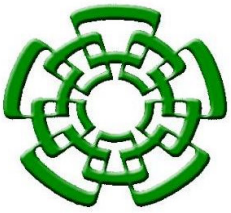


CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco
Departamento de Ingeniería Eléctrica
Sección de Computación

Desarrollo del Sistema Micro500, con Enfoque hacia una Base de Datos Activa

Tesis que presenta:

Lic. Joaquín Sergio Zepeda Hernández

Para obtener el grado de:

Maestro en Ciencias

En la especialidad de:

Ingeniería Eléctrica con opción en Computación

Director de la Tesis: Dr. Sergio V. Chapa Vergara
Profesor Investigador
Sección de Computación
Departamento de Ingeniería Eléctrica
CINVESTAV-IPN

México, Distrito Federal. Noviembre 2003

Resumen

Actualmente existen bases de datos que pueden ser consultadas desde el web, para desarrollar un sistema de este tipo muchos factores intervienen en su construcción como: el lenguaje con que se programara, la arquitectura que se utilizara, el software que intervendrá en su implementación. Parte de este desarrollo involucra un amplio estudio para obtener un sistema por demás eficiente. Una característica deseable en un sistema es que pueda extender sus capacidades y servicios de forma practica y a un bajo costo. Además de poder ofrecer un alto rendimiento y un fácil mantenimiento. Lograr todo esto en un sistema, implica considerar un buen diseño que permita disponer de todas estas características.

En el CINVESTAV existe una colección de microorganismos llamada CDBB500, y que actualmente se desea extender hacia un proyecto de bases de datos activas. Este proyecto contempla obtener una base de conocimiento a nivel nacional, para permitir facilitar el descubrimiento de información en el campo de la microbiología. El llevar a cabo este proyecto implica obtener un nuevo contexto enfocado hacia este tipo de nueva tecnología.

Debido a la semántica de los datos de los microorganismos, es necesario obtener un diseño sólido, eficiente y practico, que permita el desarrollo de una base de datos activa. El objetivo de esta tesis es obtener este diseño, permitiendo además el acceso de esta información por Internet. El trabajo aquí presentado, es parte de una primera etapa del proyecto “Ampliación de la Base de Datos Microbiológica CDBB500 y su enfoque como Base de Datos Activa”.
Esta primera etapa comprendió el desarrollo del sistema “Micro500”, que fue construido con una arquitectura modular y bajo el concepto de extensión de capacidades, para permitir en una segunda etapa la consolidación de la base de datos activa a partir de este sistema.

Contenido

Capítulo 1

Introducción al proyecto Micro500

El presente capítulo esta constituido de dos partes principales: antecedentes y presentación del proyecto de tesis. El objetivo es dar un entorno de la evolución de los sistemas de información de la base de datos de la colección de cultivos microbianos, denominada según su acrónimo CDBB-500; la cual da motivación al proyecto Micro500.

En los antecedentes se hace una reseña breve de la colección de microorganismos CDBB-500, y la evolución de los sistemas de información CDBB-500 en sus primeras versiones. El proyecto soportado inicialmente por CONABIO y CINVESTAV, tiene el resultado de un diseño de base de datos y su implementación parcial "stand- alone". El proyecto de tener una base de datos nacional y un sitio de Internet dentro de la red de CONABIO, da origen al lanzamiento de un segundo proyecto.

La segunda parte de este capítulo consiste en presentar el trabajo en cuestión, llamado Micro500. El proyecto tiene por objetivo el desarrollo de un sistema basado en conocimiento, sustentado con una base de datos abierta y tecnología cliente-servidor. Este proyecto es apoyado por el propio CINVESTAV.

1.1 Antecedentes del proyecto Micro500

La primera parte de este capítulo muestra parte de la historia de la colección de cultivos microbianos y cómo fue denominada con el acrónimo CDBB500. Así también se señalan los objetivos fundamentales bajo los cuales fue creada. Esta colección, actualmente tiene el propósito de colaborar con el desarrollo de la enseñanza, ciencia y tecnología. De igual forma se muestra parte de las actividades y servicios que brinda esta colección. En una segunda parte se describen y analizan los diferentes proyectos que han surgido a través de los años, a fin de obtener un sistema que brinde de manera eficaz una alta disponibilidad de datos.

1.1.1 La colección nacional de microorganismos

La colección de Cultivos Microbianos fue creada en el CINVESTAV, con el objeto de contar un acervo de cultivos puros para su aplicación en la docencia e investigación. A fines del año 1972, se inició su creación con el objetivo de contar con cultivos puros debido al tipo de actividades que se llevan a cabo en el departamento, y quedó formalmente integrada como colección de importancia industrial y agrícola en 1974. En 1977 fue reconocida y aceptada por el Centro Mundial de Datos Microbianos (WDC), con el acrónimo "Colección del Departamento de Biotecnología y Bioingeniería" y el numero 500, formando el acrónimo para su identificación internacional CDBB-500, que es como actualmente se conoce. Posteriormente, en 1981, por las actividades y servicios que la colección brinda tanto dentro como fuera del país, quedó afiliada a la World Federation

Culture Collections (WFCC), en 1992 la WFCC postuló a la colección a formar parte del Comité de Educación Internacional de Colecciones de Microorganismos.

Distribuidas en 56 países, existen más de 500 colecciones de microorganismos registrados en la WDC, cada una es de especial importancia cubriendo diversas actividades y con una amplia cantidad de microorganismos, virus, líneas celulares y vectores.

En los países en desarrollo existen colecciones pequeñas pero muy especializadas, con organismos únicos e interesantes de distribución limitada. Tanto las colecciones más estructuradas como las que se encuentran en desarrollo, han generado una gran cantidad de datos, originando una creciente demanda de información histórica y datos acerca de las cepas. Por lo que de esta manera, se tienen bases de datos microbianas en los países más avanzados, que se interrelacionan a través de redes de información como son: WDC, MSN, ATCC, INFO, JFCC, ECCAC, entre muchos otros. El resultado de tener vastas bases de datos y eficientes sistemas de explotación, se ha visto reflejado en un avance acelerado y homogéneo de la investigación de la microbiología de las naciones involucradas, obteniendo por consiguiente grandes beneficios de aplicación a la industria y a la biotecnología.

A la fecha, la unidad de la colección de microorganismos ha atendido a más de 300 instituciones nacionales y extranjeras en el suministro, depósito y resguardo de las cepas; obteniéndose una colección de aproximadamente 3,000 cultivos microbianos. Cada uno de ellos, tiene características especificas para ser utilizados en procesos biotecnológicos para aplicaciones industriales e investigación.

En este orden de ideas, la misión de la unidad de cultivos microbianos a nivel nacional es:

1) Depósito y resguardo seguro de las cepas.
2) Tratamiento de las cepas para procesos industriales y la investigación.
3) Apoyo a la docencia y la investigación en el área de microbiología.

Para cumplir con la misión de la unidad se tienen principalmente tres objetivos:

1) Obtener y conservar microorganismos.
2) Identificar, caracterizar y clasificar los cultivos microbianos.
3) Estudiar y documentar de manera completa.

Por tal motivo, se vio la necesidad de incorporar la tecnología de la información a una gran cantidad de tareas que fueron obtenidas de una recolección de requerimientos. En 1994, la colección nacional de microbiología dirigida por la Maestra Jovita Martínez Cruz y el Dr. Sergio V. Chapa Vergara, jefe de la sección de computación. Iniciaron un proyecto interdisciplinario para la creación de una base de datos llamada CDBB-500.

1.1.2 El proyecto del sistema CDBB-500

La primera etapa del proyecto fue el diseño e implementación de la base de datos de microbiología. Dentro del diseño se procedió a la primera etapa de recolección y análisis de los requerimientos llevada conjuntamente por el Dr. Sergio V. Chapa V. y la M. en C. Jovita Martínez Cruz, el biólogo Juan C. Estrada Mora y el estudiante de maestría Jesús M. Olivares Ceja. El objetivo fue un diseño conceptual visto desde dos enfoques interrelacionados.

a) Análisis funcional.
b) Diseño conceptual basado en el modelo.

Desde el diseño original la base de datos ha mantenido dos vistas principales:

a) Vistas de usuarios externos (requerimientos de Conabio)
   Aquí se incluyen datos taxonómicos como género, especie, rango y nombre infrasubespecífico, sinónimos, acrónimos, fuente de aislamiento, identificador, años de identificación y depósito, nombre de instituciones, depositantes, datos geográficos, bibliográficos, etc.

b) Vista de usuarios internos (de interés y estudios de biotecnología)
   A la colección se le adicionó información concerniente a la aplicación industrial de cada una de las cepas, así como también medios de cultivo, temperaturas de crecimiento, parámetros fisicoquímicos y biológicos que intervienen en la conservación de un estado viable y estable del acervo microbiano.

Obteniendo el esquema conceptual mostrado en la siguiente figura 1.1.

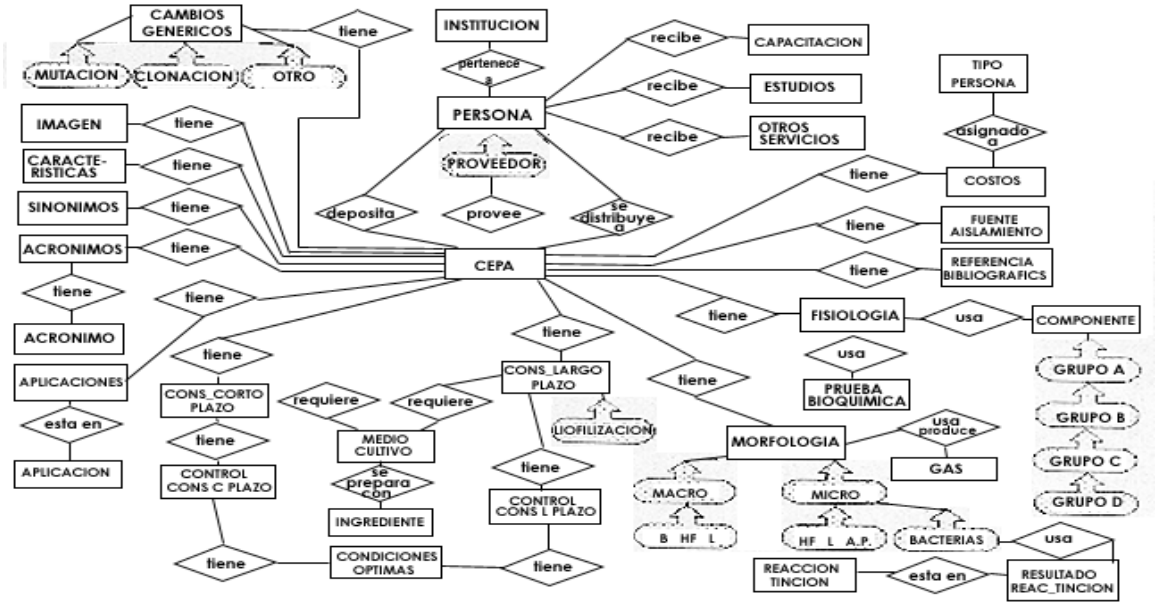


Figura 1.1 Esquema Conceptual de la base de datos CDBB-500

1.1.2.1 Sistema de Información CDBB-500 (Versión 1.0)

La primera versión del sistema se concluyo a finales del año 1995, con el proyecto de nombre "Construcción de una base de datos pictográficos con aplicación a la cole cción de microorganismos CDBB-500" [1], este sistema fue desarrollado VisualWorks, que utiliza como lenguaje de programación SmallTalk. En este proyecto se modelaron diferentes vistas, que podrían en algún momento ser implementadas al sistema. También se muestra un diseño lógico y conceptual del sistema, obteniendo un modelo relacional de la base de datos y modelando parte de un sistema orientado a objetos. Este primer sistema cuenta con las primeras interfaces de usuario para servicios de altas y bajas de microorganismos en la base de datos. Las aportaciones en este primer proyecto son:

a) Obtención del esquema conceptual.
b) Modelación de la base de datos orientada a objetos.
c) Modelación de diferentes vistas que podrán ser incluidas al sistema.

En la figura 1.1, se muestra la primera vista del sistema de información y parte de la información que estará disponible al publico. Esta es la versión 1 del sistema, y en ella se realizó un estudio de la incorporación posterior de la morfología y fisiología a futuro.

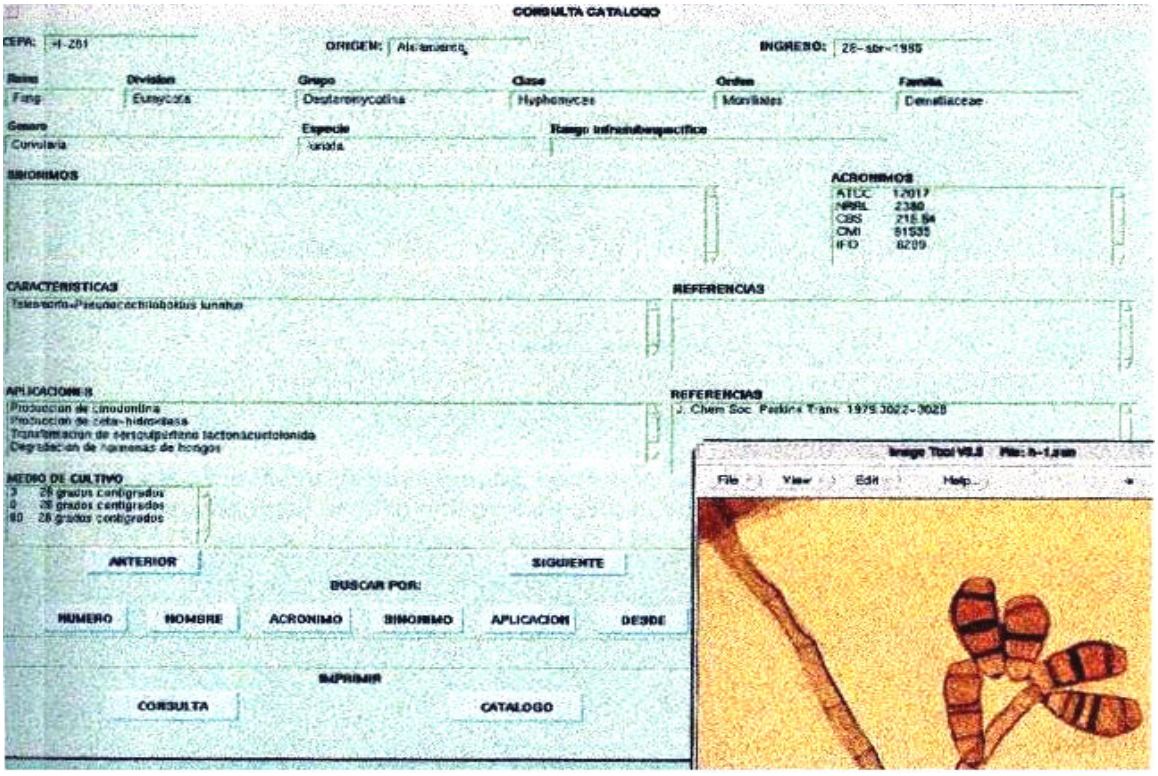


Figura 1.2 Vista del primer Sistema de Información

1.1.2.2 Sistema de Información CDBB-500 (Versión 1.1)

La siguiente versión del sistema es de gran importancia para el manejo y control integral de la colección, por estar constituida de datos procedentes de organismos vivos que deben mantener su estabilidad, para ser empleados por el hombre como una alternativa en la solución de problemas ecológicos, salud, alimentación, contaminación, agrícolas, forestales, etc. Así como su aplicación en la docencia e investigación. Diversos proyectos del curso de bases de datos conformaron los resultados de la segunda etapa:

a) Refinamiento del modelo conceptual de datos, y su diseño con SYSARCH y EVE, obteniendo un modelo relacional de datos en tercera forma normal.
b) Diseño lógico en el modelo relacional de datos.
c) Primera implantación de la base de datos en Microsoft Access 2.0, como Sistema de Información.

Los responsables de la colección junto con el Ing. Ángel E. Llanas Soto, verificaron y rediseñaron la información ya existente, concluyendo con una base de datos con 1004 cepas distribuidas en 51 tablas. A petición de CONABIO, la base de datos fue implementada en Access.

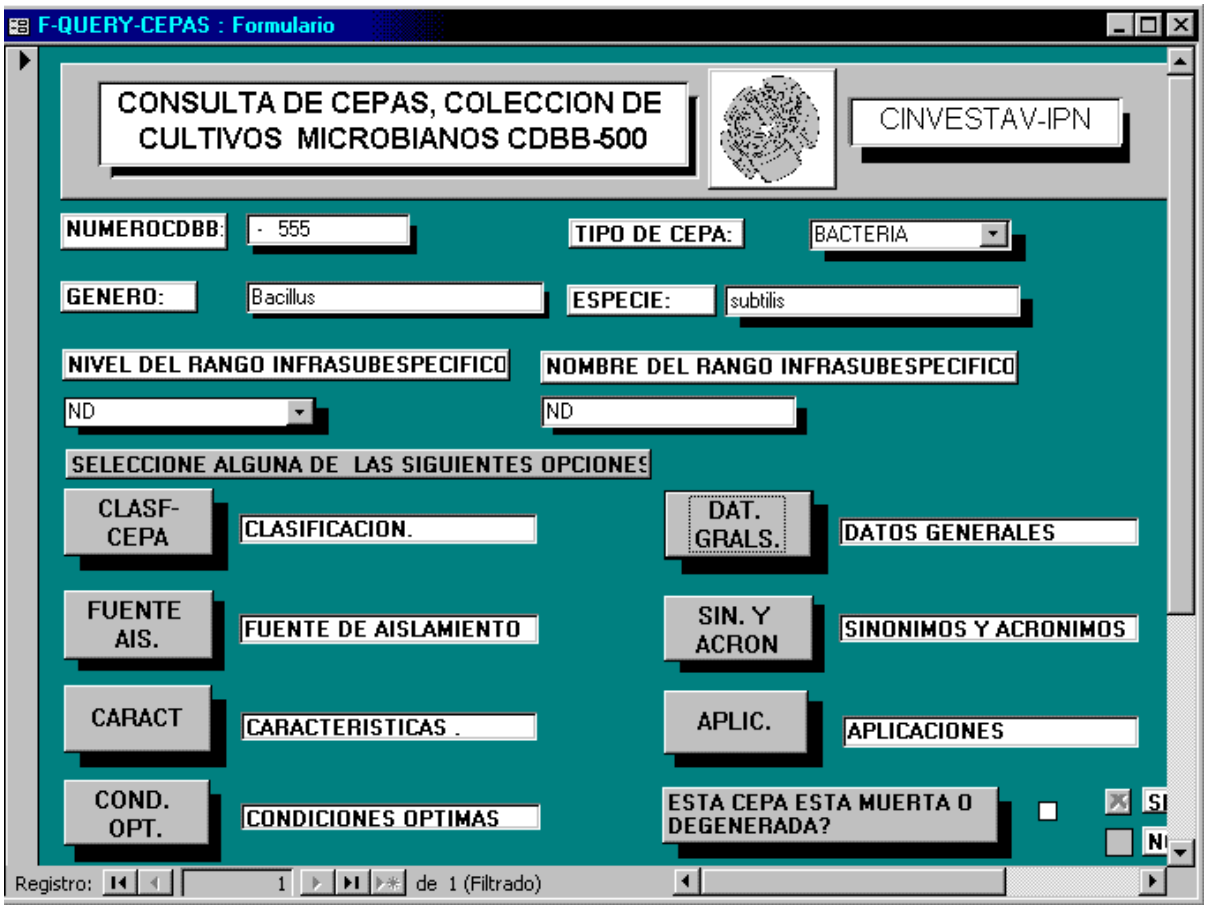


Figura 1.3 Vista del Segundo Sistema de Información

El resultado final es la Versión 1.1 del Sistema de Información CDBB-500 [2], que ya puede gestionar altas, modificaciones, bajas, consultas con referencias cruzadas y elaboración de reportes de los microorganismos.

