerca del sitio.
Información acerca de los servicios.
Los servicios que se proporcionan dentro del sitio a los usuarios son:
- Graficación de datos.
- Cálculo de cifras importantes.
- Datos sobre las estaciones.
- Descarga de los archivos de datos.
Información acerca del soporte.
En caso de que tenga algún problema por favor diríjase a la zona de contacto donde podrá transmitir su comentario al administrador, quien le responderá a la brevedad posible.

**Nombre:** login.jsp

**Página Anterior:** inicioGeo2.html
registro.jsp
contacto.jsp

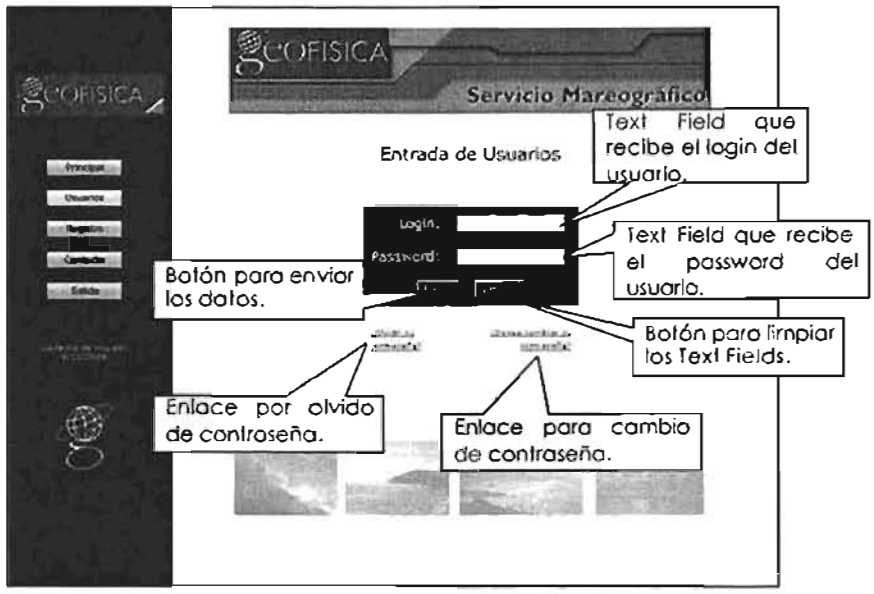
**Página Siguiente:** login.jsp
olvido.jsp
cambioPasswd.jsp
Error.jsp
registro.jsp
contacto.jsp
Salida.jsp
Mapa.jsp

**Descripción:** Página que permite el ingreso de los usuarios autorizados al sitio.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| loginUsuario | Text Field | |
| passwdUsuario | Text Field | |
| Enviar | Botón | |
| Limpiar | Botón | |
| paginaAnterior | Hidden | |

**Pantalla:**



**Nombre:** olvido.jsp
**Página Anterior:** login.jsp
**Página Siguiente:** Ninguna
**Descripción:** Página que provee información sobre el procedimiento a seguir en caso de olvido de contraseña.

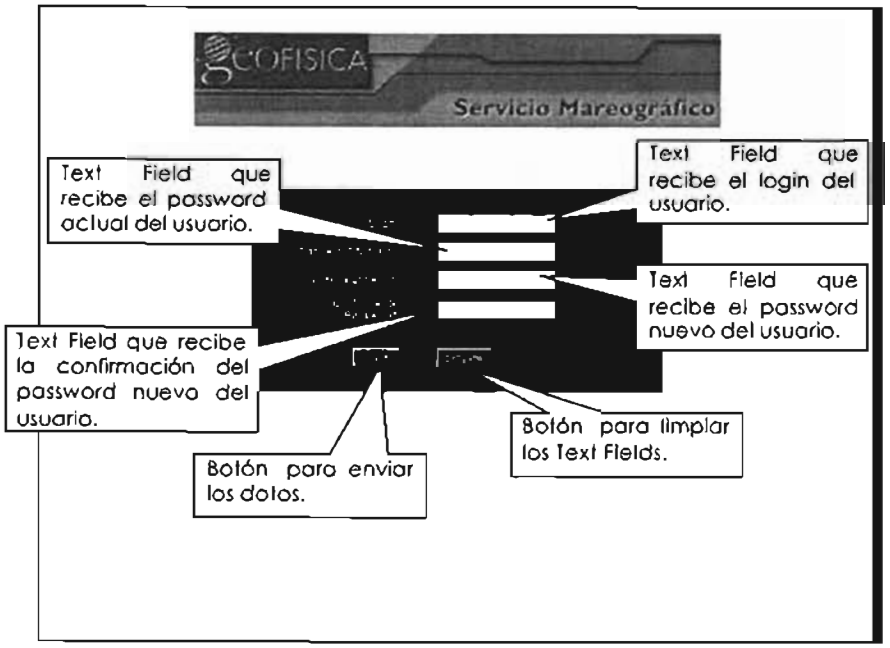
**Pantalla:**

**Nombre:** CambioPasswd.jsp
**Página Anterior:** login.jsp
**Página Siguiente:** passwdCambiado.jsp
PaginaDeError.jsp
**Descripción:** Página que permite el cambio de contraseña de los usuarios.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| loginUsuario | Text Field | 20 |
| passwdUsuario | Text Field | 20 |
| nuevoPasswd | Text Field | 20 |
| confirmacionPasswd | Text Field | 20 |
| Enviar | Botón | |
| Limpiar | Botón | |

**Pantalla:**



**Nombre:** passwdCambiado.jsp
**Página Anterior:** CambioPasswd.jsp
**Página Siguiente:** Ninguna
**Descripción:** Página que informa el éxito en el cambio de contraseña.

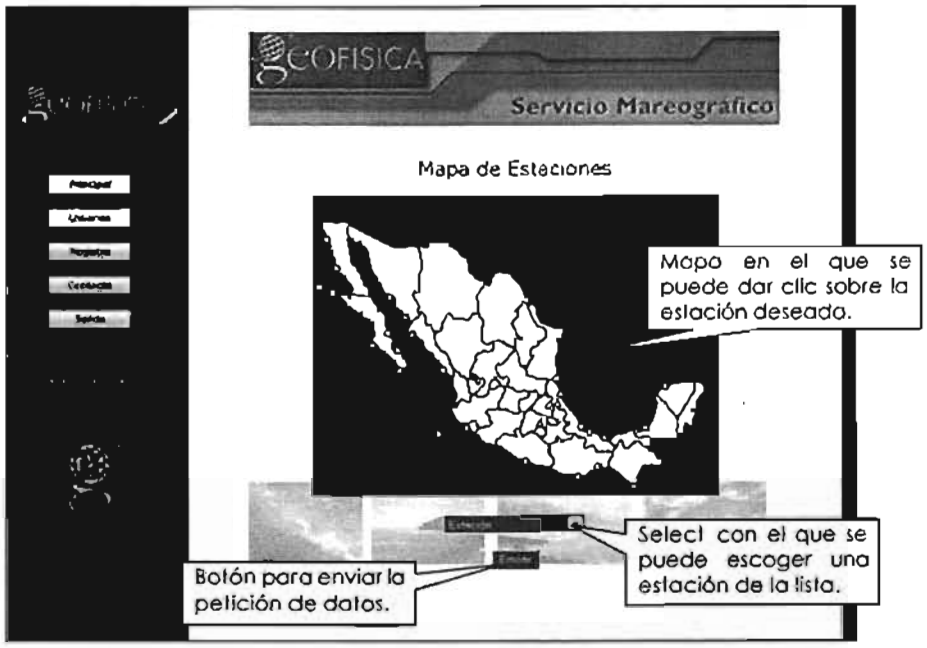
**Pantalla:**

**Nombre:** Mapa.jsp
**Página Anterior:** login.jsp
**Página Siguiente:** inicioGeo2.html
login.jsp
registro.jsp
contacto.jsp
Salida.jsp
Descripción.jsp
**Descripción:** Página en la que se puede seleccionar la estación de la cual se desea ver información.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| Map | map | |
| estacion | select | |
| Enviar | botón | |

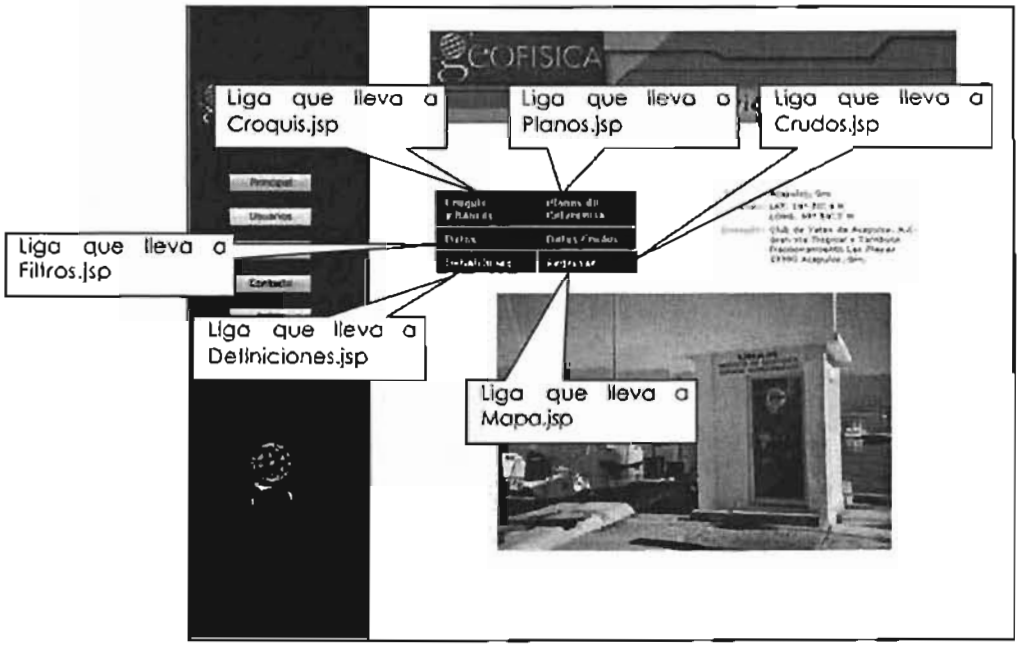
**Pantalla:**



**Nombre:** Descripcion.jsp
**Página Anterior:** Mapa.jsp
**Página Siguiente:** inicioGeo2.html
login.jsp
registro.jsp
contacto.jsp
Salida.jsp
Croquis.jsp
Planos.jsp
Definiciones.jsp
Crudos.jsp
Filtros.jsp
Mapa.jsp

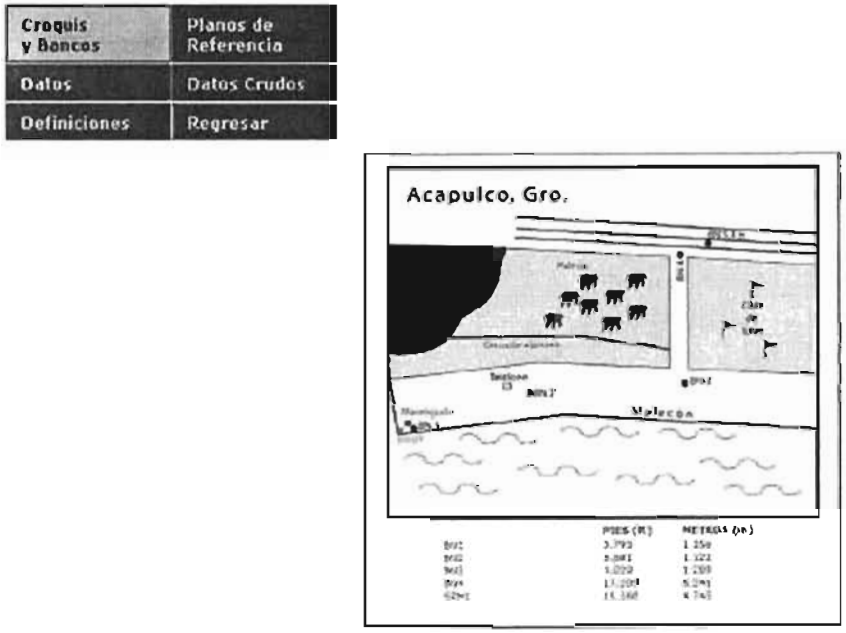
**Descripción:** Página en la que se puede ver información de la estación seleccionada en Mapa.jsp (localización, dirección, foto). También se puede tener acceso al croquis, planos de nivel, definiciones, datos crudos y datos procesados.

**Pantalla:**



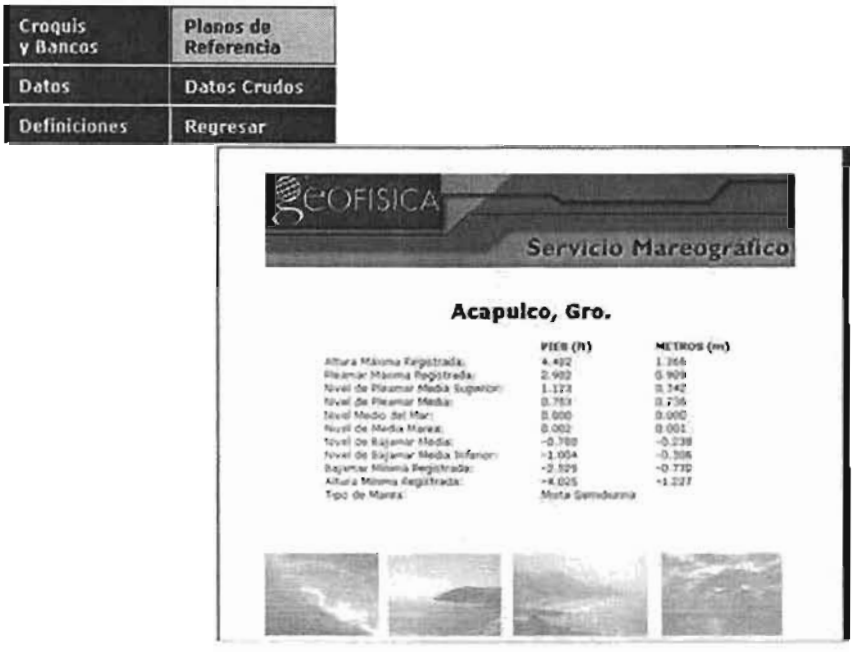
**Nombre:** Croquis.jsp
**Página Anterior:** Descripcion.jsp
**Página Siguiente:** Ninguna
**Descripción:** Página que muestra el croquis y los bancos de nivel de la estación seleccionada.

**Pantalla:**



**Nombre:** Planos.jsp
**Página Anterior:** Descripcion.jsp
**Página Siguiente:** Ninguna
**Descripción:** Página que muestra los planos de referencia de la estación seleccionada.

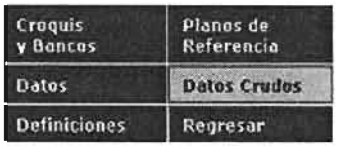
**Pantalla:**

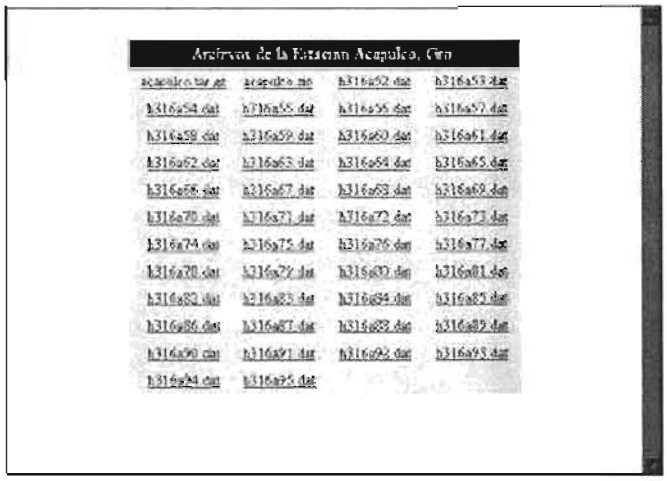


**Nombre:** Definiciones.jsp
**Página Anterior:** Descripcion.jsp
**Página Siguiente:** Ninguna
**Descripción:** Página que muestra definiciones de utilidad.
**Pantalla:**


**Nombre:** Crudos.jsp
**Página Anterior:** Descripcion.jsp
**Página Siguiente:** Ninguna
**Descripción:** Página que muestra una lista de los archivos de valores horarios disponibles de la estación seleccionada.

**Pantalla:**



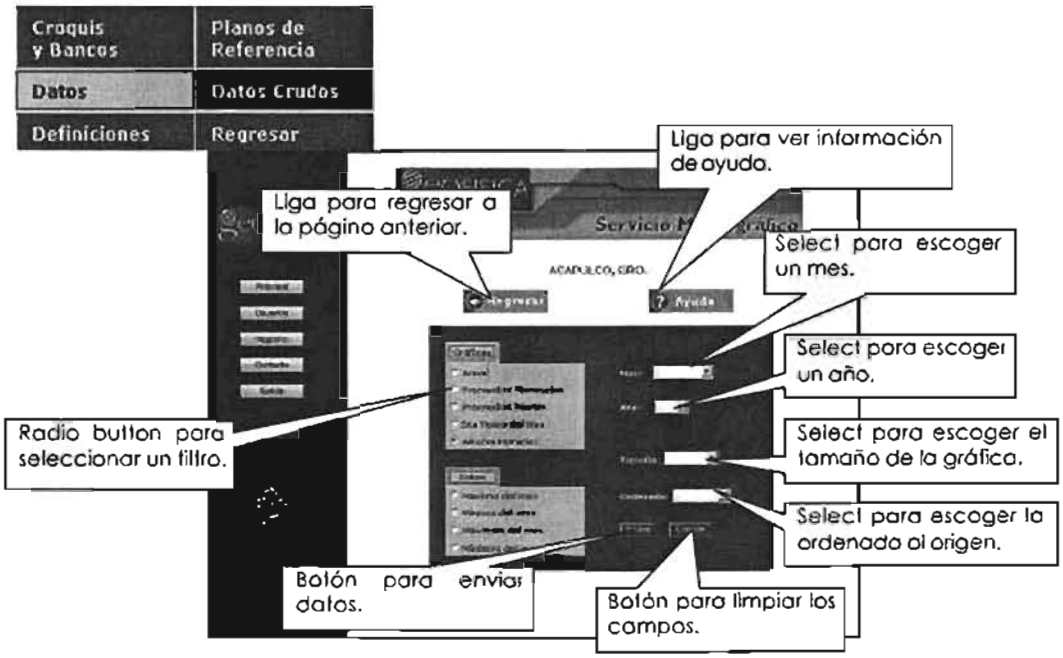



**Nombre:** Filtros.jsp
**Página Anterior:** Descripcion.jsp
**Página Siguiente:** Grafica.jsp
Datos.jsp
registro.jsp
contacto.jsp
Salida.jsp
Descripción.jsp
**Descripción:** Página en la que se puede seleccionar el tipo de información que se desea ver.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| Selección | Radio button | |
| lmes | select | |
| lanio | select | |
| ltamanio | select | |
| lordenada | select | |
| Limpiar | botón | |
| Enviar | botón | |

**Pantalla:**

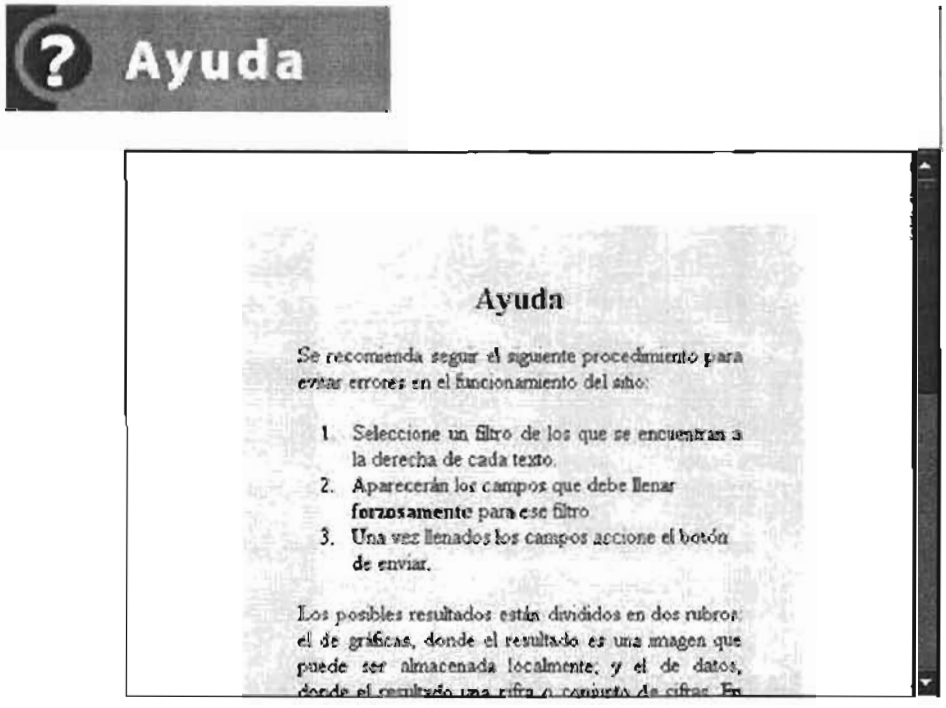


**Nombre:** ayuda.jsp
**Página Anterior:** Filtros.jsp
**Página Siguiente:** Ninguna
**Descripción:** Página que proporciona ayuda sobre el uso de la página de filtros.

**Campos:**

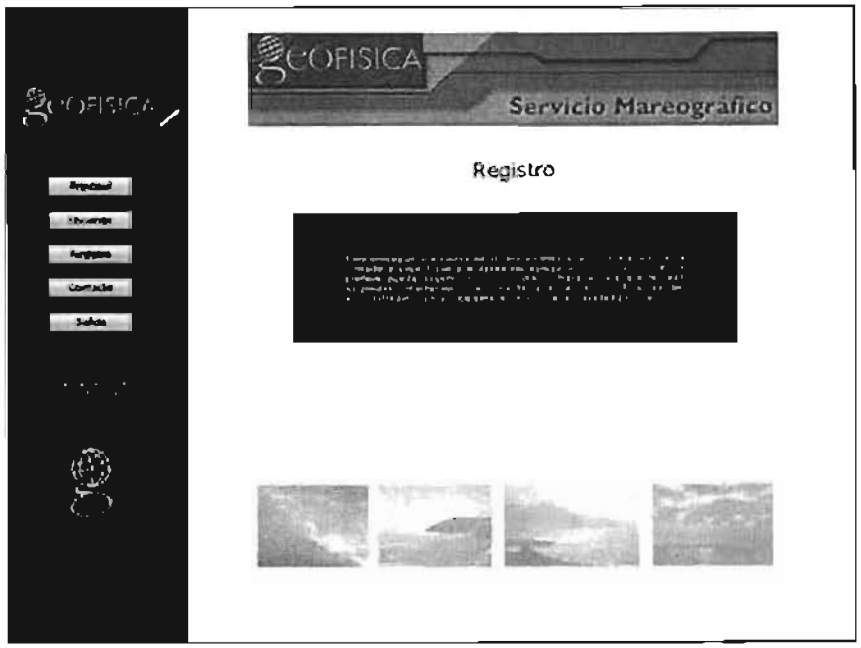
| Campo | Representación | Tamaño |
|---|---|---|
| Cerrar | botón | |

**Pantalla:**



**Nombre:** registro.jsp
**Página Anterior:** inicioGeo2.html
login.jsp
contacto.jsp
Salida.jsp
**Página Siguiente:** inicioGeo2.html
login.jsp
contacto.jsp
Salida.jsp
**Descripción:** Página que proporciona información del registro de nuevos usuarios.