1.1.2.3 Sistema de Información CDBB-500 y LIDAWEB

Con el objetivo de tener un sistema cliente-servidor se inicio una nueva versión, fundamentalmente con la idea de usar una tecnología de C-objetivo, Web Objetcs Builder e Interprise. El resultado fue la migración de la base de datos al DBMS PrimeBase 3.5 [3].

En esta etapa el propósito principal fue la de integrar los resultados de lenguajes visuales y bases de datos. LIDAWEB constituyó el prototipo de lenguajes de consulta para la base de datos CDBB-500 (figura 1.3). Este sistema resulta ser muy interesante desde el punto de vista de "Dinamic and Visual Queries" debido a que es una nueva forma de hacer consultas y permitir en modo de ejecución consultas no programadas.

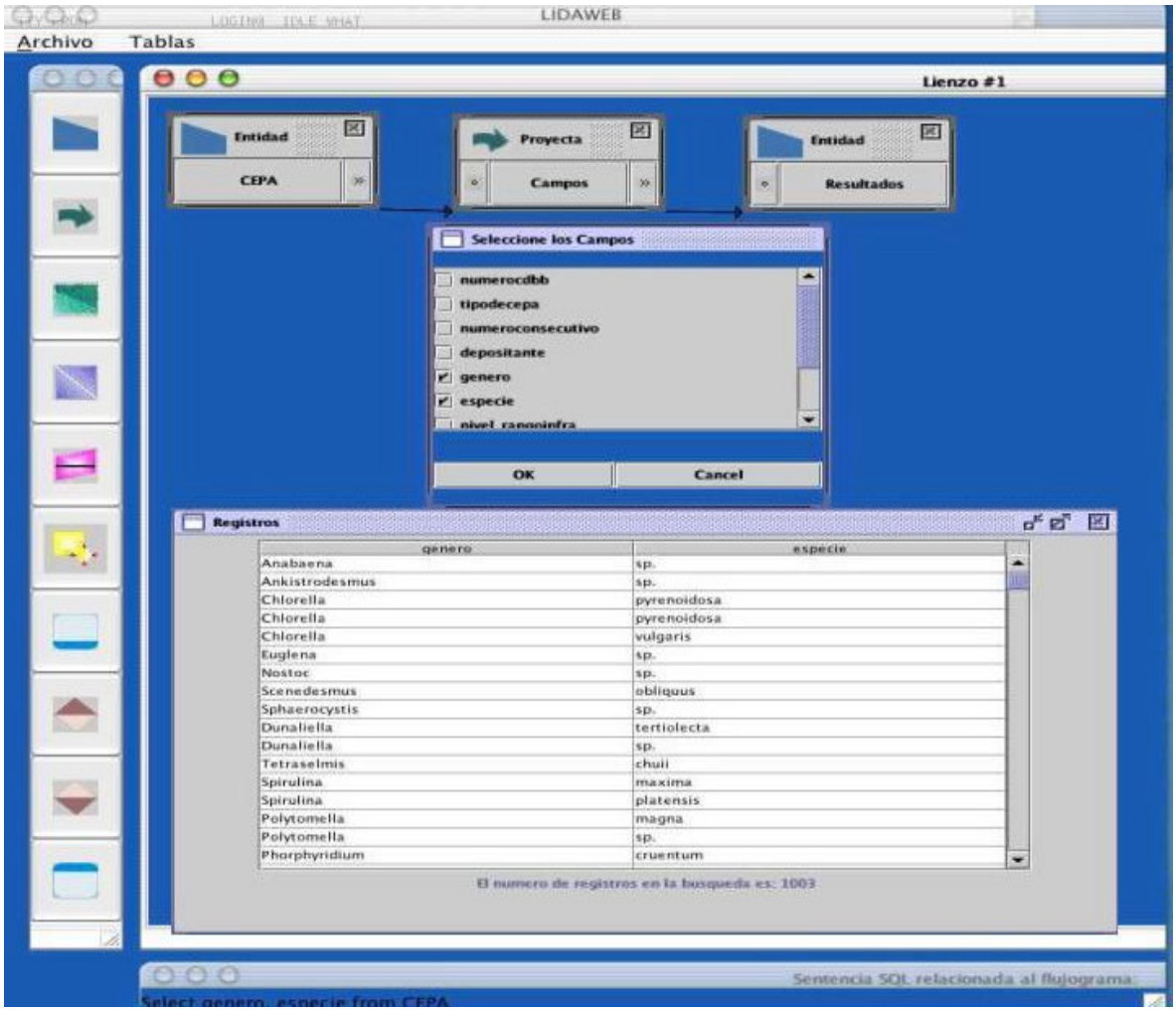


Figura 1.4 Vista del Sistema LIDAWEB

El sistema tiene una arquitectura cliente-servidor y fue implementado bajo la plataforma MacOSX 10.1, con el lenguaje visual LIDAWEB. La característica principal de LIDAWEB es la de arrastrar objetos, así como también de copiarlos y cortarlos en cualquier momento y unir a los mismos por medio de líneas, al encontrar un objeto de conexión estas se convierten en flechas con su respectiva dirección, también otra característica muy atractiva es que se tienen iconos, los cuales son usados para construir flujogramas y que al terminar de construir este tipo de diagramas visuales, el flujograma construye un query a partir de este diagrama, lo cual nos brinda la posibilidad de tener consultas no programas y creadas en tiempo de ejecución. Podemos observar en la figura 1.5 la arquitectura general del sistema, se tiene una computadora donde se encuentra el manejador de base de datos, en otra la aplicación ejecutándose en forma local y otra computadora intermedia, la cual se encarga de traducir el flujograma construido en la aplicación, una vez que es traducido como una consulta SQL, se solicitan los datos a la base de datos y se devuelve el resultado a la computadora local donde se encuentra la aplicación, de esta forma son mostrados los resultados en una tabla, donde es posible modificar su posición y tamaño. Este sistema tiene la característica de realizar consultas y operaciones sobre los datos como: adición, sustracción, intercepción, unión entre otras, pero solo funcionando sobre una red local. Un único inconveniente de este sistema, es que al inicio de su ejecución y debido a su diseño, se carga la base de metadatos para poder mostrar las opciones de cada objeto del sistema y pueda realizar su proceso, lo que hace un poco lento el arranque del sistema, debido a la carga de los metadatos en memoria.

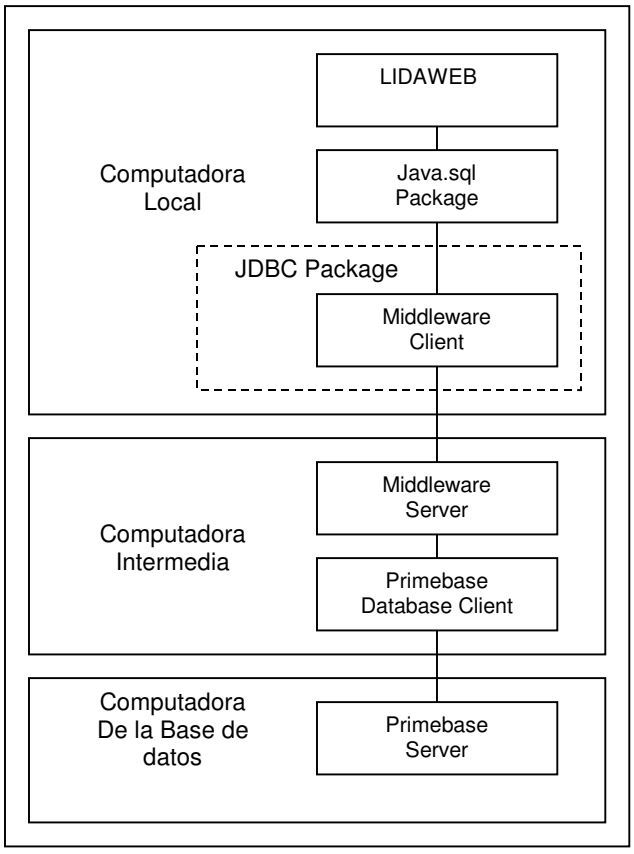


Figura 1.5 Arquitectura del Sistema LIDAWEB

1.2 Proyecto Micro500

En la descripción anterior hemos visto la evolución de diferentes sistemas aplicados a la base de datos CDBB-500, debido al gran avance de la tecnología se tiene la necesidad de diseñar un nuevo sistema con características más especiales, por las condiciones que los nuevos tiempos imponen a los sistemas de software.
Entre estas nuevas características se pueden considerar la capacidad de consulta por el web, la independencia de plataforma, diseño orientado a objetos, nuevas tecnologías orientadas al web, nuevas capacidades de los sistemas operativos, nuevos lenguajes de programación, nuevos sistemas manejadores de bases de datos (DBMS) con nuevas características, etc.

Actualmente se desea extender la investigación sobre los datos de CDBB-500, ya que se están planeando futuras investigaciones en varios campos como: Bases de Datos de Imágenes, Bases de Datos Activas, Análisis Estadístico de Datos, Visualización de datos, Simulación y Experimentación con Autómatas Celulares. De todo esto se hizo un estudio para plantear la posible incorporación de futuros proyectos al sistema Micro500, y que este brinde nuevos servicios diferentes de los que se tienen en esta etapa de desarrollo.

En la Sección de Computación del Depto. de Ingeniería Eléctrica, en conjunción con la Unidad de Servicios de la Colección Nacional de Cepas Microbianas y Cultivos Celulares del CINVESTAV. Se hizo un estudio para plantear nuevos campos de investigación sobre la base de datos CDBB-500, y se llegó a la conclusión de desarrollar un nuevo sistema, incorporando nueva tecnología de vanguardia. Para obtener así, un nuevo sistema que se adecue a los tiempos actuales y permita la incorporación paulatina de otros proyectos, bajo un estándar establecido por el propio sistema. No sin antes hacer un riguroso estudio de las nuevas tecnologías que existen hoy en día, y seleccionar entre todas ellas la más adecuada, con la visión de que no sea obsoleta en poco tiempo con el avance tecnológico.

Además, se desea obtener una independencia de plataforma para la incorporación subsecuente de nuevos proyectos, que puedan ser incorporados a futuro sin afectar lo antes ya construido. Ante todo este planteamiento, surge el desarrollo del proyecto con el nombre de un nuevo sistema llamado "Sistema Basado en Conocimiento Micro500", que pretende ser la base, donde se sumaran distintos proyectos en relación a la base de cultivos microbianos. El sistema deberá contar con una amplia facilidad de uso intuitivo, y una recomendable rapidez de procesamiento en la presentación de resultados, de forma transparente a los usuarios.

El desarrollo de este sistema deberá incorporar una amplia independencia de los datos, y de componentes de software, para ello se buscara un estándar que no dependa de un software propietario o sistema operativo especifico, de manera que si el sistema desea migrarse a otro sistema diferente al cual se construyó, deberá hacerse de manera sencilla y sin modificar la funcionalidad y objetivos del sistema.

1.2.1 Presentación de Micro500

La presente tesis es parte del proyecto "Ampliación de la Base de Datos Microbiológica CDBB-500 y su enfoque como Base de Datos Activa", soportada por el CONACYT según ref: R38479-B y responsable el Dr. Sergio Chapa Vergara.

Figura 1.6 Vista Principal del Sistema Micro500

En términos generales el proyecto Micro500 tiene los siguientes objetivos o alcances:

1.- Desarrollar un sistema que facilite el intercambio de información taxonómica, por medio de una red nacional entre las diversas instituciones.

2.- Incrementar la accesibilidad a la base de datos CDBB-500, por parte de la comunidad de microbiólogos, tratando de mantenerla actualizada y con garantía de calidad de la información.

3.- Contener directorios, catálogos y descriptores semánticos, para la vasta y creciente información microbiológica.

4.- Proveer una serie de servicios en un ambiente de interoperabilidad, que permita compartir recursos entre la comunidad y los estudios de microbiología.

5.- Implementar proyectos asociados, que permitan la investigación y el desarrollo tecnológico en computación y microbiología.

6.- Poner a disponibilidad de la comunidad la información de la base de datos, permitiendo llevar a cabo estudios de investigación en microbiología.

7.- Apoyar la docencia y formación de recursos humanos con la tecnología desarrollada.

8.- Tener un sistema con tecnología de punta , veraz, objetivo, completo, sencillo y fácil de consultar.

Para alcanzar los objetivos antes mencionados es necesario una nueva visión de tecnología de base de datos. La propuesta de base de datos activas permite gestionar al servidor de una mejor e independiente manera con sus clientes y con otros servidores. El sistema basado en la base de datos CDBB-500, requiere una amplia interoperabilidad como nodo de CONABIO. Un servidor de base de datos activas ofrece una amplia gama de posibilidades como: ejecución en SQL, programación automática en base de datos y la gestión de eventos. De esta manera se planteó un proyecto de reingeniería del sistema CDBB-500, basada en una mejor tecnología de la información con un enfoque hacia un sistema basado en conocimiento denominado Micro500.

1.2.2 Objetivo de la Tesis

El objetivo del trabajo fue desarrollar un sistema de información basado en el modelo de datos CDBB-500, que trabajara en un ambiente de interoperabilidad en Internet.

Las principales características del sistema son las siguientes:

a) Sistema con tecnología abierta, confiable y de alto rendimiento.
b) Modelo de computación basado en mensajes cliente/servidor.
c) Modelo conceptual de datos entidad-vinculo-extendido.
d) Modelo lógico relacional con Interfaz independiente ODBC o JDBC.

El sistema Micro500 ha sido diseñado con un modelo de computación basado en mensajes, en donde se divide la aplicación de procesamiento entre el proceso del cliente y el proceso del servidor. Los procesos entre el cliente y el servidor se lleva a cabo mediante una intercomunicación de capas de software. La arquitectura cliente/servidor ofrece ventajas en la optimización del hardware/software, mostrando un alto desempeño en el procesamiento de transacciones e interoperabilidad. Así nuestro principal objetivo es tener un desarrollo abierto cliente/servidor con independencia de la plataforma. El sistema basado en conocimiento Micro500, mostrado en la figura 1.6 es parte de un proyecto que podrá ser concluido en múltiples etapas.

La primera de ellas y la mas difícil comprende:

1) La recolección de los requerimientos por parte de los usuarios que serán beneficiados con el sistema.
2) Estudio de los requerimientos.
3) Conocer el funcionamiento y uso que deberá tener el sistema y con qué capacidades deberá contar.
4) Estudio de nuevas tecnologías para el desarrollo de sistemas basados en el web, y así obtener beneficios y desventajas de cada uno de ellos.
5) Seleccionar la mejor tecnología que se adecué a las necesidades del sistema.
6) Verificar que la tecnología seleccionada realmente nos ayude a implementar cada uno de los servicios que se desean incorporar.
7) Verificar la independencia de plataforma.
8) Diseñar la arquitectura del sistema.
9) Seleccionar un manejador de bases de datos para el sistema, teniendo como prioridad todas las investigaciones que se desean hacer en diferentes áreas y la incorporación de proyectos futuros.
10) Instalar y configurar todo el software que intervendrá en el sistema.
11) Migración al nuevo manejador de base de datos y verificar la integridad de que estos datos, no sean alterados al momento de la migración.
12) Implementar la aplicación en pequeños módulos independientes.
13) Incorporar la reutilización de código.
14) Probar y validar los módulos implementados.
15) Mantener una arquitectura abierta e independiente a nuevos proyectos que serán incorporados paulatinamente.

1.2.3 Comentarios Finales

La información contenida muestra parte de la historia de la Unidad de Servicios Microbianos, como empezó el desarrollo del sistema, y la modelación de una base de datos muy extensa, para dar pie a un primer sistema; que fue construido a partir de los requerimientos existentes en ese momento. Como lo ha mostrado el avance tecnológico en los diferentes ámbitos, se puede ver una evolución en este sistema. Esta evolución se ha enfocado en disponer la información de una forma rápida y sencilla, pero la construcción del sistema por la complejidad de los datos contenidos, no a resultado fácil. Por ello, se han desarrollado diferentes sistemas. Pero hoy en día debido al avance tecnológico, se hace necesario contar con un nuevo sistema orientado al web, y poner a disposición pública parte de la información existente, para apoyar la docencia e investigación en diferentes áreas vinculadas con la microbiología.

Capítulo 2

CAPÍTULO 2

La arquitectura cliente/servidor y las tecnologías web

En este capítulo se presenta un panorama general de la arquitectura cliente servidor, y las tecnologías que existen actualmente para programar un servidor de aplicaciones web, así como los servidores de bases de datos que son utilizados para desarrollar este tipo de aplicaciones. Se analizan algunas virtudes y desventajas entre diferentes tecnologías y se trata de encontrar la más adecuada, para la construcción de nuestro sistema, a fin de que cuente con las características que se necesitan como: portabilidad, reutilización de código, facilidad de programación, facilidad de localización de errores, entre otras.

2.1 Introducción a la arquitectura cliente/servidor

En la década de los ochenta las organizaciones tenían que elegir entre computadoras personales o grandes mainframes para su procesamiento de datos, las computadoras personales cada vez mas económicas y poderosas carecían de los servicios que un mainframe podía ofrecer, con el paso del tiempo se hizo necesario que los diferentes usuarios pudieran compartir información en forma instantánea. Para lograr que las computadoras personales pudieran compartir información se crearon las redes, a las que también se integraron los mainframes y a finales de la década de los ochenta se empezó a ocupar el termino cliente/servidor, refiriéndose a computadoras físicamente separadas para realizar diferentes procesos y prestaciones de servicios.
El término cliente/servidor se refiere a la relación cooperativa entre dos procesos de software, conectados mediante un medio de comunicación. En la computación cliente/servidor, un proceso servidor que actúa en respuesta a un requerimiento de mensaje enviado por otro proceso cliente, que solicita un servicio ofrecido por el servidor. Así de manera formal podemos definir el termino en dos componentes: un cliente, como un solicitante de servicios y un servidor como el proveedor de esos servicios. Una computadora puede ser tanto un cliente como servidor, en un momento determinado o dependiendo de su funcionalidad para la cual fue establecido.
La arquitectura cliente/servidor nos brinda facilidad de uso, flexibilidad, interoperabilidad y escalabilidad. Esta arquitectura de software ha evolucionado conforme a la resolución de distintos problemas.

2.1.1 Computo cliente/servidor

El computo cliente/servidor es lo opuesto a la computación independiente. Un entorno independiente puede imaginarse como algo monolítico. En un entorno de estas características tenemos una computadora que acepta la entrada de datos, procesa los datos y después muestra los resultados de ese procesamiento en su propio monitor. Durante mucho tiempo las computadoras operaron de forma independiente, contenían una aplicación, la información, la entrada del usuario, la presentación en pantalla y la salida en una única

máquina, para trasladar la información a otra computadora se debía copiar la información en un disquete y llevarla a otra computadora donde se deseaba tener la información.
Con el paso del tiempo, aparecieron redes como Novell, que posibilitaron compartir información entre distintas máquinas, este fue el comienzo de la computación cliente servidor. En los primeros tiempos de este tipo de computación, el servidor era una computadora que nos servia archivos a medida que los necesitábamos y algunas veces servicios de impresión. Este modelo aumento la eficiencia de la computación, debido a la habilidad de las máquinas para dialogar, ya que este proceso era muy limitado porque la entrada de datos, el procesamiento y la salida en pantalla, todavía estaban contenidos dentro de la PC monolítica. Pero esta arquitectura fue evolucionando hasta lograr diferentes modelos de esta Arquitectura entre las que se encuentran:

- Arquitectura de servidor de archivos.
- Arquitectura de dos capas.
- Arquitectura de tres capas.
    - Tres capas con tecnología de monitoreo en el procesamiento de transacciones.
    - Tres capas con servidor de mensajes.
    - Tres capas con un servidor de aplicaciones.
    - Tres capas con arquitectura ORB.
- Arquitectura de n- capas.

Para entender como funciona este tipo de arquitectura deberemos de tener claro como trabaja un cliente y un servidor, para ello mostraremos una definición y funcionamiento de cada uno de ellos.

2.1.2 Clientes, Servidores y middleware

En la arquitectura cliente/servidor se tienen tres principales estratos: cliente, servidor y middleware; los cuales son diferentes capas funcionalmente integradas. No solamente esos estratos son físicamente distribuidos y funcionalmente distintos, si no también suministrados por vendedores diferentes.
En cierto sentido, la estratificación inherente a la arquitectura cliente/servidor, parece ser más compleja que otras arquitecturas; sin embargo esta estratificación sirve para que la complejidad sea puesta en el paradigma divide y vencerás por integraciones complejas aisladas en cada capa. Por ejemplo, un servidor de base de datos incluye algoritmos muy complicados para el manejo de procesamiento de transacciones, pero esto es ocultado con la capa del servidor.

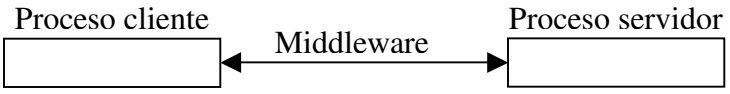


Figura 2.1 Proceso cliente/servidor

A continuación mostramos una breve descripción de estos elementos:

Cliente.- Un cliente es una computadora o dispositivo de computación, que utiliza los servicios de otra computadora. Como se explico anteriormente esta PC independiente podía estar conectada a una red, y utilizaba la administración de archivos de otra computadora para obtener cierta funcionalidad, una computadora cliente debía tener una gran cantidad de funciones integradas para que se le pudiera usar con alguna finalidad. La mayor parte del procesamiento realizado por una aplicación tenia lugar en el cliente; también residía en él gran parte de la información. Además en cliente conocía en detalle el servidor que usaba, en términos del sistema operativo. Pero con el surgimiento de la Tecnología de Internet se ha modificado la forma del trabajo del cliente, ya que el navegador se convirtió en el principal vehículo a través del cual un cliente realiza su actividad. Y esta actividad esta más relacionada con la presentación en pantalla que con el procesamiento de la información o la lógica computacional.

Servidor.- Un servidor es una computadora que provee servicios a otra. De esta manera tenemos servidores que provean servicios de administración de archivos, servidores de impresión, servidores de administración de red, servidores de bases de datos, etc. Algo muy importante a tener en cuenta cuando se trabaja con servidores es la diferencia entra hardware servidor y software servidor. La mayoría de las personas imagina al servidor como un equipo que esta allí, en algún lugar de la red. Si bien ahí existe un equipo claramente distinguible que contiene los dispositivos de almacenamiento y las tarjetas de red, la verdadera acción se produce en el software servidor.
El software servidor es un servicio, de esta forma es posible que un equipo servidor pueda ofrecer soporte para muchos tipos de software servidor. Podemos tener un servicio que provea seguridad de sistemas de archivo, otro que facilite el acceso a una impresora, otro que acepte peticiones de datos en Internet y las responda, servidores que cooperen con otros y aporten cierta interoperabilidad entre servidores, y esto puede ser tan complejo como queramos. Así cuando alguien dice que tiene un servidor de correo electrónico en realidad quiere decir que tiene un software de correo, cuando alguien dice que tiene un servidor de Internet, en realidad quiere decir que tiene una computadora en la cual se ejecuta un software servidor de Internet, debe quedar claro que ambos servidores podrían estar ejecutándose en la misma computadora.

Middleware.- El middleware es un módulo intermedio que actúa como conductor entre sistemas, permitiendo a cualquier usuario del sistema de información comunicarse con varias fuentes de información que se encuentran conectadas por una red. Desde un punto de vista amplio, una solución basada en productos middleware, debe permitir conectar entre sí a una variedad de productos procedentes de diferentes proveedores. De esta forma se puede separar la estrategia de sistemas de información de soluciones propietarias de un sólo proveedor, el concepto de middleware no es un concepto nuevo. Los primeros monitores de teleproceso de los grandes sistemas basados en tecnología cliente servidor ya se basaban en él, pero es con el nacimiento de la tecnología basada en sistemas abiertos que el concepto de middleware toma su máxima importancia.

2.1.3 El Protocolo HTTP

El protocolo de trasferencia de Hipertexto es un protocolo de nivel de aplicación para sistemas de información distribuidos y colaborativos de hipermedios. Es un protocolo genérico y sin manejo de estados, el cual puede ser utilizado para varias tareas más allá de su uso para la transferencia de hipertexto. El protocolo HTTP es un protocolo basado en el esquema solicitud/respuesta. El servidor envía una respuesta al cliente basado en la solicitud hecha por éste, cerrando inmediatamente la conexión entre ambos, el servidor no mantiene información del cliente, por ello se le conoce como protocolo sin manejo de estados; no existe el concepto de sesión como ocurre con otros protocolos. Una conexión bajo el protocolo HTTP puede expresarse de la siguiente manera:

1.- Un cliente envía una solicitud al servidor en forma de un método de solicitud, una URL y una versión del protocolo utilizado, seguido de un mensaje tipo MIME, conteniendo los modificadores de la solicitud, la información del cliente y un posible cuerpo de contenido a través de la conexión con el servidor.

2.- El servidor responde con una línea de estado, incluyendo la versión del protocolo, del mensaje y un código de éxito o error, seguido por un mensaje tipo MIME, conteniendo la información del servidor, entidades de metainformación, y posible contenido cuerpo/entidad, una vez que el servidor ha respondido la petición del cliente, se rompe la conexión entre ambos y no se guarda memoria del contexto de la conexión para siguientes conexiones. La mayor parte de las comunicaciones HTTP es iniciada por un agente del usuario y consiste en una solicitud que será aplicada a un recurso en algún servidor origen. En el caso más simple, puede ser realizada a través de una sola conexión entre el agente del usuario y el servidor origen.

Una situación más complicada ocurre cuando uno o más intermediarios esta presente en la cadena solicitud/respuesta. Los servidores HTTP responden a cada solicitud del cliente sin relacionar tal solicitud con alguna solicitud previa o subsecuente. Para solucionar este problema Netscape introdujo el concepto de cookie, el cual es un grupo de cabeceras que se agrega a las solicitudes de los clientes y a las respuestas de los servidores, llevando consigo información. Esta técnica permite a los clientes y a los servidores intercambiar información sobre el estado y colocar solicitudes y respuestas del HTTP dentro de un contexto más largo, lo cual lo llamaremos sesión. No obstante existen diferentes contextos potenciales y por lo tanto hay varios tipos potenciales de sesión.

2.1.4 Aplicaciones web

Con la introducción de Internet y del web en concreto, se han abierto infinidad de posibilidades en cuanto al acceso de la información desde casi cualquier sitio. Esto representa un desafió a los desarrolladores de aplicaciones, ya que los avances de la tecnología demandan cada vez aplicaciones más rápidas, ligeras y robustas que permitan utilizar el web.

Las aplicaciones cliente-servidor a diferencia de las aplicaciones tradicionales, no proveen una respuesta inmediata, todo depende del ancho de banda con que se disponga entre el cliente y el servidor en el momento de la ejecución.

En ocasiones en que hay poco tráfico en la red, la respuesta puede ser casi inmediata, no así en ocasiones cuando hay mucho tráfico.Cuando se desarrolla una aplicación web, se deben de tener en cuenta separar el software en tres diferentes capas que son:

La capa de presentación.- Se encuentra en el borde del sistema de software, su trabajo es capturar los estímulos de eventos externos, y realizar un grado de edición sobre los datos entrantes. También esta encargada de la presentación de la respuesta del evento hacia el mundo exterior. Los clientes pueden utilizarse para emular pantallas del servidor con muy poca lógica de presentación residiendo en el cliente. El paradigma del lenguaje de la capa de presentación está cada vez más orientado a objetos, el ambiente de ventanas en la mayoría de los sistemas operativos cliente, tiende por si mismo en forma natural a las estructuras de objetos.

La capa lógica.- Contiene el código que ejecuta y hace cumplir las políticas para mantener la integridad de los datos, las reglas y las regulaciones, así como los procesos internos, el software que ejecuta la lógica es la capa más cambiante. Puede encontrarse en el servidor central o en cualquier otro lugar intermedio. La tendencia es un movimiento hacia las estructuras orientadas a objetos, el grado de orientación a objetos empleado en la capa lógica es altamente dependiente del lenguaje seleccionado o de la herramienta de desarrollo, es posible tener componentes 3GL, 4GL, y objetos mezclados dentro de la misma aplicación.

La capa de administración de datos.- Proporciona acceso a los datos corporativos, maneja las peticiones simultaneas de lectura y escritura a la base de datos, así como la sincronización de los elementos de datos distribuidos, gran parte de la capa de administración de datos seguirá a la ubicación física de los datos. La decisión de distribuir o centralizar la base de datos determinará gran parte de la ubicación de la capa de administración de datos.
Para la mayoría de los sistemas, el paradigma de base de datos es la base de datos relacional, la mayoría de los datos recopilados por los sistemas se ajusta perfectamente al formato de columnas y renglones del paradigma relacional. Los proveedores de bases de datos relacionales también están respondiendo a la presión de extender sus bases de datos para que manejen datos no estructurados, tales como los de multimedia, sonido, video y objetos de hipertexto.

A continuación mostraremos el software utilizado para la implementación de las diferentes capas, en primer lugar se analiza la capa de administración de datos y seguidamente por la capa lógica y las diferentes tecnologías que se pueden utilizar para su implementación.

2.2 Servidor de Base de Datos

En los últimos años, el software de bases de datos ha experimentado un auge extraordinario a raíz de la progresiva informatización de casi la totalidad de las empresas de hoy día, por la gran ventaja de tener bases de datos en la industria, investigación, comunicación y muchas áreas más, no es extraño que en la actualidad existan multitud de gestores de bases de datos compitiendo por obtener el primer sitio en cuanto a sistemas administradores de bases de datos. Muchos sistemas han entrado a esta fuerte competencia y muchos han sido desplazados, y algunos otros ya han encontrado un lugar en la industria del software. La competencia es tan férrea en este campo, que los sistemas han evolucionado de manera muy rápida; cada uno de estos sistemas contiene diferentes características que los hace entrar en competencia con sus rivales. Estos sistemas son llamados DataBase Management System (DBMS). Entre estos sistemas manejadores de bases de datos mas conocidos tenemos a Oracle, Microsoft SQL Server, Borland Interbase, Informix, PrimeBase Server, todos ellos comerciales, lo que implica que tienen un costo y este costo representa una buena cantidad de dinero, además de los costos por licencias de uso en diferentes equipos o número de usuarios a los que proporcionarán servicio. Algunas veces también delimitando el tiempo de uso. Sin embargo existe la contraparte de estos sistemas con versiones libres y de muy buena reputación, que incluso sobrepasan a las versiones comerciales, tal es el caso de PostgreSQL y MySQL, que pueden ser utilizados con muy pocas restricciones, y sin ninguna remuneración económica por parte de quien los utiliza.

2.2.1 Características de un servidor de Base de datos

Una aplicación de base de datos cliente/servidor distribuye las tareas con una comunicación de infraestructura. La arquitectura de base de datos cliente servidor balancea la carga de trabajo de la aplicación entre los procesos del cliente y los procesos del server.

Un servidor de base de datos se concentra en las porciones relativas de la base de datos de la aplicación para centralizar la funcionalidad de la gestión de datos. Esta división de funcionalidad que existe entre clientes, servidores y la infraestructura de comunicación que los integra, es semejante a la división de trabajo que existe en una empresa, cada elemento tiene su propia especialidad y todos los elementos trabajan de manera conjunta, con el mejor desempeño dependiendo del medio de comunicación.

En general, el medio de comunicación de la red es mucho más lento que la velocidad de transferencia interna de los datos de la máquina del servidor. El desempeño de la red de comunicación tiende a degradarse con un tránsito de datos pesado. Los datos conjuntos de dos archivos de datos en un aplicación del servidor de archivos es controlada por un programa que se ejecuta en el cliente y no en el servidor de archivos donde residen los datos. Todo lo que potencialmente se requiera de los datos de cada archivo deberá cruzar la red para obtener del servidor de archivos ordenando por el programa lógico del cliente para decidir que incluye y que excluye.

El servidor de base de datos extiende el concepto de un servidor de archivo, mediante la implementación de un manejo lógico de datos con las capacidades propias de ejecución del

servidor de base de datos. Un cliente de base de datos envía un corto mensaje de requerimiento al servidor. Los comandos SQL dicen al servidor que servicio deberá ser ejecutado, pero no como deberá ser ejecutado. El servidor responde al mensaje requerido de los diferentes procesos de los clientes, los cuales por si mismo no tienen un significado directo de acceso a la base de datos física. El cliente y el servidor residen en hardware diferente y separado por una red de comunicación, un servidor de base de datos reside en una máquina pero no es una máquina. El servidor es mejor entendido como el proceso de software que provee los accesos a los servicios de la base de datos. La máquina donde el proceso de server reside se denomina máquina servidor. La combinación de la máquina servidor, su sistema operativo y el software de comunicación se denomina plataforma del servidor.

A pesar que las aplicaciones de base de datos cliente/servidor aparecen mas complejas que su equivalente a una arquitectura monolítica, las características de las aplicaciones de base de datos cliente/servidor potencialmente ofrecen ventajas significativas.

- Optimización hardware/software.
- Desempeño en el procesamiento de transacciones.
- Concurrencia.
- Multiversiones.
- Sistema abierto.
- Portabilidad.
- Interoperabilidad.

El servidor de bases de datos es asistido por la optimización de hardware/software, mediante la relevante gestión de datos centralizada, mientras otras características son distribuidas. La administración y la seguridad es centralizada en funciones del servidor de base de datos y servicios de la base de datos. Esos servicios son programas del cliente que usualmente se ejecutan en la máquina servidor y estos servicios ayudan a la optimización del ambiente de la base de datos como:

- Monitoreo del desempeño.
- Velocidad del servidor.
- Utilización de datos.
- Carga y descarga de datos.
- Transferencia de datos entre base de datos y archivos.
- Inicio, suspensión y caídas del servidor.
- Control de la seguridad en la integridad de datos.
- Acceso restringido a personal autorizado.
- Acceso restringido a computadoras autorizadas.

2.2.2 PostgreSQL y MySQL

El buen rendimiento y poder que han mostrado estos dos administradores de bases de datos, son indiscutibles y debido ha esto, existe una fuerte controversia sobre cual es el mejor. Las diferentes experiencias de los programadores, han desatado una gran discusión y cada quien ha tomado partido por uno o por otro, dependiendo de su experiencia y de cuál de ellos usan actualmente. Los dos son muy buenos, pero por diferentes artículos, comentarios y experiencia de diversos programadores, resaltan muchas características de PostgreSQL que MySQL no contiene, a continuación se muestra la tabla de comparación:

Nombre	**MySQL**	**PostgreSQL**
Versión	3.23.41	7.1.3
Licencia	GPL	BSD
Plataformas	Linux, Solaris,HP-UX, AIX IRIX, FreeBSD, Mac OS X, Windows 9X/NT/2000/XP, NetBSD,OpenBSD,BSDI, Compact Tru64,DEC,OS/2.	Linux, Solaris,HP-UX, AIX IRIX, FreeBSD, Mac OS X, Windows NT Server 2000 NetBSD,OpenBSD,BSDI, Compact Tru64,BeOS,SCO OpenServer,Unixware,QNX.
Cumplimiento SQL standar	Media	Alta
Velocidad	Media/Alta	Media
Estabilidad	Alta	Alta
Integridad de datos	No	Si
Seguridad	Alta	Alta
Concurrencia	Media	Alta
Soporte de Vistas	No	Si
Soporte Subconsultas	No	Si
Replicación	Si	Si
Procedimientos Almacenados	No	Si (PL/pgSQL,PL/Perl, PL/TCL)
Soporte Unicode	No	Si
Soporte Disparadores	No	Si
Integridad Referencial	No	Si
Interfaces de Programación	ODBC,JDBC, C/C++,PHP, Perl,Python,OLEDB,Delphi.	ODBC,JDBC, C/C++,PHP, Perl,Python,Tcl/Tk,SQL Embebido en C
Claves Foráneas	No	Si
Transacciones	Si	Si
Tipos de tablas Alternativas	ISAM,MYISAM,Gemini, BerkeleyDB,InnoDB,MERGE HEAP.	PostgreSQL mantiene su propio sistema de tipos de tablas.
Backups en caliente	Si	Si

Tabla 1 Comparativa entre PostgreSQL y MySQL

En la tabla 1 podemos observar una comparación de MySQL contra PostgreSQL, en donde PostgreSQL cuenta con características un poco más avanzadas que MySQL. Se resaltaron en gris las más importantes y las que se desean para nuestro sistema, ya que se pretende hacer investigación con bases de datos activas, para lo cual se necesita soporte para disparadores de eventos, y se quiere tener una alta integridad de los datos por su importancia.

En comparación con MySQL, PostgreSQL es más lento en inserciones y actualizaciones porque cuenta con cabeceras de transacción que no tiene MySQL, además la velocidad de respuesta que ofrece PostgreSQL con bases de datos relativamente pequeñas, puede parecer un poco deficiente, pero esa misma velocidad se mantiene cuando la base de datos es realmente grande, cosa que resulta muy conveniente y atractiva desde el punto de vista de eficiencia y estabilidad, este comportamiento no lo aseguran otros DBMS, ante el cambio de tamaño de la base de datos.

PostgreSQL es uno de los mejores sistemas de administración de bases de datos que compite con los mejores gestores de datos comerciales, como Oracle, Informix, Microsoft SQL Server, Borland Interbase entre muchos otros. Cuenta además con todo lo que algunos de estos servidores comerciales, excepto por una interfaz para interactuar con el administrador. Una vez familiarizado con los comandos resulta muy práctico, pero también se pude integrar con una herramienta que se distribuye con el código de PostgreSQL, esta herramienta esta en PHP, así que para integrarla deberemos tener el servicio PHP, esta no es la única forma ya que fácilmente se puede crear una con código java y la tecnología JSP.

Una de las formas para decidir si se utiliza MySQL o PostgreSQL, es poner en la balanza y considerar si se quiere " Rapidez o Potencia", en este trabajo nosotros queremos potencia de programación y se decidimos optar por PostgreSQL.

2.2.3 Antecedentes de PostgreSQL

Todo comenzó como un proyecto denominado Ingress, y poco más tarde fue desarrollado comercialmente por Relational Technologies Ingres Corporation. En 1986 un equipo de programadores dirigido por Michael Stonebraker de Beckerley, continuó el desarrollo del código de Ingres para crear un Object Relational Database System llamado Postgres, pero no hubo una versión operativa hasta 1987. La versión 1.0 fue liberada para unos pocos usuarios en junio de 1989, un año más tarde en 1990, se liberó la versión 2.0, pero hubo muchas críticas sobre su sistema de reglas, lo que obligo a replantear el sistema y lanzar la versión 3.0 en 1991. La versión incluía una serie de mejoras como mayor eficiencia en el ejecutor de peticiones. Más adelante las demás versiones se centraron en la portabilidad del sistema, y el proyecto se dio por finalizado con la versión 4.2 debido la imposibilidad de mantenimiento por parte de los desarrolladores.
En 1994 se retomo el proyecto y se le añadió un interprete de SQL y se le renombró como Postgres95 y además liberado en Internet como un proyecto Open Source; todo el sistema estaba programado totalmente en el lenguaje C y se mejoró su motor interno. En 1996 los desarrolladores decidieron nuevamente cambiar el nombre a PostgreSQL que era la versión 6.0. Actualmente se tiene disponible la versión 7.3.2.