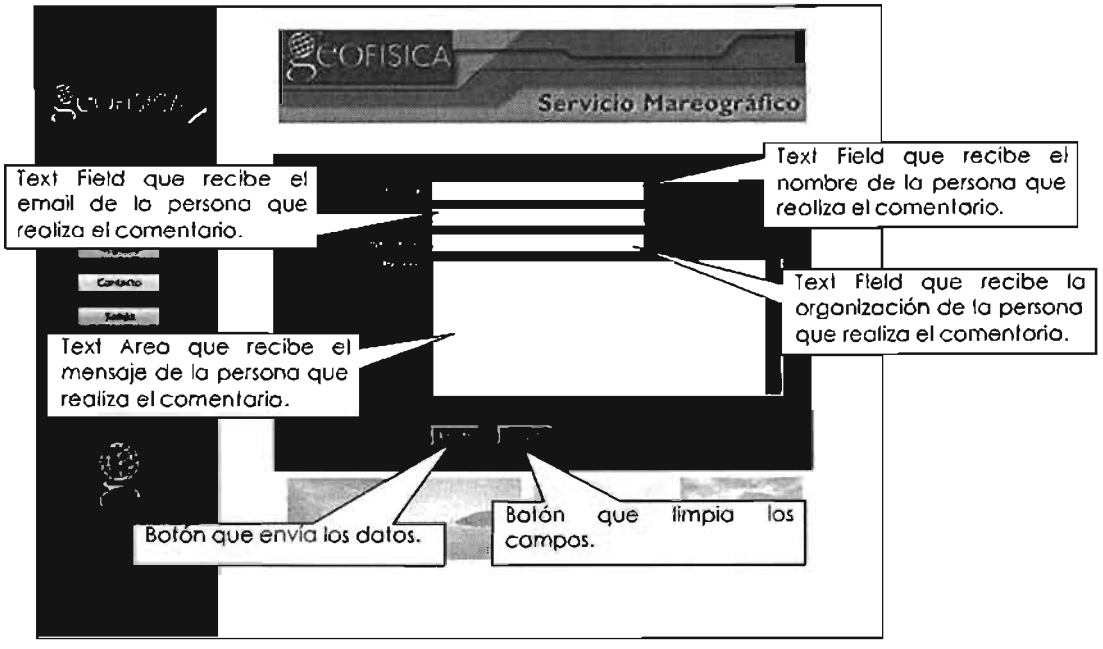
**Pantalla:**



**Nombre:** contacto.jsp
**Página Anterior:** inicioGeo2.html
login.jsp
registro.jsp
Salida.jsp
**Página Siguiente:** inicioGeo2.html
login.jsp
registro.jsp
Salida.jsp
**Descripción:** Página que permite realizar comentarios.
**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| nombreContacto | Text Field | 35 |
| emailContacto | Text Field | 35 |
| organizacionContacto | Text Field | 35 |
| mensajeContacto | Text Area | 50 x 10 |
| Enviar | Botón | |
| Limpiar | Botón | |

**Pantalla:**



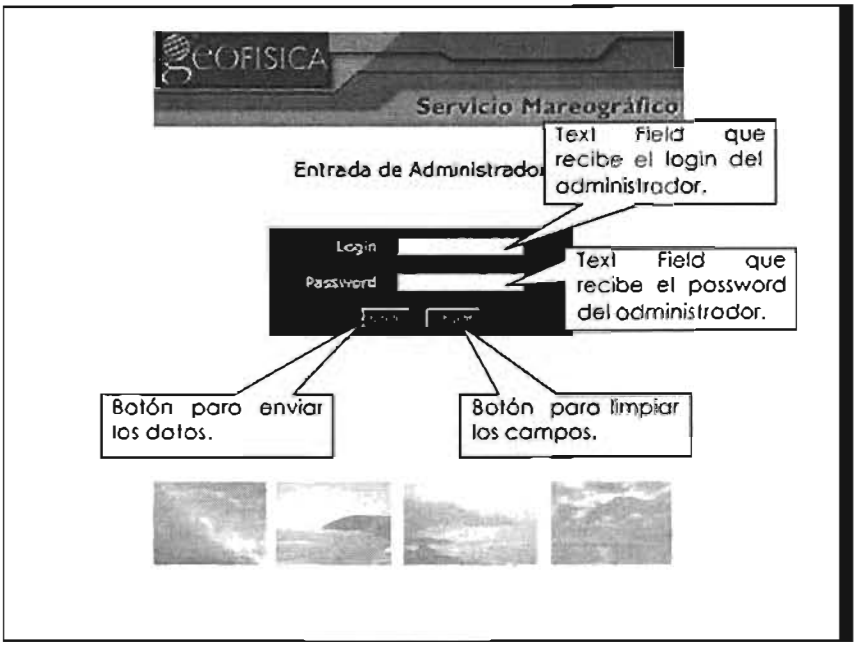
### 3.3.2 Zona de Administración

**Nombre:** index.jsp
**Página Anterior:** Ninguna
**Página Siguiente:** index2.html
**Descripción:** Página que permite el ingreso de los administradores autorizados al sitio.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| loginAdministrador | Text Field | |
| passwdAdministrador | Text Field | |
| Enviar | Botón | |
| Limpiar | Botón | |
| paginaAnterior | Hidden | |

**Pantalla:**



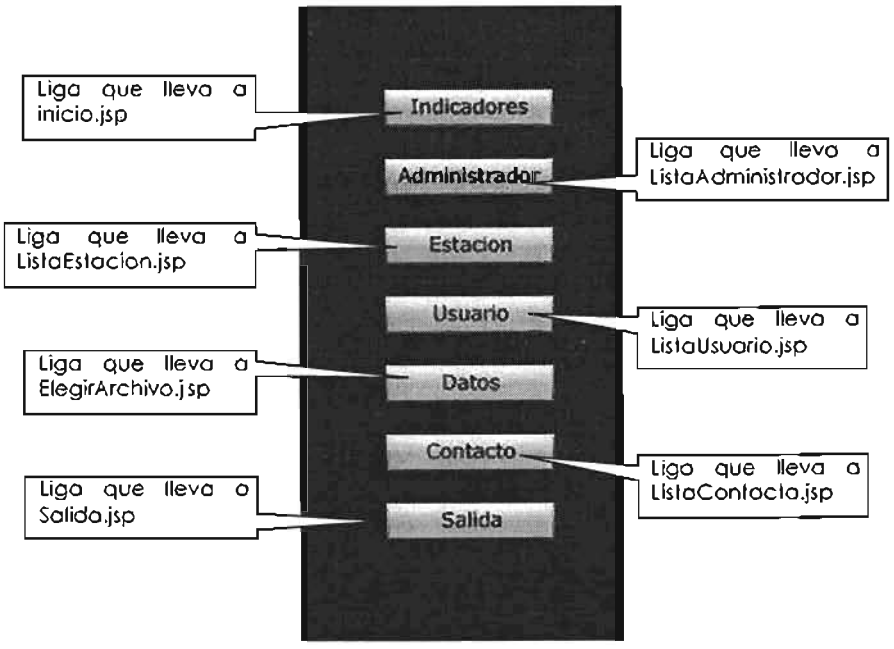
**Nombre:** menu.html

**Página Anterior:** Ninguna

**Página Siguiente:** Ninguna

**Descripción:** Menú de opciones

**Pantalla:**



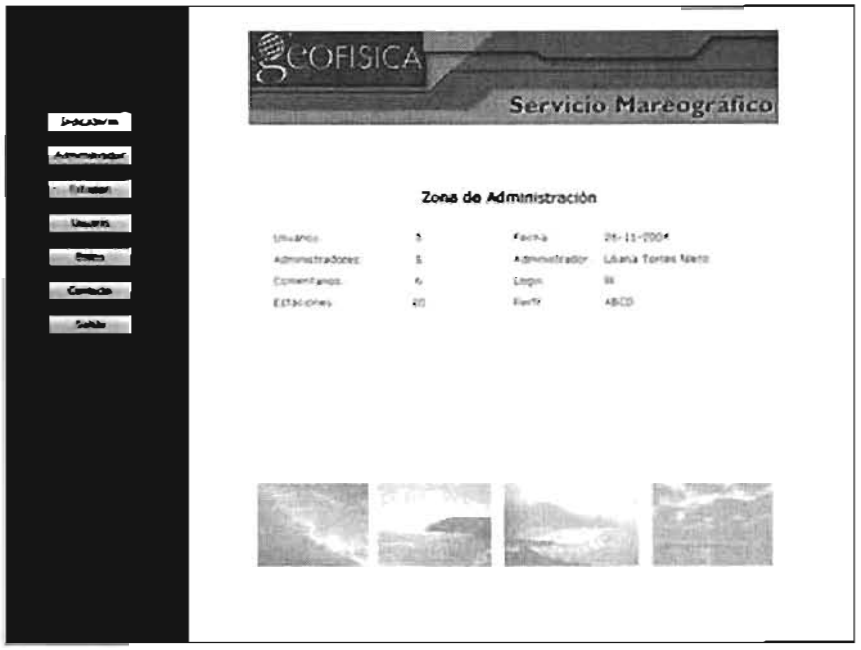
**Nombre:** inicio.jsp

**Página Anterior:** index.jsp

**Página Siguiente:** ListaAdministrador.jsp
ListaEstacion.jsp
ListaUsuario.jsp
ElegirArchivo.jsp
ListaContacto.jsp
Salida.jsp

**Descripción:** Página que muestra indicadores.

**Pantalla:**



**Nombre:** ListaAdministrador.jsp

**Página Anterior:** inicio.jsp
ListaEstacion.jsp
ListaUsuario.jsp
ElegirArchivo.jsp
ListaContacto.jsp
Salida.jsp

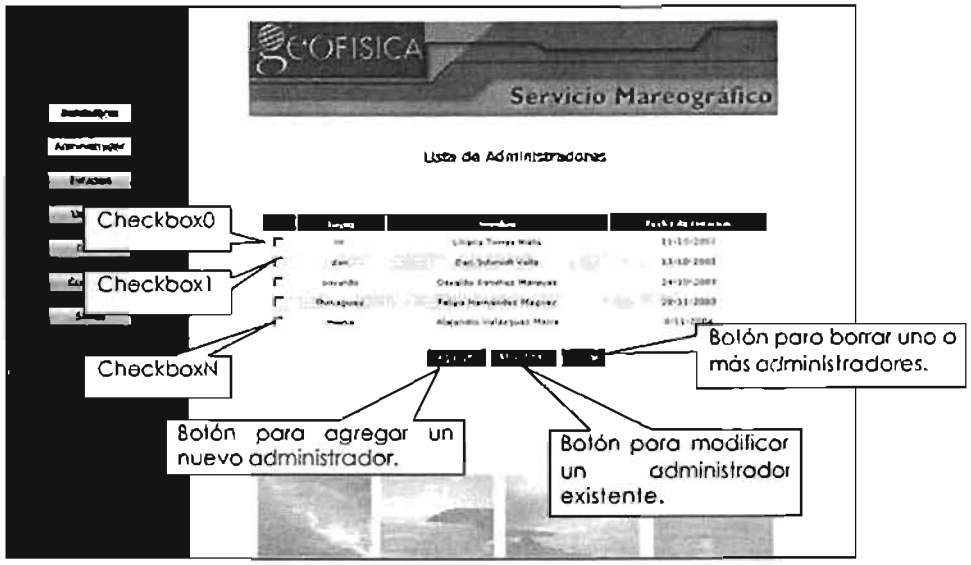
**Página Siguiente:** inicio.jsp
AltasAdministrador.jsp
ModificacionAdministrador.jsp
ListaEstacion.jsp
ListaUsuario.jsp
ElegirArchivo.jsp
ListaContacto.jsp
Salida.jsp

**Descripción:** Página que muestra una lista de los administradores existentes.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| checkboxN | checkbox | |
| Agregar | Botón | |
| Modificar | Botón | |
| Borrar | Botón | |
| paginaAnterior | Hidden | |

**Pantalla:**




**Nombre:** AltasAdministrador.jsp

**Página Anterior:** ListaAdministrador.jsp

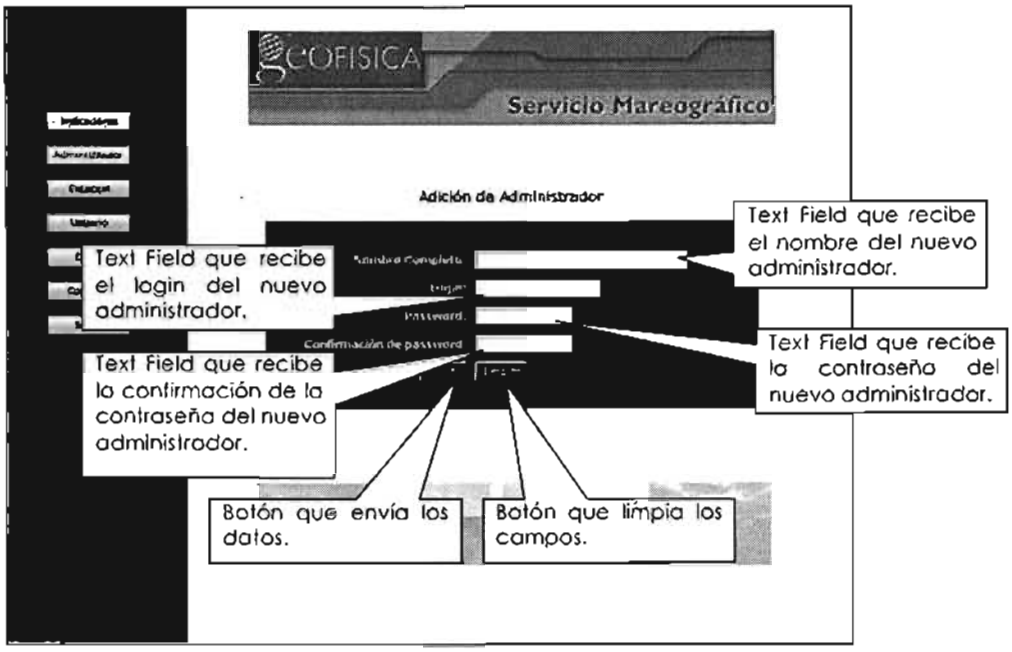
**Página Siguiente:** ListaAdministrador

**Descripción:** Página que permite agregar un nuevo administrador.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| nombreAdministrador | Text Field | 35 |
| loginAdministrador | Text Field | 20 |
| passwordAdministrador | Text Field | 15 |
| passwordAdministrador2 | Text Field | 15 |
| Enviar | Botón | |
| Limpiar | Botón | |

**Pantalla:**




**Nombre:** ModificacionAdministrador.jsp

**Página Anterior:** ListaAdministrador.jsp

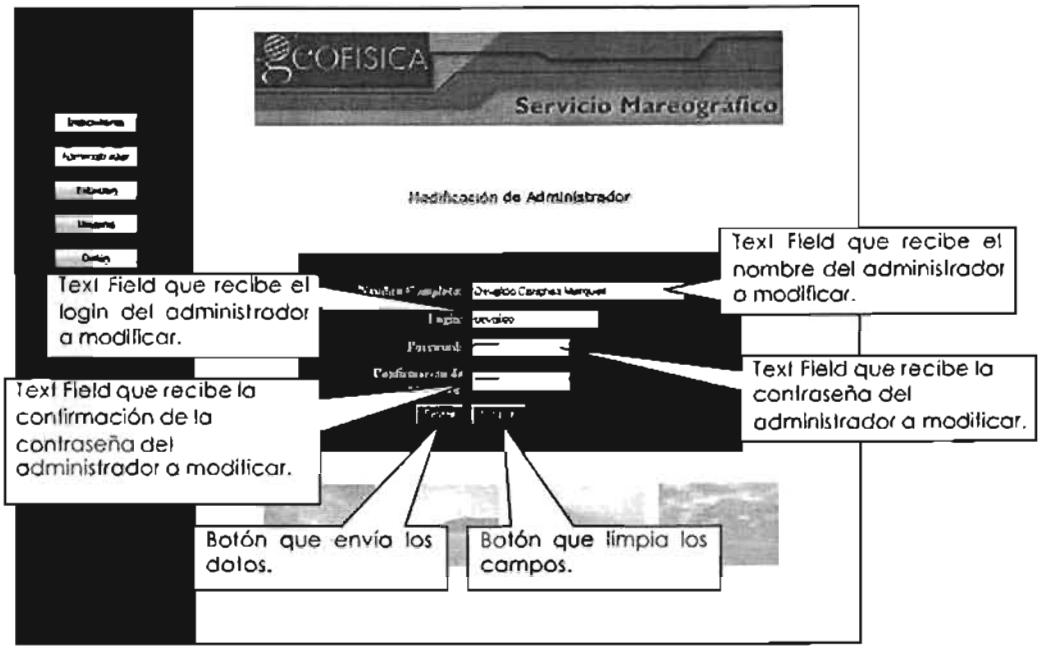
**Página Siguiente:** ListaAdministrador.jsp

**Descripción:** Página que permite modificar un administrador existente.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| nombreAdministrador | Text Field | 35 |
| loginAdministrador | Text Field | 20 |
| passwordAdministrador | Text Field | 15 |
| passwordAdministrador2 | Text Field | 15 |
| Enviar | Botón | |
| Limpiar | Botón | |

**Pantalla:**



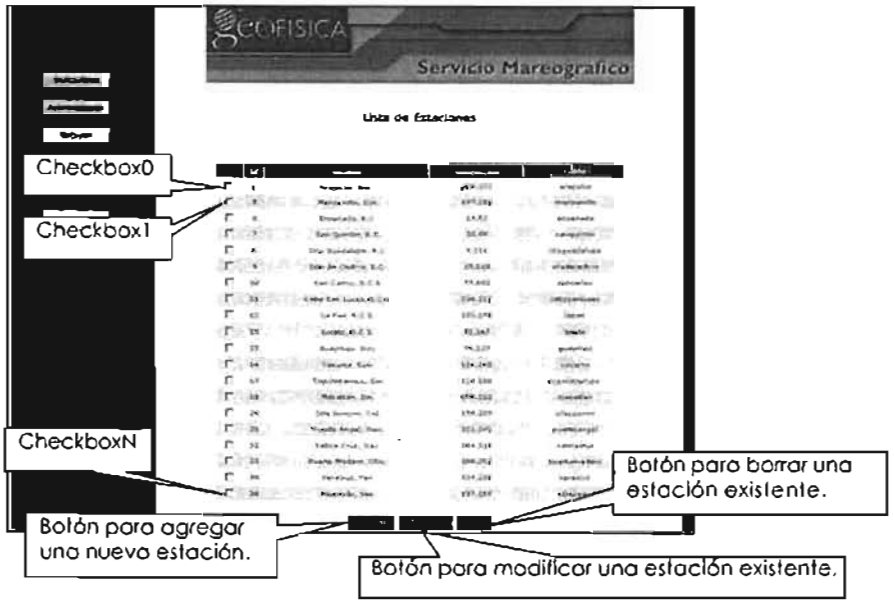

| | |
|---|---|
| **Nombre:** | ListaEstacion.jsp |
| **Página Anterior:** | inicio.jsp<br>ListaAdministrador.jsp<br>AltasEstacion.jsp<br>ModificacionEstacion.jsp<br>ListaUsuario.jsp<br>ElegirArchivo.jsp<br>ListaContacto.jsp<br>Salida.jsp |
| **Página Siguiente:** | inicio.jsp<br>ListaAdministrador.jsp<br>ListaUsuario.jsp<br>ElegirArchivo.jsp<br>ListaContacto.jsp<br>Salida.jsp |

**Descripción:** Página que muestra una lista de las estaciones existentes.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| checkboxN | checkbox | |
| Agregar | Botón | |
| Modificar | Botón | |
| Borrar | Botón | |
| paginaAnterior | Hidden | |

**Pantalla:**



**Nombre:** AltasEstacion.jsp

**Página Anterior:** ListaEstacion.jsp

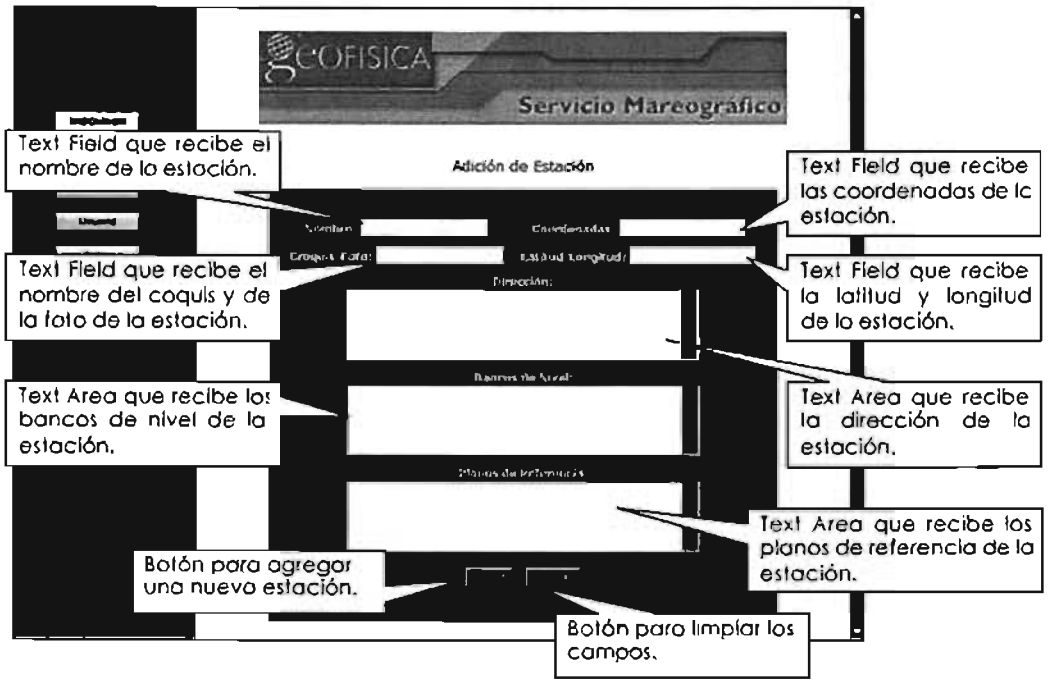
**Página Siguiente:** ListaEstacion.jsp

**Descripción:** Página que permite agregar una nueva estación.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| nombreEstacion | Text Field | |
| coordenadasEstacion | Text Field | |
| croquisFotoEstacion | Text Field | |
| geografiaEstacion | Text Field | |
| direccionEstacion | Text Area | 50 x 5 |
| bancosEstacion | Text Area | 50 x 5 |
| planosEstacion | Text Area | 50 x 5 |
| Enviar | Botón | |
| Limpiar | Botón | |

**Pantalla:**



**Nombre:** ModificacionEstacion.jsp

**Página Anterior:** ListaEstacion.jsp

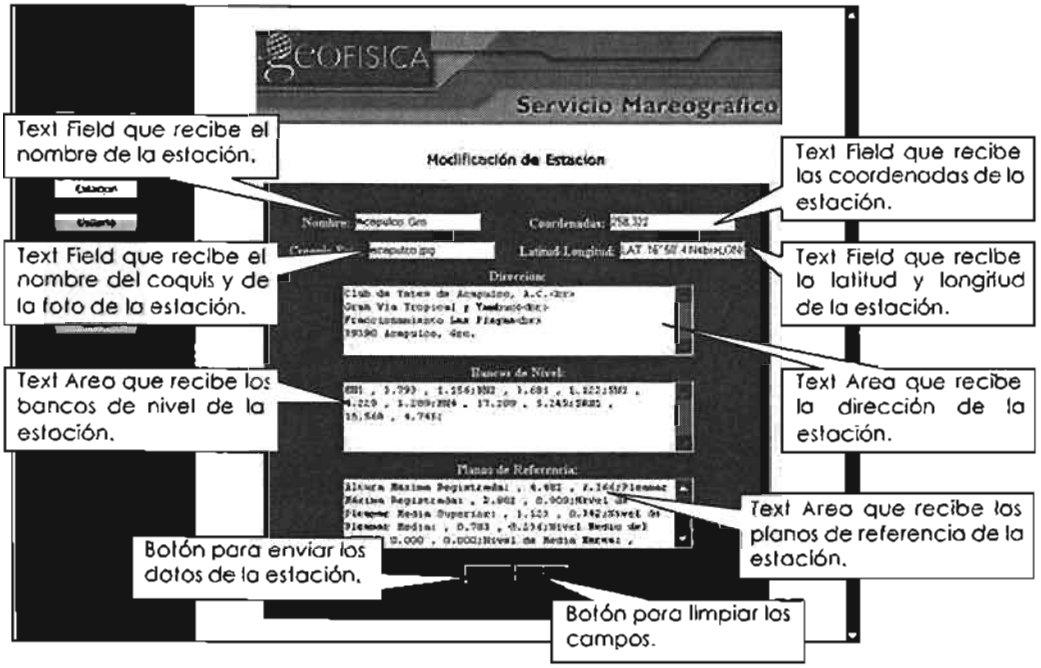
**Página Siguiente:** ListaEstacion.jsp

**Descripción:** Página que permite modificar los datos de una estación existente.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| nombreEstacion | Text Field | |
| coordenadasEstacion | Text Field | |
| croquisFotoEstacion | Text Field | |
| geografiaEstacion | Text Field | |
| direccionEstacion | Text Area | 50 x 5 |
| bancosEstacion | Text Area | 50 x 5 |
| planosEstacion | Text Area | 50 x 5 |
| Enviar | Botón | |
| Limpiar | Botón | |

**Pantalla:**



**Nombre:** ListaUsuario.jsp

**Página Anterior:** inicio.jsp
ListaAdministrador.jsp
ListaEstacion.jsp
AltasUsuario.jsp
ModificacionUsuario.jsp
ElegirArchivo.jsp
ListaContacto.jsp
Salida.jsp

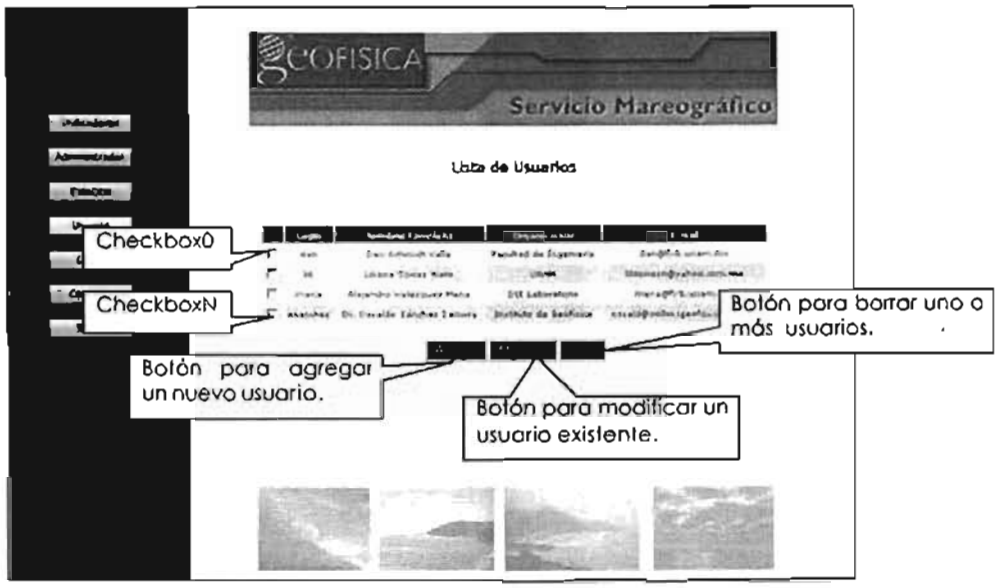
**Página Siguiente:** inicio.jsp
ListaAdministrador.jsp
ListaEstacion.jsp
ElegirArchivo.jsp
ListaContacto.jsp
Salida.jsp

**Descripción:** Página que muestra una lista de los usuarios existentes.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| checkboxN | checkbox | |
| Agregar | Botón | |
| Modificar | Botón | |
| Borrar | Botón | |
| paginaAnterior | Hidden | |

**Pantalla:**




**Nombre:** AltasUsuario.jsp

**Página Anterior:** ListaUsuario.jsp

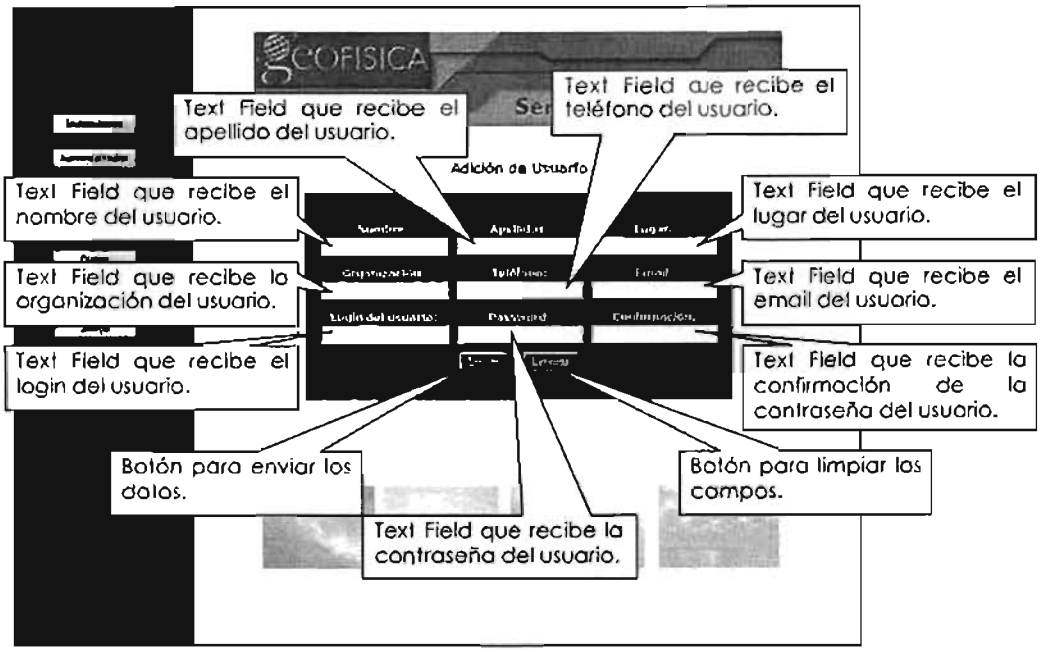
**Página Siguiente:** ListaUsuario.jsp

**Descripción:** Página que permite agregar un nuevo usuario.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| nombreUsuario | Text Field | 20 |
| apellidosUsuario | Text Field | 20 |
| lugarUsuario | Text Field | 20 |
| organizacionUsuario | Text Field | 20 |
| telefonoUsuario | Text Field | 20 |
| emailUsuario | Text Field | 20 |
| loginUsuario | Text Field | 20 |
| passwordUsuario | Text Field | 20 |
| passwordUsuario2 | Text Field | 20 |
| Enviar | Botón | |
| Limpiar | Botón | |