2.2.4 Características PostgreSQL

Entre las características más avanzadas con las que cuenta PostgreSQL tenemos:

- Transacciones.
- Disparadores.
- Restricciones Avanzadas.
- Replicación.
- Backup y Recuperación.
- Reglas.
- Procedimientos Almacenados y Funciones.
- Integridad Referencial.
- Sintaxis ANSI SQL 89, 92 y 98.
- Logging.
- Extensivo y programable.
- Orientado a Objetos.
- Características sofisticadas de integridad de datos.
- Tipos de datos y funciones definidos por el usuario.
- Máximo tamaño de una base de datos: ilimitado, sólo limitado por la capacidad de almacenamiento del hardware.
- Máximo tamaño de una tabla: hasta 64f Tb (terabytes).
- Máximo tamaño de un campo: 1Gb.
- Máxima cantidad de tuplas o registros: Ilimitado.
- Máxima cantidad de columnas en un tabla: hasta 1600.
- Máxima cantidad de índices por tabla: Ilimitado

2.3 Tecnologías Web

En la actualidad existen diferentes tipos de tecnologías para crear aplicaciones web y conectarse con sistemas de bases de datos para mostrar la información requerida. El tener que escoger alguna tecnología en especifico para crear un nuevo sitio, depende en gran parte del enfoque desde el cual será construido. Existen múltiples tecnologías, algunas de ellas son propietarias, otras son libres y se pueden utilizar páginas estáticas o dinámicas para dar servicio a los usuarios.
Estos sistemas usan diferentes tecnologías y diseños, dependiendo del grado de seguridad que el mismo sistema requiere, así tenemos que no es lo mismo diseñar un sistema que publique noticias, a otro que sólo muestre cierta información, o un sistema para una tienda virtual, donde se manejan números de tarjetas de crédito para comprar; en este caso se requiere alta seguridad en la transferencia de datos, ya que se pueden tener grandes pérdidas de dinero, si esta información es usada por terceros con fines maliciosos.

Todas estas características son tomadas en cuenta para el desarrollo de un nuevo sistema, muchas veces el diseño puede ser rígido y no se pueda permitir sumarle capacidades, viéndose en la necesidad de rediseñarlo y recodificarlo en su totalidad. En la actualidad, al

tratar de desarrollar un nuevo sistema, se debe tener la visión de qué lenguaje o sistema operativo sobreviva este rápido avance tecnológico.

A través de estos últimos años, han existido múltiples lenguajes de programación que en determinado momento han ocupado un lugar importante, pero en muy poco tiempo se han vuelto obsoletos ante otros de nuevo surgimiento. Pocos lenguajes han podido sobrevivir este continuo avance tecnológico, y han tomado una nueva visión. Muchos de ellos trabajan actualmente bajo el concepto de orientación a objetos, y trabajan como herramientas de desarrollo rápido de aplicaciones, las conocidas herramientas RAD; entre las que se encuentra Delphi, Borland C++, Visual C++, JBuilder, J++, Kylix, etc.

Algunos mas utilizan la Interfaz de programación de aplicaciones conocida como API, para construir aplicaciones visuales, algunas veces la herramienta de desarrollo nos limita y obliga a trabajar bajo cierto sistema operativo, esto se vuelve un problema al paso del tiempo, porque en algunos casos puede existir una incompatibilidad con nuevas versiones del sistema operativo, o quizás si la herramienta es propietaria dependerá de licencias, y si en determinado momento ya no se pudieran pagar, se estaría obligado a hacerlo si se desea seguir utilizando la aplicación. Muchas veces uno se puede quedar atado también al sistema operativo, ya que la aplicación no puede ser migrada debido a incompatibilidades con otros sistema operativo diferente a donde se construyo.

2.3.1 Elementos de presentación

La implementación de un sistema con cierto lenguaje de programación, depende de un estudio del uso del mismo sistema y hacia que fin está orientada su construcción. Una vez ubicado el uso se debe hacer un estudio sobre que tecnología usar y definir el tipo de sitio web a implementar, ya sea con páginas estáticas o dinámicas.

Página estática.- Es aquella que al ser mostrada, su contenido nunca cambia, esta página puede ser construida con diferentes herramientas disponibles para crear páginas web. Cuando un servidor web recibe la petición de una página estática, esta deberá disponible en el servidor y su contenido se mantendrá sin cambios entre una petición y otra. Esto se traduce simplemente en encontrar el archivo solicitado y presentarlo al usuario, el archivo también puede contener imágenes, gif animados, archivos flash, scripts, applets, archivos de audio, archivos de video, o algún otro tipo de contenido que las puede hacer muy vistosa. De está forma se pueden construir sitios web y moverse entre páginas estáticas para mostrar cierta información. En este tipo de páginas se pueden contener ligas a otros sitios, o poner a disposición archivos para que puedan ser bajados desde el web a la máquina del cliente, aunque podamos hacer este tipo de operaciones, sigue siendo una página estática.

Paginas dinámicas.- Este tipo de páginas se construyen en tiempo de ejecución, y presentan información que cambia rápidamente o que constantemente se esta actualizando. Con este tipo de páginas, es necesario que se procese la solicitud de un cliente para poder obtener una respuesta.

Para devolver una respuesta, se puede depender de varios factores que pueden ir más allá de la simple localización de un archivo que la contiene, como puede ser la fecha o la hora de petición, el país de donde parte, la identificación del usuario, o los datos proporcionados por el usuario mismo. Después de una petición la página es construida en tiempo de ejecución y los datos son incorporados hasta mostrar un formato preestablecido.
Este tipo de páginas dinámicas web son muy útiles en la consulta de bases de datos, ya que la información de una base de datos es muy grande, y las solicitudes de información de los usuarios pueden ser muy variadas, para ello se puede obtener una pregunta del cliente en tiempo de ejecución, pasar la petición a la base de datos, obtener la respuesta y esta ser presentada al usuario. Si la información cambia o se actualiza momento antes de que se realice la petición, entonces al ser solicitada se mostrara la información actualizada. Al desarrollar páginas dinámicas, los elementos dinámicos se dividen en dos tipos, los cuales son Cliente-Side y Server-Side.

Cliente-Side.- Son elementos embebidos en el código HTML, viajan hasta el navegador del cliente y se ejecutan en la máquina del cliente, brindando una funcionalidad definida sin comunicarse nuevamente con el servidor. Ejemplo de esto podría ser una calculadora en Internet, o la comparación de un formato requerido en una forma antes de ser enviada al servidor. Los lenguajes utilizados para este propósito son Java Applets, VBScript, Java Script.

Server-Side.- Estos elementos son ejecutados en el servidor y generan datos que son enviados al cliente como lenguaje HTML, algunos de ellos son embebidos en código HMTL reenviando al servidor una petición de respuesta, para este tipo de elementos se han desarrollado múltiples tecnologías que algunos web server soportan, sin necesidad de incluirle cosas adicionales. Algunas veces es necesario tener un servidor independiente que soporte una tecnología especifica, entre este tipo de tecnología se encuentra el CGI, FAST CGI, ISAPI, ColdFusion, ASP, PHP, Servlet/JSP, ahora se describen brevemente, las características de cada una de estas tecnologías.

2.3.2 Tecnología CGI

La primera forma de creación de contenido dinámico en páginas web, fue a través del mecanismo conocido como Common Gateway Interface (CGI), a través del cual los servidores web pueden pasar información a páginas externas, que serán ejecutadas en el servidor web para generar respuestas en tiempo de ejecución. CGI no es un lenguaje de programación, sino un protocolo que funciona por medio de shell scripts, que invocan a una aplicación que puede ser escrita en cualquier lenguaje de programación y que pueda ser ejecutada por el servidor web como Python, C, C++, Java, Perl y así devolver un archivo con una cabecera estándar de página web.

En esta tecnología existen muchas deficiencias que la hace un poco desaconsejable en aplicaciones medianamente complejas, ya que por cada petición realizada se genera un nuevo proceso externo, esto no quiere decir que cada vez que un cliente hace una solicitud al CGI, el servidor debe crear un proceso a nivel del sistema operativo, carga el interprete, el script y luego al terminar debe deshacerlo todo, esto se vuelve un verdadero problema

cuando tenemos muchas peticiones simultaneas, por que se genera una sobrecarga difícil de soportar, y se consume mucha memoria RAM, lo que puede ocasionar un bloqueo de recursos que el propio web server necesite acceder.

FastCGI.-Es una extensión del CGI y mejora su rendimiento reduciendo la cantidad de procesos, ya que se encarga del manejo de recursos, memoria y mantiene un sólo proceso por usuario, sin importar las múltiples conexiones que él mismo tenga. Las páginas web se generan embebiendo HTML directamente en el lenguaje de programación, esto ocasiona que sea muy difícil manejar la capa de presentación que queda incrustada en la lógica.

ISAPI.- Esta tecnología sólo sirve para la plataforma Windows NT, ya que se utiliza en una librería dinámica DLL, que es cargada la primera vez para que quede a disposición en memoria sobre el espacio de direcciones del servidor web, posibilitando que los recursos estén disponibles. Cuando estas DLL no sean utilizadas durante cierto tiempo de inactividad pueden ser descargadas de memoria, hasta que ocurra una solicitud hacia ellas.

2.3.3 Tecnologías ASP, ASP.NET, ColdFusion y PHP.

Estas tecnologías tienen como característica común que son software propietario excepto PHP, y que de una u otra forma se deben pagar ciertas licencias por su uso. El que tengan un costo no indica que sean las mejores y que no tenga cierto inconveniente su uso. También es sabido de algunos problemas fuertes de seguridad, sin embargo esto no ha impedido que en la actualidad sean utilizados en una gran cantidad de sitios web. A continuación se describen brevemente, algunas características de cada una de ellas.

ASP.- La tecnología Active Server Pages de Microsoft, está basada en la inclusión de etiquetas y permite utilizar el lenguaje VBScript. Este lenguaje da soporte a componentes ActiveX que puede encapsular prácticamente cualquier tipo de funcionalidad, incluyendo la manipulación de archivos y acceso a una base de datos. A pesar de todo esto, existe una muy fuerte limitación al uso de esta tecnología ya que es propietaria de Microsoft, y esta diseñada para trabajar sólo sobre el servidor web Internet Information Server IIS; que se ejecuta sólo sobre plataformas Microsoft Windows.

Algunos vendedores han desarrollado herramientas para trabajar sobre otros servidores como Apache en servidores UNIX, pero el poder de esta tecnología deriva precisamente por la inclusión de ActiveX, y no esta disponible para otra plataforma que no sea Microsoft, lo cuál la hace totalmente dependiente de la plataforma, siendo esto algo no muy deseable para el desarrollo de algunos sistemas.
ASP.NET.- Es una extensión de ASP, sólo que muchas características nuevas son adicionadas, como la devolución de código HTML puro al cliente, el soporte para otros lenguajes de programación como VB.Net, JScript, C++, VBScript entre otros, pero tiene las mismas desventajas de ASP en cuanto a dependencia de plataforma.

ColdFusion.- Conocido como ColdFusion Markup Language (CFML), creado por Allaire, se basa en una serie de etiquetas HTML/XML, que soportan una gran variedad de acciones para la generación de contenido dinámico, y elementos predefinidos para acceso a bases de

datos, servidores de correo y también Servlets o Enterprise Java Beans (EJB). Puede correr en varios servidores como Server ColdFusion, JRun de Macromedia, IIS de Microsoft, WebSphere, etc. Se pueden encapsular funciones especificas de una aplicación que pueden ser desarrolladas en C++ o Java.

Un gran inconveniente es la existencia de algunos problemas en sistemas que no son de Microsoft, y por si fuera poco, se han reportado varios fallos de seguridad importantes que pueden poner en riesgo la integridad de los datos, aunque no deja de ser una buena tecnología de desarrollo rápido de sistemas web. No debemos olvidar que es software propietario y su costo es elevado ya que también se requiere de licencias para utilizarlo.

PHP.- Personal Home Page (PHP), es una tecnología de código abierto en pleno crecimiento y utiliza una sintaxis muy parecida al lenguaje C, con características adicionales tomadas de C++, Perl y Java. El código se puede convertir en algo un poco complejo, ya que se tiene una gran cantidad de funciones para acceso a bases de datos o servidores de correo y se dispone de extensiones para comunicarse con otros recursos.

PHP esta disponible para múltiples plataformas y es compatible con Windows, Mac OS X, UNIX, LINUX, y se puede utilizar con un gran número de servidores Web como Apache, Microsoft IIS, WebSphere, entre otros, y tiene acceso a diferentes sistemas administradores de bases de datos como MySQL, PostgreSQL, Access etc.

Esta tecnología ha sido de las más utilizadas en sitios web y ha demostrado ser de las mejores, gracias a su total independencia de plataforma y la conexión con diferentes bases de datos. Al igual que las otras tecnologías, utiliza su propia sintaxis y es muy parecida al lenguaje de programación C, pero una desventaja que mostró recientemente y que causó mucha molestia a los programadores, fue la incompatibilidad entre la versión 4.0 de reciente lanzamiento y las versiones anteriores, aunque no se muestra una incompatibilidad total, si se modificó parte de la sintaxis drásticamente, causando problemas de migración a la versión 4.

2.3.4 Java y la Tecnología JSP/Servlets

El lenguaje de programación Java es un lenguaje joven en comparación con muchos otros, al inicio de su lanzamiento muchos desconfiaban de él por ser un lenguaje nuevo, frente al poderoso C o al reciente en esos tiempos C++, muchos trataron de resistirse al cambio, pero poco a poco lo tuvieron que aceptar si querían que sus aplicaciones sobrevivieran al desarrollo tecnológico. A través de este corto tiempo fue ganado popularidad mucha de ello debido a su independencia de plataforma, ya que no es necesario rescribir las aplicaciones si no sólo migrarlas. En la actualidad java empezó con applets, aplicaciones pequeñas, y a nivel local, extendiéndose a infinidad de aplicaciones, como aplicaciones en red, aplicaciones web, aplicaciones de tiempo real, acceso a casi todo tipo de bases de datos vía ODBC o vía JDBC desde casi cualquier sistema operativo, entre otras.

Java es un lenguaje de programación orientado a objetos diseñado para ser portable en diversas plataformas y sistemas operativos, esta característica lo hace muy poderoso en

comparación con otros lenguajes de programación. Desarrollado por Sun Microsystem se diseñó con base en el lenguaje de programación C++, e incluye características especiales que lo hacen ideal para crear programas para Internet. De acuerdo con el creciente soporte y popularidad de java junto con el crecimiento explosivo de Internet, nos da paso a una nueva generación de programas en computadora.

ServletsContainers.- Para poder utilizar la tecnología JSP, es necesario un contenedor de servlets, el cual proporciona la conexión entre el servidor web y los servlets. También un entorno de ejecución y es responsable de cargar e invocar a los servlets generados cuando son solicitados. Además de incluir páginas JSP, servlets, applets y páginas HTML estáticas, administra parte de los recursos para que sean utilizados por los servlets y generar una respuesta. Para poder hacer uso de un contenedor de servlets existen varias formas, la primera de ellas es por medio de adicionar módulos a algunos servidores web existentes como Apache o IIS, aunque no todos los servidores web pueden contar con esta opción. Otra forma es por medio de servidores contenedores de servlets como JRun, ServerColdFusión, Tomcat. Los dos primeros son comerciales, mientras que el ultimo es la especificación oficial del proyecto Jakarta, entre Sun Microsystem y Apache Software Foundation, cuyo objetivo es la implementación de código abierto de las especificaciones Servlet y JSP.

Servlets.- En 1996 fueron introducidos los servlets como pequeñas aplicaciones que añaden funcionalidad interactiva a los servidores web. Los servlets java son más eficientes, fáciles de utilizar, poderosos, transportables, seguros y económicos, que el CGI tradicional y con muchas características parecidas, pero a diferencia de este no necesita lanzar un nuevo proceso por cada petición, si no que los servlets asociados a un servidor web se ejecutan en un único proceso, siendo la maquina virtual de java la que crea una tarea especifica para atender cada petición. Además se dispone de todas las ventajas de java como son, programación orientada a objetos, manejo automático de memoria, portabilidad, independencia entre plataformas, disponibilidad de una gran colección de APIs java, como acceso a bases de datos, telefonía, correo electrónico, voz, etc. Por lo tanto la ventaja que ofrecen los servlets es tremenda desde este punto de vista.
Las desventajas que presentan es que se necesita una gran cantidad de código para realizar una tarea simple, el servlet se puede volver inmanejable e inmantenible cuando contiene una gran cantidad de código, destinado a la presentación de resultados. El contenido completo de la página es generado a través del código incrustado en el propio servlet y cualquier modificación por simple que sea, requiere la recompilación y volver a cargar los ficheros al servidor, esto complica en gran medida la reusabilidad del código y complica las tareas de desarrollo entre la parte de diseño gráfico y el desarrollo de la aplicación.

JSP.- La tecnología Java Server Pages, es el resultado de Sun Microsystem para separar la programación, de la presentación del contenido. En 1998 apareció la especificación 1.0 dándola a conocer como un híbrido, por qué por un lado se soporta código embebido en las páginas al igual que en ASP, PHP, y por otro, se permite el uso de etiquetas que interactúan con objetos java en el servidor, para brindar una respuesta como en el caso de ColdFusion.
La tarea de presentación de contenido dinámico es separada en dos partes para facilitar la programación y reducir el coste de creación y mantenimiento. Estas partes son la lógica de aplicación y la lógica de presentación. La primera se enfoca a la creación de contenidos y a

los datos de entrada, al procesamiento y los nuevos datos de salida. La segunda parte comprende sólo la presentación del contenido o diseño gráfico en que se presentaran los resultados obtenidos. De esta forma los diseñadores de la página web pueden hacer modificaciones y editar la parte estática de la página, tantas veces como se quiera sin afectar la lógica de la aplicación. Del mismo modo, los desarrolladores pueden introducir cambios a la aplicación a nivel de programación o de un componente JavaBean, sin tener que editar las páginas que hagan uso de este componente.

La especificación JSP 1.1 que muestra más características avanzadas que su predecesor, fue liberada a finales de 1999, e introdujo una nueva idea llamada etiqueta de usuario y se basa en la arquitectura utilizada en ColdFusion. La tecnología JSP nos permite incluir scripts embebidos en el código HTML, y código java que puede ser ejecutado por el servidor, o en su caso hacer llamados a objetos java que están en el servidor y esperar una repuesta de ellos, además de permitir el uso de etiquetas y directivas propias de JSP, que pueden brindar una gran funcionalidad que otras tecnologías no pueden ofrecer.
Así después de probar diferentes tecnologías en este proyecto, se opta por utilizar la tecnología JSP como tecnología intermedia, para tener la capacidad de mostrar páginas dinámicas, y tener una conectividad con nuestra base de datos por medio de JDBC, y que éstos puedan ser mostrados hacia el exterior; por medio de una página web creada en tiempo de ejecución.

2.3.5 Comentarios Finales.

Parte del desarrollo de un sistema, consiste en plantear diferentes tipos de lenguajes de programación y software que intervendrán para obtener la funcionalidad deseada. Una parte muy importante es el hardware sobre el cual funcionará nuestro sistema, al igual que el sistema operativo sobre el cual estará implementado. Por el software elegido podemos observar que tenemos una total independencia de plataforma, ya que cada elemento está disponible para los diversos sistemas operativos más comunes y utilizados en estos días, tales como Windows NT, Unix, Linux, MacOSX. Así mismo, se ha comprobado la calidad y rendimiento de cada elemento utilizado, a través de la experiencia de múltiples programadores que han utilizado estos servidores de base de datos, páginas web y servlets.

Capítulo 3

Capítulo 3

Arquitectura de Micro500

En este capítulo se mostrará como se fueron integrando las diferentes partes que se necesitaron para el desarrollo del sistema, se explica el funcionamiento y características de cada uno de estos elementos involucrados, así como los detalles de la integración del servidor web con el contenedor de servlets y a su vez, con el servidor de la base de datos por medio de un controlador JDBC. Se resalta la importancia del modelo MVC y el funcionamiento interno de un contenedor de servlets, para construir las páginas en forma dinámica y ser devueltas al servidor web como páginas estáticas, al final se muestra una arquitectura general del sistema Micro500.

3.1 La plataforma del servidor

En los últimos años los sistemas de información han experimentado un auge extraordinario y las bases de datos han sido parte de este auge. Muchas empresas e institutos han desarrollado aplicaciones cliente-servidor y han puesto a disposición información concerniente a sus bases de datos. Estas aplicaciones han aportado gran cantidad de información ya sea con fines de investigación, ayuda, salud, comercial, entre otros.
Para que una aplicación pueda ejecutarse en modo cliente-servidor, debe existir comunicación entre varios elementos, y tener en cuenta que la comunicación sea la más eficiente. En primer lugar, debemos saber en dónde va a correr nuestro servidor, lo que implica conocer las características más comunes de un equipo de computo, como son:

a) Capacidad de almacenamiento.
b) Velocidad del procesador.
c) Cantidad de memoria RAM.
d) Sistema operativo.
e) Protocolo de comunicación.
f) Dispositivos de comunicación.
g) Cantidad aproximada de usuarios.
h) Tipo de información que se manejará.
i) Seguridad ante ataques informáticos.
j) Ubicación del servidor.
k) Objetivos del servidor.

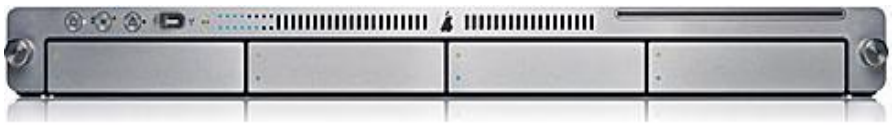
Otros detalles a tomar se cuenta son:

- El tipo de clientes a los que será dirigida la aplicación.
- Como los clientes encontraran fácilmente la información solicitada.
- Como será devuelta la información por el servidor.
- Como los datos serán presentados al cliente.

3.1.1 Características de un XServer

En la Sección de Computación se cuenta con un laboratorio de computadoras Mac Power con diferentes características, de entre el equipo con que se cuenta, se asigno para este proyecto un XServer MacOSX de muy resaltables características, este equipo cuenta con procesador dual a 1.33 GHz Power PC G4, 512 MB SDRAM, high-bandwidth I/O, puertos FireWire 800 ports, y dual Gigabit Ethernet interfaces.

42U Rack



XServer

Figura 3.1 Rack 42U y un Xserver MacOSX

La figura 3.1 nos muestra un XServer y un Rack que sirve para contener 42 unidades Xservers en un espacio reducido, cuando este rack esta completo podemos tener la potencia de 84 procesadores, debido a que los Xserver son duales y se puede alcanzar una capacidad de almacenamiento de 20 Terabytes por cada rack 42U.

Esta servidor incorpora un procesador de muy poderosas características llamado G4, este chip incorpora una nueva tecnología llamada PowerPC G4 con Velocity Engine, capaz de procesar información en enormes bloques de 128 bits, en contraste con los bloques de 32 o 64 bits usados por los procesadores tradicionales.

3.1.2 Sistema Operativo MacOSX

El haber mostrado la infraestructura con la que se cuenta, nos lleva a describir algunas características con las que cuenta este sistema operativo de reciente lanzamiento, y el cual es el sistema operativo nativo del Power Mac Dual G4. El 24 de marzo del 2001, Apple Computers liberó su esperada versión del sistema operativo Mac OS X, donde se combina la solidez, fiabilidad, seguridad, estabilidad, resistencia a fallos, protección de memoria y capacidad de desarrollos a nivel industrial de UNIX, con la facilidad de interacción de los usuarios de un Macintosh.

La empresa Apple resalta muchas de las características de este nuevo sistema operativo, como las numerosas mejoras al kernel nativo de UNIX, y una biblioteca ampliada para aumentar en gran medida el rendimiento, planificación inteligente por hilos basada en preferencias y afinidades, uso de la memoria para evitar dobles paginaciones, asignación de 1 GB de memoria a una tarea determinada para el acceso y manipulación de grandes cantidades de datos. Según la empresa Apple, con las características mencionadas anteriormente, las aplicaciones se pueden ejecutar mucho más rápido, ya que el almacenamiento agresivo en memoria cache evita la escritura en bloques no contiguos. También se permite la multitarea preventiva, multiproceso simétrico y compatibilidad con estándares de redes y seguridad como IPv6, IPSec, SSL y SSH2, por si fuera poco, también cuenta con un FireWare que se puede configurar con las debidas restricciones que deseen implantar para seguridad.
Este sistema operativo cuenta con herramientas avanzadas de gestión remota, con seguridad desde cualquier lugar de la red o a través de Internet. También cuenta con el recurso de memoria protegida, para asegurar que una aplicación no pueda tomar el control de la maquina dejándola inerte o bloqueada.
Mac OS X presenta una implementación optimizada para servidores de Java 2, que muestra lo ultimo en desarrollo para el web, y un complejo conjunto incorporado de servicios de grupo de trabajo e Internet, basados en estándares abiertos. Debido al soporte de estos estándares, se incluye un servidor web Apache, optimizado para ofrecer hospedajes de alto rendimiento para sitios web dinámicos y seguros, publicando el contenido web mucho más rápido que la versión corriente de Apache.

3.1.3 Red de Comunicación

El laboratorio de computadoras Mac cuenta con cuatro diferentes modelos, estos son: G3, G4, G4 dual y dos XServers como se muestra en la figura 3.2. Un XServer es utilizado para brindar servicio a la red local del laboratorio Mac, el otro XServer es utilizado para tener residente a nuestro sistema, y algunos otros desarrollos de diferentes proyectos que se están realizando actualmente en la Sección de Computación.

Seguridad en la red.- La seguridad es un punto esencial para cualquier sistema que se implemente, ya que nos protege de que gente no autorizada tenga acceso a datos restringidos para que estos no puedan ser modificados y puedan provocar un daño a la información contenida. Los XServers cuentan con fireware propio, y permite restringir el acceso a ciertos puertos y direcciones IP especificas. Parte de esta funcionalidad es utilizada para proteger nuestros sistema, y se configuraron los accesos para el servidor de bases de datos, ya que solo se permite el acceso de un usuario autorizado, previendo el caso en que se obtuvieran el login y password correspondiente, se configuro para que solo desde un IP estrictamente autorizado se puedan tener accesos a la base de datos, de esta forma no se permite el acceso externo a la red bajo ninguna circunstancia. Además de que internamente también se restringieron los permisos de creación y modificación de las tablas aun usuario diferente al que usa el contenedor de servlets para comunicarse con el servidor de la base de datos.

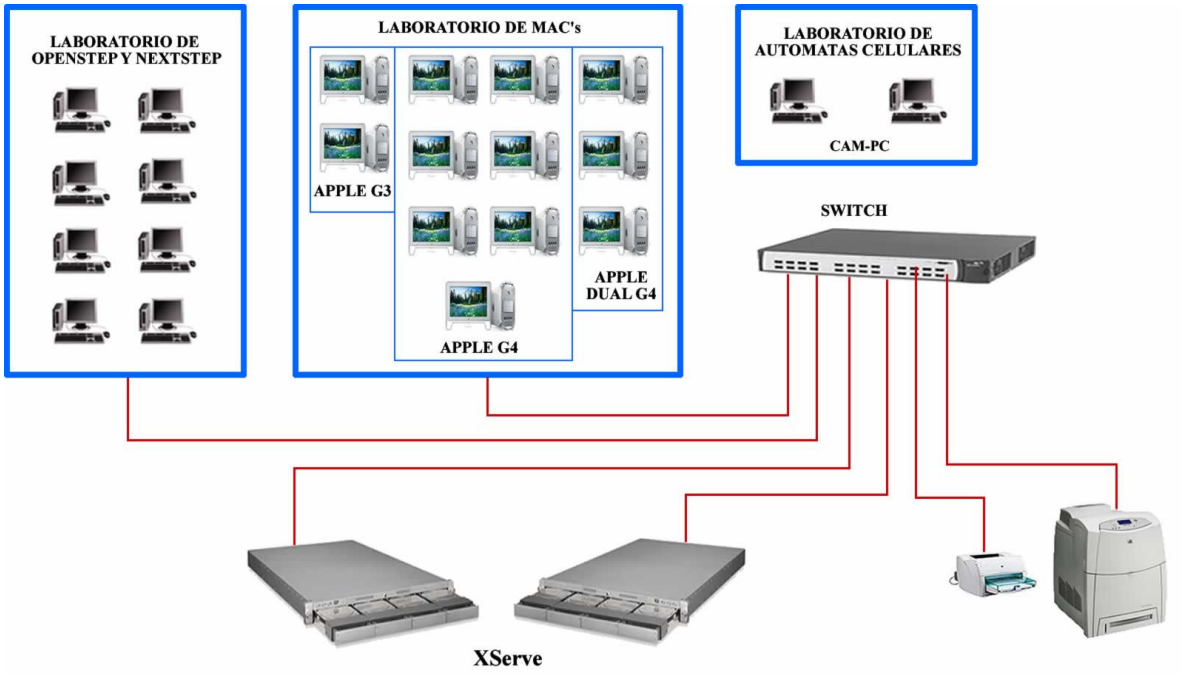


Figura 3.2 Red local y los MacOSX XServers

La figura 3.2 nos muestra un diagrama de la red local desde donde el XServer proporcionara los servicios, se puede observar que se cuenta con un switch que conecta tres redes locales y los XServers son conectados directamente hacia el switch, cada uno de ellos con una velocidad de un Gigabit Ethernet, parte de la velocidad depende en gran medida del trafico que se de en ese momento.

3.2 El Servidor Web

Apache es el servidor web más popular desde Abril de 1996. Las estadísticas presentadas por diferentes empresas, lo señalan que entre el 65% y 58% como servidor, del total de los sitios web que funcionan actualmente en todo el mundo. Esto lo hace el más usado en comparación con el resto de todos los servidores web juntos, haciendo hincapié en que en la actualidad existe una gran variedad de ellos, como IIS de Microsoft, Server Nestcape, como entre los más fuertes competidores de Apache.

El Proyecto Apache HTTP Server es un esfuerzo para desarrollar y mantener un servidor HTTP Open Source para diversos sistemas operativos, tales como UNIX y Windows NT. La finalidad de este proyecto es la de proporcionar un servidor seguro, eficiente y extensible, que proporcione servicios HTTP que se ajusten a los estándares HTTP actuales.

Apache cuenta con muchas características especiales, entre las más resaltables se encuentra que es un software libre, y su uso no requiere de licencias especiales, ni mucho menos de un costo por su adquisición, aunque pudiera pensarse que por no contar con un costo, este pudiera no ser confiable. En diversas pruebas su uso ha comprobado un alto rendimiento,

capacidad, potencia y gran rapidez en sitios web, a veces muy por encima de los servidores web comerciales existentes hoy en día. Otra característica muy importante es su independencia de plataforma, ya que existe Apache tanto para Linux, Windows, Mac OS X, UNIX, entre otros, esto lo hace un software muy versátil. La arquitectura de apache está conformada módulos, los cuales son incorporados dependiendo de las necesidades y capacidades del sitio web, convirtiéndolo en un servidor ideal para muchas aplicaciones web.

3.2.1 Contenedor de Servlets

El contenedor de Servlets es quizás el elemento más importante, aparte de la base de datos, ya que por medio de él se construirán las páginas dinámicas. Este contenedor es la parte intermedia que comunica al servidor web con la base de datos, en términos más generales, este contendor hace posible que se puedan obtener respuestas de la base de datos hacia los usuarios que hacen las peticiones desde Internet.
Pero ¿Cómo funciona este contenedor de servlets? la figura 3.3 explica este proceso.

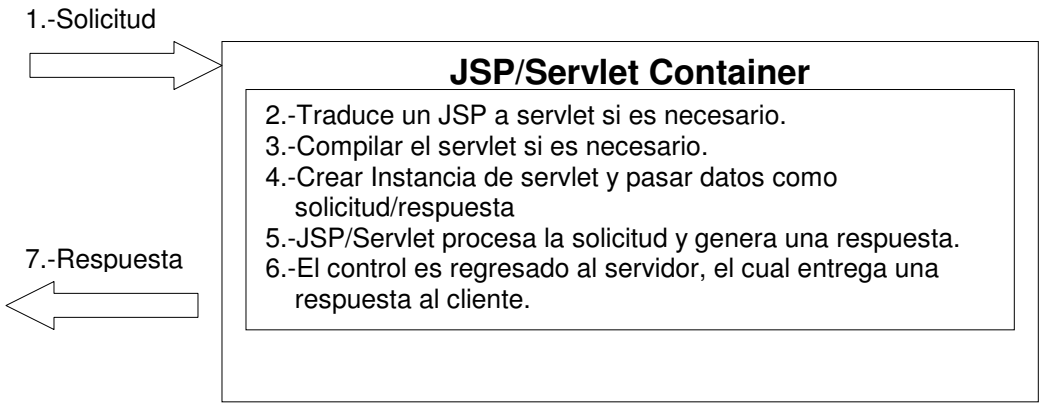


Figura 3.3 Funcionamiento de un Contenedor de Servlets

Podemos observar, en primer término, que se realiza una solicitud, la que es enviada al contenedor, una vez que se encuentra ahí, se busca la página JSP solicitada, si es la primera vez que se hace la solicitud a esta página, entonces esta es traducida a un servlet y compilada en tiempo de ejecución y cargada a memoria, esto sólo ocurre una sola vez, ya que en las solicitudes posteriores ya se encontrará en memoria. Una vez creada una instancia en memoria, se toman los datos y se procesa esta solicitud y se trata de obtener una respuesta. Algunas veces para obtener respuesta, se debe consultar a una base de datos o quizás a otro servidor, una vez que se ha obtenido la respuesta, se construye una nueva página si es que se tienen páginas dinámicas o se muestra una página estática dependiendo del caso y se regresa el control al servidor del contenedor, para presentar los datos al cliente que hizo la solicitud. La gran ventaja que tiene un JSP, es que cuando se realizan cambios ya sea de presentación o sobre la lógica del programa, solo se modifica el archivo que contiene el código y sólo hasta que se haga la solicitud a ese archivo por vez primera, si se tiene una modificación entonces se compilara automáticamente y mostrarán las modificaciones hechas, es de suponerse que la primera vez se pueda tardar un poco más de

lo normal en mostrar la página JSP. Para poder tener un contenedor se tienen dos opciones, la primera de ellas es incorporar los módulos o archivos necesarios al servidor web y configurarlos, la segunda opción es obtener un servidor contenedor de Servlets como JRun, TOMCAT, por mencionar algunos y acceder a ellos por el puerto asignado por default o configurar un nuevo puerto.

3.2.2 Integración del Contenedor de servlets y servidor web

Una vez mostrado el funcionamiento interno de un contenedor de servlets tal y como se mencionó antes, se pueden tener servidores con este tipo de funcionalidad como en el caso de JRun, software propietario o Tomcat software libre, pero ninguno de ellos tiene la robustes, desempeño, capacidad y rapidez del servidor web APACHE, razones por las que deseamos sea nuestro servidor de cabecera para este sistema.
Es indispensable mencionar que se puede tener a los dos servidores web trabajando en la misma computadora sin producir conflictos, como se muestra en la figura 3.4

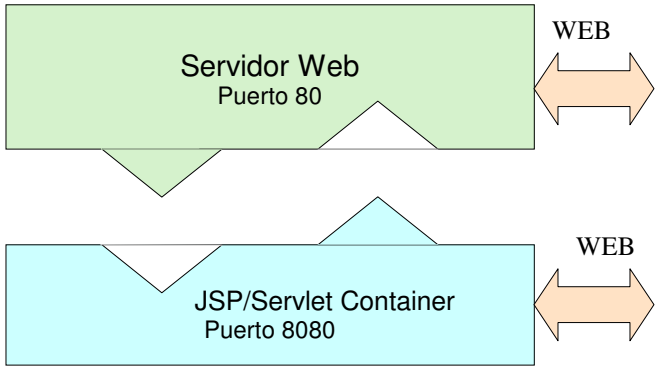


Figura 3.4 Servidores web trabajando de forma independiente

No existe conflicto debido a que los servicios se estan prestando por puertos diferentes, incluso si se quiere, se pueden comunicar entre ellos y hacerse solicitudes de páginas, pero esto podría causar algunas confusiones cuando se quiera dar mantenimiento o se deseen hacer cambios al sistema, aúnque esta puede ser una forma de trabajar con un contenedor de servlets y un servidor web, también se pueden crear problemás de seguridad al tener otro puerto a disposición de servicios web.

Ninguno de los contenedores de servlets cuenta con la solidez de Apache, asi que es deseable en un buen sistema contar con esta solidez, y mucho más si este servidor cuenta con soporte para servlets, ya que nos brindaría todas las ventajas que proporciona java. Esto es posible con la integración del contenedor y el servidor Apache. Para lograrlo se debe configurar internamente un puerto desde el cual Apache pueda comunicarse con el contenedor y obtener las respuestas y mostrarlas como si el proceso lo hubiera hecho Apache; para ello es necesario configurar adecuadamente algunos archivos.

Esto se puede lograr con el contenedor Tomcat, ya que pertenece a un proyecto adyacente de Apache Software Foundation y por lo mismo se ha creado una forma de unir estos dos servidores para unir las caracteristicas de cada uno y funcionar como si fuera uno solo brindando las capacidades de los dos.

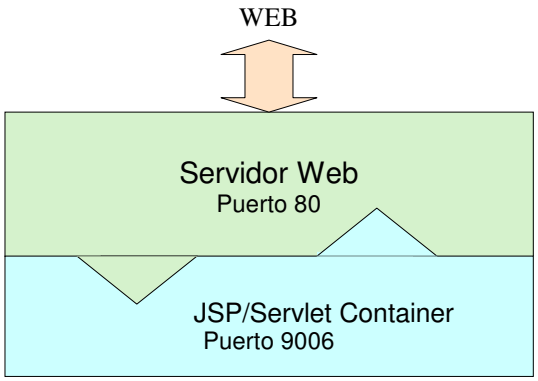


Figura 3.5 Integración de Apache y Tomcat

La configuración de Apache reconoce el puerto 9006 en este caso para tener una comunicación interna y transparente a los usuarios, todas las peticiones se hacen por medio del puerto 80, que es el único que estará disponible desde el web y las respuestas serán nuevamente enviadas por este puerto. Apache envía una solicitud y obtiene la respuesta por medio del puerto 9006. Para que esto funcione de forma adecuada se deberán de respetar las rutas que proporciona el contenedor para almacenar los archivos JSP o HTML.