**Pantalla:**




**Nombre:** ModificacionUsuario.jsp

**Página Anterior:** ListaUsuario.jsp

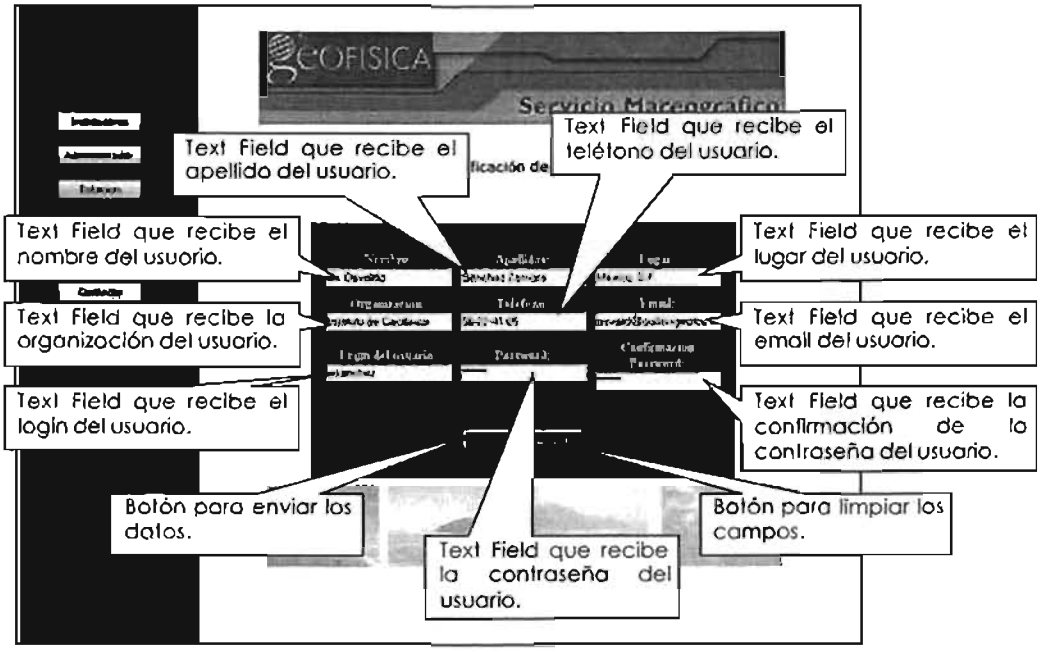
**Página Siguiente:** ListaUsuario.jsp

**Descripción:** Página que permite modificar un usuario existente.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| nombreUsuario | Text Field | 20 |
| apellidosUsuario | Text Field | 20 |
| lugarUsuario | Text Field | 20 |
| organizacionUsuario | Text Field | 20 |
| telefonoUsuario | Text Field | 20 |
| emailUsuario | Text Field | 20 |
| loginUsuario | Text Field | 20 |
| passwordUsuario | Text Field | 20 |
| passwordUsuario2 | Text Field | 20 |
| Enviar | Botón | |
| Limpiar | Botón | |

**Pantalla:**




**Nombre:** ElegirArchivo.jsp

**Página Anterior:** inicio.jsp
ListaAdministrador.jsp
ListaEstacion.jsp
ListaUsuario.jsp
ListaContacto.jsp
Salida.jsp

**Página Siguiente:** inicio.jsp
SubirArchivo.jsp
ListaAdministrador.jsp

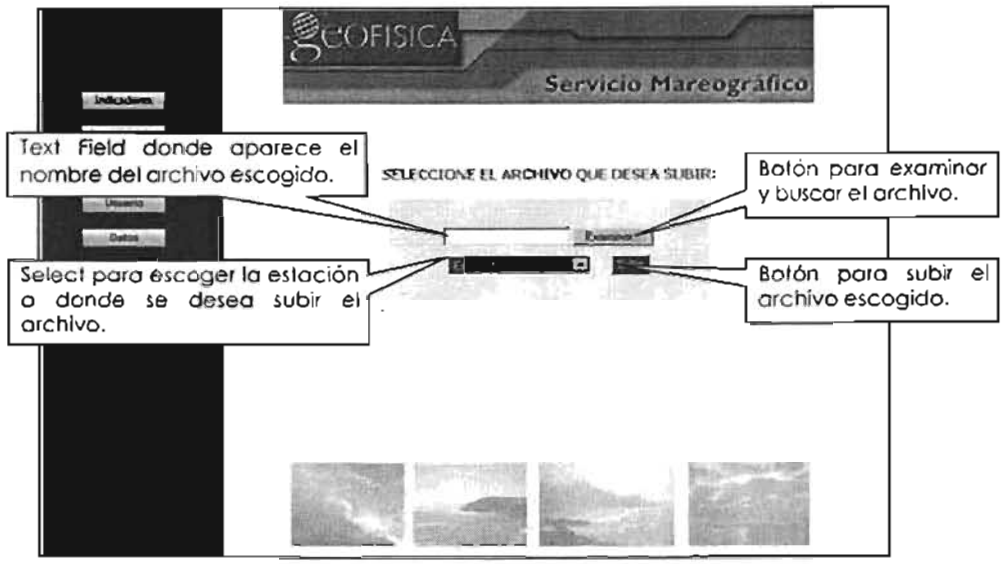
ListaEstacion.jsp
ListaUsuario.jsp
ListaContacto.jsp
Salida.jsp

**Descripción:** Página que permite agregar archivos de datos de estaciones existentes.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| FILE1 | file | |
| estacion | select | |
| Subir | botón | |

**Pantalla:**



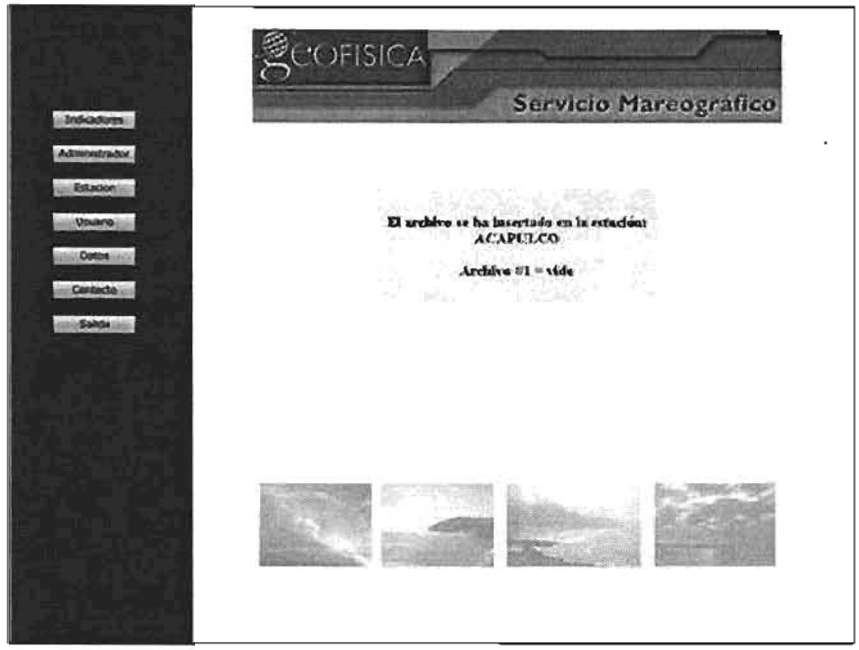
**Nombre:** SubirArchivo.jsp

**Página Anterior:** inicio.jsp
ListaAdministrador.jsp
ListaEstacion.jsp
ListaUsuario.jsp
ElegirArchivo.jsp
ListaContacto.jsp
Salida.jsp

**Página Siguiente:** inicio.jsp
ListaAdministrador.jsp
ListaEstacion.jsp
ListaUsuario.jsp
ElegirArchivo.jsp
ListaContacto.jsp
Salida.jsp

**Descripción:** Página que muestra información sobre los archivos guardados hacia el servidor.

**Pantalla:**



**Nombre:** ListaContacto.jsp

**Página Anterior:** inicio.jsp
ListaAdministrador.jsp
ListaEstacion.jsp
ListaUsuario.jsp
ElegirArchivo.jsp
Salida.jsp

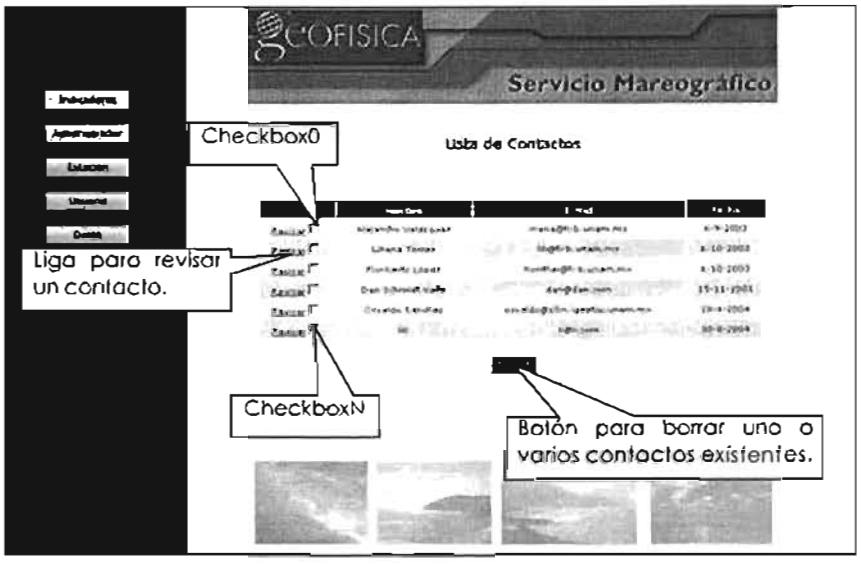
**Página Siguiente:** inicio.jsp
ListaAdministrador.jsp
ListaEstacion.jsp
ListaUsuario.jsp
ElegirArchivo.jsp
Salida.jsp

**Descripción:** Página que muestra una lista de los contactos existentes.

**Campos:**

| Campo | Representación | Tamaño |
|---|---|---|
| checkboxN | checkbox | |
| Borrar | Botón | |
| paginaAnterior | Hidden | |

**Pantalla:**



# 4. Programación

## 4.1 Java

**Acerca de la versión de JAVA**

Para la programación en Java, se tomó la decisión de utilizar la versión 1.3.1 de la JDK, dado que era la más estable al momento. Estaba disponible la versión 1.4.1, pero aún era una versión beta, por lo que se optó por utilizar la más estable. Las nuevas tecnologías que interesan de la JDK 1.3 para el desarrollo de este proyecto son:

- **Java Beans**: Son un componente que implementa exitosamente la capacidad de la plataforma Java para "escribir una vez, ejecutar donde sea". De hecho, los *Java Beans* llevan la interoperabilidad hacia un nuevo nivel, ya que el código corre bajo cualquier sistema operativo y bajo el ambiente de cualquier aplicación.
- **Mejoras del java.util**: Este paquete ha sido mejorado para la comodidad del programador con mejores herramientas para el manejo de eventos, colecciones de objetos y fechas/horas.
- **Precisión matemática:** Se incluyen dos clases (BinInteger y BigDecimal) con manejo de precisión arbitrario. En especial la clase BigDecimal es excelente para el manejo de una cantidad fija de decimales y para dar formato a números.
- **Reflexión**: Permite al código de Java obtener información sobre los atributos, métodos y constructores de las clases cargadas.

- **JDBC**: La JDK incluye el paquete java.sql, que permite la conectividad con casi cualquier base de datos, hoja de cálculo o archivo de datos plano; y javax.sql, que agrega funciones para utilizar del lado del servidor.

### Acerca de los nombres para los componentes

Con el objetivo de estandarizar la programación, se determinaron ciertos lineamientos a seguir para la generación de código. Como ya es costumbre dentro de la comunidad de programadores, se utilizaron mayúsculas para el nombre de las clases, así como para los *JSPs*. La convención incluye, además, el iniciar cada palabra del nombre de una clase con letra mayúscula también. De esta manera, en caso de existir una clase cuya función sea dar formato a números, un nombre pudiese ser DarFormatoANumeros, dentro del archivo DarFormatoANumeros.java. Nótese que se respeta la ortografía de las palabras, excepto por los acentos, diéresis, eñes y guiones. No es posible, asimismo, incluir espacios en blanco dentro del nombre de una clase. En caso de ser necesario puede sustituirse los mismos por guiones bajos ( _ ) , pero esto no es acostumbrado.

En lo referente a los nombres de atributos y métodos, el criterio es el mismo, pero se utiliza letra minúscula en la primera palabra. Como ejemplo un método que transforme una cadena a entero podría tener la firma siguiente: cadenaAEntero( ). Asimismo, una cadena que almacenase el número de cuenta de un alumno pudiera llamarse numeroDeCuentaDeAlumno.

Un estándar particular al que hemos recurrido para el nombrado de objetos nuevos es el utilizar, en letra minúscula, la primera letra de cada palabra de la clase de la cual el objeto es instancia. Por lo tanto, un objeto de la clase antes mencionada, (DarFormatoANumeros), tendría como nombre de referencia "dfan". En el caso de que sea necesario generar un segundo objeto de la misma clase, se optó por agregar letras de la primera palabra del nombre de la clase. Como ejemplo, el primer objeto de la clase Usuario tendrá como nombre de referencia "u", el segundo "us", el tercero "usu", y así sucesivamente. Si bien es cierto que puede ser conveniente dar siempre a un objeto un nombre descriptivo, el uso repetido dentro del código de tal objeto puede hacer que un nombre descriptivo, pero largo, sea inapropiado. Por ende, este estándar que se ha implementado nos parece adecuado.

# 4.2 SQL

### Postgres

PostgreSQL es la base de de datos gratuita más popular del mundo. Implementa algunos de los estándares ISO-*SQL*: ANSI *SQL*/99, *SQL*/92 Y ANSI *SQL* 89. PostgreSQL es una base de datos objeto-relacional de próxima generación. Los futuros estándares ANSI/*SQL*, como *SQL* 1999 (*SQL*-3) y siguientes, incrementarán el acercamiento entre las bases de datos de tipo relacional y las orientadas a datos. PostgreSQL es el único manejador gratuito del mundo que soporta bases de datos objeto y de datos.

### Sobre SQL y los estándares

*SQL* se ha convertido en el lenguaje de consulta relacional más popular. El nombre "*SQL*" es una abreviatura de Structured Query Language (Lenguaje de consulta estructurado). En 1974, Donald Chamberlain y otros definieron el lenguaje SEQUEL (Structured English Query Language) en IBM Research. Este lenguaje fue implementado inicialmente en un prototipo de IBM llamado SEQUEL-XRM en 1974-75. Entre 1976 y 1977 se definió una revisión de SEQUEL llamada SEQUEL/2 y el nombre se cambió a *SQL*.

IBM desarrolló un nuevo prototipo llamado System R en 1977. System R implementó un amplio subconjunto de SEQUEL/2 (ahora *SQL*) y un número de cambios que se le hicieron durante el proyecto. System R se instaló en algunas estaciones de trabajo para usuarios, tanto internos en IBM como en algunos clientes seleccionados. Gracias al éxito y aceptación de System R en los mismos, IBM inició el desarrollo de productos comerciales que implementaban el lenguaje *SQL* basado en la tecnología System R. Durante los años siguientes, IBM y bastantes otros vendedores anunciaron productos *SQL* tales como *SQL*/DS (IBM), D82 (IBM), ORACLE (Oracle Corp.), DG/*SQL* (Data General Corp.), y SYBASE (Sybase Inc.).

*SQL* es también un estándar oficial hoy. En 1982, la American National Standards Institute (ANSI) encargó a su Comité de Bases de Datos X3H2 el desarrollo de una propuesta de lenguaje relacional estándar. Esta propuesta fue ratificada en 1986 y consistía básicamente en el dialecto de IBM de *SQL*. En 1987, este estándar ANSI fue también aceptado por la Organización Internacional de Estandarización (ISO). Esta versión estándar original de *SQL* recibió informalmente el nombre de "*SQL*/86". En 1989, el estándar original fue extendido, y recibió el nuevo nombre, también informal, de "*SQL*/89". También en 1989, se desarrolló un estándar relacionado llamado Database Language Embedded *SQL* (ESQL).

Los comités ISO y ANSI han estado trabajando durante muchos años en la definición de una versión muy ampliada del estándar original, llamado informalmente SQL2 o *SQL/92*. Esta versión se convirtió en un estándar ratificado durante 1992: International Standard ISO/IEC 9075:1992, Database Language *SQL*. *SQL/92* es la versión a la que normalmente la gente se refiere cuando habla de «*SQL* estándar».

**SQL3**

Los tipos de datos comúnmente referidos como tipos *SQL3* son los nuevos tipos de datos que se han adoptado en la nueva versión del estándar ANSI/ISO de *SQL*. El JDBC 2.0 proporciona interfaces que representan un mapeado de estos tipos de datos *SQL3* dentro del lenguaje Java. Con estas nuevas interfases, es posible trabajar con tipos *SQL3* igual que con otros tipos.

Los nuevos tipos *SQL3* le dan a una base de datos relacional más flexibilidad en lo referente a los tipos utilizables para una columna de una tabla. Por ejemplo, una columna podría ahora almacenar el nuevo tipo de dato *BLOB* (Binary Large Object), que puede almacenar grandes cantidades de datos como una fila de bytes. Una columna también puede ser del tipo *CLOB* (Character Large Object), que es capaz de almacenar grandes cantidades de datos en formato caracter. El nuevo tipo ARRAY hace posible el uso de un array como un valor de columna. Incluso las nuevas estructuras de "tipos definidos por el usuario" (*UDT*: User Defined Types) de *SQL* pueden almacenarse como valores de columna.

La siguiente lista tiene las interfases del JDBC 2.0 que mapean los tipos *SQL3*.

- Un ejemplar Blob mapea un *BLOB* de *SQL*.
- Un ejemplar Clob mapea un *CLOB* de *SQL*.
- Un ejempar Array mapea un ARRAY de *SQL*
- Un ejemplar Struct mapea un tipo Estructurado de *SQL*.
- Un ejemplar Ref mapea un REF de *SQL*.

Se recuperan, almacenan, y actualizan tipos de datos *SQL3* de la misma forma que los otros tipos. Se utilizan los métodos ResultSet.get o CallableStatement.get para recuperarlos, los métodos PreparedStatement.set para almacenarlos y update para actualizarlos. Probablemente el 90% de las operaciones realizadas con tipos *SQL3*

implican el uso de los métodos get, set, y update. La tabla 4.1 muestra qué métodos utilizar:

| Tipo SQL3 | Método get | Método set | Método update |
|---|---|---|---|
| *BLOB* | getBlob | setBlob | updateBlob |
| *CLOB* | getClob | setClob | updateClob |
| ARRAY | getArray | setArray | updateArray |
| Tipo Structured | getObject | setObject | updateObject |
| REF(Tipo Structured) | getRef | setRef | updateRef |

**Tabla 4. 1 Nuevos tipos de datos del estándar SQL 3**

Por ejemplo, el siguiente fragmento de código recupera un valor ARRAY de *SQL*. Para este ejemplo, la columna SCORES de la tabla STUDENTS contiene valores del tipo ARRAY. La variable stmt es un objeto Statement.

```
ResultSet rs = stmt.executeQuery ("SELECT SCORES FROM
STUDENTS WHERE ID = 2238");
rs.next();
Array scores = rs.getArray("SCORES");
```

La variable scores es un apuntador lógico al objeto ARRAY de *SQL* almacenado en la tabla STUDENTS en la fila del estudiante 2238. Si se desea almacenar un valor en la base de datos, se emplea el método set apropiado. Por ejemplo, el siguiente fragmento de código, en el que rs es un objeto ResultSet, almacena un objeto Clob:

```
Clob notes = rs.getClob("NOTES");
PreparedStatement pstmt = con.prepareStatement("UPDATE
MARKETS SET
COMMENTS = ? WHERE SALES < 1000000",
ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);
pstmt.setClob(1, notes);
```

Este código configura notes como el primer parámetro de la sentencia de actualización que está siendo enviada a la base de datos. El valor *CLOB* designado por notes será almacenado en la tabla MARKETS en la columna COMMENTS en cada columna en que el valor de la columna SALES sea menor de un millón.

Una característica importante sobre los objetos Blob, Clob, y Array es que se pueden manipular sin tener que traer todos los datos desde el servidor de la base de datos a la máquina cliente. Un ejemplar de cualquiera de esos tipos es realmente un puntero lógico al objeto en la base de datos que representa el ejemplar. Como los objetos *SQL BLOB, CLOB* o ARRAY pueden ser muy grandes, esta característica puede aumentar drásticamente el rendimiento.

Se pueden utilizar los comandos *SQL* y los *API* JDBC 1.0 y 2.0 con objetos Blob, Clob, y Array como si se estuviese operando realmente con el objeto de la base de datos. Sin embargo, si se desea trabajar con cualquiera de ellos como un objeto Java, será necesario traer todos los datos al cliente, con lo que la referencia se materializa en el objeto. Por ejemplo, si el fin es utilizar un ARRAY de *SQL* en una aplicación como si fuera un array Java, es pertinenete materializar el objeto ARRAY en el cliente para convertirlo en un array de Java. Entonces, podrá utilizarse los métodos de arrays de Java para operar con los elementos del array. Todas las interfases Blob, Clob, y Array tienen métodos para materializar los objetos que representan.

Los tipos estructurados y distinct de *SQL* son dos tipos de datos que el usuario puede definir. Normalmente se hace referencia a ellos como *UDTs* y se crean con una sentencia CREATE TYPE de *SQL*. Un tipo estructurado de *SQL* se parece a los tipos estructurados de Java en que tienen miembros, llamados atributos, que puede ser cualquier tipo de dato. De hecho, un atributo podría ser a su vez un tipo estructurado. Aquí se presenta un ejemplo de una simple definición de un nuevo tipo de dato *SQL*:

```
CREATE TYPE PLANE_POINT
(
X FLOAT,
Y FLOAT
)
```

Al contrario que los objetos Blob, Clob, y Array, un objeto Struct contiene valores para cada uno de los atributos del tipo estructurado de *SQL* y no es sólo un puntero lógico al objeto en la base de datos. Por ejemplo,

supóngase que un objeto PLANE_POINT está almacenado en la columna POINTS de la tabla PRICES.

```
ResultSet rs = stmt.executeQuery("SELECT POINTS FROM PRICES
WHERE PRICE > 3000.00");
while (rs.next())
{
Struct point = (Struct)rs.getObject("POINTS");
}
```

Si el objeto PLANE_POINT recuperado tiene un valor X de 3 y un valor Y de -5, el objeto Struct, point contendrá los valores 3 y -5.

Puede objetarse que Struct es el único tipo que no tiene métodos get y set. Debe utilizarse getObject y setObject con ejemplares Struct. Esto significa que cuando se recupere un valor utilizando el método getObject, se obtendrá un Object Java, al cual se hará un cast explícito a Struct, como se ha hecho en el ejemplo de código anterior.

El segundo tipo *SQL* que un usuario puede definir con una sentencia CREATE TYPE de *SQL* es un tipo Distinct. Este tipo se parece al typedef de C o C++ en que es un tipo basado en un tipo existente. Aquí tenemos un ejemplo de creación de un tipo distinct:

```
CREATE TYPE MONEY AS NUMERIC(10, 2)
```

Esta definición crea un nuevo tipo llamado MONEY, que es un número del tipo NUMERIC que siempre está en base 10 con dos dígitos después de la coma decimal. MONEY es ahora un tipo de datos en el esquema en el que fue definido y puede almacenarse ejemplares de MONEY en una tabla que tenga una columna del tipo MONEY.

Un tipo distinct *SQL* se mapea en Java al tipo en el que se mapearía el tipo original. Por ejemplo, NUMERIC mapea a java.math.BigDecimal, porque el tipo MONEY mapea a java.math.BigDecimal. Para recuperar un objeto MONEY, se emplea ResultSet.getBigDecimal o CallableStatement.getBigDecimal; para almacenarlo se recurre a PreparedStatement.setBigDecimal.

Algunos aspectos del trabajo con los tipos *SQL3* pueden ser bastante complejos. Para mencionar alguna de las características más avanzadas, la interfase Struct es el mapeo estándar para un tipo estructurado de *SQL*.

Si se quiere trabajar fácilmente con un tipo estructurado de Java, puede mapearse a una clase Java. El tipo estructurado se convierte en una clase, y sus atributos en campos. No es forzoso utilizar un mapeado personalizado, pero normalmente es más conveniente.

Algunas veces es deseable trabajar con un apuntador lógico a un tipo estructurado de *SQL*, en vez de con todos los valores contenidos en el propio tipo. Esto podría suceder, por ejemplo, si el tipo estructurado tiene muchos atributos o si los atributos son muy grandes. Para hacer referencia a un tipo estructurado, se declarara un tipo REF de *SQL* que represente un tipo estructurado particular. Un objeto REF de *SQL* se mapea en un objeto Ref de Java, y se operara con el como si se tratara con el objeto del tipo estructurado al que representa.

**Uso en la Tesis**

La base de datos utilizada en el servidor es la versión 7.2 de Postgres. Esta base de datos es gratuita como todo el demás software utilizado en este proyecto a excepción del sistema operativo Solaris. Como soporta la mayor parte de los estándares ANSI, no hubo problema en la utilización de las funciones y lenguaje *SQL* estándar.

Para la generación de tablas existen *scripts* que permiten la ejecución del código *SQL*. Estos *scripts* ofrecen la facilidad de poder corregir cualquier error sobre un archivo de texto, en vez de tener que recapturar un comando que puede ser muy extenso. Además, la utilización de *scripts* permite la combinación de comandos en *SQL* con comandos de shell. En particular, esta faceta nos ayudó enormemente al momento de insertar nuevos datos a una estación en particular. Copiar datos desde un archivo hacia una tabla es un proceso sencillo si el archivo se encuentra en un formato apropiado. No obstante, los archivos de texto plano que son convención dentro del ambiente de los analistas del Servicio Mareográfico cuentan con un formato que no es conveniente para las bases de datos. De esta forma, una combinación del lenguaje AWK, el shellscript y las herramientas de la base de datos permiten, desde el área de administración, agregar un nuevo archivo de datos cuando sea necesario.