También se pueden tener archivos HTML en la ruta de Apache y al solicitar información reenviarla al contenedor de servlets y obtener la respuesta. Una vez hecha la integración obtenemos mucha potencia de parte de nuestro servidor y podemos hacer uso de todo el lenguaje de programación java, y se puede hacer uso de clases con java puro por medio de los java beans, ya que desde un archivo JSP se pueden hacer llamados a clases java o servlets.

La figura 3.6 nos muestra el alcance que tiene esta integración, podemos ver que incluso se pueden hacer extensiones al propio servidor y ser utilizadas por el contenedor según sea el caso. A su vez se tiene una máquina virtual de java en el servidor y se pueden hacer extensiones y nos facilita hasta la reutilización de código, ya que se pueden crear clases con funcionalidades específicas y ser utilizadas por cualquier archivo JSP o clase java.

También se pueden crear servlets con este mismo concepto, todo esto gracias al concepto de programación orientado a objetos de java.

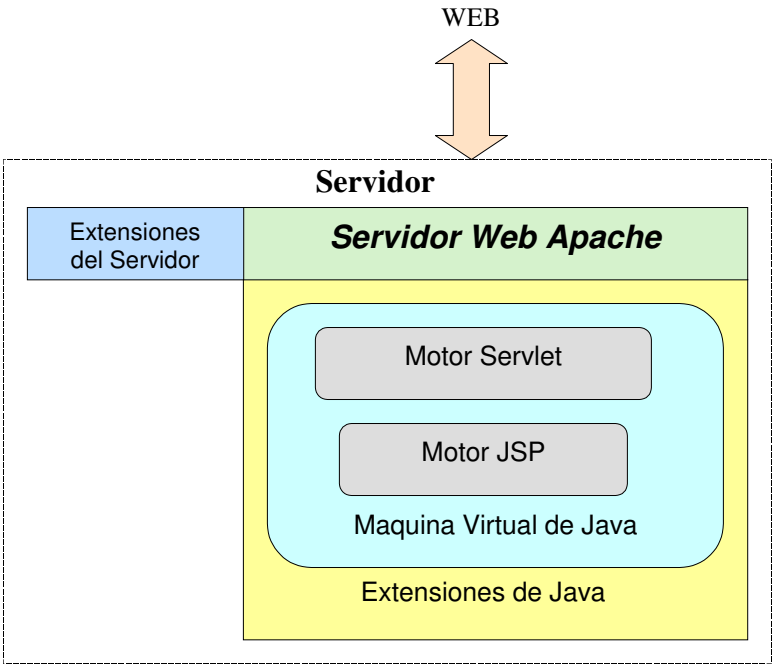


Figura 3.6 Extensiones de servidor web Apache

3.2.3 Integración del contenedor y el servidor de base de datos

Teniendo la integración del contenedor con el servidor apache, se debe buscar la forma en que el contenedor se comunique con la base de datos para poder obtener los datos deseados. La comunicación se debe realizar por medio de un drive específico dependiendo de la base de datos a la que se quiera conectar y también de qué contenedor estamos utilizando. Algunos contenedores comerciales ya cuentan con diferentes controladores y con tags específicos para realizar la conexión sin mucho problema. En este caso se está utilizando el contenedor Tomcat, el cual sólo utiliza código java para generar la comunicación con los diferentes elementos involucrados, lo que representa una gran ventaja, ya que java ofrece soporte de diferentes controladores y mecanismos para conectarse a diferentes sistemas de bases de datos.

Un DataBase Management System (DBMS) contiene una interfaz de comunicación diferente. Para solucionar este tipo de problemas la empresa Sun Microsystem desarrolló una Application Programming Interface (API) llamada Java DataBase Connectivity (JDBC), y permite que desde cualquier programa que hayamos desarrollado en java se pueda acceder a cualquier DBMS en forma transparente.
La figura 3.7 muestra cómo JDBC está construido en varias capas; la primera de ellas es la aplicación java que programamos y emplea la API JDBC para comunicarse con el administrador JDBC. Una vez logrado esto, el administrador transfiere la información entre la aplicación y un controlador JDBC, aquí también se muestra como existen diferentes controladores para cada DBMS distinto.

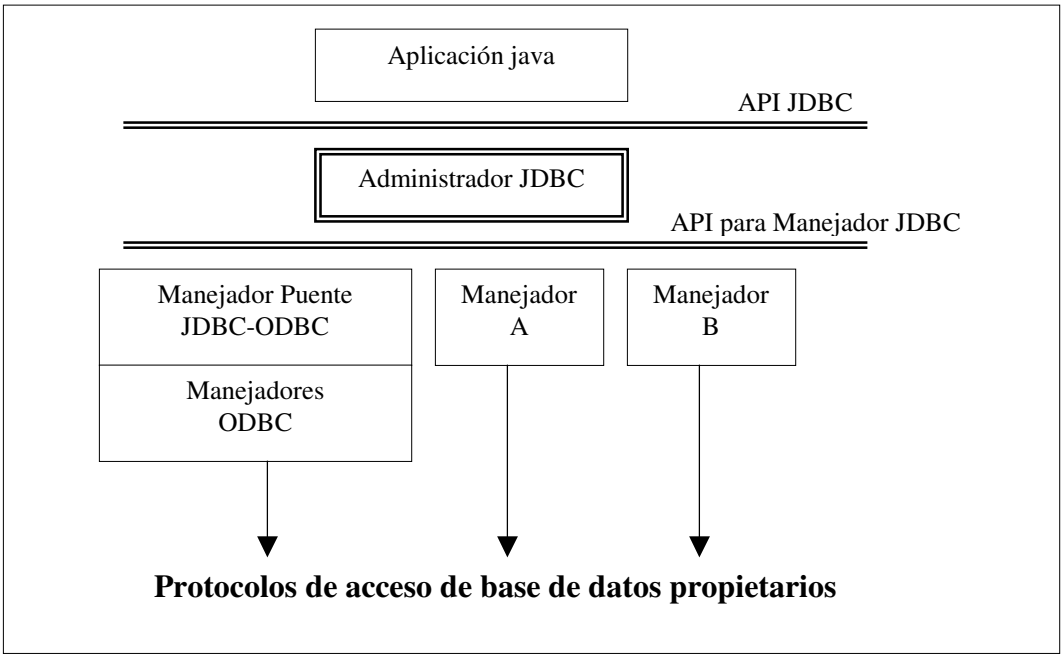


Figura 3.7 Capas del Protocolo JDBC

Una vez que se tiene el controlador específico de la base de datos a la cual se quiere acceder, el controlador convierte las instrucciones SQL que han sido enviadas desde la aplicación java al protocolo propietario de acceso al DBMS.

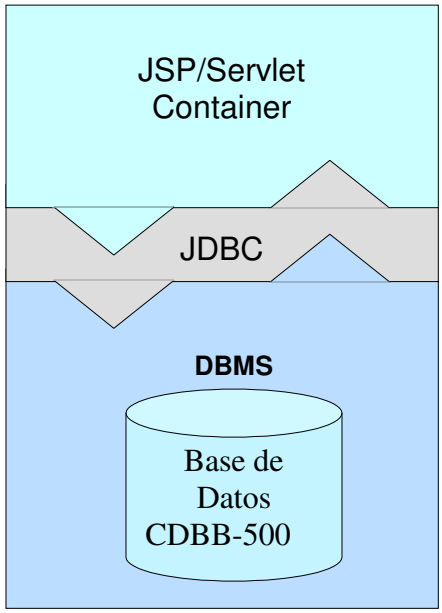


Figura 3.8 Integración del Contenedor con el DBMS

De este modo tenemos que la API, es consistente sin importar cuál DBMS y cuáles controladores JDBC se estén utilizando, para obtener una conectividad de java con el DBMS. La API y el administrador JDBC nos proporcionan una alta independencia entre la aplicación y la instrumentación del DBMS, lo cual es muy práctico porque podemos conectarnos a diferentes tipos de bases de datos con los protocolos existentes, o solicitando los controladores del DBMS al que deseamos acceder. Las compañías de software al crear nuevos mecanismos de conectividad, registran los nombres de sus protocolos de conectividad con JavaSoft, para asignar un nombre único al software y evitar la duplicidad de éstos y se pone a disposición de la comunidad Java para ser utilizado en futuras aplicaciones.

JDBC se puede comunicar con un DBMS de forma remota si los permisos del DBMS lo permiten. Esto significa que podemos establecer no sólo comunicación de manera local, si no tener el DBMS en la misma máquina que se tiene la aplicación. También se puede establecer una comunicación, aún cuando el DBMS se encuentre en un lugar diferente de donde tenemos nuestra aplicación; para ello debemos especificar el URL o la dirección donde se encuentra, y el puerto por medio del cual nos podemos comunicar para establecer comunicación. De esta forma se puede tener un servidor para la aplicación y otro para el DBMS. En este proyecto utilizamos el controlador de PostgreSQL para conectar nuestra base de datos desde el contenedor, todo por medio de código java nativo.

3.3 Integración de la arquitectura por capas

Existen diferentes tipos de arquitecturas utilizadas en la creación de aplicaciones con páginas JSP, todas ellas involucran diferentes capas para facilitar la escalabilidad y el mantenimiento. Para ello se utilizó el Modelo-Vista-Controlador (MVC) y la integración con diferentes capas de software servidor, a continuación mostramos la utilidad de cada uno de los elementos del modelo y la arquitectura general obtenida.

3.3.1 Modelo MVC

En este modelo existen diferentes componentes interrelacionados, que intervienen en la generación de una respuesta del servidor. El modelo MVC (Modelo-Vista-Controlador) consta de tres partes y cada una de ellas es explicada a continuación:

Controlador.- Este elemento es el que reacciona a las acciones del usuario y el encargado de crear y asignar valores al modelo para su funcionamiento. La figura 3.9 nos muestra cómo una petición del usuario es capturada por el controlador y establece la forma en que estos datos serán asignados para proveer una respuesta favorable.

*Modelo.-*El modelo proporciona el control de la funcionalidad de la aplicación y encapsula el estado de la aplicación sin saber nada de los otros dos elementos vista –controlador. Una vez que el controlador ha asignado los valores el modelo, puede formatear estos valores para hacer una solicitud a la base de datos en forma adecuada. Cuando se obtienen estos datos como respuesta, el mismo modelo puede formatear nuevamente los datos para que

puedan ser mostrados al usuario en forma conveniente o solicitada y son enviados al elemento vista para su presentación hacia el exterior.

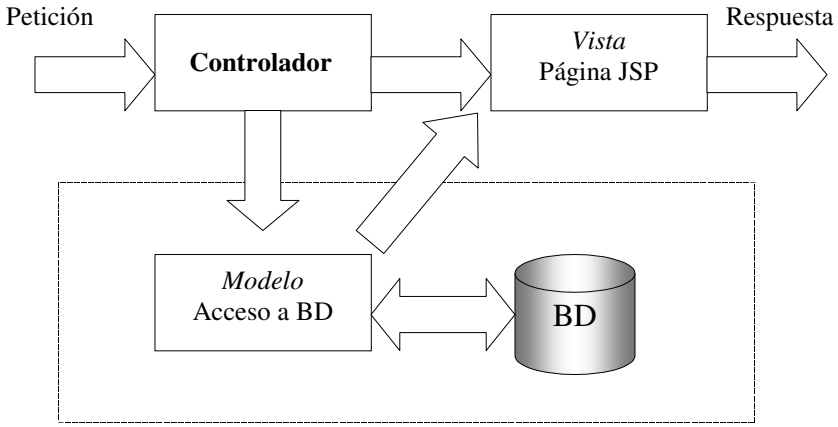


Figura 3.9 Modelo MVC

Vista.- La vista proporciona la presentación del modelo, presentando a su vez la apariencia de la aplicación. La vista puede acceder a métodos get y obtener la información del modelo, esto es factible cuando se utilizan los JavaBeans en el modelo. También la vista es notificada cuando se producen cambios en el modelo y al siguiente llamado se mostraran los cambios que se han realizado. De esta forma se puede controlar la transferencia de información entre páginas JSP y todas las acciones realizadas en cada una de ellas. Muchas de estas características pueden permanecer bajo el control del programador java sin que el diseñador de la página web intervenga en lo absoluto, ya que se separa el diseño y presentación de la lógica de la aplicación. Nuestro propósito es contar con la posibilidad de incorporar nuevos recursos en forma paulatina y también nuevos proyectos con distintas funcionalidades. Por ello el recurso de optar por el modelo MVC y hacer una separación del código de la aplicación y la presentación, la forma en que realizamos este proceso es por medio de la directiva include que permite incluir archivos externos en la página JSP.
Esta circunstancia incrementa el tiempo de procesado de la página pero a cambio nos brinda una forma muy flexible de modificar el texto en todas las páginas a la vez, sin tener que recompilar cada una de ellas. También nos permite incluir elementos estáticos HTML, y recursos como otras páginas JSP, teniendo la posibilidad de incluir varios niveles de anidamiento creando varias capas con niveles distintos dentro del servidor con una separación bien definida. Mucho de esto es utilizado en nuestro sistema y debido a diferentes niveles de anidamiento nosotros podemos tener embebido las capas de Modelo y Vista como se muestra en la figura 3.10 Así nuestro proceso, aunque es en tres pasos utilizando el modelo MVC, incorporando la reutilización de código y el anidamiento de niveles se muestra como si fueran solamente dos capas siendo tres en realidad, y generando una página dinámica en tiempo de ejecución, fijando los valores con el controlador y generando una respuesta y la forma de presentación en una sola vez, creando valores dinámicos que pueden ser accesados posteriormente, generando otras páginas dinámicas a su vez.

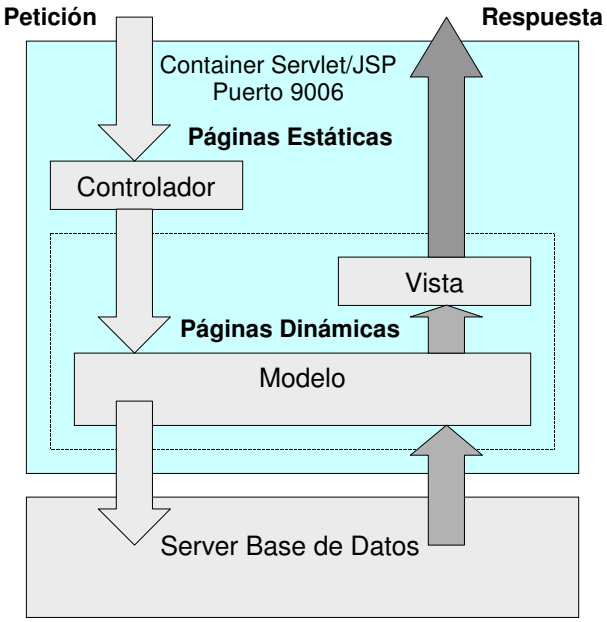


Figura 3.10 Modelo MVC utilizado

3.3.2 Integración de la arquitectura general de Micro500

Una vez que se ha analizado cada una de las partes que se necesitan para que nuestro sistema funcione de la mejor forma, desde la petición que se realiza en Internet, hasta la comunicación con la base de datos; ahora mostramos una arquitectura general donde se suman todos los elementos que se han visto en forma separada. La figura 3.11 muestra la arquitectura adoptada por Micro500 de la cual se puede observar que tenemos una arquitectura que consta de varias capas que interactúan una con la otra, y existen capas intermedias por medio de las cuales las capas de los extremos se puedan comunicar. En el diagrama mostrado se pueden observar triángulos que están hacia abajo y otros hacia arriba, unidos a una determinada capa. Con esto se quiere mostrar el flujo de los datos. Los triángulos que miran hacia abajo muestran un concepto de entrada de datos, se puede ver en cada capa desde que la solicitud viene de Internet y es tomada por el servidor web Apache, procesada y enviada a nuestro contenedor de servlets. Después el contenedor se comunica con el drive JDBC de java y este a su vez establece una comunicación con el DBMS en este caso PostgreSQL. Después se obtiene una respuesta y se tiene todo este proceso a la inversa. Con este tipo de diagrama de la arquitectura queremos resaltar cómo es que estamos logrando una unión entre diferentes elementos involucrados en el sistema, y a su vez, también sirve para señalar como es que todos estos elementos son independientes unos de otros y que en determinado momento si alguno de ellos no satisface las expectativas por las cuales fue elegido, se tiene la facilidad de sustraerlo y sustituirlo por otro nuevo. Suponiendo que en determinado momento nuestro contenedor de servlets no cumple las expectativas, podemos prescindir de éste y colocar otro que hayamos elegido, y mantener nuestro código inalterable y sólo cambiar al elemento contenedor de Servlets.

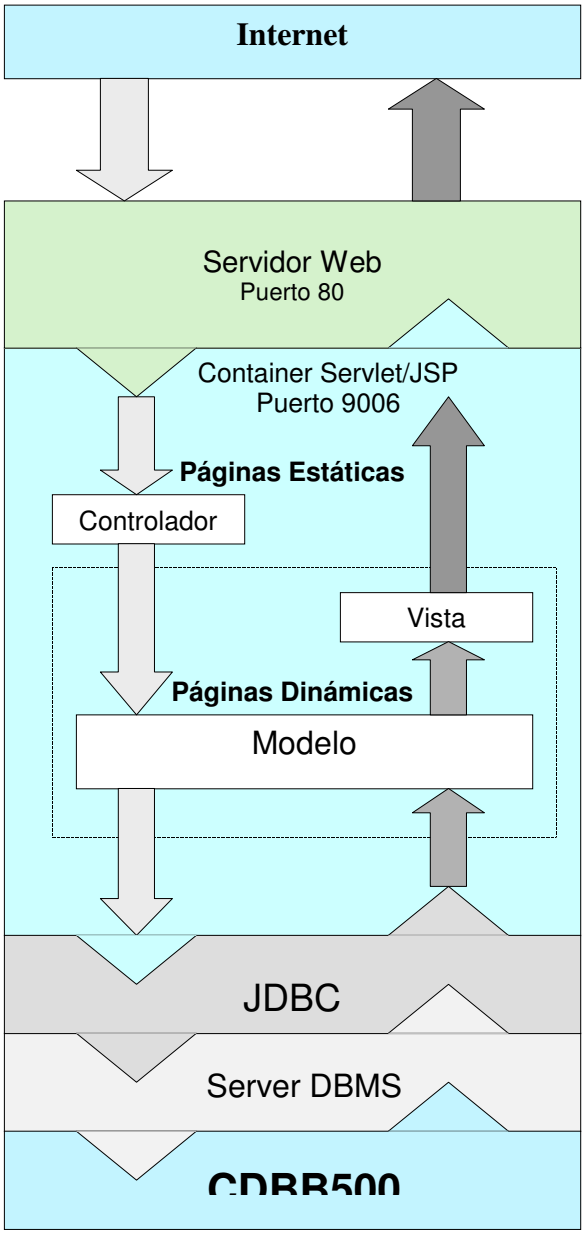


Figura 3.11 Arquitectura general de Micro500

3.3.3 Comentarios Finales

La arquitectura adoptada por Micro500, nos permite tener elementos independientes que en determinado momento se pueden sustituir por otros, proporcionando desde este punto de vista una arquitectura escalable, ya que por las características de cada uno de los elementos es independiente de cualquier plataforma, y cada elemento está disponible para diferentes sistemas operativos. El uso del modelo MVC, facilita la programación y el desempeño del servidor web. La integración de los distintos elementos involucrados nos permitieron construir una arquitectura sofisticada con gran rendimiento y de alta calidad.