El código *SQL* utilizado dentro de la programación para obtener datos es casi por completo transparente. Se utilizan únicamente dos funciones particulares de Postgres para la optimización del proceso de selección en particular. Una es la función *extract*, que permite tomar de una fecha

únicamente el año, mes o día. La otra función es *distinct*, que permite eliminar tuplas repetidas de una selección.

Como es normal, cada tabla cuenta con una llave primaria y sus atributos están definidos por un nombre, tipo y característica especiales de nulidad, obligatoriedad, valores por defecto y secuencias. Es necesario resaltar que hemos decidido conveniente el generar una tabla especial por cada estación para el almacenamiento de los valores horarios. De esta manera, además de la tabla donde se almacenan los particulares de las N estaciones deseadas, otras N tablas existen como recipiente de los datos de las tantas estaciones. Uno de los valores almacenados en la tabla de datos de las estaciones es el nombre de la tabla donde se encuentran sus datos. Dentro de estas tablas recipientes, el valor de la fecha es siempre la llave primaria y le acompañan los 24 valores horarios para ese día en particular, asegurándose así la consistencia, y al mismo tiempo evitándose la necesidad de tener una columna que identifique la estación a la que pertenecen los datos o, peor aún, tener una sola tabla con todos los datos de todas las estaciones.

Las demás tablas son relativamente sencillas, máxime que no tienen relación especial entre sí. Funcionan como entes separados para efectos prácticos y permiten las operaciones que se desea realizar con competencia. Dos de ellas, la de estaciones y la de contactos, cuentan con una llave primaria que es un número secuencial generado dentro de la misma base de datos. Este número es consecutivo, aumenta en razón de uno cada vez que se acciona la secuencia y permite identificar tanto a estaciones como a comentarios de una forma práctica.

## 4.3 Gráficas

La compañía de ingeniería de software Visual Engineering, Inc.[1], fundada en 1983 para la generación de herramientas de diseño para aplicaciones gráficas, es la proveedora de un *API* que permite la generación de gráficas de distintos tipos en archivos de imagen o de animación. Asimismo, el *API*, bastamente difundido entre los servidores de aplicaciones gráficas, permite el tratamiento de datos en tiempo real.

La esencia de la herramienta es la generación de imágenes a partir de las herramientas existentes de *AWT* y *SWING*, obteniéndose un *GUI* en el escritorio del entorno gráfico de la máquina servidor. Se deduce, por tanto, que la máquina servidor necesita tener instalado un entorno gráfico

[1] Para mayor información visitar www.ve.com

para el sistema operativo. Una vez se genera la imagen, se exporta a la máquina cliente. Esto precisa que el servidor de sesión gráfico pueda ser accesible desde todos los clientes donde se pretenda utilizar el sitio.

Desde el punto de vista de la programación JAVA, el *API* de KavaChart (el producto de VE) ha evolucionado enormemente. Con la creciente popularidad de la tecnología JAVA en los ambientes de red, los desarrolladores de VE se han preocupado por mantener al día su producto, proporcionando al implementador la amplia gama de soluciones que se utilizan para realizar los productos Web en la actualidad.

De tal manera, se comprenderá con facilidad que las versiones iniciales del *API* KavaChart incluyesen únicamente documentación y soporte para el trabajo con *applets*, pues fue el primer producto de JAVA que dominó el mercado. Después, con el desarrollo de la tecnología de lado del servidor, se generó una nueva versión que incluía la posibilidad de generar elementos en el servidor como los *servlets*. Esto evitaba la necesidad de transmitir un programa completo desde el servidor para ejecutar un *applet* en la máquina cliente. El trabajo se realiza en un equipo de mayor capacidad que la que poseían los clientes, enviando únicamente la imagen para ser mostrada en un navegador. La última versión, la cuarta (actualmente para descarga la 4.2.2), es realmente una herramienta revolucionaria. Incluye funcionalidad para la generación de gráficas en imagen, en *applet* , en *servlet*, en *JSP*, en *ASP* o en animación. Asimismo, se provee una colección de *JavaBeans* para facilitar la especificación de las características de las gráficas.

La variedad de las gráficas realizables es notable, incluyéndose algunas de uso muy específico, como las gráficas de burbuja, hasta otras con gran procesamiento de datos, como las gráficas de dispersión. Por cierto, estas últimas incluyen la posibilidad de encontrar la recta adecuada para el conjunto de puntos, inclusive la ordenada al origen y la pendiente. La lista completa de posibles gráficas se muestra a continuación:

| De Línea | De Barra | De Columna |
|---|---|---|
| Línea | Barra | Columnas |
| Línea con Leyenda | Barra Fechada | Columna Fechada |
| Línea Discontinua | Barra Apilada | Columna Apilada |
| Línea Fechada | | |
| Regresión Lineal | | |

| Otras | | |
|---|---|---|
| Area | Ejes gemelos | Polar |
| Burbuja | Gant | Tacómetro |
| Combinadas | Pie | |

**Tabla 4. 2 Tipos de Gráficas**

Es conveniente en este momento hacer una descripción más detallada de la estructura del *API* KavaChart. Una gráfica está compuesta siempre, sin importar el tipo, por los siguientes elementos: Ejes (Axis), área de mapeo de datos (Plotarea), fondo (Background), representación de los datos (DataRepresentation), leyenda (Legend), datos (Datum y Datasets). Cada uno de estos elementos se añade para crear un conjunto que se expresa a la postre como una gráfica. Como puede comprenderse, esta modularización es resultado de varias versiones de trabajo y permite una optimización al momento de particularizar una gráfica.

La herramienta KavaChart acomoda los datos dentro de las clases Datum y Dataset. Un objeto de la clase Datum almacena la información de una vista en particular, inclusive los valores, etiquetas e información gráfica. Es decir los elementos Datum representan las rebanadas de una gráfica de Pie, las barras de una de barras verticales u horizontales, los puntos de unión en una de línea o de dispersión y los vértices en una de área. Ahora bien, los objetos Datum se agrupan en vectores para formar los Datasets, lo cual puede asimilarse a un contenedor, con el añadido de que, además, sirve para fijar algunos atributos para cada serie de datos como el nombre, el color de línea o relleno, o la imagen icónica que se desea para representar el grupo de datos. No importando el método por el que se especifiquen los objetos Datum y Dataset (propiedades de *JSP*, parámetros de *applet* , utilerías de servidor o *beans*) la información invariablemente termina organizada en estos contenedores para su graficación.

Los datos se mapean dentro del área de ploteo creando escalas en los ejes y transformando los datos geométricamente dentro de un área específica de la gráfica. Esta región normalmente posee ejes X y Y, dado que la mayoría de las gráficas son bidimensionales. No obstante, este no es siempre el caso, como sucede con las gráficas de Pie, donde no hay ejes, o con las gráficas polares (de Kiviat), donde hay más de dos ejes. El tipo de representación de datos (DataRepresentation) es el que determina a la postre la vista del producto terminado.

Varios objetos cooperan para la construcción del resultado gráfico de estas representaciones de datos. Primeramente, el "eje Y" accede a las clases Dataset existentes para determinar cómo escalar al eje. Busca y encuentra los valores mínimo y máximo, generando una escala que

reporte una buena vista, así como una proporción útil. Naturalmente, en caso de que así se desee, es posible especificar el valor mínimo y máximo que se desea para cada uno de los ejes. En especial, puede resultar interesante el relocalizar el valor mínimo de una gráfica de manera que el cero del eje sea el punto de referencia inferior. Acto seguido, el "eje X" realiza la misma rutina para los valores de las clases Dataset.

El siguiente paso corre a cargo del objeto DataRepresentation, quien utiliza los valores calculados por las clases de los ejes, combinados con información de localización y tamaño del objeto Plotarea, para determinar la geometría adecuada. Finalmente, el objeto DataRepresentation utiliza los valores, colores y etiquetas de los Dataset para acomodar los elementos dentro del área de mapeo (Plotarea). Para las gráficas más sencillas, usualmente hay nada más un área de mapeo, un tipo de representación de datos y un par de ejes. Esta configuración básica puede complicarse para formar productos más complejos que incluyan varios conjuntos como el descrito.

Casi todos los atributos gráficos del *API* están almacenados en objetos de tipo Gc. Esta clase Gc contiene colores y tipos de línea, colores y tipos de relleno, imágenes, gradientes y texturas. Cada componente de los diagramas utiliza uno o más objetos para almacenar esta información. Por ejemplo, un Dataset utiliza un objeto Gc para definir los colores de una serie de datos, el color de línea para una representación de datos de tal estilo está definida por la propiedad LineColor; el objeto Legend utiliza la propiedad Icon del mismo objeto para definir la imagen que representa a la serie de datos; también la clase Axis depende de los colores y atributos almacenados en Gc para generar las líneas de los ejes, los marcadores de valores y las líneas auxiliares; incluso Plotarea y Background, para definir el área de mapeo y fondo, recurren a la clase Gc para obtener los valores de colores de relleno y borde.

De la característica de *JavaBeans*, incorporada al *API* KavaChart, hemos de argumentar a favor. En versiones anteriores de esta herramienta, el particularizar una gráfica de acuerdo con necesidades específicas era un proceso tedioso e incluso confuso, ya que era imperativo seguir un camino tortuoso para la modificación de cada elemento de la gráfica. La misma estructura llena de interfases, clases abstractas y herencias dificultaba cualquier intento de modificación una vez generado el código de la gráfica. La existencia de este escollo es atribuible principalmente a la deficiencia en la generación de las representaciones. Una vez incorporados los datos y leyendas al dibujo, éste quedaba definido en su totalidad y era indispensable recurrir a un complicado proceso para modificar cualquier atributo.

Como ejemplo puede considerarse el caso en que se desease indicar a la gráfica que el mínimo valor de las ordenadas fuese cero. Para lograr este objetivo, se debía recurrir a los métodos getter del tipo de gráfica generado. Del dibujo, se obtenía al objeto "eje Y", pero no relacionado con una referencia de tipo "eje Y", puesto que el método setter para esta propiedad recibe un objeto "eje Y" genérico (para cualquier gráfica), de modo que el objeto "eje Y" debía ser acogido por una referencia de tipo interfase para los "ejes Y". Una vez hecho esto, el alterar cualquier propiedad del eje es relativamente sencillo, a través de los distintos métodos disponibles. Finalmente, el objeto "eje Y" apuntado por la referencia de interfase "eje Y" se incorpora nuevamente al dibujo mediante uno de los métodos setter.

Este proceso dolorosamente complejo, principalmente debido a la falta de documentación adecuada de la estructura de clases, ha sido simplificado enormemente gracias a la inclusión de los *Beans* en el esquema del KavaChart. Los *Beans* permiten la definición de propiedades mediante métodos getter y setter con asombrosa facilidad. Esto se traduce en la oportunidad de generar un objeto de gráfica de cualquier tipo con una referencia de tipo *Bean*. Esta referencia apunta al objeto y contiene todos los métodos adecuados para parametrizar la gráfica. Debido a esta previsión, el proceso de alterar el menor valor de un eje, como se discutió previamente, se simplifica hasta el punto de simplemente llamar al método que modifica la propiedad deseada. El método tiene como firma "public void setProperty(String propiedad, String valorDePropiedad)".

A pesar de las bondades mostradas, la optimización que es realmente asombrosa y útil es la que se refiere a la generación de las gráficas una vez que éstas han sido particularizadas. Entiéndase que hasta ahora el proceso de generación de la gráfica mediante valores y etiquetas era previo a la modificación de cualquier parámetro. Esto implicaba que cualquier alteración redundase en la recreación del dibujo para incorporar las nuevas facetas. Esto es un procedimiento ineficiente, si bien eficaz. Los *Beans* permiten la fabricación de elementos gráficos predefinidos. Cada clase asignable a una referencia *Bean* cuenta con un constructor sobrecargado para poder crear objetos a partir de las propiedades definidas en un objeto de la clase Properties, de tal curso que, al invocar al constructor, éste obtenga las distintas propiedades con las que se ha de generar el dibujo para no alterarlo posteriormente. Esto elimina la doble generación de la representación de datos.

Como último comentario acerca de esta herramienta es necesario hacer referencia a la posibilidad de almacenar las gráficas generadas en el servidor. Esta faceta puede resultar útil para casos de alta demanda, ya que, en vez de generar el dibujo cada vez, éste se almacena en algún

repositorio determinado para ser enviado cada ocasión que se requiera. De otro modo, cada vez que existiese una petición habría necesidad de generar la imagen. Al respecto es adecuado señalar que puede decidirse un tiempo de vida en el servidor para cada gráfica. Asìmismo, dado que al almacenarlas forman parte del sistema de archivos, la gráficas son susceptibles a borrarse, renombrarse, copiarse, ligarse y modificarse de acuerdo con los permisos del sistema operativo.

Para utilizar varias de las propiedades especiales de la herramienta generadora, se requiere definir un sitio donde las imágenes sean almacenadas. Este es el caso de la generación de los comentarios de posición sobre una imagen. De manera que un usuario pueda obtener una gráfica en la que se muestren etiquetas al colocar el ratón sobre alguna zona, debe haber una copia de la imagen en el servidor. La ruta dentro del servidor donde se almacenan las imágenes se puede especificar como una más de las propiedades de los *Beans*. No obstante, en este caso, es importante recordar que la escritura del mapa está sujeta a los permisos existentes para el sistema de archivos. La mejor estrategia para contar con una ruta adecuada es utilizar la clase Magic, que es proporcionada por VE dentro del archivo war. Esta clase permite evaluar la posibilidad de escritura de la imagen sin tener que recurrir necesariamente al desarrollo completo vía Web.

## 4.4 Formato y Estilo

### 4.4.1 JavaScript

*JavaScript* es el lenguaje que permite interactuar con el navegador de manera eficiente y eficaz, proporcionando a las páginas Web dinamismo y vida. Fue desarrollado por Netscape para incrementar las funcionalidades del lenguaje HTML. Sus características más importantes son:

- Es un lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias ***JavaScript*** contenidas en una página HTML y ejecutarlas adecuadamente.
- Es un lenguaje orientado a eventos. Cuando un usuario acciona un enlace o mueve el apuntador del mouse sobre una imagen, se produce un evento. Mediante ***JavaScript*** se pueden desarrollar ***scripts*** que ejecuten acciones en respuesta a estos eventos.

- Es un lenguaje orientado a objetos. El modelo de objetos de ***JavaScript*** está reducido y simplificado, pero incluye los elementos necesarios para que los ***scripts*** puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador.

**El problema de las versiones**

*JavaScript* fue desarrollado por Netscape, y la primera versión, la 1.0, fue introducida por primera vez en su navegador Netscape Navigator 2.0. *JavaScript* 1.1 representó una mejora en las características del lenguaje, y se incluyó en el Navigator 3.0. Microsoft intentó dar soporte a la primera versión de *JavaScript* en el Internet Explorer 3.0 (con el nombre de Jscript). Sin embargo, Jscript resultó poco fiable y estaba plagado de ***bugs***.

ECMA impulsó un estándar para *JavaScript*, que Microsoft introdujo en el Internet Explorer 4.0. Este estándar comparte la mayoría de las características con *JavaScript* 1.2, la nueva versión que apareció con el Netscape Navigator 4.0. Sin embargo, no son totalmente equivalentes, lo que dificulta la creación de ***scripts*** que funcionen correctamente en los dos navegadores.

Las últimas versiones del Navigator (a partir de la 4.06) implementan ya *JavaScript* 1.3, que es totalmente compatible con ECMA-262. Ver tabla 4.3.

| Navegador | Versión de JavaScript | Soporte ECMA |
|---|---|---|
| Nestcape 2 | JavaScript 1.0 | - |
| Internet Explorer 2 | No soporta JavaScript | - |
| Nestcape 3 | JavaScript 1.1 | - |
| Internet Explorer 3 | JavaScript 1.0 | - |
| Nestcape 4 | JavaScript 1.2 - 1.3 incompleta | ECMA-262-compliant hasta la version 4.5 |
| Internet Explorer 4 | JavaScript 1.2 | ECMA-262-compliant |
| Nestcape 6 | ECMA compliant JavaScript 1.4 | Full ECMAScript-262 |
| Internet Explorer 5 o Superior | ECMA compliant JavaScript 1.3 | Full ECMAScript-262 |

**Tabla 4. 3 Navegadores y JavaScript**

Las diferentes versiones de *JavaScript* han sido finalmente integradas en un estándar denominado ECMAScript-262. Dicho estándar ha sido realizado por la organización ECMA dedicada a la estandarización de información y sistemas de comunicación. Las versiones actuales de los navegadores soportan este estándar.

**Uso en la Tesis**

Para cubrir nuestras necesidades de integridad de datos, recurrimos al *JavaScript* principalmente para validación en los distintos formularios. De esta manera, aseguramos que el usuario o administrador los llene apropiadamente, evitando que los errores se generen en el lado del servidor. La validación se realiza en el lado del cliente, ahorrando tiempo y asegurando la consistencia de los datos en la base de datos.

Por lo tanto, utilizamos un *script* genérico para validar todas las situaciones donde el usuario se encuentra con un formulario que debe llenar forzosamente. Un ejemplo puede ser el formulario de entrada tanto al área de usuarios como al área de administración. En ambos casos debe proporcionarse un nombre de usuario y contraseña al formulario para que los datos sean enviados al servidor y sean validados. El *script* genérico toma a todos los campos del formulario como parte de un arreglo y repasa tal arreglo hasta encontrar un campo vacío. Al momento de realizar el hallazgo, el programa arroja un mensaje de alerta indicando cuál campo está vacío y debe ser llenado. En caso de que no se produzca un error, la forma es enviada normalmente.

Otro *script*, siempre útil, es aquel que valida las cadenas de los correos electrónicos. Todo correo electrónico cumple con ciertas características, entre las cuales se incluyen: tener el carácter arroba (@), tener al menos un punto en la cadena posterior a la arroba, no tener comas o diagonales, no tener dominios que terminen en números, entre otras. Generamos un *script* que realiza esta función; muestra una alerta con la explicación del error si existe algún fallo, o permite la inserción o modificación, si no lo existe.

Por último, dada la tendencia a que los usuarios se decidan por contraseñas fáciles, cortas y poco diversas, generamos un *script* para la verificación de la contraseña de usuario o administrador. Al momento de dar de alta, el administrador proporciona una contraseña en el formulario apropiado, junto con su confirmación. Al accionar el botón de enviar, el *script* verifica que la contraseña y la verificación sean iguales y que ambas tengan al menos seis caracteres. En caso de que exista un error en las contraseñas, se muestra una alerta con la causa. En caso contrario, se permite continuar con el proceso.

### 4.4.2 Hojas de estilo

Una hoja de estilo permite aumentar el control del diseñador sobre cómo se verán sus páginas Web, asociando un conjunto de propiedades de formato físico a los elementos estándar del lenguaje HTML. Ver figura 4.1.


**Figura 4.1 Diagrama de la relación HTML - CSS**

De esta forma, el resultado final que observa la persona en su navegador es una mezcla entre las características predefinidas para cada comando HTML y la hoja de estilo. Una gran virtud de esta técnica es que permite lograr una presentación muy uniforme, ya que basta definir un estilo una sola vez para que éste se aplique a todos los elementos del mismo tipo que existan en una página.

**Ventajas**

Si bien puede parecer engorroso al principio escribir una serie de instrucciones en código para lograr un propósito, la idea que está detrás es que se escriba este conjunto de instrucciones una sola vez, en un archivo separado, y esta hoja de estilo sea usada por una serie de páginas (por ejemplo, por todas las páginas de un mismo servicio Web). De este modo, se consigue una presentación uniforme con un mínimo de esfuerzo, ya que en el caso de que se desee cambiar algo, basta editar un archivo y este cambio se aplicará a todas las páginas que utilicen ese estilo. El uso de hojas de estilo evita el uso excesivo de imágenes, que, además de que cuesta mucho generarlas y mantenerlas a lo largo de las actualizaciones, ocupan bastante más tiempo de transferencia que unas cuantas líneas de código extra.

**Navegadores que lo soportan**

Esta tecnología no es universal, por lo que no todos los navegadores la soportan. En concreto, sólo los navegadores de Netscape versiones de la 4 en adelante y de Microsoft a partir de la versión 3 son capaces de comprender los estilos en sintaxis *CSS*. Además, cabe destacar que no todos los navegadores implementan las mismas funciones de hojas de estilo, por ejemplo, Microsoft Internet Explorer 3 no soporta todo lo relativo a capas. Esto quiere decir que se debe usar esta tecnología con cuidado, ya que muchos usuarios no podrán ver los formatos que se apliquen a las páginas con *CSS*. Así pues, las hojas de estilos deben emplearse cuando se esté seguro de que no supondrán un problema.

Dentro del creciente desarrollo del ambiente de las páginas Web, siempre es necesario comprobar que cualquier aspecto de las hojas de estilo que se piense utilizar funcione en los principales navegadores. Normalmente, toma algún tiempo que las nuevas ideas se conviertan en estándar. Posteriormente, esas ideas son incorporadas a los intérpretes de los navegadores y pueden utilizarse sin problema.

**Uso en la Tesis**

Las hojas de estilo nos permitieron dar un formato unificado a la interfaz gráfica de este proyecto. Generalizando, cuatro aspectos principales son los que fueron manejados con hojas de estilo. Primeramente, el tipo, color y tamaño de letra; segundo, el fondo de las páginas; tercero, las flechas y barrar de navegación; cuarto y último, los enlaces o ligas.

En el caso de los formatos de la fuente utilizada, determinamos utilizar letra Verdana, que es suficientemente clara, dentro de un estilo adornado, que permite evitar una impresión de máquina de escribir y al mismo tiempo legibilidad. El color para la letra, excepto en los encabezados de las tablas (donde se utilizó el blanco), se asimiló lo más posible al color del logo del Instituto de Geofísica, que es un tono de azul marino. Para los títulos, se determinó un tamaño y otro más pequeño para todo el demás texto.

El fondo de las páginas surgió como una necesidad para los momentos en los que la información de las páginas no era suficiente y existía demasiado espacio en blanco. Se generó a partir de varias fotografías relacionadas con el tema, que se agruparon y se degradaron para dar un aspecto de fondo de agua. Las hojas de estilo se utilizaron en esta sección para lograr que el fondo apareciese siempre en la parte inferior de la pantalla y centrado.

Para el caso de resoluciones de pantalla inferiores a 1024 x 768 pixeles (recomendada para la mejor apreciación del sitio), la navegación por las

pantallas del sitio puede requerir de las flechas y barras de navegación. Éstas aparecen siempre que existe información fuera del campo de visión para una pantalla en un momento dado. Al accionarlas puede recorrerse hacia los lados o hacia arriba y abajo el código HTML interpretado. No obstante, si no se indica de otra forma, las flechas y barras aparecen en color gris, con contornos en negro. Nuevamente, estos colores desentonaban con el formato gráfico general del sitio, por lo que se recurrió a las hojas de estilo para la especificación de los colores.

Finalmente, una forma de dar vitalidad y dinamismo a un sitio es el empleo de las hojas de estilo. Evita la necesidad de los molestos "plug-ins" y la necesidad de transmitir elementos que pueden ser de tamaño poco conveniente y que se comporten de forma distinta en varios navegadores. Para no utilizar botones que pueden resultar ineficaces, si son de formato Flash, u otros que pueden resultar poco atractivos, como los de HTML, se puede recurrir a las hojas de estilo para convertir a las ligas o enlaces en botones. Este procedimiento proporciona la ventaja de que se utiliza un elemento del HTML con un comportamiento extremadamente regular en la vasta mayoría de los navegadores. Además, las ligas son fácilmente combinables con el *JavaScript* para accionar formularios, abrir nuevas ventanas o borrar información.

El tema general de color y forma del sitio no está especificado por hojas de estilo, pero debe tratarse dentro de este apartado. De hecho, la mayor parte de los colores para el sitio fueron escogidos en base al concepto del mar y sus tonalidades. El azul, además de ser un color sobrio y elegante, transmite la calma, la constancia y el lento evolucionar de los fenómenos oceánicos.

# 5. Pruebas

## 5.1 Servidor Web (Concurrencia)

**Conexiones en concurrencia**

Las pruebas de stress para conexiones concurrentes en el servidor fueron realizadas mediante un programa de software llamado Webserver Stress Tool de la compañía Paessler. La versión 6.20 de este software puede obtenerse como demo en el sitio http://www.solutionsreview.com/Paessler_Web_Stress_Tool_Professional_Edition.asp.

Las pruebas se realizaron conjuntamente tanto a la raíz del servidor, como a la zona de administración. Se configuraron los parámetros de la siguiente manera:

- Terminar hasta que cada usuario realizase 10 clicks.
- Número de usuarios: 10
- Tiempo entre clicks aleatorio.
- Escribir resultados a la bitácora cada 10 segundos.

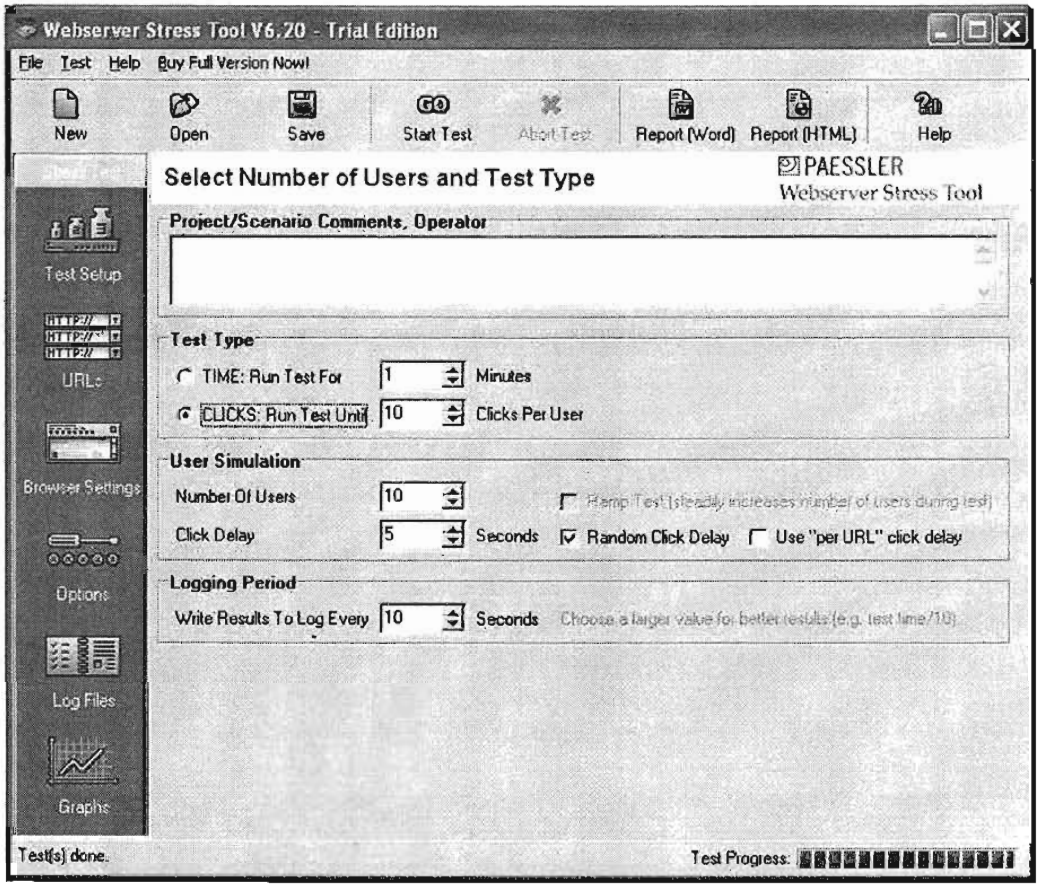



**Figura 5.1 Configuración de parámetros para pruebas de stress**

Los resultados de la prueba incluyen distintas gráficas y comentarios que se presentan a continuación, pero baste el resumir para efectos de evaluación que el servidor parece gozar de excelente salud en lo que respecta al manejo de peticiones simultáneas.

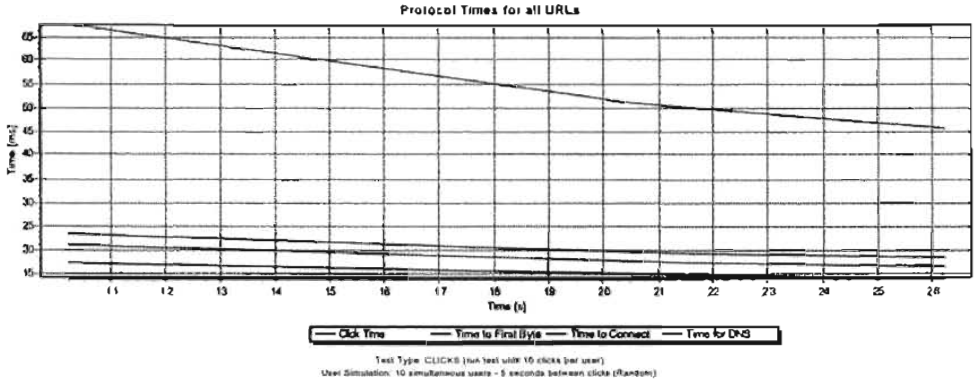



**Figura 5.2 Tiempos de Protocolo para cada URL**

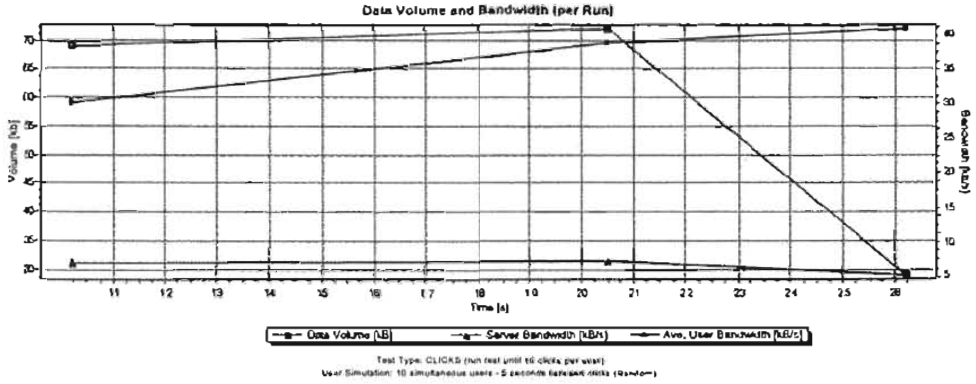



**Figura 5.3 Volumen de datos y ancho de banda**

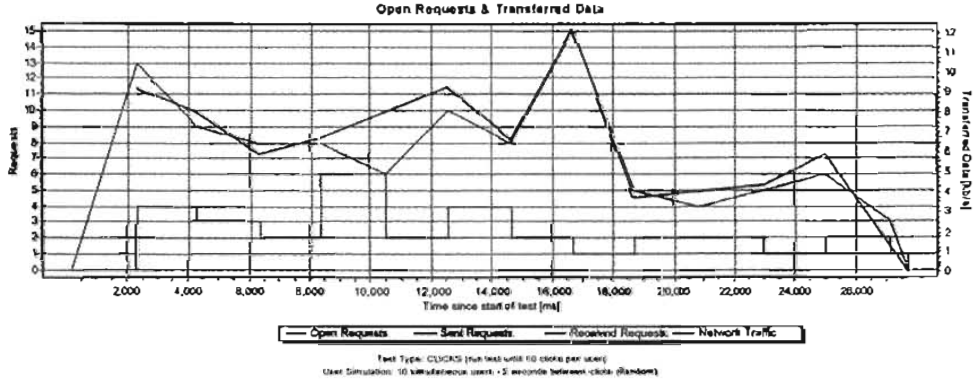



**Figura 5.4 Peticiones abiertas y transferencia de datos**

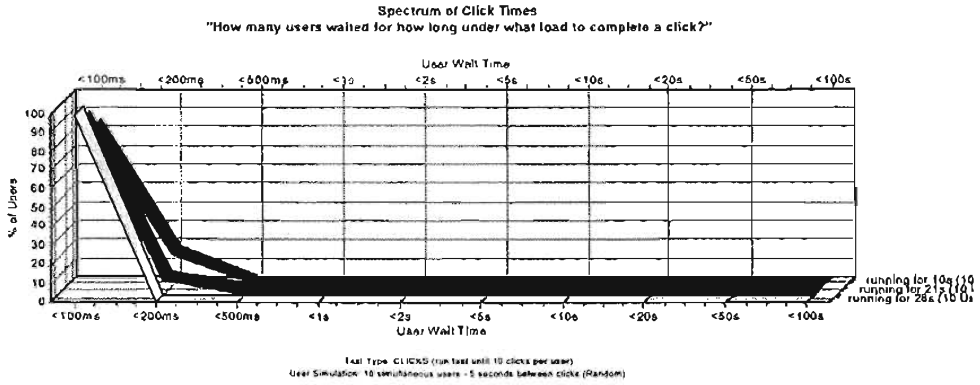



**Figura 5.5 Tiempo de espera después de la petición**

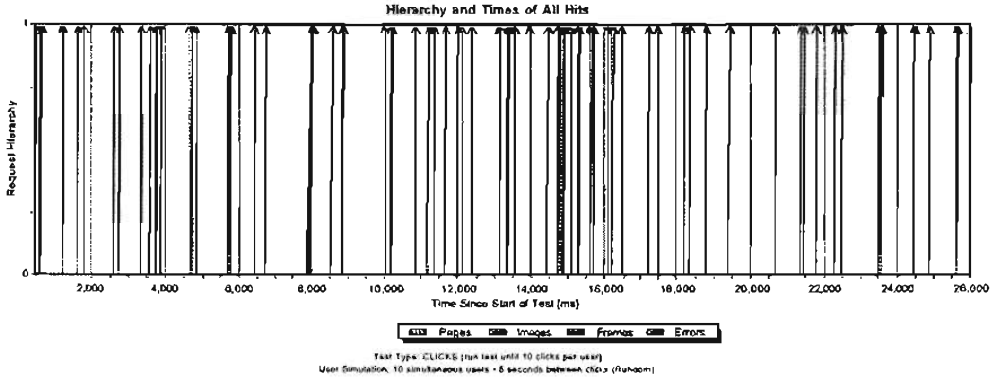



**Figura 5.6 Instantes de las peticiones realizadas**

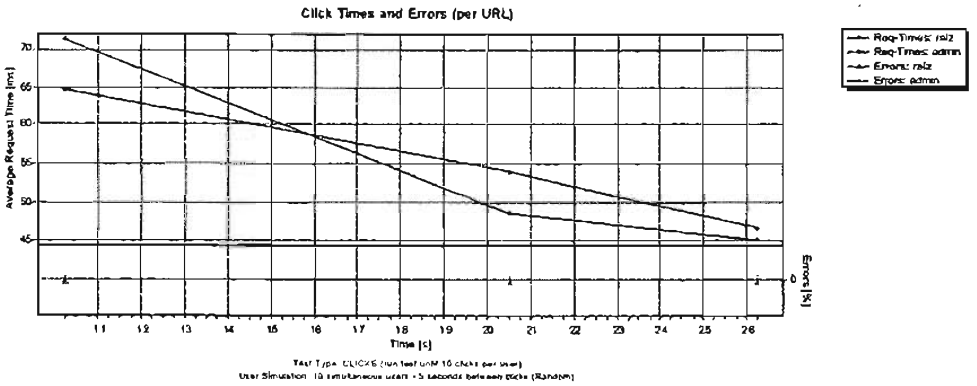



**Figura 5.7 Instantes de petición y errores**

**Summary Log**

** Test Logfile by Webserver Stress Tool, Version 6.20 Trial Edition **

Test run on 04/10/2004 01:06:41 p.m.

Results of period #1:

**********

Completed Clicks: 43 with 0 Errors (=0%)

Average Click Time for 10 Users: 68 ms

Successful clicks per Second: 4.267 (equals 15360.172 Clicks per Hour)

Results of period #2:

**********

Completed Clicks: 41 with 0 Errors (=0%)

Average Click Time for 10 Users: 51 ms

Successful clicks per Second: 3.989 (equals 14359.03 Clicks per Hour)

Results of period #3:

**********

Completed Clicks: 16 with 0 Errors (=0%)

Average Click Time for 10 Users: 46 ms

Successful clicks per Second: 2.807 (equals 10105.852 Clicks per Hour)

Results of complete test

**********

** Results per URL for complete test **

URL#1 (raiz): Average Click Time 58 ms, 50 Clicks, 0 Errors

URL#2 (admin): Average Click Time 57 ms, 50 Clicks, 0 Errors

Total Number of Clicks: 100 (0 Errors)

Average Click Time of all URLs: 57 ms

**URLs to Test**

| URL# | Name | URL |
|---|---|---|
| 1 | raíz | http://132.248.182.198/dan/ |
| 2 | admin | http://132.248.182.198/dan/administracion/ |

**Tabla 5. 1 URLs probados**

**Results per User**

| User No. | Clicks | Hits | Errors | Avg. Click Time [ms] | Bytes | kB/s | Cookies |
|---|---|---|---|---|---|---|---|
| 1 | 10 | 10 | 0 | 66 | 8320 | 12.668 | |
| 2 | 10 | 10 | 0 | 58 | 25910 | 44.911 | |
| 3 | 10 | 10 | 0 | 77 | 8320 | 10.823 | |
| 4 | 10 | 10 | 0 | 66 | 25910 | 39.081 | |
| 5 | 10 | 10 | 0 | 65 | 8320 | 12.859 | |
| 6 | 10 | 10 | 0 | 46 | 25910 | 56.636 | |
| 7 | 10 | 10 | 0 | 45 | 8320 | 18.552 | |
| 8 | 10 | 10 | 0 | 51 | 25910 | 51.272 | |
| 9 | 10 | 10 | 0 | 38 | 8320 | 21.96 | |
| 10 | 10 | 10 | 0 | 63 | 25910 | 41.347 | |

**Tabla 5. 2 Resultados por Usuario**

**Results per URL**

| URL No. | Name | Clicks | Errors | Errors [%] | Time Spent [ms] | Avg. Click Time [ms] |
|---|---|---|---|---|---|---|
| 1 | geofisica | 20 | 0 | 0 | 1036 | 52 |
| 2 | otro | 15 | 0 | 0 | 576 | 38 |
| 3 | admin | 15 | 0 | 0 | 669 | 45 |

**Tabla 5. 3 Resultados por URL**

Test Logfile by Webserver Stress Tool, Version 6.20 Trial Edition

18/06/2004 10:24:24:

Test run on 18/06/2004 10:24:24

Project and Scenario Comments, Operator **

Test Setup **

Test Type: CLICKS (run test until 5 clicks per user)

User Simulation: 10 simultaneous users- 20 seconds between clicks (Random)

Logging Period: Log every 10 seconds

URLs

Sequencing: Users always click the same URL (to spreads load evenly on all URLs, set number of users to number of URLs!)

3 URLs

URL#1: GET http://132.248.182.198 POSTDATA= Click Delay=

URL#2: GET http://132.248.182.198/dan/ POSTDATA= Click Delay=

URL#3: GET http://132.248.182.198/dan/administracion/ POSTDATA= Click Delay=

Browser Settings **

Browser Simulation:

User Agent: Mozilla/4.0 (compatible; MSIE 6.0b; Windows NT 5.0)

HTTP Request Timeout: 120 s

Recursive Browsing / HTML Parsing:

Options

Advanced Settings:

Logging:

Write detailed log(s)

Timer: not enabled

Using Local IPs:

192.9.200.113

Advanced Data Merging Features:

Client System

Windows 2000 V5.0 (Build 2195) Service Pack 4, CPU Proc. Type 586 (Rev. 772) at 2793 MHz,

543 MB available RAM of 795 MB total physical RAM, 894 MB available pagefile, 9391 MB free disk space on C:

Client's MAC address: 00-0D-56-D6-81-9E

Time measurement resolution: 2,793651 µsec, clock runs at 4MHz

Test is starting

---

Creating Users...

Setting Priority of WebStress.exe Process to "HIGH"

Test preparations done

Controller thread started

Switching to 10 users.

18/06/2004 10:24:34:

Results of period #1:

**********

Analyzed Time span of this period: 10193 msec

Sent/Received Requests: 11 / 10 (=9,09% not answered in this period)

Completed Hits: 10 (including images, frames etc.)

Completed Clicks: 10 with 0 Errors (=0%)

Results for Images for this period:

Image Hits: 0 with 0 Errors

Spectrum of Click Times

80% of All Users waited <100ms

20% of All Users waited <200ms

0% of All Users waited <500ms

0% of All Users waited <1s
0% of All Users waited <2s
0% of All Users waited <5s
0% of All Users waited <10s
0% of All Users waited <20s
0% of All Users waited <50s
0% of All Users waited <100s

Measured Times:
Average *DNS* Time for 10 Users: 23 ms
Average Time to Connect for 10 Users: 32 ms
Average Time to First Byte for 10 Users: 28 ms
Average Click Time for 10 Users: 65 ms

Hits per Second: 0,981 (equals 3531,84 Hits per Hour)
Successful clicks per Second: 0,981 (equals 3531,84 Clicks per Hour)

Results per URL for this Period:
URL#1 (geofisica): Average Click Time 83 ms, 4 Clicks, 0 Errors,
URL#2 (otro): Average Click Time 49 ms, 3 Clicks, 0 Errors,
URL#3 (admin): Average Click Time 59 ms, 3 Clicks, 0 Errors,
Average Click Time of all URLs: 65 ms

Resulting page statuscodes for this period:
10x "200" (=OK)

Datavolume and Bandwidth for this period:
Average User Bandwidth: 30,674 kByte/s
Total Bytes: 13581 Bytes (13 kB) (Throughput ~1 kByte/sec)

18/06/2004 10:24:45:
Results of period #2:
**********

Analyzed Time span of this period: 10240 msec
Sent/Received Requests: 13 / 14 (=-7,69% not answered in this period)
Completed Hits: 14 (including images, frames etc.)
Completed Clicks: 14 with 0 Errors (=0%)

Results for Images for this period:
Image Hits: 0 with 0 Errors

Spectrum of Click Times
100% of All Users waited <100ms
0% of All Users waited <200ms
0% of All Users waited <500ms
0% of All Users waited <1s
0% of All Users waited <2s
0% of All Users waited <5s
0% of All Users waited <10s
0% of All Users waited <20s
0% of All Users waited <50s
0% of All Users waited <100s

Measured Times:
Average *DNS* Time for 10 Users: 10 ms
Average Time to Connect for 10 Users: 14 ms
Average Time to First Byte for 10 Users: 13 ms
Average Click Time for 10 Users: 34 ms

Hits per Second: 1,367 (equals 4921,963 Hits per Hour)
Successful clicks per Second: 1,367 (equals 4921,963 Clicks per Hour)

Results per URL for this Period:
URL#1 (geofisica): Average Click Time 33 ms, 5 Clicks, 0 Errors,
URL#2 (otro): Average Click Time 33 ms, 6 Clicks, 0 Errors,
URL#3 (admin): Average Click Time 38 ms, 3 Clicks, 0 Errors,
Average Click Time of all URLs: 34 ms

Resulting page statuscodes for this period:
14x "200" (=OK)

Datavolume and Bandwidth for this period:
Average User Bandwidth: 34,919 kByte/s
Total Bytes: 16905 Bytes (16 kB) (Throughput ~2 kByte/sec)

18/06/2004 10:24:55:
Results of period #3:
**********
Analyzed Time span of this period: 10247 msec
Sent/Received Requests: 11 / 11 (=0% not answered in this period)
Completed Hits: 11 (including images, frames etc.)
Completed Clicks: 11 with 0 Errors (=0%)

Results for Images for this period:
Image Hits: 0 with 0 Errors

Spectrum of Click Times
90,91% of All Users waited <100ms
9,09% of All Users waited <200ms
0% of All Users waited <500ms
0% of All Users waited <1s
0% of All Users waited <2s

0% of All Users waited <5s
0% of All Users waited <10s
0% of All Users waited <20s
0% of All Users waited <50s
0% of All Users waited <100s

Measured Times:
Average *DNS* Time for 10 Users: 17 ms
Average Time to Connect for 10 Users: 21 ms
Average Time to First Byte for 10 Users: 25 ms
Average Click Time for 10 Users: 50 ms

Hits per Second: 1,073 (equals 3864,364 Hits per Hour)
Successful clicks per Second: 1,073 (equals 3864,364 Clicks per Hour)

Results per URL for this Period:
URL#1 (geofisica): Average Click Time 62 ms, 5 Clicks, 0 Errors,
URL#2 (otro): Average Click Time 37 ms, 2 Clicks, 0 Errors,
URL#3 (admin): Average Click Time 40 ms, 4 Clicks, 0 Errors,
Average Click Time of all URLs: 50 ms

Resulting page statuscodes for this period:
11x "200" (=OK)

Datavolume and Bandwidth for this period:
Average User Bandwidth: 36,251 kByte/s
Total Bytes: 16168 Bytes (16 kB) (Throughput ~2 kByte/sec)

18/06/2004 10:25:05:
Results of period #4:
**********

Analyzed Time span of this period: 10248 msec
Sent/Received Requests: 8 / 8 (=0% not answered in this period)
Completed Hits: 8 (including images, frames etc.)
Completed Clicks: 8 with 0 Errors (=0%)

Results for Images for this period:
Image Hits: 0 with 0 Errors

Spectrum of Click Times
100% of All Users waited <100ms
0% of All Users waited <200ms
0% of All Users waited <500ms
0% of All Users waited <1s
0% of All Users waited <2s
0% of All Users waited <5s
0% of All Users waited <10s
0% of All Users waited <20s
0% of All Users waited <50s
0% of All Users waited <100s

Measured Times:
Average *DNS* Time for 10 Users: 12 ms
Average Time to Connect for 10 Users: 16 ms
Average Time to First Byte for 10 Users: 15 ms
Average Click Time for 10 Users: 40 ms

Hits per Second: 0,781 (equals 2810,344 Hits per Hour)
Successful clicks per Second: 0,781 (equals 2810,344 Clicks per Hour)

Results per URL for this Period:
URL#1 (geofisica): Average Click Time 38 ms, 3 Clicks, 0 Errors,
URL#2 (otro): Average Click Time 39 ms, 3 Clicks, 0 Errors,
URL#3 (admin): Average Click Time 42 ms, 2 Clicks, 0 Errors,
Average Click Time of all URLs: 40 ms

Resulting page statuscodes for this period:
8x "200" (=OK)

Datavolume and Bandwidth for this period:
Average User Bandwidth: 31,502 kByte/s
Total Bytes: 10162 Bytes (10 kB) (Throughput ~1 kByte/sec)

18/06/2004 10:25:15:
Results of period #5:
**********
Analyzed Time span of this period: 10248 msec
Sent/Received Requests: 5 / 5 (=0% not answered in this period)
Completed Hits: 5 (including images, frames etc.)
Completed Clicks: 5 with 0 Errors (=0%)

Results for Images for this period:
Image Hits: 0 with 0 Errors

Spectrum of Click Times
100% of All Users waited <100ms
0% of All Users waited <200ms
0% of All Users waited <500ms
0% of All Users waited <1s
0% of All Users waited <2s

0% of All Users waited <5s
0% of All Users waited <10s
0% of All Users waited <20s
0% of All Users waited <50s
0% of All Users waited <100s

Measured Times:
Average *DNS* Time for 10 Users: 13 ms
Average Time to Connect for 10 Users: 17 ms
Average Time to First Byte for 10 Users: 16 ms
Average Click Time for 10 Users: 41 ms

Hits per Second: 0,488 (equals 1756,485 Hits per Hour)
Successful clicks per Second: 0,488 (equals 1756,485 Clicks per Hour)

Results per URL for this Period:
URL#1 (geofisica): Average Click Time 39 ms, 2 Clicks, 0 Errors,
URL#2 (otro): Average Click Time 39 ms, 1 Clicks, 0 Errors,
URL#3 (admin): Average Click Time 44 ms, 2 Clicks, 0 Errors,
Average Click Time of all URLs: 41 ms

Resulting page statuscodes for this period:
5x "200" (=OK)

Datavolume and Bandwidth for this period:
Average User Bandwidth: 36,161 kByte/s
Total Bytes: 7670 Bytes (7 kB) (Throughput ~1 kByte/sec)

18/06/2004 10:25:25:
Results of period #6:
**********

Analyzed Time span of this period: 9435 msec
Sent/Received Requests: 2 / 2 (=0% not answered in this period)
Completed Hits: 2 (including images, frames etc.)
Completed Clicks: 2 with 0 Errors (=0%)

Results for Images for this period:
Image Hits: 0 with 0 Errors

Spectrum of Click Times
100% of All Users waited <100ms
0% of All Users waited <200ms
0% of All Users waited <500ms
0% of All Users waited <1s
0% of All Users waited <2s
0% of All Users waited <5s
0% of All Users waited <10s
0% of All Users waited <20s
0% of All Users waited <50s
0% of All Users waited <100s

Measured Times:
Average *DNS* Time for 10 Users: 14 ms
Average Time to Connect for 10 Users: 18 ms
Average Time to First Byte for 10 Users: 17 ms
Average Click Time for 10 Users: 42 ms

Hits per Second: 0,212 (equals 763,104 Hits per Hour)
Successful clicks per Second: 0,212 (equals 763,104 Clicks per Hour)

Results per URL for this Period:
URL#1 (geofisica): A verage Click Time 40 ms, 1 Clicks, 0 Errors,
URL#2 (otro): 0 Clicks, 0 Errors,
URL#3 (admin): Average Click Time 44 ms, 1 Clicks, 0 Errors,
Average Click Time of all URLs: 42 ms

Resulting page statuscodes for this period:
2x "200" (=OK)

Datavolume and Bandwidth for this period:
Average User Bandwidth: 39,395 kByte/s
Total Bytes: 3419 Bytes (3 kB) (Throughput ~0 kByte/sec)

**********
Test Done, now waiting for simulated users to stop surfing
**********
Waiting for users to cool down and stop surfing...
18/06/2004 10:25:26:
All surfers died...
Setting Priority of WebStress.exe Process to "NORMAL"
Controller thread halted

Results of complete test
**********

** Results per User for complete test **

User #1: Avg. Click Time: 51 ms,5 Clicks, 5 Hits, 0 Errors, 4140 Bytes, 16,221 kB/s
User #2: Avg. Click Time: 47 ms,5 Clicks, 5 Hits, 0 Errors, 4160 Bytes, 17,792 kB/s
User #3: Avg. Click Time: 54 ms,5 Clicks,5 Hits, 0 Errors,12955 Bytes, 47,624 kB/s
User #4: Avg. Click Time: 50 ms,5 Clicks, 5 Hits, 0 Errors, 4140 Bytes, 16,724 kB/s
User #5: Avg. Click Time: 34 ms,5 Clicks, 5 Hits, 0 Errors, 4160 Bytes, 24,212 kB/s
User #6: Avg. Click Time: 39 ms,5 Clicks,5 Hits,0 Errors, 12955 Bytes, 65,622 kB/s
User #7: Avg. Click Time: 73 ms,5 Clicks, 5 Hits, 0 Errors, 4140 Bytes, 11,389 kB/s
User #8: Avg. Click Time: 34 ms, 5 Clicks, 5 Hits, 0 Errors, 4160 Bytes, 24,48 kB/s
User #9: Avg. Click Time: 40 ms,5 Clicks,5 Hits,0 Errors, 12955 Bytes, 64,806 kB/s
User #10: Avg. Click Time: 34 ms,5 Clicks,5 Hits,0 Errors, 4140 Bytes, 24,336 kB/s

** Results per URL for complete test **

URL#1 (geofisica): Average Click Time 52 ms, 20 Clicks, 0 Errors
URL#2 (otro): Average Click Time 38 ms, 15 Clicks, 0 Errors
URL#3 (admin): Average Click Time 45 ms, 15 Clicks, 0 Errors

Total Number of Clicks: 50 (0 Errors)
Average Click Time of all URLs: 46 ms

Saving summary log to "C:\Archivos de programa\Webserver Stress Tool 6 (Trial)\logs\Summary Log.txt"
Saving detailed log to "C:\Archivos de programa\Webserver Stress Tool 6 (Trial)\logs\Detailed Log.txt"
Closing "data.dat" files
Cleaning up
Done

## 5.2 Pruebas Unitarias

**Conexión a la base de datos**

- Requerimientos a cubrir

  La conexión a la base necesita de los siguientes datos:

  - Dirección del servidor
  - Puerto de la base de datos
  - Nombre de la base de datos
  - Usuario
  - Contraseña

- Condiciones prerrequisito

  Conocer los datos especificados en el punto anterior y estar habilitado como cliente en el servidor.

- Entradas de prueba

  Las pruebas a realizar en este caso incluyen verificar la conexión, realizar operaciones de la base de datos (inserts, selects, joins, deletes, updates, máximos, mínimos, promedios) y cerrar la conexión.

- Resultados de prueba

  Los datos insertados conservan la integridad deseada, excepto por aquellos datos que contienen el caracter " ' " (comilla simple). Este caracter tiene conflictos con el que es utilizado por la base de datos para delimitar las cadenas de caracteres y las fechas, por lo que es necesario anteponer una diagonal invertida (\) antes de insertar ese caracter.

- Evaluación de resultados

  La conexión es correcta. Las operaciones se realizan de manera adecuada, arrojando el resultado de cada operación para ser evaluado. Quedó claro que es necesario cerrar la conexión a la base de datos después de utilizarla, ya que, de no hacerlo, los servicios de red del servidor se dan de baja por exceso de tráfico.

**Funcionamiento de los beans de entidad**

- Requerimientos a cubrir

  Los *beans* de entidad son componentes destinados a manejar la información proveniente de o destinada a la base de datos. De esta manera, se espera de ellos el poder manejar la información correctamente una vez que ésta se ha extraído de la base de datos o bien ser un eficiente vehículo para llevarla a la misma.

- Condiciones prerrequisito

  Establecer una conexión con la base de datos.