Capítulo 4

Capítulo 4

Reingeniería del sistema CDBB500

En esta parte se muestra el concepto modular utilizado para la implementación de todas las funcionalidades del sistema. Primeramente se muestra la complejidad de la base de datos CDBB500, y con ello la dificultad que existe al tratar de codificar y controlar el sistema. Debido a esto, se hizo la partición la base de datos en módulos que son integrados por entidades relacionadas, y de esta forma la base de datos se va complementando de manera paulatina, conforme el sistema lo requiere. El concepto se extiende para modelar la lógica de programación, para que las capacidades del sistema se vayan extendiendo a partir de la incorporación de servicios independientes.

4.1 Diseño del modelo de datos

En este punto se muestra el estudio y las modificaciones que se hicieron a la base de datos CDBB500. Como sabemos una base de datos es un conjunto, colección o depósito de datos almacenados en un soporte informático no volátil. Los datos están interrelacionados y estructurados de acuerdo a un modelo que es construido para obtener el máximo contenido semántico, y la redundancia en ellos debe de ser controlada, de forma que no existan duplicidades innecesarias o perjudiciales. El modelo de la base de datos nos permite suprimir todas las redundancias lógicas, que pudieran afectar a nuestra base de datos. Se puede permitir cierta redundancia física, que a veces puede ser necesaria para responder a ciertos objetivos de eficiencia y ser tratadas por el sistema. Se denomina como redundancia controlada por el sistema, cuando un dato se actualiza lógicamente por el usuario en forma única, y el sistema cambia físicamente todos aquellos campos en los que el dato esta repetido, en caso de existir redundancia física.

4.1.1 Independencia lógica y física entre datos

Un aspecto muy importante de una base de datos, es su independencia tanto física como lógica entre datos y tratamientos. Esto significa la separación entre el almacenamiento y la organización lógica de los datos, se contemplan a través de los programas de aplicación que hacen uso de la base de datos, consiguiendo una doble finalidad. La primera de ellas es que los datos se presentarán en distintas formas según las necesidades de los usuarios. La segunda es que el almacenamiento de los datos, su estructura lógica y los programas de aplicación, serán independientes unos de otros, de modo que un cambio en alguno de ellos no obligue a modificar los demás. Así un cambio en el tratamiento de los datos, la inclusión de nueva información, desaparición de otra, cambios en la estructura física, cambios en los caminos de acceso, no deberán alterar los programas, ni llevar a cabo un rediseño de la base de datos. Esto supone una gran ventaja al evitar la reprogramación de una aplicación cuando se producen cambios en los datos. Conseguir una mutua independencia entre datos y programas, es un objetivo esencial de las bases de datos. Esto nos brinda la capacidad de

conseguir, sin excesivo costo, la continua adaptación del sistema de información a la evolución de la organización.

4.1.2 Modelo conceptual y relacional de CDBB500

Una vez que se han resaltado las características de una base de datos, se procederá a mostrar el esquema conceptual bajo el cual fue diseñada la base de datos. Como hemos resaltado en el primer capítulo, este diseño fue construido junto con los primeros sistemas de información, pero se mantiene vigente; debido a su buen diseño e independencia lógica alcanzada por el modelo, por lo cual no ha tenido que ser modificado. Para la construcción de este modelo primero se colectó la información, después se hizo un análisis y se propuso un primer modelo lógico de la base de datos, el cual se sometió a revisión, obteniendo una serie de depuraciones y estructurándose hasta poder obtener el diseño final.

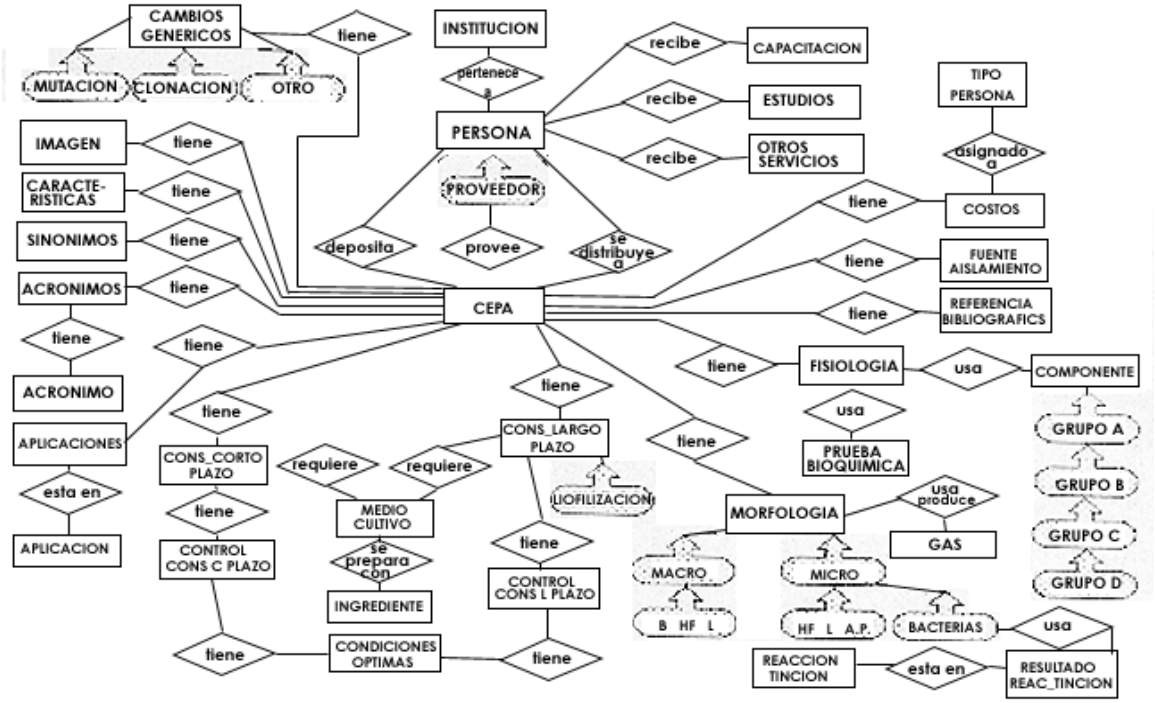


Figura 4.1 Esquema Conceptual de CDBB500

En la figura 4.1 podemos observar el modelo relacional de la base de datos CDBB500, en donde se utiliza un diagrama Entidad-Vínculo-Extendido, en el que las entidades de las diferentes vistas fueron expresadas en términos del modelo relacional. Para ello se aplicaron las primeras tres formas normales eliminando la redundancia y facilitando la integridad de los datos. Como modelo base se utilizaran los diagramas propuestos por Chen[5] sobre el modelo entidad-relación. De esta forma los rectángulos nos representan las entidades y los rombos los vínculos entre las entidades. Observando el diagrama podemos apreciar que se tiene una entidad central llamada Cepa, la cual sirve de base para las otras entidades y este concepto se utilizará más tarde en la implementación del sistema. También se obtuvo un modelo orientado a objetos de la base de datos CDBB500, este

modelo objeto relacional se define como una extensión objetual del modelo relacional, permitiendo la definición de nuevos tipos y de relaciones de herencia, entre otras cosas.

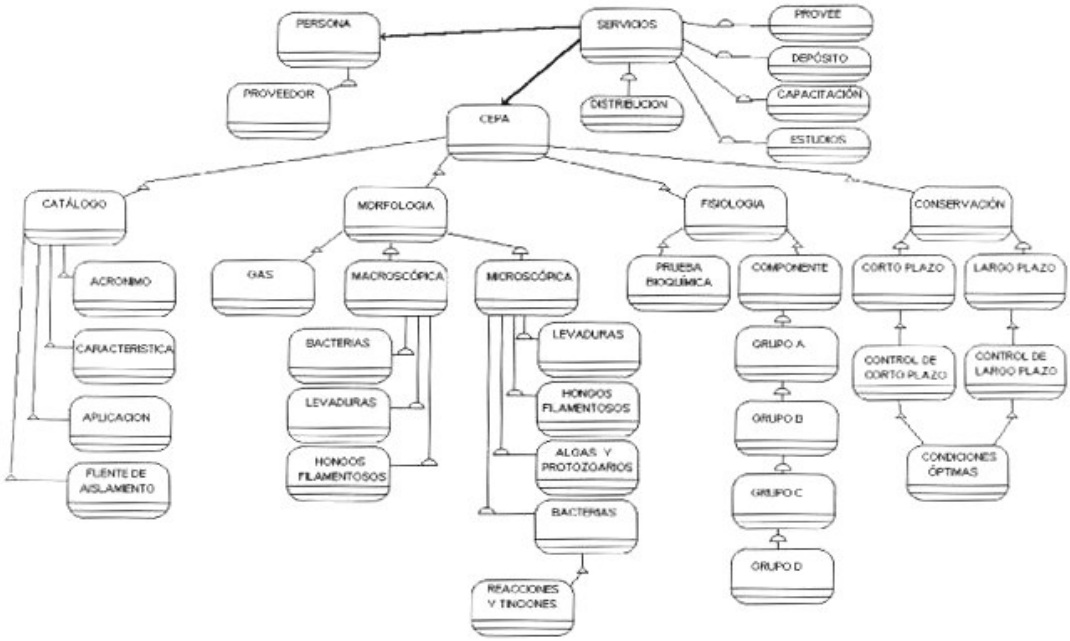


Figura 4.2Esquema Orientado a Objetos de CDBB500

Existen dos forma de aplicar la orientación a objetos a un sistema de bases de datos. La primera es por medio del esquema de la base de datos, que podemos modelar totalmente orientada a objetos o como un modelo objeto-relacional; esto significa tener parte en forma relacional y parte orientada a objetos.

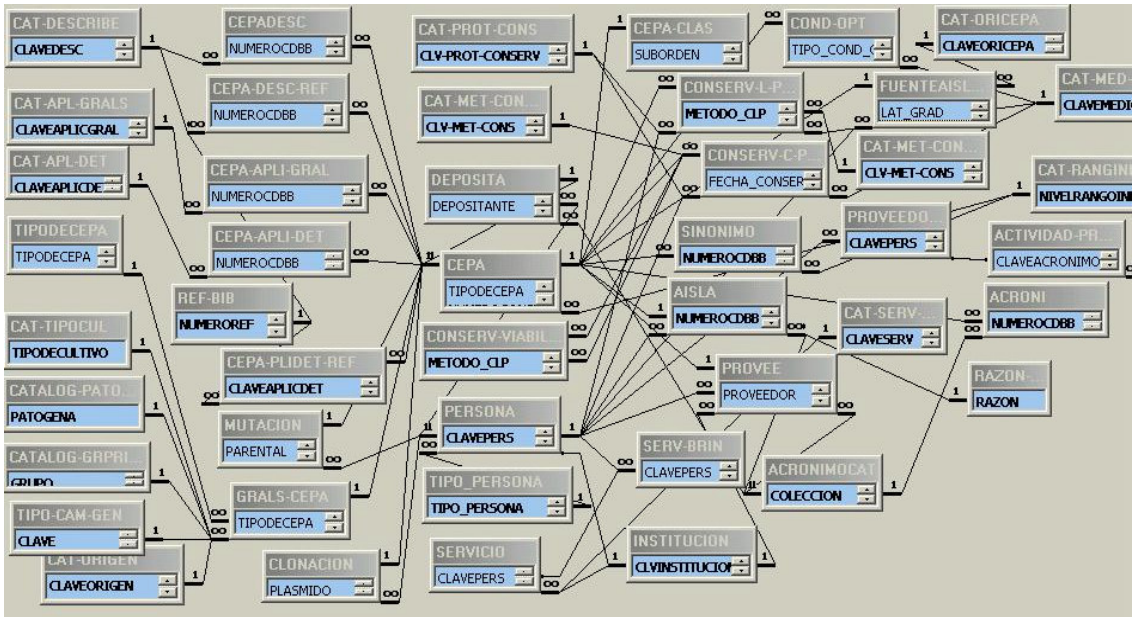


Figura 4.3 Relaciones de CDBB500 en Access

La segunda forma es tener el esquema de la base de datos, sea cual sea, y trabajar la implementación del sistema con un lenguaje orientado a objetos que son los mas usados actualmente.
Una vez que mostramos los esquemas construidos para la base de datos CDBB500, poco después en 1996, se implementó esta base de datos en el manejador de bases de datos ACCESS de Microsoft por petición de CONACYT. Para ello se construyó todo el esquema de la base de datos, para cada entidad se establecieron llaves primarias y llaves foráneas, así como todo un esquema de relaciones, el cual se muestra en la figura 4.3, en el que se puede observar parte de las relaciones existentes, ya que la figura sólo muestra una parte de la base de datos. También se muestran algunas tablas, aunque sólo en esta figura cada una muestra un atributo, pero cada una de ellas cuenta con diferente número de atributos.

4.2 Reingeniería del modelo de datos

De las relaciones entre las tablas, se puede ver que la base de datos contiene una cierta complejidad lo que dificulta en gran parte la implementación del sistema y la forma en que será mostrada la información, esto requiere un amplio desarrollo y un cuidado minucioso en el tratamiento de los datos, para que éstos puedan ser procesados adecuadamente por el sistema. La implementación del nuevo sistema requirió un cambio de DBMS de Access a PostgreSQL, debido a ello fue necesario hacer un estudio sobre la conveniencia de realizar algunos cambio a la base de datos, para tratar de optimizar su acceso y facilitar la búsqueda de la información vía web. En los puntos siguientes se muestra parte de cómo fue realizado este proceso.

4.2.1 Modificación a la base de datos

La modificación a la base de datos se realizo en varias etapas que son:

1.- Estudio y análisis del modelo conceptual de la base de datos

En este punto se hace un análisis detallado del modelo así como bajo qué concepto fue construido, la utilidad que representa cada una de las partes del modelo en el sistema y de que forma opera cada una de las partes involucradas y su importancia. Este análisis también incluye verificar el esquema de tablas construido, las relaciones existentes entre diversas entidades y qué tipo de propiedades contienen cada una de ellas. El esquema de tablas fue visualizado por medio de Access y se buscaron puntos de refinamiento en base a lo ya construido y usado hasta el momento.

2.- Estudio de nuevas propuestas

Una vez obtenido un análisis del esquema conceptual y del esquema de tablas construido al momento, se llegó a la conclusión que se podían hacer pequeñas modificaciones a la base de datos a nivel de tablas, como el cambio de formato de algunos atributos para facilitar el procesamiento de los mismos, la eliminación de un atributo en la tabla de cepa. También se propuso suprimir dos tablas llamadas Cepa_Desc y Cepa_Plidet_Ref y parte de sus

atributos fueron adicionados a otras tablas para optimizar parte de las búsquedas. Parte de esta propuesta también consideraba el cambio de los nombres de campo de algunas tablas, y su estandarización, para mayor facilidad de manejo, tratamiento y localización de la información de la base de datos CDBB500.

3.- Análisis de conveniencia y utilidad de las nuevas propuestas y comportamiento de los nuevos cambios.

Una vez que se obtuvieron algunas propuestas de refinamiento de la base de datos, se analizaron los beneficios o inconvenientes que se podrían obtener al realizar dichos cambios, y se determino que se obtenían obtener múltiples beneficios en el tratamiento de los datos, y que dichos cambios no modificaban el esquema conceptual con que fue construida, ya que sólo se refinaban aspectos a nivel de atributos de las tablas.

4.- Hacer efectivos los cambios propuestos.

Al ver que no se alteraba el comportamiento de la base de datos con los nuevos cambios, se procedió a su realización. Parte de estos cambios involucraron suprimir las dos tablas antes mencionadas y cambiar algunos nombres de atributos.

CEPA

númerocdbb	**tipodecepa**	**númeroconsecutivo**	**depositante**	**genero**
A-000007	A	7	2	Anabaena
B-000019	B	19	15	Nocardiodes

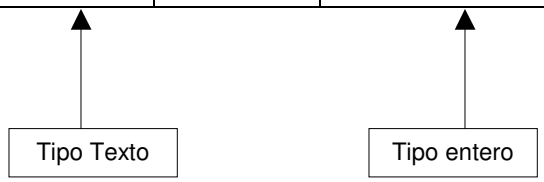


Figura 4.4 Tabla de cepa original

La figura 4.4 muestra parte de la tabla CEPA, que como se mencionó en puntos anteriores, es la entidad principal de nuestra base de datos que se muestra la implementación original. Se puede observar que el campo númerocdbb cuenta con un formato especial que fue definido en algún momento sobre cierto concepto de no sobrepasar cierto rango de numeración.

En la figura 4.5 se muestran los cambios realizados, entre los más resaltables, podemos citar primeramente que el nombre del campo es modificado de númerocdbb a sólo cdbb y esto se estandarizó para todas las demás tablas existentes. Otro cambio resaltable es que se suprime el campo númeroconsecutivo, y este es contenido en el campo cdbb, que cambia de ser un atributo de tipo texto a ser de tipo entero, conteniendo un número de cdbb, que es el mismo consecutivo en la base de datos.

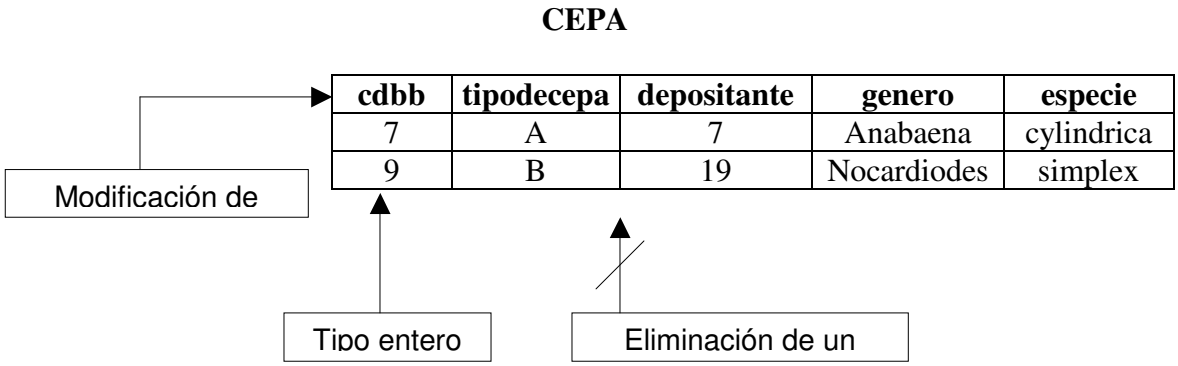


Figura 4.5 Modificaciones a tabla cepa

Como el campo cdbb no puede ser repetido en ningún momento, ni siquiera vacío y es único, en la tabla es considerado un campo llave primaria y es utilizado para encontrar datos en otras tablas, así este campo tiene doble finalidad, contener el número de registro en la base de datos CDBB500 y el lugar consecutivo ocupado. Otro beneficio mas inmediato es la facilidad de búsqueda y mantenimiento, ya que es más fácil recordar un número que toda una cadena de caracteres con un formato especial, así como cuantos ceros se tienen a la izquierda como en el formato anterior.
Otros cambios realizados a la base de datos, fueron la estandarización de los nombres de campos, dependiendo de la tabla en que se encontraban y la función que desempeñaban. Aunque la mayor parte de ellos estaban nombrados de esta forma, algunos pocos no conservaban en forma estricta esta característica, por lo que se dio a la tarea de obtener un sólo formato para todas las tablas y así evitar confusiones en la solicitud de datos. Una vez realizados todos los cambios propuestos, el siguiente problema era la migración de los datos de Access a PostgreSQL, este proceso se muestra en el siguiente punto.

4.2.2 Migración de la base de datos

Una vez que se refinó la base de datos existente, se procedió a su migración, y aunque pudiera pensarse que este es un proceso sencillo, no es así debido a múltiples problemas que pudieran parecer insignificantes, pero que en realidad son aspectos muy importantes a tomar en cuenta.