- Entradas de prueba

  Datos extraídos de la base de datos de cada una de las tablas de entidad. Estos datos son enviados a los *beans* de entidad siempre como tipo cadena de caracteres.

- Resultados de prueba

  Los *beans* fueron capaces de generar un objeto del tipo correspondiente (con todos los atributos) a partir de un arreglo de datos de tipo cadena de caracter o un vector. Asimismo, realizaron correctamente la tarea de generar la cadena apropiada para insertar los datos en la tabla correspondiente. Finalmente, los métodos get y set de cada *bean* funcionaron adecuadamente para asignar y recuperar valores de los atributos.

- Evaluación de resultados

  Los *beans* de entidad entregaron el funcionamiento que se esperaba de ellos, facilitando el tratamiento, almacenamiento y despliegue de los datos almacenados dentro de ellos.

**Generación de comentarios**

- Requerimientos a cubrir

  Para realizar un comentario se necesitan los datos siguientes:

  - Nombre
  - Correo Electrónico
  - Organización
  - Mensaje

- Condiciones prerrequisito

  Ingresar al sitio y accionar el botón de Contacto.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | Nombre, organización, mensaje. |
| Email | Caracteres del código ASCII, conteniendo los caracteres "@" y "." | email |

**Tabla 5. 4 Entradas de prueba. Generación de comentarios.**

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Texto | Vacío. | "Hay algún campo vacío." |
| Email | Sin "@" o "."<br>Con caracteres prohibidos.<br>Formato inadecuado. | "El correo electrónico proporcionado tiene algún error." |

**Tabla 5. 5 Resultados de prueba. Generación de comentarios.**

- Evaluación de resultados

  El procedimiento es exitoso al proporcionar todos los datos. En caso de que algún campo esté vacío, el mensaje adecuado se muestra inmediatamente advirtiendo al usuario el error. En caso de que el correo electrónico proporcionado no cumpla con el formato adecuado (contener "@" y " . ", no tener espacios en blanco, tener un dominio adecuado, etc.), se muestra un mensaje de error.

**Validación de login**

- Requerimientos a cubrir

  Para realizar el proceso de login se necesitan los datos siguientes:

  - Login
  - Password

- Condiciones prerrequisito

  Ingresar al sitio y accionar el botón de entrada de usuarios o ingresar a la entrada de administradores.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | login, password. |

**Tabla 5. 6 Entradas de prueba. Validación de login.**

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Texto | Vacío. | "Hay algún campo vacío" |

**Tabla 5. 7 Resultados de prueba. Validación de login.**

- Evaluación de resultados

  El procedimiento es exitoso al proporcionar el login y password. En caso de que algún campo esté vacío, el mensaje adecuado se muestra inmediatamente advirtiendo al usuario el error.

**Mapa de estaciones**

- Requerimientos a cubrir

  Para seleccionar una estación se necesita:

  - Estación

- Condiciones prerrequisito

  Ingresar al sitio, accionar el botón de entrada de usuarios y proporcionar login y password correctos.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Lista | Números. | Estación |

**Tabla 5. 8 Entradas de prueba. Mapa de estaciones.**

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Lista | Vacío. | "Elija una estación." |

**Tabla 5. 9 Resultados de prueba. Mapa de estaciones.**

- Evaluación de resultados

  El procedimiento es exitoso al seleccionar una estación de la lista. En caso de que no se seleccione una estación, el mensaje adecuado se muestra inmediatamente advirtiendo al usuario el error.

**Lista de archivos de datos**

- Requerimientos a cubrir

  Para obtener la lista de archivos de datos crudos de una estación se necesita:

  - Estación

- Condiciones prerrequisito

  Ingresar al sitio, accionar el botón de entrada de usuarios, proporcionar login y password correctos. Posteriormente, seleccionar una estación de la lista y accionar el botón de Datos Crudos.

- Entradas de prueba

  Selección del archivo deseado para abrir o guardar.

- Resultados de prueba

  Archivo deseado abierto en el navegador o guardado localmente en el lugar especificado.

- Evaluación de resultados

  El procedimiento es exitoso al abrir o guardar un archivo de datos crudos de la estación deseada.

**Evaluación de filtros**

- Requerimientos a cubrir

  Para realizar la evaluación de filtros se necesita:

  - Selección
  - Mes
  - Año
  - Tamaño
  - Ordenada

- Condiciones prerrequisito

  Ingresar al sitio, accionar el botón de entrada de usuarios, proporcionar login y password correctos. Posteriormente, seleccionar una estación de la lista y accionar el botón de Datos.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Lista | Caracteres. | Mes, año, tamaño y ordenada. |
| Lista excluyente | Número. | Selección. |

**Tabla 5. 10 Entradas de prueba. Evaluación de filtros.**

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Lista | Caracteres. | "Seleccione un mes." "Seleccione un año." |
| Lista excluyente | Vacío. | "Asegúrese de seleccionar un filtro." |

**Tabla 5. 11 Resultados de prueba. Evaluación de filtros.**

- Evaluación de resultados

  El procedimiento es exitoso al seleccionar un filtro. En caso de que no se seleccione un mes, se muestra el mensaje adecuado de error. En caso de que no se seleccione un año, se muestra el mensaje adecuado de error. En caso de que no se seleccione un filtro, el mensaje adecuado se muestra inmediatamente advirtiendo al usuario el error.

**Obtención de datos de la base**

- Requerimientos a cubrir

  El dato o los datos solicitados en el área de filtros deben mostrarse con el día y hora en el cuál se suscitaron. Se necesitan los datos siguientes para obtener la información de la base:

  - Estación
  - Mes
  - Año

- Condiciones prerrequisito

  Seleccionar un filtro que redireccione a la zona de datos (máximo, mínimo, máximos o mínimos).

- Entradas de prueba

  Se proporciona una estación, un mes y un año para verificar si el o los datos resultantes son correctos.

- Resultados de prueba

  El mínimo y el máximo de un mes de un año son obtenidos correctamente por la base de datos. Los máximos y mínimos definidos según el criterio de la primera y segunda derivadas son obtenidos correctamente a partir del algoritmo ideado (puede revisarse en los apéndices, en el anexo E).

- Evaluación de resultados

  Si bien el mínimo y el máximo de un mes y año los obtiene la base de datos con facilidad, el saber el día y hora exactos se complica, por lo que se optó por generar un algoritmo para obtenerlos también sin utilizar las herramientas de la base de datos. En caso de que no existan datos para un mes y año particulares, se muestra un mensaje apropiado.

**Generación de gráficas**

- Requerimientos a cubrir

  Para obtener una gráfica pueden ser necesarios los siguientes datos:

  - Selección
  - Mes
  - Año
  - Tamaño
  - Ordenada

- Condiciones prerrequisito

  Seleccionar un filtro que redireccione a la zona de gráficas (promedios anuales, promedios mensuales por año, promedios diarios por mes, día típico, alturas horarias por mes).

- Entradas de prueba

  Se proporciona una estación, un mes, un año, la ordenada y el tamaño para verificar si el o los datos resultantes son correctos.

- Resultados de prueba

  Se obtiene la gráfica con título (con el nombre de la estación y el tipo de gráfica), leyenda en los ejes X y Y, marcadores para cada valor graficado, el valor de la ordenada para cada valor graficado y la posibilidad de guardar la imagen para análisis posterior.

- Evaluación de resultados

  Una vez conseguida la nueva herramienta para generar las gráficas, se pudo observar una gran mejora en la calidad de las mismas. También, la parametrización de las gráficas eliminó una buena parte de los errores que se presentaban tanto en la representación de los datos, como en la generación de las imágenes.

**Adición de nuevos elementos**

- Requerimientos a cubrir

  Se insertan todos los datos del formulario correspondiente: usuario, estación o administrador.

- Condiciones prerrequisito

  Ingresar mediante el botón correspondiente a la zona deseada: usuario, estación o administrador. No debe haber checkbox seleccionados.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | Todos. |

**Tabla 5. 12 Entradas de prueba. Adición de nuevos elementos.**

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Texto | Vacío. | "Hay un campo vacío." |

**Tabla 5. 13 Resultados de prueba. Adición de nuevos elementos.**

- Evaluación de resultados

  Los datos se insertan correctamente, en cuanto los campos no estén vacíos y tengan el formato adecuado.

**Modificación de elementos**

- Requerimientos a cubrir

  Se actualizan los datos del formulario correspondiente: usuario, estación o administrador.

- Condiciones prerrequisito

  Ingresar mediante el botón correspondiente a la zona deseada: usuario, estación o administrador. Debe haber sólo un checkbox seleccionado.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | Todos |

**Tabla 5. 14 Entradas de prueba. Modificadón de elementos.**

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Texto | Vacío. | "Hay un campo vacío." |

**Tabla 5. 15 Resultados de prueba. Modificación de elementos.**

- Evaluación de resultados

  Los datos se actualizan correctamente, en cuanto los campos no estén vacíos y tengan el formato adecuado.

**Borrado de elementos**

- Requerimientos a cubrir

  Se borran todos los registros de los elementos seleccionados para los datos almacenados de usuario, estación, comentario o administrador.

- Condiciones prerrequisito

  Ingresar mediante el botón correspondiente a la zona deseada: usuario, estación o administrador. Debe haber al menos un checkbox seleccionado.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Checkbox | Caracteres del código ASCII.<br>Números. | Todos. |

**Tabla 5. 16 Entradas de prueba. Borrado de elementos.**

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Checkbox | Vacío. | "Debe seleccionar al menos un registro para borrar." |

**Tabla 5. 17 Resultados de prueba. Borrado de elementos.**

- Evaluación de resultados

  Los datos se eliminan correctamente. El mensaje de error se muestra cuando no hay registros seleccionados para borrar.

**Inserción de nuevos datos**

- Requerimientos a cubrir

  Para insertar nuevos datos a una estación se necesita:

  - Estación
  - Archivo

- Condiciones prerrequisito

  Ingresar al sitio, ingresar a la entrada de administradores, proporcionar login y password correctos, accionar el botón de Datos.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Lista | Número. | Estación. |
| File | Caracter. | Archivo. |

**Tabla 5. 18 Entradas de prueba. Inserción de nuevos datos.**

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Lista | Vacío. | "Elija una estación." |
| File | Vacío. | "Elija un archivo." |

**Tabla 5. 19 Resultados de prueba. Inserción de nuevos datos.**

- Evaluación de resultados

  El procedimiento es exitoso al seleccionar una estación y proporcionar un archivo. En caso de que algún campo esté vacío, el mensaje adecuado se muestra inmediatamente advirtiendo al usuario el error.

## 5.3 Pruebas Integrales

**Ver información de una estación**

- Requerimientos a cubrir

  Para ver la información de una estación se necesita:

  - Login
  - Password
  - Estación

- Condiciones prerrequisito

  Ingresar al sitio y accionar el botón de entrada de usuarios.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | login, password. |
| Lista | Números. | Estación. |

**Tabla 5. 20 Entradas de prueba. Ver información de una estación.**

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Texto | Vacío. | "Hay algún campo vacío" |
| Lista | Vacío. | "Elija una estación" |

**Tabla 5. 21 Resultados de prueba, Ver información de una estación.**

- Evaluación de resultados

  El procedimiento es exitoso al proporcionar el login y password. En caso de que algún campo esté vacio, el mensaje adecuado se muestra inmediatamente advirtiendo al usuario el error. Posteriormente, en caso de que no se seleccione una estación, el mensaje de error adecuado se muestra.

**Obtener archivos de una estación**

- Requerimientos a cubrir

  Para obtener archivos de una estación se necesitan los datos siguientes:

  - Login
  - Password
  - Estación

- Condiciones prerrequisito

  Ingresar al sitio, accionar el botón de entrada de usuarios, proporcionar login y password correctos, posteriormente seleccionar una estación de la lista y accionar el botón de datos crudos.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | login, password. |
| Lista | Números. | Estación. |

**Tabla 5. 22 Entradas de prueba. Obtener archivos de una estación.**

Selección del archivo deseado para abrir o guardar.

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Texto | Vacío. | "Hay algún campo vacío." |
| Lista | Vacío. | "Elija una estación." |

**Tabla 5. 23 Resultados de prueba. Obtener archivos de una estación.**

Archivo deseado abierto en el navegador o guardado localmente en el lugar especificado.

- Evaluación de resultados

El procedimiento es exitoso al abrir o guardar un archivo de datos crudos de la estación deseada después de ingresar con un login y password correctos y seleccionar una estación de la lista. En caso de que no se proporcione un login y password correctos o que no se seleccione una estación, el mensaje adecuado se muestra inmediatamente advirtiendo al usuario el error.

**Ver datos procesados de una estación**

- Requerimientos a cubrir

  Para ver los datos procesados de una estación se necesitan los datos siguientes para obtener la información de la base:

  - Login
  - Password
  - Estación
  - Selección
  - Mes
  - Año

- Condiciones prerrequisito

  Ingresar al sitio, accionar el botón de entrada de usuarios, proporcionar login y password correctos. Posteriormente, seleccionar una estación de la lista y accionar el botón de datos. Finalmente, seleccionar un filtro que redireccione a la zona de datos (máximo, mínimo, máximos o mínimos).

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | login, password. |
| Lista | Números. | Estación. |
| Lista | Caracteres. | Mes y año. |
| Lista excluyente | Número. | Selección. |

**Tabla 5. 24 Entradas de prueba. Datos procesados de una estación.**

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Texto | Vacío. | "Hay algún campo vacío." |
| Lista | Vacío. | "Elija una estación." |
| Lista | Caracteres. | "Seleccione un mes."<br>"Seleccione un año." |
| Lista excluyente | Vacío. | "Asegúrese de seleccionar un filtro." |

**Tabla 5. 25 Resultados de prueba. Ver datos procesados estación.**

El mínimo y el máximo de un mes de un año son obtenidos correctamente por la base de datos. Los máximos y mínimos definidos según el criterio de la primera y segunda derivadas son obtenidos correctamente a partir del algoritmo ideado.

- Evaluación de resultados

El procedimiento es exitoso al proporcionar el login, password y al seleccionar una estación de la lista y un filtro. En caso de que no se seleccione un mes, se muestra el mensaje adecuado de error. En caso de que no se seleccione un año, se muestra el mensaje adecuado de error. En caso de que no se seleccione un filtro, el mensaje adecuado se muestra inmediatamente advirtiendo al usuario el error. En caso de que no existan datos para un mes y año particulares, se muestra un mensaje apropiado.

**Ver gráfica de una estación**

- Requerimientos a cubrir

  Para ver la gráfica de una estación se necesitan los datos siguientes para obtener la información de la base:

  - Login
  - Password
  - Estación
  - Selección
  - Mes
  - Año
  - Tamaño
  - Ordenada

- Condiciones prerrequisito

  Ingresar al sitio, accionar el botón de entrada de usuarios, proporcionar login y password correctos, posteriormente seleccionar una estación de la lista y accionar el botón de datos. Posteriormente seleccionar un filtro que redireccione a la zona de gráficas (promedios anuales, promedios mensuales por año, promedios diarios por mes, día típico, alturas horarias por mes).

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | login, password. |
| Lista | Números. | Estación. |
| Lista | Caracteres. | Mes, año, tamaño y ordenada. |
| Lista excluyente | Número. | Selección. |

**Tabla 5. 26 Entradas de prueba. Ver gráfica de una estación.**

- Resultados de prueba

| Formatos | Valores | Mensajes |
|---|---|---|
| Texto | Vacío. | "Hay algún campo vacío." |
| Lista | Vacío. | "Elija una estación." |
| Lista | Caracteres. | "Seleccione un mes."<br>"Seleccione un año." |
| Lista excluyente | Vacío. | "Asegúrese de seleccionar un filtro." |

**Tabla 5. 27 Resultados de prueba. Ver gráfica de una estación.**

Se obtiene la gráfica con título (con el nombre de la estación y el tipo de gráfica), leyenda en los ejes X y Y, marcadores para cada valor graficado, el valor de la ordenada para cada valor graficado y la posibilidad de guardar la imagen para análisis posterior.

- Evaluación de resultados

El procedimiento es exitoso al proporcionar el login, password y al seleccionar una estación de la lista y un filtro. En caso de que no se seleccione un mes, un año o un filtro, el mensaje adecuado se muestra inmediatamente advirtiendo al usuario el error. Una vez conseguida la nueva herramienta para generar las gráficas, se pudo observar una gran mejora en la calidad de las mismas. También, la parametrización de las gráficas eliminó una buena parte de los errores que se presentaban tanto en la representación de los datos, como en la generación de las imágenes.

**Administrar usuario**

- Requerimientos a cubrir
  - Login de administrador
  - Password de administrador
  - Nombre
  - Apellidos
  - Lugar de origen
  - Organización
  - Teléfono
  - Email
  - Login de usuario
  - Contraseña

- Condiciones prerrequisito

  Ingresar a la zona de administración e identificarse como administrador.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | Login de administrador, password de administrador, nombre, apellidos, lugar, organización, teléfono, login de usuario contraseña de usuario. |
| Email | Caracteres del código ASCII. | Email. |
| Checkbox | Caracteres del código ASCII. | Selección. |

**Tabla 5. 28 Entradas de prueba. Administrar usuario.**

- Resultados de prueba

| Formatos | Valores | Mensaje |
|---|---|---|
| Texto | Vacío. | "Asegúrese de llenar todos los campos." |
| Email | Sin "@" y " . " Caracteres prohibidos. | "El formato de correo electrónico parece ser incorrecto." |
| Checkbox | Vacío. | "Seleccione al menos un registro." |
| | Uno seleccionado. | "No es necesario seleccionar registros para agregar usuarios." |
| | Varios seleccionados. | "Seleccione solo un registro para modificar." |

**Tabla 5. 29 Resultados de prueba. Administrar usuario.**

- Evaluación de resultados

  La zona de administración de usuarios no presentó errores. Los procesos de agregado, borrado y modificado de usuarios se realizaron varias veces, verificándose las validaciones indicadas para los casos de fallo y realizándose la interacción con la base de datos de forma correcta.

**Administrar estación**

- Requerimientos a cubrir
  - Login de administrador
  - Password de administrador
  - Identificador numérico de la estación
  - Nombre
  - Coordenadas en el mapa de enlaces
  - Nombre del croquis de la caseta y foto del lugar
  - Dirección
  - Bancos de nivel
  - Planos de referencia

- Condiciones prerrequisito

  Ingresar a la zona de administración e identificarse como administrador.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | Login de administrador, password de administrador, nombre, coordenadas en el mapa de enlaces, nombre del croquis de la caseta y foto del lugar, dirección, bancos de nivel, planos de referencia. |
| | Números. | Identificador numérico de la estación. |
| Checkbox | Caracteres del código ASCII. | Selección. |

**Tabla 5. 30 Entradas de prueba. Administrar estación.**

- Resultados de prueba

| Formatos | Valores | Mensaje |
|---|---|---|
| Texto | Vacío. | "Asegúrese de llenar todos los campos." |
| Checkbox | Vacío. | "Seleccione al menos un registro." |
| | Uno seleccionado. | "No es necesario seleccionar registros para agregar usuarios." |
| | Varios seleccionados. | "Seleccione solo un registro para modificar." |

**Tabla 5. 31 Resultados de prueba. Administrar estación.**

- Evaluación de resultados

  La zona de administración de estaciones no presentó errores. Los procesos de agregado, borrado y modificado de estaciones se realizaron varias veces, verificándose las validaciones indicadas para los casos de fallo y realizándose la interacción con la base de datos de forma correcta.

**Administrar administrador**

- Requerimientos a cubrir
  - Login de administrador
  - Password de administrador
  - Nombre nuevo administrador
  - Login nuevo administrador
  - Contraseña nuevo administrador

- Condiciones prerrequisito

  Ingresar a la zona de administración e identificarse como administrador.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | Login de administrador, password de administrador, nombre nuevo administrador, login nuevo administrador, contraseña nuevo administrador. |
| Checkbox | Caracteres del código ASCII. | Selección. |

**Tabla 5. 32 Entradas de prueba. Administrar administrador.**

- Resultados de prueba

| Formatos | Valores | Mensaje |
|---|---|---|
| Texto | Vacío. | "Asegúrese de llenar todos los campos." |
| Checkbox | Vacío. | "Seleccione al menos un registro." |
| | Uno seleccionado. | "No es necesario seleccionar registros para agregar usuarios." |
| | Varios seleccionados. | "Seleccione solo un registro para modificar." |

**Tabla 5. 33 Resultados de prueba. Administrar administrador.**

- Evaluación de resultados

  La zona de administración de otros administradores no presentó errores. Los procesos de agregado, borrado y modificado de administradores se realizaron varias veces, verificándose las validaciones indicadas para los casos de fallo y realizándose la interacción con la base de datos de forma correcta.

**Administrar comentario**

- Requerimientos a cubrir
  - Login de administrador
  - Password de administrador
  - Fecha
  - Email
  - Remitente
  - Mensaje

- Condiciones prerrequisito

  Ingresar a la zona de administración e identificarse como administrador.

- Entradas de prueba

| Formatos | Valores | Campos |
|---|---|---|
| Texto | Caracteres del código ASCII. | Login de administrador, password de administrador, fecha, remitente, mensaje. |
| Email | Caracteres del código ASCII. | Email. |
| Checkbox | Caracteres del código ASCII. | Selección. |

**Tabla 5. 34 Entradas de prueba. Administrar comentario.**

- Resultados de prueba

| Formatos | Valores | Mensaje |
|---|---|---|
| Texto | Vacío. | "Asegúrese de llenar todos los campos." |
| Checkbox | Vacío. | "Seleccione al menos un registro." |
| | Varios seleccionados. | "Seleccione sólo un registro para modificar." |

**Tabla 5. 35 Resultados de prueba. Administrar comentario.**

- Evaluación de resultados

  La zona de administración de comentarios no presentó errores. Los procesos de revisado y borrado de comentarios se realizaron varias veces, verificándose las validaciones indicadas para los casos de fallo y realizándose la interacción con la base de datos de forma correcta.

## 5.4 SQL

### Máximo de una estación

El máximo de una estación se puede obtener, por atributo, con la instrucción MAX("atributo"). En el caso de los valores obtenidos para las estaciones fue necesario obtener el máximo de los máximos obtenidos de cada una de las 24 horas de un día por separado, dado que tantos son los atributos. Asimismo, los máximos no incluyen los valores que están definidos como el indicador de ausencia de valor (9999).

Los máximos fueron probados para todas las estaciones del Pacífico, con distintos meses y años, obteniéndose en todos los casos los resultados esperados. En el caso de escogerse un mes de un año que contuviese exclusivamente los valores de ausencia (9999), se pudo comprobar que el resultado arrojado era el conjunto vacío.

### Mínimo de una estación

El mínimo de una estación se puede obtener, por atributo, con la instrucción MIN("atributo"). En el caso de los valores obtenidos para las estaciones fue necesario obtener el mínimo de los mínimos obtenidos de cada una de las 24 horas de un día por separado, dado que tantos son los atributos. Asimismo, los mínimos no incluyen los valores que están definidos como el indicador de ausencia de valor (9999).

Los mínimos fueron probados para todas las estaciones del Pacífico, con distintos meses y años, obteniéndose en todos los casos los resultados esperados. En el caso de escogerse un mes de un año que contuviese exclusivamente los valores de ausencia (9999), se pudo comprobar que el resultado arrojado era el conjunto vacío.

### Datos de un mes y año

La obtención de datos de un año o mes en particular requiere de la instrucción extract(year from fecha) o extract(month from fecha), dependiendo del caso. Además, puede ser necesario agrupar los resultados por mes o por año, lo cual puede lograrse con la instrucción "group by" seguida del rubro por el cual se desea agrupar. Finalmente, la instrucción "distinct" es útil para evitar que los registros obtenidos se repitan.

Las pruebas que se realizaron son las siguientes:

- Obtener todos los años registrados para una estación.
- Obtener los datos correspondientes a cada año de los registrados en una estación.
- Obtener los datos correspondientes a cada mes de un año de los registrados en una estación.
- Obtener los datos correspondientes a un mes de un año de los registrados en una estación.

En todos los casos, el resultado esperado fue el obtenido al final de la prueba. En el caso de escogerse un mes o un año que contuviese exclusivamente los valores de ausencia (9999), se pudo comprobar que el resultado arrojado era el conjunto vacío.

## 5.5 Pruebas del usuario

Las pruebas realizadas por los usuarios involucraron extensivas sesiones dedicadas a la revisión de las gráficas, de la secuencia de pantallas, del diseño gráfico del sitio y de la precisión de los resultados, entre otras. Entre las sugerencias realizadas se encontraron:

- Respecto al diseño gráfico:
  - Logos inicialmente de tamaño y resolución inadecuados.
  - Menú de opciones de funcionamiento poco claro.
  - Texto de la página de inicio innecesario.
  - Mención a la UNAM necesaria para el proyecto en la página inicial.
  - Distribución de la dirección, datos geográficos, menú interno de usuario y fotografía del mareógrafo inadecuada y poco funcional.
  - Opciones de los filtros poco claras.
  - Falta de colores de diseño en algunas de las ventanas emergentes.
  - Necesidad de un fondo de agua para algunas páginas con poco contenido.

- Respecto a la secuencia de pantallas
  - Redundancia en la selección de estación para datos procesados y datos crudos.
  - Necesidad de botones de regreso para una zona.
  - Necesidad de un botón de salida para terminar la sesión de usuario.
  - Necesidad de encontrar ligas hacia las páginas del Instituto de Geofísica y de la UNAM.

- Respecto a la precisión de los resultados
  - Resultados imprecisos en algunos cálculos debido a que no deben tomarse en cuenta los valores no conocidos en los mismos.

- Respecto a las gráficas
  - Necesidad de tomar en cuenta los "huecos" en los valores dentro de la base de datos para poder observar el comportamiento con claridad.
  - Necesidad de contar con marcadores de punto para poder precisar el valor de cada punto en algunas gráficas.
  - Necesidad de mostrar el valor de ordenada.
  - Errores en el servidor con el tamaño máximo de gráfica provisto.

Todas estas sugerencias fueron atendidas e implementadas para poder entregar un producto satisfactorio al usuario. Una vez informados de las modificaciones, los usuarios confirmaron que las adecuaciones convinieran a sus necesidades y expresaron su conformidad, con lo que se dio por terminado el proceso de pruebas.

# 6. Liberación

## 6.1 Análisis

El trabajo realizado de análisis engloba la lista de requerimientos, la aportación de los conceptos básicos necesarios, el diagrama de casos de uso, la descripción de los casos de uso, el diagrama de clases y los diagramas de flujo de procesos. Esta información mencionada se puede encontrar en los capítulos primero, segundo y tercero. Esta etapa fue liberada por los Ingenieros Osvaldo Sánchez y Alejandro Velázquez Mena, en el mes de diciembre de 2003.

## 6.2 Diseño

El trabajo realizado de diseño engloba los diagramas de secuencia, diagramas de actividades, diagramas de flujo, diagramas de base de datos (modelos conceptual y físico) y secuencia de pantallas. Esta información mencionada se puede encontrar en los capítulos tercero y cuarto. Esta etapa fue liberada por los Ingenieros Osvaldo Sánchez y Alejandro Velázquez Mena, en el mes de septiembre de 2004.

## 6.3 Desarrollo

El trabajo realizado de desarrollo engloba la programación en Java, el código SQL, la programación para las gráficas, la validación en Javascript, los shellscripts y scripts de AWK y las hojas de estilo. Comentarios sobre esta información pueden encontrarse en el capítulo quinto. Esta etapa fue

liberada por el Ingeniero Alejandro Velázquez Mena, en el mes de enero de 2005. Los elementos liberados son los siguientes:

Programación Java

| | |
|---|---|
| Administrador.java | Estacion.java |
| Analisis.java | Fecha.java |
| Aplicacion.java | GraficaAlturasHorarias.java |
| ConectaGeofisica.java | GraficaAnual.java |
| Contacto.java | GraficaBean.java |
| Controlador.java | GraficaDiaTipico.java |
| ControladorAdministrador.java | GraficaDiaria.java |
| ControladorCambioPasswd.java | GraficaMensual.java |
| ControladorContacto.java | Login.java |
| ControladorEstacion.java | MiniMax.java |
| ControladorFiltros.java | NuevoControladorFiltros.java |
| ControladorListas.java | Numeros.java |
| ControladorLogin.java | ParseBancos.java |
| ControladorMapa.java | SubirArchivoServlet.java |
| ControladorUsuario.java | Usuario.java |
| EjecutaProceso.java | |

Código SQL

| | | |
|---|---|---|
| datosAdministrador.sql | selectOp1.sql | tablaContacto.sql |
| datosEstacion.sql | selectOp2.sql | tablaDatosEstacion.sql |
| datosUsuario.sql | selectOp3.sql | tablaEstacion.sql |
| selectOp0.sql | tablaAdministrador.sql | tablaUsuario.sql |

Validación Javascript

pass.js
radioButtons.js
valida.js
validaAU.js
validaContacto.js
validaMapa.js
validaRBs.js
validate.js

Shellscript y AWK

ponerComas.awk
compresion.bash
formato.bash
generadorTablas.bash
inserta.bash
script.bash

Hojas de estilo

03css001.css
bordes.css
fondocentrado.css
mystyle.css

## 6.4 Pruebas

El trabajo realizado de pruebas engloba las pruebas realizadas a los servidores Web y de base de datos, al código Java en forma unitaria e integral, al código SQL y al Sistema en conjunto. Las pruebas a los servidores se realizaron con software especializado. Las demás fueron conducidas en primera instancia por los tesistas. Posteriormente, las pruebas al Sistema fueron realizadas por el Dr. Osvaldo Sánchez Zamora, el personal del Servicio Mareográfico de Instituto de Geofísica y el Ing. Alejandro Velázquez Mena, en el mes de marzo de 2005.

# Conclusiones

Después de haber concluido la elaboración de este proyecto respetando los lineamientos indicados por las técnicas de Ingeniería de Software, este capítulo se dedica a la exposición de las siguientes conclusiones:

*Primera*. El Sistema cumple con la lista de requerimientos definidos por los usuarios.

Los requerimientos quedaron especificados en los primeros dos capítulos de este ensayo. Este objetivo es necesariamente el primario, dado que el resultado del proyecto debe evaluarse conforme a esos lineamientos. Los objetivos, algunos de los cuales se comentan en puntos posteriores, incluyen el facilitar el trabajo de quienes laboran en el Servicio Mareográfico del Instituto de Geofísica, el dar presencia al Servicio Mareográfico de Instituto del Geofísica en Internet y el generar un repositorio de datos confiable. Estos tres objetivos primordiales son los que manifestaron los usuarios para el Sistema y que nosotros comprendimos como requerimientos. En vista de que el sitio está en función plena, nosotros consideramos que los tres objetivos se cumplen. Puede realizarse las operaciones y gráficas deseadas; existe una presencia del Servicio Mareográfico del Instituto de Geofísica en Internet; los datos almacenados, tanto en la base de datos, como en el formato de texto acostumbrado, se encuentran disponibles a nivel internacional para su consulta y descarga.

*Segunda*. El análisis y diseño realizado permite la extensión, modificación, actualización y corrección del Sistema con el mínimo de esfuerzo.

El UML es una excelente herramienta para la documentación de aplicaciones de software. La documentación incluida en el tercer capítulo de la tesis comprende los casos de uso, el diagrama de clases, los diagramas de secuencia, los diagramas de actividades y los diagramas de flujo. Estos diagramas permiten a cualquier desarrollador, que en un futuro interactué con el Sistema, comprender los distintos módulos y su forma de funcionar. Nos parece que el haber generado documentación completa y estandarizada es uno de los mayores logros de este proyecto. Gran cantidad de proyectos hoy en día no están propiamente documentados y permanecen una caja negra para todo aquel que intente comprenderlo. Las extensiones, actualizaciones y correcciones que debe sufrir el Sistema necesariamente con el paso del tiempo serán más sencillas y podrán planearse mejor gracias a la documentación en UML.

*Tercera*. El software de la compañía VE parece ser óptimo para la generación de las gráficas solicitadas.

El API de la compañía Visual Engineering resultó ser de gran utilidad para la generación de las gráficas de este proyecto. No sólo permite la parametrización y modularización de las mismas, sino que permitirá, en caso de ser necesario en el futuro, la presentación de datos en tiempo real. Asimismo, la variedad de formatos en los que es factible presentar las gráficas, junto con las facilidades de administración de las imágenes dentro del servidor, permite una optimización de la herramienta para utilizar menos recursos de hardware y ofrecer a los usuarios una mejor calidad de producto.

Entre las ventajas que proporciona este API se encuentra el poder agregar etiquetas a los datos graficados, para que el usuario pueda saber el valor para cada punto graficado. Para simplificar más la tarea, se puede agregar marcadores a cada punto, lo cual hace más fácil localizarlos. Además, las discontinuidades que pueden manejarse hacen más fácil la representación correcta de los datos.

*Cuarta*. El lenguaje Java permitió una programación modular, orientada a objetos, concisa, eficiente y *ad hoc* con los avances de la Ingeniería de Software.

La elección del lenguaje Java fue muy acertada a nuestro parecer. La gran mayoría de las aplicaciones serias y a gran escala se desarrollan en este lenguaje. Por ende, existe una buena cantidad de documentación, información y ejemplos disponibles para mejorar la programación.

El que Java sea orientado a objetos permite una programación modularizada y apegada al diseño. La implementación del patrón Modelo-Vista-Controlador se facilitó con el lenguaje Java, puesto que las JSP son excelentes para realizar vistas, los servlets pueden realizar el control y la conexión JDBC puede conectar el conjunto con el back-end.

*Quinta.* La base de datos Postgres cumple con las expectativas de desempeño para la cantidad de información pronosticada.

Una de las principales preocupaciones que había en un inicio era la capacidad de la base de datos Postgres para almacenar la gran cantidad de datos y realizar las operaciones concurrentes necesarias. Nos tranquilizamos a este respecto puesto que no parece haber problemas al respecto. Acertadamente, decidimos generar una tabla para los datos de cada estación y esto permitió un mejor desempeño al realizar búsquedas y operaciones.

El único problema que pudimos percibir fue al permitir que las conexiones a la base de datos permaneciesen abiertas indefinidamente. Este error a la larga ocasionaba la caída de los servicios de base de datos y posteriormente de los de red. Resolvimos este problema asegurando que toda vez que se iniciase una conexión para extraer datos se cumpliese con cerrarla.

*Sexta.* El servidor Web Tomcat pasó las pruebas de concurrencia.

El servidor Web es susceptible de ser sobrecargado por peticiones en un momento de extrema demanda. Tal sobrecarga resulta en un alto a las funciones del servidor temporal o permanentemente. Para evaluar tal posibilidad hicimos las pruebas de concurrencia detalladas en el punto 6.1. Concluimos que el servidor es capaz de soportar la demanda esperada.

*Séptima*. El proyecto nos sirvió para aplicar los conceptos aprendidos en las distintas materias de la carrera.

Indudablemente, para nosotros como estudiantes, uno de las principales alicientes para involucrarnos en un proyecto de tal magnitud es el poder emplear y probar los conocimientos adquiridos durante los diez semestres de nuestra instrucción universitaria. Es de gran valor para nosotros el saber que el tiempo empleado no fue en vano y que lo asimilado tiene una utilidad para la sociedad. Utilizamos conocimientos de las materias Cálculo, Computadoras y Programación, Ingeniería de Software, Bases de Datos y Redes de Computadoras.

*Octava*. El equipo de hardware designado por el Servicio Mareográfico de Instituto del Geofísica es de capacidad suficiente para albergar el Sistema.

Dado que este proyecto no incluyó para nosotros la elección de una computadora, debimos esperar que el equipo provisto fuese de suficiente capacidad para manejar los servicios y para almacenar los datos ya existentes y una buena cantidad de los que se generen al pasar del tiempo. El equipo provisto ya fue discutido en los capítulos previos.

El Sistema se ha desempeñado de excelente manera durante el tiempo de desarrollo y una vez que se ha comenzado la fase de pruebas. La conexión es rápida, así como la respuesta una vez que se solicitan datos. Las gráficas se obtienen con prontitud. El espacio en disco es suficiente y lo será por bastante tiempo, puesto que al momento se han insertado los datos de treinta estaciones, manteniéndose el espacio en la partición de la base de datos libre en un 94%.

*Novena*. El Sistema es competente para representar al Servicio Mareográfico del Instituto de Geofísica en Internet.

Una de las finalidades del Sistema es el darle presencia al Servicio Mareográfico del Instituto de Geofísica. Esta presencia es por el momento discreta comparada con la especialización del sitio de la Universidad de Hawaii. Otro ejemplo que puede citarse es el del CICESE, que posee un sitio que se encuentra bastante bien organizado, pero que carece de dinamismo para la generación de gráficas de datos. No obstante, nos

parece que el sitio actual puede representar dignamente a la Universidad, aceptando además que pueden realizarse mejoras y modificaciones. Posiblemente pueda dedicarse otro grupo de trabajo como el nuestro para poder mejorar lo existente.

*Décima.* Las gráficas resultantes de los datos son útiles para el trabajo que realizan los usuarios del sistema.

Las gráficas que pueden generarse hasta el momento son: el promedio de cada año existente en la base de datos, el promedio de cada mes de un año específico, el promedio de cada día de un mes y un año específico, el promedio de cada hora de un mes y un año específico, y todos los valores horarios para un mes y un año específico. Esta variedad es la que por el momento se nos indicó que era suficiente y permite realizar el análisis del nivel del mar.

Las gráficas se generan como imágenes que pueden almacenarse en disco duro. Es posible escoger el tamaño de la gráfica e indicar si se desea ver el origen de las ordenadas para darle mayor perspectiva a los datos. Para proporcionar mayor información, se agregaron etiquetas a los valores graficados para tener el valor exacto de cada punto y no tener que recurrir a los ejes coordenados. En caso de no existir valores para un rango de fechas, se visualiza en las gráficas el espacio en blanco, lo cual evita suponer que los datos de un periodo no continuo con otro son fehacientes.

*Décima primera.* El mismo principio de tratamiento de datos puede utilizarse para otras variables mareográficas como son la salinidad y la temperatura del agua.

A pesar de que en un inicio se tenía contemplado el realizar gráficas similares para otras variables mareográficas. Por ejemplo, el sitio del CICESE maneja datos de temperatura del agua, temperatura del aire, velocidad del viento, presión atmosférica y humedad relativa.

En caso de que se contara con estos datos, el realizar un agregado al sitio para graficarlos y mostrarlos sería sencillo, puesto que el análisis y diseño se realizó con cuidado y dedicación. Las nuevas gráficas deberán seleccionar datos de distintas tablas, pero tendrán la misma funcionalidad.

El almacenamiento de los nuevos datos no será problema, como ya se discutió en la parte de base de datos. Deberán crearse nuevas tablas, siguiendo el patrón establecido que sugiere una tabla para cada tipo de datos y por cada estación.

*Décimo segunda*. El Sistema es escalable con facilidad para el reporte de datos en tiempo real.

Uno de los principales avances que debe implementar el Servicio Mareográfico de Instituto del Geofísica es incorporar estaciones de medición que puedan transmitir datos en tiempo real, para evitar el penoso procedimiento de tener que dar formato a los datos que llegan por correo con bastante tiempo de retraso. Fenómenos como el tsunami que azotó las costas del Sureste asiático pueden detectarse a tiempo sólo de esta manera.

El API de VE permite el tratamiento de datos con applets. Esta faceta hace posible el estar actualizando el contenido de las gráficas permanentemente, en tanto que el flujo de datos se mantenga. De esta manera, se podría implementar un sistema de alarma para los momentos en los que sucediese una variación significativa en los valores recibidos.

Este suplemento al Sistema actual sí pudiera significar modificaciones más drásticas al mismo, debido a que el algoritmo para generar las gráficas puede cambiar bastante. En todo caso, un proceso de análisis y diseño previo al desarrollo debe minimizar las incongruencias y facilitar el acoplamiento del nuevo módulo.

*Décimo tercera*. La zona de administración del Sistema permite eliminar, casi por completo, el contacto del administrador con el Sistema operativo del servidor.

A pesar de no ser uno de los objetivos principales, la zona de administración del sitio funciona y es parte importante del desarrollo que realizamos. Esta zona es accesible sólo para el personal del Servicio Mareográfico de Instituto del Geofísica. Como la mayoría de las zonas de administración está destinada a la realización de altas, bajas y cambios.

Los rubros administrables son los usuarios, estaciones, comentarios y otros administradores. La administración de estaciones es la que requiere de mayor atención, ya que involucra crear una tabla para los datos de la estación y un directorio en el repositorio de datos para almacenar los formatos.

Además, como el objetivo de esta zona es eliminar el contacto directo de los administradores con el servidor, es posible subir los nuevos archivos de datos que se generen para que sean agregados en el repositorio correspondiente y se agreguen a la base de datos. Este proceso es posiblemente el más complejo dado que el formato en el que se proveen los datos originalmente es inadecuado para insertar en la base de datos. Para este efecto debimos emplear el lenguaje AWK y el shell script para lograr una funcionalidad óptima.

*Décimo cuarta.* Una zona de pronósticos intentó incorporarse al Sistema para completarlo en su funcionamiento como herramienta.

La parte complementaria al estudio de las bitácoras de los niveles del mar es el pronóstico de mareas. Este proceso es largo e involucra una buena cantidad de cálculos. Actualmente, se calculan a mano unos coeficientes que se alimentan a un programa hecho en FORTRAN. El programa pondera los coeficientes y arroja los pronósticos de mareas.

Este proceso podría simplificarse si se automatizase por completo. Desafortunadamente, el programador del software que pronostica no estuvo dispuesto a cooperar para la implementación de un programa que completase el proceso. En realidad, intentamos contactarlo en varias ocasiones y nunca obtuvimos respuesta.

Si este módulo puede implementarse en un futuro, el Sistema mejorará mucho. Muy posiblemente, en ese caso, se necesite un rediseño para decidir la solución a problemas como dónde se almacenarán los datos pronosticados y si el proceso se realizará periódicamente de modo automático.

*Décimo quinta.* El Sistema no es aún completo para competir con las referencias existentes de otros.

La Universidad de Hawai, la Universidad Estatal de Florida, el Centro de Observación de la Costa de Texas, el Instituto Nacional de Mareas de Australia, la Secretaría de Marina, el Centro de Datos Oceanográficos Británico y el Centro de Prevención de Tsunamis del Pacífico son instituciones con sitios funcionales que realizan alguna o varias funciones similares a la del Sistema del Servicio Mareográfico de Instituto del Geofísica. A la luz de esos sitios, pensando en el status que nos gustaría que alcanzara el Servicio Mareográfico de Instituto del Geofísica, nos parece que debe trabajarse aún más para poder ofrecer más servicios de mejor calidad a la comunidad científica nacional e internacional, dejando en alto el nombre de la Universidad y del país.

# Apéndices

## Anexo A: Instalación Solaris

La instalación del sistema operativo se realizó mediante un CD que contiene Solaris 8, para arquitectura SPARC. Este CD nos fue proporcionado por los administradores de red del Instituto de Geofísica, a quienes debe entregarse el software del equipo que llega a cualquier departamento. Son estos administradores quienes proporcionan datos como el nombre del equipo, la dirección de *IP*, la máscara de red y los servicios permitidos para el servidor.

Los pasos para instalar son, de manera general, los siguientes:

a) Planeación de la instalación

Averiguamos, con los administradores de red, los siguientes datos:

| | |
|---|---|
| IP | 132.248.182.198 |
| Máscara | 255.255.255.0 |
| Puerta de enlace | 132.248.182.254 |
| Nombre del equipo | tsunami1 |
| Servicios permitidos | *http, ftp, telnet , ssh, dns, nis, nfs.* |

Con esta información procedimos al análisis de la distribución de espacio del los disco duro, con 80 Gb de espacio. Las particiones se configuraron con el tamaño siguiente:

- Dado que el equipo cuenta con 1 Gb de memoria RAM, se destinaron dos particiones de 1 Gb cada una como área de SWAP. El realizar dos particiones mejora el desempeño de administración del área de SWAP, mientras se respeta la recomendación de generar un espacio de por lo menos el doble de la cantidad de memoria RAM que exista en el equipo de cómputo. Estas dos particiones son primarias dentro del disco.
- La tercera partición primaria del disco se destinó a la raíz del sistema, donde se almacena el kernel de sistema operativo y distintos comandos y utilerías. Se asignó a esta partición la cantidad de 3 Gb. No es necesario que sea muy grande dado que todos los programas extras cuentan con una partición especial.
- Las siguientes particiones son forzosamente extendidas, dado que un disco puede ser dividido únicamente en cuatro particiones primarias. Por lo tanto, la cuarta y última partición, se utiliza para alojar una unidad lógica que puede subdividirse a su vez. A esta unidad le restan 75 Gb por repartir.
- Las particiones que se asignaron para este espacio son el /usr, que se utiliza para los comandos de usuario; el /opt, donde se almacena todo el software adicional u opcional del sistema; el /users, donde decidimos almacenar tanto el repositorio de datos de la base de datos; y el /export/home, donde los sistemas Solaris ubican los directorios casa de los usuarios. A cada partición se le asignó el siguiente tamaño:

| | |
|---|---|
| /usr | 10 Gb |
| /opt | 15 Gb |
| /users | 25 Gb |
| /export/home | 25 Gb |

Con esta distribución se concluye la partición del disco.

b) Inicio de la instalación

Se presiona la tecla STOP del teclado de la SunBlade y se ejecuta la sentencia boot cdrom. Esta instrucción indica a la computadora que debe iniciar la secuencia de arranque desde el cdrom.

c) Reconocimiento de hardware.

d) Selección del idioma: Español.

e) Creación de particiones y formateo del disco para la instalación.

f) Configuración de parámetros de red.

g) Configuración de fecha y hora.

h) Determinación de la contraseña de administrador.

i) Especificación de paquetes de software deseados.

j) Copiado de paquetes e instalación.

k) Instalación de parches

Se obtiene un paquete con las actualizaciones y ajustes de seguridad que el equipo de Sun recomienda para evitar problemas de intrusos o mal funcionamiento. Es recomendable instalar este paquete antes de configurar otros servicios, debido a que las modificaciones de los parches probablemente los reconfiguren.

l) Instalación de paquetes indispensables.

Se instalaron los paquetes wget, lynx, gcc y make. Estos paquetes se obtuvieron en modo binario (del sitio sunfreeware.com/indexsparc8.html), es decir, son ejecutables específicos para el sistema operativo. Para descargarlos del servidor, se utilizó *ftp*. El wget y lynx facilitan la búsqueda posterior de software. El gcc permite obtener software para ser compilado, que generalmente es más seguro.

m) Instalación de los paquetes de servicios

Los servicios que ofrece el servidor son *ssh*, base de datos postgres y Web Tomcat. Los dos últimos se discuten en secciones posteriores. Del *ssh* baste comentar que se prefirió la versión de *SSH* Communications Security sobre la provista por OpenSSH, por presentar la primera muchos menos huecos de seguridad que la segunda. Esta versión del *SSH*, la 3.2.3, se compiló y permite el acceso al servidor de una forma segura y remota.

Para mayor detalle sobre la instalación de Solaris 8, dirigirse al sitio http://sunsite.unam.mx/archivos/instala.pdf

# Anexo B: Instalación Tomcat

El servidor Web Tomcat puede obtenerse del sitio jakarta.apache.org. El Tomcat 4.x utiliza un contenedor de *servlets* nuevo llamado Catalina, basado en una arquitectura completamente nueva. Asimismo, estas versiones utilizan los *Servlets* 2.3 y los *JSP* 1.2. Las mejoras que resaltan de la versión original 4.0 a la versión 4.1.18, que es la utilizada en este proyecto, son:

- Facilidades para administración vía Web.
- Nuevo conector Coyote (con acceso para *HTTP*/1.1, AJP 1.3 y JNI)
- El compilador Jasper (para transformar *JSPs* en *Servlets*) se ha reescrito.
- Mejoras en el desempeño y uso de memoria.
- Mayor facilidad para conexión con herramientas de diseño.

Una vez se obtiene el servidor del sitio de Jakarta, se procede a descomprimirlo donde se desea tener la aplicación. Una de las ventajas que tiene este servidor es que al generar el árbol en la descompresión, ya nada más es necesario configurar algunos archivos para ponerlo en funcionamiento.

La estructura que se genera tiene los siguientes directorios relevantes:

- bin — Donde se almacenan los ejecutables para iniciar y detener el servicio.
- common — Donde se almacenan los componentes comunes a las distintas aplicaciones, como pueden ser librerías o archivos jar.
- conf — Donde se alojan los archivos de configuración.
- logs — Donde se almacenan los archivos de bitácora, inclusive el catalina.out.
- Webapps — El directorio base que se puede utilizar para publicar.
- work — Donde se almacenan los *servlets* obtenidos de la compilación de los *JSPs*, de todas las instancias que se creen en el servidor.

Dentro del directorio Webapps, el directorio base utilizable es el ROOT. Si no se altera la configuración, este es el directorio donde se almacena todo lo que se desea que permanezca directamente sobre la raíz del sitio, al

acceder al servidor vía Web. En el directorio examples/ se encuentran todos los ejemplos que se facilitan al desarrollador.

De los archivos que puede desearse configurar, los principales son: server.xml y Web.xml. En el primero de ellos, se configuran aspectos relativos al servidor, como pueden ser el puerto para el *HTTP*/1.1, puerto para *SSL*, puerto para el conector Coyote, el número máximo de hilos que soportará el servidor, el archivo de bitácora y los drivers para algunas bases de datos. En el segundo, la configuración que se realiza es la concerniente a las distintas instancias del servidor Web. Puede configurarse residentes virtuales, el directorio de mapeo de los *servlets*, los tipos *MIME (*Multipurpose Internet Mail Extensions), el tiempo de caducidad de las sesiones (en caso de que no se especifique al crearlas) y los archivos que buscará automáticamente el servidor como página de bienvenida en un directorio.

Para el caso presente de instalación de servidor, las líneas modificadas a cada archivo fueron las siguientes:

**server.xml**

**Para el puerto 80:**

```
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
        port="80" minProcessors="5" maxProcessors="75"
        enableLookups="true" redirectPort="8443"
        acceptCount="100" debug="0" connectionTimeout="20000"
        useURIValidationHack="false" disableUploadTimeout="true" />
```

**Web.xml**

**Para el mapeo de servlets:**

```
<servlet-mapping>
  <servlet-name>invoker</servlet-name>
  <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

**Para el mapeo de JSPs:**

```
<servlet-mapping>
  <servlet-name>jsp</servlet-name>
  <url-pattern>*.jsp</url-pattern>
```

```
</servlet-mapping>
```

**Para tener sesiones de 10 minutos:**

```
<session-config>
  <session-timeout>10</session-timeout>
</session-config>
```

**Para configurar esos archivos como páginas de bienvenida:**

```
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

Finalmente, para terminar con la configuración del servidor, agregamos los archivos kcServlet.jar, mail.jar, pg73jdbc2ee.jar y *servlet*-2.0.jar a la carpeta /opt/jakarta-tomcat-4.1.18/common/lib. Posteriormente, agregamos todos esos archivos JAR al CLASSPATH.

# Anexo C: Instalación Postgres

La base de datos Postgres puede obtenerse de cualquiera de los servidores *FTP* provistos por el sitio www.postgresql.org. Se obtiene un archivo postgresql-7.2.X.tar.gz, donde la X se refiere a la versión dentro del release 7.2 que se utiliza. Este archivo debe descomprimirse y expandirse con los siguientes comandos:

```
# gunzip postgresql-7.2.X.tar.gz
# tar -xvf postgresql-7.2.X.tar
```

Ambas acciones generan un directorio de nombre postgresql-7.2.X donde se descomprimen los distintos componentes de la base de datos. El siguiente paso es comenzar el proceso de instalación. Se utiliza el comando configure para revisar que todo el software accesorio esté instalado y para especificar ciertos parámetros, como pudieran ser el directorio de instalación, el directorio de almacenamiento de datos, el puerto del servidor, la comunicación con lenguajes como perl o python, el servidor de autenticación para la base y el directorio de librerías. Para el caso de nuestra base de datos se realizó el siguiente comando:

```
# ./configure --prefix=/opt/pgsql –datadir=/users/data
```

Esta configuración se utilizó debido a que se generó una partición especial para el software opcional, como es una base de datos, pero la partición de usuarios (/users) contaba con más espacio libre al momento de instalar. Dado que se tiene pensado que la base de datos crezca constantemente a través de los años, consideramos prudente localizar el repositorio de datos en el lugar que le ofreciese mayor posibilidad de expansión.

Acto seguido, se procede a la compilación del código vía la ejecución de la secuencia de comandos de shell. Esto se logra con el comando make:

```
# make
```

Finalmente, se realiza la copia y configuración de variables de ambiente con el mismo comando make:

```
# make install
```

Al terminar esta parte, la base de datos está instalada, pero aún no está lista para ser iniciada. Hay que cumplir con algunos requisitos más. El primero es que necesita existir un usuario dentro del sistema operativo llamado "postgres". Esto se logra de manera sencilla con el comando "adduser" o "useradd":

```
# adduser postgres
```

El *daemon* que mantiene el puerto abierto para los clientes que deseen conectarse a la base de datos debe ser iniciado con este usuario recién creado. Además, el usuario "postgres" necesita crear el directorio que fue especificado como repositorio de datos para poder iniciar la base de datos. Estas acciones se consiguen con los siguientes comandos:

```
# su – postgres
# mkdir /users/data
```

Finalmente, se ejecuta el comando initdb para que se generen los archivos adecuados para el manejo de las bases de datos. Nuevamente, este comando debe ejecutarse como usuario "postgres".

```
# /opt/pgsql/bin/initdb -D /users/data
```

Se generan los archivos: pg_hba.conf, pg_ident.conf, postgresql.conf y postmaster.opts. Posteriormente, es necesario iniciar el *daemon* que controla la conexión a la base. Es recomendable remitir los comandos para el inicio y detención de la base de datos a un *script* que permita la configuración de los niveles de arranque. No obstante, si aún no se tuviese, se utiliza el siguiente comando:

```
# /opt/pgsql/bin/postmaster -D /users/data >logfile 2>&1 &
```

que indica que se ejecuta el *daemon* postmaster en segundo plano, con el repositorio de datos /users/data, y con el archivo logfile como salida estándar y de error.

De los archivos generados ya mencionados, merecen especial atención el pg_hba.conf y el postgresql.conf. En el archivo pg_hba.conf, se configura con las líneas de los clientes a quien se desea permitir la conexión al servidor. Las líneas de configuración presentan el siguiente formato:

```
TYPE    DATABASE    IP_ADDRESSMASK  AUTH_TYPE
```

Donde los campos se definen según la significación siguiente:

| | |
|---|---|
| TYPE (tipo) | Tipo de un cliente. Este campo puede hacer referencia a la máquina local (local), a un cliente en específico o grupo de clientes accediendo mediante el protocolo *IP* (*host*) o conexiones seguras con SSL*IP* (hostssl). |
| DATABASE (base de datos) | Es la base de datos a la que se permite el acceso al cliente especificado en el siguiente campo. |
| IP_ADDRESS | Dirección de *IP* del cliente al que se desea permitir el acceso. Conformada por tres números separados por un punto, entre 0 y 255 cada uno. |
| MASK (máscara de red) | Máscara de red que, al aplicarse con la operación AND a la dirección de *IP*, genera la primera dirección de la red a la que pertenece el cliente. |
| AUTH_TYPE | Hace referencia al método de verificación de usuarios para acceder a la base de datos especificada. Estos métodos pueden ser: |

Trust – que no requiere contraseña para acceder.

Reject – que limita el acceso por completos.

Password – que requiere una crontraseña, que no es almacenada encriptada.

Md5 – que utiliza este algoritmo de cifrado en las versiones 7.2 en adelante.

Crypt – que utiliza este método para versiones anteriores a 7.2.

Krb5 – que utiliza Krerberos V5 para la validación desde las conexiones remotas. No aplica en sockets locales.

Para el caso particular del servidor de bases de datos, por el momento, solo interesa permitir las conexiones bcales, dado que el procesamiento de datos se hace dentro del mismo entorno. Las líneas de configuración se escriben como sigue:

```
# Para permitir el acceso local
local   all             password

# Para permitir la conexión del servidor Tomcat con el de bases de datos
host    all127.0.0.1  255.255.255.255  password
```

El archivo postgresql.conf contiene líneas de configuración relativas a la faceta de servidor de la base de datos. Algunos parámetros definibles son el puerto del servidor, el número de conexiones permitidas, las conexiones por *SSL*, el directorio, grupo y servicios para los sockets, el tamaño de memoria compartida que puede usar la base de datos, estadísticas de acceso y de consultas y algunos parámetros de optimización. Para levantar inicialmente la base de datos son solo unos cuantos parámetros los que hay que definir:

```
# Habilitar las conexiones remotas por TCP/IP
tcpip_socket = true

# Máximo de conexiones a la base de datos
max_connections = 1000

# Puerto para el servidor
port = 5432

#     Tamaño de la memoria compartida
shared_buffers = 2000                   # 2*max_connections, min 16
max_fsm_relations = 100                 # min 10, fsm is free space map
max_fsm_pages = 10000                   # min 1000, fsm is free space map
max_locks_per_transaction = 64          # min 10
wal_buffers = 8                         # min 4
```

Una vez configurados estos dos archivos, la base de datos es funcional y no presenta problemas de desempeño.

# Anexo D: Instalación Java

La instalación del compilador de Java comprende la obtención, descompresión y ejecución del software para que éste se incorpore a las herramientas existentes en el sistema. El sitio java.sun.com proporciona un archivo comprimido, el cual puede descargarse con facilidad. Este archivo se descomprime con utilerías como el zip o el gzip. Al realizar la extracción, un archivo binario, con extensión sh, ejecutable es quien realiza el trabajo de instalación. Debe asegurarse que el archivo binario cuente con permisos de ejecución, lo cual se logra con el comando:

```
# chmod +x j2sdk-1_3_1-solaris-sparc.sh
```

para todos los usuarios del sistema, o bien con el comando:

```
# chmod u+x j2sdk-1_3_1-solaris-sparc.sh
```

para el usuario al cual pertenezca el archivo. Normalmente, este tipo de instalaciones son realizadas por el superusuario *root*. Posteriormente, se ejecuta el archivo, estando en el mismo directorio, con el comando:

```
# ./j2sdk-1_3_1-solaris-sparc.sh
```

La ejecución de este archivo genera la carpeta /usr/java donde se almacenan las librerías, los ejecutables, las páginas de manual, el código fuente y otras herramientas. De igual manera, el proceso de instalación genera varias ligas dentro del directorio /usr/bin, dentro de las cuales se encuentra una hacia el compilador de Java "javac", hacia el ejecutor de bytecodes "java", hacia el generador de cabeceras de C y archivos fuente necesario para implementar métodos nativos "javah", hacia el descriptor de clases "javap" y hacia el generador de documentación *API* "javadoc". Estas últimas ligas permiten al usuario utilizar las herramientas de java sin necesidad de modificar el PATH de comandos al directorio recientemente creado /usr/java/bin.

# Anexo E: Algoritmos de máximos y mínimos

### Algoritmo para máximo

```
limpiar lista de días
máximo = 0
para cada elemento de la tabla de datos hacer
   si el elemento actual de la tabla es distinto de 9999
      si el elemento actual es mayor que máximo
         limpiar la lista de días
         máximo igual al elemento actual
         añadir fecha y hora a la lista de días
      si no
         si el elemento actual es igual a máximo
            añadir fecha y hora a la lista de días
```

### Algoritmo para mínimo

```
limpiar lista de días
mínimo = 10000
para cada elemento de la tabla de datos hacer
   si el elemento actual de la tabla es distinto de 9999
      si el elemento actual es menor que mínimo
         limpiar la lista de días
         mínimo igual al elemento actual
         añadir fecha y hora a la lista de días
      si no
         si el elemento actual es igual a mínimo
            añadir fecha y hora a la lista de días
```

**Algoritmo para máximos**

```
convertir tabla en vector
limpiar la lista de días
índice del elemento siguiente = 2
para cada elemento del vector de datos hacer
   si el elemento actual de la tabla es distinto de 9999
      mientras el elemento actual sea igual al elemento siguiente
         aumentar el índice del elemento siguiente
      si el elemento actual es mayor que el elemento anterior
      y  el elemento actual es mayor que el elemento siguiente
         añadir a la lista el elemento actual
      índice del elemento actual = índice del elemento siguiente – 1
   índice del elemento siguiente + 1
```

**Algoritmo para mínimos**

```
convertir tabla en vector
limpiar la lista de días
índice del elemento siguiente = 2
para cada elemento del vector de datos hacer
   si el elemento actual de la tabla es distinto de 9999
      mientras el elemento actual sea igual al elemento siguiente
         aumentar el índice del elemento siguiente
      si el elemento actual es menor que el elemento anterior
      y  el elemento actual es menor que el elemento siguiente
         añadir a la lista el elemento actual
      índice del elemento actual = índice del elemento siguiente – 1
   índice del elemento siguiente + 1
```

# Glosario

**ASP:** Véase *Proveedor de Servicios para Aplicaciones*.

**API:** Véase *Interfaz para la Programación de Aplicaciones*.

**Applet:** Programa escrito en Java que se almacena en Internet y que se ejecuta en un navegador compatible con Java.

**AWT:** Iniciales de Abstract Windowing Toolkit, AWT es una herramienta de Java (de plataforma independiente) para la graficación y generación de ventanas e interfaces de usuario.

**Back End:** Estos sistemas se encargan del almacenamiento de los datos de las aplicaciones y de ejecutar los procesos necesarios para responder a las solicitudes de los clientes.

**Bean:** Componente de software reutilizable. Estos componentes pueden combinarse para crear una aplicación.

**BLOB:** Binary Large Object. Tipo de dato en bases de datos orientadas a objetos.

**Bugs:** es el resultado de una falla de programación introducida en el proceso de creación de programas de computadora. El término *bug* fue acreditado erróneamente a Grace Murria Hopper, una pionera en la historia de la computación, pero Thomas Edison ya lo empleaba en sus trabajos para describir defectos en sistemas mecánicos por el año 1870.

**Business Intelligence:** Proceso de obtención de información en un campo determinado.

**Byte-Code:** Cuando en una computadora se compila un programa escrito en Java, lo que se genera es un byte-code, o archivo binario cuyo contenido, a diferencia de un archivo ejecutable, no puede ser interpretado directamente por la computadora, sino que necesita ser ejecutado en una máquina virtual de Java, que es un programa que interpreta el byte-code y transmite las instrucciones a la computadora.

**CLOB:** Character Large Object. Tipo de dato en bases de datos orientadas a objetos.

**Common Language Runtime (CLR):** Aplicación similar a un máquina virtual que se encarga de gestionar la ejecución de las aplicaciones para ella escritas.

**CSS:** Las hojas de estilo en cascada (Cascading Style Sheets) son un lenguaje formal de computadora usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirá de estándar para los agentes de usuario o navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

**Daemon:** Un daemon es un término de UNIX que se refiere a un programa ejecutado en segundo plano y que espera solicitudes. No hace falta estar en UNIX para que un programa se comporte como un daemon.

**Data warehouse:** Frase sin traducción que literalmente quiere decir depósito o almacén (warehouse) de información o datos (data). Es un depósito de datos donde una organización puede poner información al alcance de sus clientes. La información allí guardada permite análisis o búsquedas posteriores. El concepto de "hacer *warehousing*" ha ganado aceptación porque permite realizar un provechoso minado de datos (o sea, búsqueda de datos relacionados).

**Data warehousing:** Véase *Data warehouse*.

**DBMS:** DataBase Management System o Sistema de administración de base de datos. Software que permite la interfaz con bases de datos localizadas en varios lugares de la red de computación.

**DNS:** Domain Name Server. El servidor local o servidor distante que decide de los nombres de domino en las direcciones *IP*.

**ECMA:** Siglas de European Computer Manufacturers Association. Fue fundada en 1961 para estandarizar los sistemas de cómputo en Europa.

**Enterprise Java Bean:** Agrupa funcionalidades para una aplicación. Sin embargo, a diferencia de un *Java Bean,* un *EJB* es un *"deployable component"*, el término *"deployable component"* implica que existe un ambiente de ejecución. Este ambiente es precisamente un *"EJB* Container" parte de un*java application server* .

**EJB:** Véase *Enterprise Java Bean*.

**Extensible Markup Language:** Se le ve como el sucesor de HTML. Permite personalizar las etiquetas que describen la presentación y el tipo de los elementos de datos. Es muy útil para los sitios que mantienen grandes volúmenes de datos y para una intranet. Es un lenguaje de marcado, basado en una sintaxis.

**File Transfer Protocol:** Protocolo para la transferencia de archivos. El *FTP* puede ser utilizado para copiar documentos de la red a la computadora del usuario y viceversa. Los navegadores de *WWW* pueden hacer transferencias de *FTP,* pero existen programas diseñados específicamente para esta tarea. Los usuarios deben darle al programa *FTP* la dirección del servidor. También es necesario tener una cuenta en el servidor y con nombre de usuario (username) y contraseña (password), a menos que se trate un servidor de *FTP* anónimo.

**Front End:** Colección de servidores que proveen el núcleo de los servicios Web, tales como *HTTP, FTP,* a las aplicaciones cliente de *workflow*

**FTP:** Véase *File Transfer Protocol*.

**Fully J2EE Compliant:** Este término implica que se cumplen todas las especificaciones *J2EE* definidas por Sun.

**GUI:** Interfaz gráfica de usuario (GUI) es un método para facilitar la interacción del usuario con el ordenador a través de la utilización de un conjunto de imágenes y objetos pictóricos (iconos, ventanas..) además de texto. Surge como evolución de la línea de comandos de los primeros sistemas operativos y es pieza fundamental en un entorno gráfico. GUI es un acrónimo del vocablo inglés Graphical User Interface. Douglas Engelbart, además de inventor del ratón de ordenador, desarrollo la primera interfaz gráfica en los años 1960 en E.U.

**Host:** Un servidor (o host) en informática o computación tiene dos acepciones:

- Una aplicación informática que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes. Algunos servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de una computadora y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final. Este es el significado original del término. Es posible que una computadora cumpla simultáneamente las funciones de cliente y de servidor.
- La computadora en el que se ejecutan dichos programas, tanto si se trata de una computadora central (*mainframe*), una computadora personal, un PDA o un sistema integrado. Sin embargo, hay computadoras destinadas únicamente a proveer los servicios de estos programas: estos son los servidores por antonomasia.

**HTML:** El HTML, acrónimo inglés de Hyper Text Markup Language (lenguaje de marcación de hipertexto), es un lenguaje informático diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web. Gracias a Internet y a los

navegadores del tipo Explorer o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos. Contrariamente a otros lenguajes de programación, el HTML utiliza etiquetas o marcas, que consisten en breves instrucciones de comienzo y final, mediante las cuales se determina la forma en la que deben aparecer el texto, así como también las imágenes y los demás elementos, en la pantalla de la computadora.

**HTTP:** *(Hypertext Transfer Protocol)* Protocolo de Transferencia de HiperTexto. Protocolo de comunicación que posibilita las conexiones entre los clientes de *WWW* y los Web Sites. Las siglas *HTTP* se encuentran en las direcciones de las páginas electrónicas, seguidas de ://. El prefijo *HTTP* informa al servidor de qué forma debe ser atendido la petición del cliente.

**HTTPD:** Protocolo de Transferencia de HiperTexto. La "D" de *HTTPD* es la inicial de *daemon*.

**Interfaz para la Programación de Aplicaciones (API):** Especificación que determina la forma en que un programador que está escribiendo una aplicación accede al comportamiento y estado de clases y objetos.

**IP:** Protocolo de Internet. Cada computadora está identificada en Internet por una dirección numérica (por ejemplo: 188.147.6.90). Cada dirección *IP* tiene una dirección *DNS* correspondiente (por ejemplo: www.xyz.com).

**Java 2 Enterprise Edition (J2EE):** *J2EE* es un grupo de especificaciones diseñadas por Sun que permiten la creación de aplicaciones empresariales. Es importante notar que *J2EE* es sólo una especificación. Esto permite que diversos productos sean diseñados alrededor de estas especificaciones. Algunos son Tomcat y Weblogic.

**Java Application Servers:** Los Java Application Servers, hoy en día ya denominados *Application Servers* o Servidores de Aplicación, ofrecen una manera de integrar y ofrecer las funcionalidades requeridas por la gran mayoría de sistemas. Una de las razones por las cuales el mercado ha sido inundado con estos

*Application Servers* es que están diseñados alrededor de *J2EE*, que es sólo un grupo de especificaciones definidas por Sun.

**Java Bean:** Véase *Bean*.

**Java Server Pages (JSP):** Una página *JSP* es un tipo especial de página HTML que contiene unos pequeños programas (también llamados *scripts*) que son ejecutados en servidores antes de ser enviados al usuario para su visualización en forma de página HTML. Habitualmente, esos programas realizan consultas a bases de datos y los resultados de esas consultas determinan la información personalizada que se envía a cada usuario específico.

**Java Scripts (JS):** Lenguaje de programación para *WWW* desarrollado por Netscape. Al igual que VBScript, pertenece a la familia Java pero se diferencia de este último en que los programas están incorporados en el archivo HTML.

**Local Area Network (LAN):** Red de Área Local. Red de datos para dar servicio a un área geográfica máxima de unos pocos kilómetros cuadrados, por lo cual pueden optimizarse los protocolos de señal de la red para llegar a velocidades de transmisión de Gbps (gigabits por segundo).

**Mainframe:** Tipo de computadora que capaz de gestionar muchas terminales y de utilizar sistemas de almacenamiento con capacidades superiores a decenas de Gigabytes.

**Middleware:** Software de comunicaciones que reside físicamente en el cliente remoto y en un servidor de comunicaciones, localizado entre el cliente y el servidor de aplicaciones. Es el software que actúa como un traductor universal entre distintas tecnologías de radiofrecuencia y protocolos.

**MIME:** MIME (Multi-Purpose Internet Mail Extensions, Extensiones de correo Internet multipropósito) son una serie de convenciones o especificaciones dirigidas a que se puedan intercambiar a través de Internet todo tipo de archivos (texto, audio, vídeo, etc.) de forma transparente para el usuario. Una

parte importante del MIME está dedicada a mejorar las posibilidades de transferencia de texto en distintos idiomas y alfabetos.

**NIS:** El servicio de información de red (NIS) es el protocolo de Sun Microsystems equivalente a un directorio telefónico. Es utilizado para la distribución de información a través de una red, como puede ser nombres de usuario o de equipo.

**NFS:** Network File System (NFS) es un sistema de archivos distribuido para un entorno de red de área local. Hace posible que distintos sistemas conectados a una misma red accedan a archivos remotos como si se tratara de locales. Originalmente desarrollado por Sun Microsystems.

**Open Source:** Código Abierto por definición significa que puede obtenerse el código fuente, usar el programa, y modificarlo libremente sin las limitaciones del software propietario.

**PDA:** La sigla significa Asistente Personal Digital, y es el nombre con el que se designa a los dispositivos de almacenamiento de datos portátiles, como agendas, PCs de bolsillo, handhelds, palmtops, agendas electrónicas, etc. Poseen una pantalla o visor y algún método de ingreso de datos, como un teclado, un stylus (lápiz plástico) o ambos. Funcionan con sistemas operativos propios. Algunos modelos vienen con versiones reducidas de otros sistemas operativos de escritorio, como Windows, OS/2 o Linux.

**Protocolo de Aplicación Inalámbrica:** El Protocolo de Aplicación Inalámbrica (Wireless Application Protocol, *WAP*) es la primera norma mundial para los servicios de Internet a través de las redes de telefonía móvil. Permite ver "minisitios Web" que parecen sencillos comparados con los sitios Web normales, pero que ofrecen ya una gran variedad de servicios útiles, como operaciones bancarias, compra de entradas, noticias, etc.

**Proveedor de Servicios para Aplicaciones:** Un Proveedor de Servicios para Aplicaciones (*ASP*) es una compañía que ofrece a individuos o empresas acceso, a través de Internet, a aplicaciones de software y a servicios relacionados

que, de no ser así, tendrían que estar ubicados en sus propias computadoras.

**Popup window:** Ventanas emergentes o no solicitadas

**Recovering:** Facilidad de aprender a usar un sistema, preparar los datos de entrada, interpretar sus resultados.

**RDBMS:** Sistema de Gestión de Base de Datos Relacional.

**Script:** Pequeños programas en un lenguaje que típicamente es interpretado al momento de ejecución, en vez de compilarse antes.

**Stand-Alone:** Sistema que realiza su función requiriendo poca o nada de asistencia de sistemas de interfaz. Esta constituido por una computadora que funciona en forma aislada.

**Servlet:** Un programa de Java del lado del servidor que ofrece funciones suplementarias al servidor.

**Servlet Engine:** Dentro de este se ejecutan exclusivamente las clásicas aplicaciones de Servidor (*JSP*'s y Servlets).

**SGA:** Área global de sistema. Las SGAs son áreas de memoria usadas por la información compartida por la base de datos y por los usuarios.

**SSH:** SSH es el nombre de un protocolo y del programa que lo implementa. Este protocolo sirve para acceder a máquinas a través de una red, de forma similar a como se hacía con telnet. La diferencia principal es que SSH usa técnicas de cifrado para que ningún atacante pueda descubrir el usuario y contraseña de la conexión ni lo que se escribe durante toda la sesión.

**SSL:** SSL (Secure Sockets Layer) es un protocolo diseñado por la empresa Netscape Communications, que permite cifrar la conexión, incluso garantizando la autentificación. Se basa en la criptografía asimétrica y en el concepto de los certificados. La versión estandarizada por el IETF se conoce como TLS. Su mayor ventaja es que funciona entre la capa TCP y la capa de aplicación, por esto es muy fácil usarlo para proteger los protocolos de la capa de aplicación (por ejemplo FTP y HTTP) sin tener que realizar cambios importantes en los mismos.

**Structured Query Language (SQL):** Lenguaje de consulta estructurado.

**SWING:** Swing es una librería de gráficas para Java. Forma parte de las clases fundamentales e incluye componentes como cajas de texto, botones, páneles y tablas.

**TCP/IP:** TCP/IP son las siglas de Transmission Control Protocol/Internet Protocol, el lenguaje que rige todas las comunicaciones entre todas las computadoras en Internet. TCP/IP es un conjunto de instrucciones que dictan cómo se han de enviar paquetes de información por distintas redes. También tiene una función de verificación de errores para asegurarse que los paquetes llegan a su destino final en el orden apropiado.

**Telnet:** Es el nombre de un protocolo (y del programa informático que implementa el cliente) que permite acceder mediante una red a otra máquina, para manejarla como si se estuviese sentado delante de ella. Para que la conexión funcione, como en todos los servicios de Internet, la máquina a la que se accede debe tener un programa especial que reciba y gestione las conexiones. Sólo sirve para acceder en modo terminal, es decir, sin gráficos, pero fue una herramienta muy útil para arreglar fallos a distancia, sin necesidad de estar físicamente en el mismo sitio que la máquina que los tenga. También se usa para consultar datos a distancia, como datos personales en máquinas accesibles por red, información bibliográfica, etc. Su mayor problema es de seguridad, ya que todos los nombres de usuario y contraseñas necesarias para entrar en las máquinas viajan por la red sin cifrarse (en texto claro). Esto permite que cualquiera que espíe el tráfico de la red pueda obtener los nombres de usuario y contraseñas, y así acceder él también a todas esas máquinas.

**Token:** Palabras clave, identificadores, etc.

**UDT:** User Defined Types. Tipos de dato definidos por el usuario.

**Wide Area Network (WAN):** Redes de Área Extensa. Red de computadoras conectadas entre sí en un área geográfica relativamente extensa. Este tipo de redes suelen ser públicas, es decir, compartidas por muchos usuarios.

**WAP:** Véase *Protocolo de Aplicación Inalámbrica*.

**Web-Container:** Dentro de este se ejecutan exclusivamente las clásicas aplicaciones de Servidor (*JSP*'s y *Servlets*).

**Workflow:** Ritmo de trabajo o la manera de trabajar. Cuando se dice que un programa tiene un workflow intuitivo, significa qua la manera de trabajar es fluida e intuitiva.

**WWW:** World Wide Web.

**XML:** Véase *Extensible Markup Language*.

# Referencias

http://about.reuters.com/latam/productos/risk/kondor.htm
http://arcadia.inf.udec.cl/db/lab3/
http://ascii.eii.us.es/cursos/php/php4.html
http://campus-llamaquique.uniovi.es/virtual/recursos/comun/WebHTML/servidorWeb
http://cvs.berlios.de/cgi-bin/viewcvs.cgi/curso-sobre/introsobre/casos.xml?rev=1.12
http://delta.cs.cinvestav.mx/~pmejia/softeng
http://es.tldp.org/Postgresql-es/Web/navegable/todopostgresql/intro.htm#AEN38
http://es.wikipedia.org/wiki/BD_Relacional
http://kasuga.csce.kyushu-u.ac.jp/codesign/papers/doctoral-thesis-akaboshi/
http://polaris.dit.upm.es/~jcduenas/patrones/Modelo.htm
http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci211796,00.html
http://searchvb.techtarget.com/sDefinition/0,290660,sid8_gci293919,00.html
http://usuarios.lycos.es/andyhincapie/bases_de_datos/modelo_red/modelo_red.htm
http://usuarios.lycos.es/froufe/parteJ1/capa-1.html
http://Webmaster.lycos.es/glossary/X/
http://wp.netscape.com/es/eng/mozilla/3.0/handbook/authoring/advanced.htm
http://www.ahciet.net/tecnologia/dic.asp?letra=m&Significante=MIDDLEWARE
http://www.austral.edu.ar/Web/ingenieria/ing-sist/INTRODUCCION.ppt
http://www.brujula.net/cursos/
http://www.builderhouse.cl/support/temas_inicio/Websphere.jsp
http://www.ciberpais.elpais.es/d/20010823/cibersoc/soc9.htm

http://www.cicesa.com/informatica/formacion/vs_net2003_02.html
http://www.deguate.com/infocentros/gerencia/glosario/d.htm
http://www.desarrolloWeb.com/articulos/592.php?manual=15
http://www.eclac.cl/deype/noticias/proyectos/7/5497/mecovi1/prop7.htm
http://www.elisoft.net/glosario
http://www.fismat.umich.mx/~elizalde/tesis/node15.html
http://www.galileo.edu/wp/display/3832/
http://www.iespana.es/marting/Glosario/glosario.htm
http://www.inei.gob.pe/cpi-mapa/bancopub/libfree/lib619/GLOSA.HTM
http://www.ingesis.com.gt/AcercaDe.htm
http://www.interware.com.mx/secciones/java/iwejava04.html
http://www.itver.edu.mx/comunidad/material/ing-software/unidad1.ppt
http://www.iwaysoftware.com/definition/middleware.html
http://www.lasalle.edu.co/csi_cursos/informatica/glosario
http://www.learnthenet.com/spanish/glossary/tcpip.htm
http://www.lindavista.com.mx/Web/desarrolloWeb.php?subSec=Tecno
http://www.monografias.com/trabajos6/intert/intert.shtml#que
http://www.nokia.es/soporte/glosario/w.jsp
http://www.osmosislatina.com/aplicaciones/servidor_Web.htm#javaserv
http://www.osmosislatina.com/java
http://www.osmosislatina.com/jboss/basico.htm
http://www.pmforum.org/library/glossary/PMG_S06.htm
http://www.pro-3.com.mx/ab29.htm
http://www.programacion.com/
http://www.proz.com
http://www.red-tecnica.com.ar/tutor/serverftp.html
http://www.sceu.frba.utn.edu.ar/e-learning/glosario.htm
http://www.servidores.unam.mx/tutoriales/postgres.html
http://www.sobl.org/traducciones/practical-postgres/node13.html
http://www.tectimes.com/cda/glosario.asp?texto=P&codpalabra=588
http://www.terra.es/tecnologia/articulo/html/tec8153.htm
http://www.tripod.lycos.es/support/glossary/
http://www.ucm.es/info/Psyap/Prieto/alum9798/intranet01/server.htm
http://www.udabol.edu.bo/biblioteca/congresos/cicc/cicc2001/datos/Papers

http://www.upv.es

http://www.utem.cl/Web/diccioa.htm

http://www.vdkolk.com/mozilla/queesmozilla.php

http://www.wikipedia.com/

http://www.xdata.com.uy/cd/Documentos%20Xflow/Arquitectura.pdf

http://www-1.ibm.com/servers/eserver/iseries/software/Websphere/wsappserver/docs/