Esta migración se desarrolló en varias etapas mismas que a continuación se describen:

1.- Instalación y configuración de PostgreSQL

La instalación de PostgreSQL no mostró mucho problema, la configuración debe hacerse con más atención y seguir ciertas instrucciones para un adecuado desempeño de este manejador de bases de datos, ya que se definen las máquinas que podrán tener acceso a PostgeSQL en forma remota. En nuestro caso solamente el servidor de la misma máquina tendrá acceso a la base de datos, de esta manera se trata de mantener cierto nivel de seguridad desde la base de datos misma. Un rasgo muy importante es que se configuró

PostgreSQL para aceptar el tipo de información en español, ya que diversas pruebas mostraron problemas con los acentos y otros caracteres especiales propios de nuestro idioma. Esta configuración se realiza en el momento de crear una nueva base de datos, y se fija el tipo de caracteres que se utilizaran para el procesamiento de la información. Una vez realizadas varias pruebas se considero listo el DBMS para migrar la base de datos.

2.- Obtención y construcción de la estructura conceptual.

Se obtuvo toda la estructura de la base de datos y los metadatos de CDBB500. Se verificaron los tipos de datos utilizados en Access, y se realizó una adecuación de tipos hacia el nuevo DBMS. Ya que cada gestor de base de datos cuenta con tipos propios y existe un problema de compatibilidad entre muchos de ellos, se adecuan los datos a los tipos existentes en el nuevo gestor. En este proceso se obtuvieron varios cambios pero sin afectar la base de datos. Una vez realizado esto se creo toda la estructura de CDBB500, con los nuevos tipos en PostgreSQL, dejándolo solo a la espera de los nuevos datos, como se muestra en el siguiente ejemplo:

```
Table "cepa"
      Column       |     Type     | Modifiers
-------------------+--------------+-----------
 cdbb              | integer      | not null Unique
 tipocepa          | character(1) | not null
 depositante       | integer      | not null
 genero            | text         | not null
 especie           | text         | not null
 nivel_rangoinfra  | text         | not null
 nombre_rangoinfra | text         | not null
 baja              | character(1) | not null
 motivobaja        | text         |
Unique keys: cepa_cdbb_key
```

3.- Exportación de tablas

La forma en que se migraron los datos fue por medio de una aplicación de exportación de Access la cual, contiene múltiples opciones de obtener los datos y vaciarlos en archivos con diferente formato. Se requirió también de un formato especial de los datos. En nuestro caso, se ocupó un archivo de texto con separación tabular y sin caracteres especiales de contención de datos. Cuando se obtuvieron estos archivos entraron a un proceso de verificación de archivos y parte de ellos tuvieron que ser reformateados con otro editor de texto, ya que se incluían formatos exclusivos de Access, que no son compatibles con ningún otro gestor de datos, como la inclusión de decimales en datos de tipo entero o un formato de fecha y hora en los de tipo date.

Una vez que estos datos eran reformateados se obtenían otros archivos de texto, pero se debía tener en cuenta también, que se debían trasladar a un sistema operativo diferente, en este caso a MacOSX, esto nos provoca la inclusión de caracteres ocultos en la información, que fue tratada más adelante.

4.- Alcanzar la compatibilidad

Con el cambio de sistema operativo se detectaron problemas con caracteres ocultos y acentos, para ello lo que se tuvo que aplicar un proceso a los archivos y obtener los datos correspondientes al sistema MacOSX. También se configuró el sistema operativo para aceptar caracteres con acento, ya que esta preconfigurado para el idioma ingles. Lograda la compatibilidad de los archivos y el sistema operativo, entonces se procedió al vaciado de datos en PostgreSQL.

5.- Vaciado de los Datos

Para que los datos pudieran ser vaciados en el DBMS, se utilizaron Scripts propios de PostgreSQL. Este mecanismo utiliza una validación de datos y no realiza la inserción de la información, si algún dato no contiene el formato especificado. Una vez realizado el vaciado de los datos, se realizó una verificación de la total integridad de todos los datos insertados en las tablas. Con el termino de todas estas etapas se dio por concluida la migración de la base de datos a PostgreSQL.

4.2.3 Diseño de la base de datos por módulos

La base de datos CDBB500 contiene una estructura de relaciones un poco compleja y difícil de seguir, cuando se desea programar una aplicación. Esto nos lleva a requerir más tiempo de análisis, y mayor dificultad en el seguimiento de rastreo de errores en el desarrollo. Además de que se desea una forma más abierta de incorporar nueva información a la base de datos. Para resolver parte de este problema se utilizo del concepto de módulos, concepto que aplicado a nuestra base de datos, nos ayuda en gran medida a tener una visión por demás clara de nuestra base de datos.

La complejidad de la base de datos se maneja con este concepto para ir creando pequeños módulos independientes, con una funcionalidad especifica. A manera de incorporarlos paulatinamente, y así brindar nuevas funcionalidades con las cuales no se contaba en un inicio. De esta manera, se separaron las entidades del esquema conceptual en módulos con funcionalidades específicas de información de la base de datos CDBB500. Así en esta primera etapa se incorporaron estos cuatro módulos que se nombraron como: Características y Aplicaciones, Características Avanzadas, Investigadores e Instituciones, Medios de cultivo y Condiciones Optimas. Quedando lista la próxima incorporación de un módulo de Imágenes de las cepas.
Se puede observar en la figura 4.6a, el módulo de Investigadores e Instituciones, para lo cual se tomaron las entidades respectivas y las separamos como un módulo independiente que puede ser agregado en algún momento. Hay que recordar que las entidades Investigadores e Instituciones, cuentan con diversas tablas y en el caso de otras entidades se hace uso de catálogos, por ejemplo el módulo de características y aplicaciones. Esto nos sirve para visualizar de una forma más sencilla la complejidad de nuestra base de datos, y separar la relación entre diferentes tablas con un propósito común.

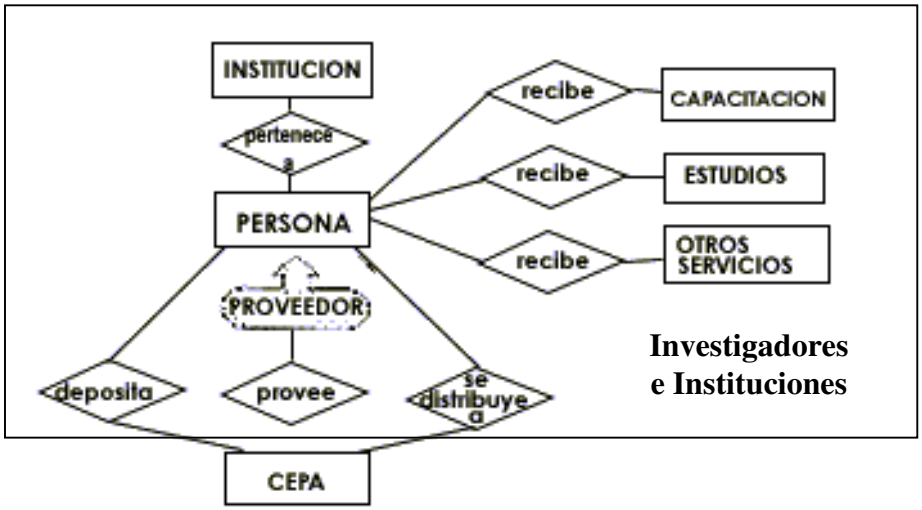


Figura 4.6 a

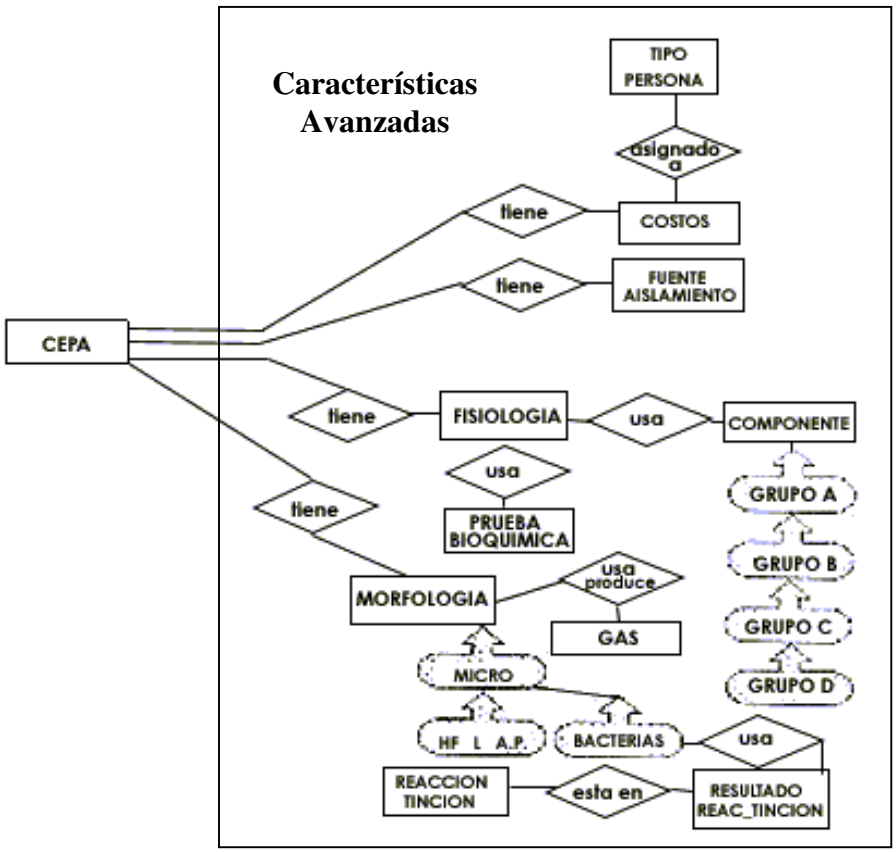


Figura 4.6 b

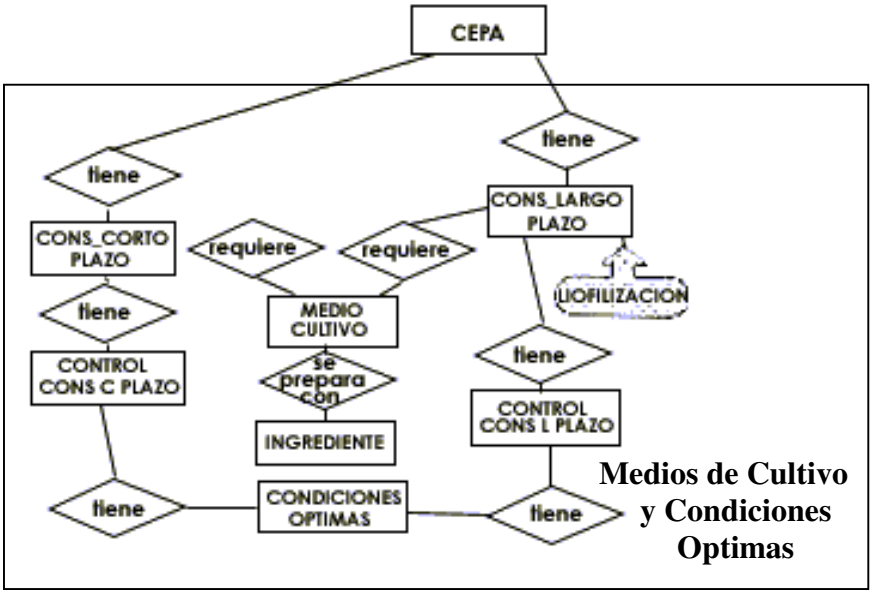


Figura 4.6 c

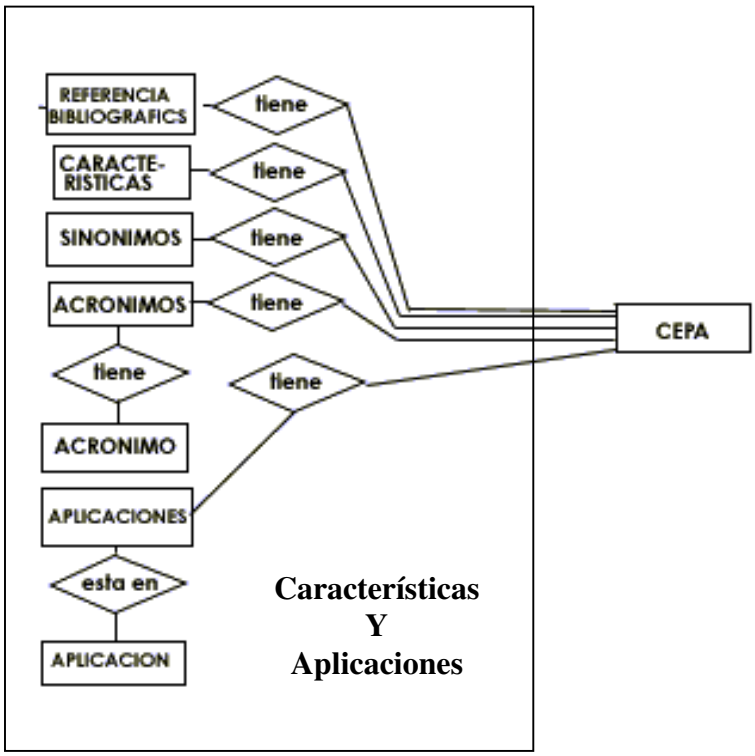


Figura 4.6 d

De esta manera el esquema conceptual fue separado en varios módulos, que son incorporados separadamente para lograr todo el sistema relacional implementado anteriormente, pero en un nuevo contexto.

Este proceso se logra gracias a que tenemos una entidad principal denominada cepa, que funciona como parte central de todo el sistema. Los módulos proveen información específica que puede ser adicionada y utilizada por la entidad principal. Pueden también existir entidades no relacionadas directamente con cepa, pero se llega a ellas por medio de una relación de otra entidad involucrada en el respectivo módulo. Así de esta forma, podemos en algún momento, extender la base de datos con nuevas entidades relacionadas directamente o indirectamente. Modelando un pequeño esquema de relación con nuestra entidad principal sin modificar nuestro esquema correspondiente. Solo incorporando el nuevo módulo y desarrollando solo la explotación de esta nueva información en forma independiente, o haciendo uso del desarrollo ya implementado.

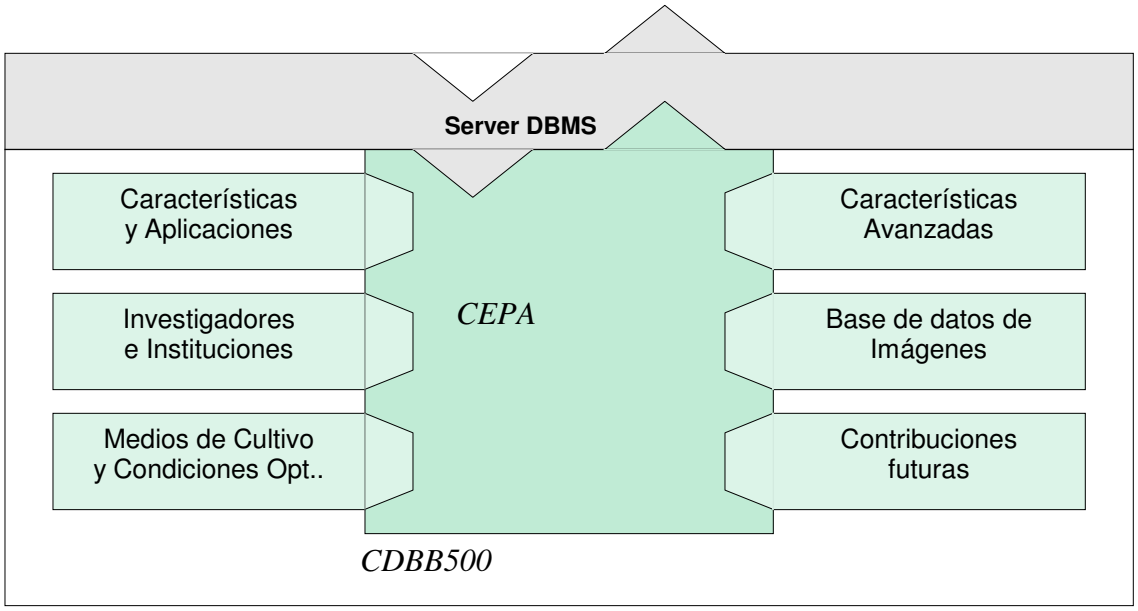


Figura 4.7 Módulos de CDBB500

La ventaja que nos ofrece trabajar de esta manera es que se pueden incorporar proyectos futuros a la base de datos, ya que podemos modelar parte de la base de datos que deseamos adicionar, desarrollar su diseño y vincularla con la entidad principal, todo esto conservando ciertos aspectos prefijados de la base de datos central para incorporarla. La figura 4.10 nos muestra los diferentes módulos que han sido incorporados para el desarrollo del sistema y cómo se vinculan y relacionan de cierta forma con la entidad cepa, además de como el servidor del DBMS nos permite el acceso a la base de datos CDBB500 y a todos los módulos que se hayan incorporado en forma paulatina, así como la espera de la conexión de una aplicación y brindar la información respectiva.

4.3 Diseño del sistema por módulos

Al igual que en la base de datos, la implementación del sistema también fue planteada con el concepto de módulos y la reutilización de código. La utilización del lenguaje de programación de java nos facilita en gran medida la implementación de este concepto debido a que es un lenguaje orientado a objetos. La programación de objetos nos permite crear funcionalidades específicas, que solo requieren datos de entrada y los devuelven ya procesados como respuesta, para ser utilizados por la aplicación para un fin especifico. Estas funcionalidades pueden ser llamadas desde diferentes clases o incluidas como paquetes, y ser utilizadas cuando se crea una entidad de dicho objeto, para envíar los parametros específicados por dicho objeto para el tratamiento de los datos. Esto permite la reutilización de código, ya que éste puede ser escrito sólo una vez y utiizado desde diferentes clases con sólo un llamado.

4.3.1 Diseño modular

Para lograr la modularidad en el sistema, se hicieron diversas pruebas y se crearon mecanismos con una función específica. Una vez desarrollado el mecanismo se trató de encontrar una atomicidad en el mecanismo, a manera de reducirlo a la más simple codificación posible, quedando reducido a pequeñas partes de código que lograban un procesamiento específico de los datos; de esta manera podemos aislar el mecanismo y llamarlo o incluirlo desde los diferentes módulos, que son implementados en forma separada e independiente de la aplicación, para ser incluidos en el sistema y así brindar un nuevo servicio.

De esta forma fueron implementados los mecanismos de acceso a la base de datos, mecanismos de presentación y de acceso a otras consultas por medio de los datos presentados al usuario. Para que los mecanismos funcionaran de manera adecuada, se fijó un estandar para el paso de parámetros; y estos fueran procesados de forma exitosa por las diferentes páginas JSP.

En el paso de parámetros entre el cliente y el servidor, se estudió la posibilidad de que fueran procesados con JavaScript, utilizándolo para la captura de errores y formateo de entrada de los datos, pero se desechó esta propuesta debido a la confusión que podía provocar la inclusión de código JavaScript y Java en la codificación de la página, ya que también podría crear confusiones en el mantenimiento del sistema en un futuro; además de que ya no se tendría una separación clara entre la programación y la presentación del contenido. Debido a este problema se decidió utlizar sólo código java para las páginas JSP y utilizar los mecanismos de tratamiento de errores soportados por JSP; por medio del redireccionamiento a páginas que muestran el error encontrado o introducido por el usuario del sistema. Esto nos proporciona una gran eficiencia desde el punto de vista de que sólo tenemos java como lenguaje de programación, y el soporte de los diferentes navegadores de internet; ya que el servidor envía código HTML puro a los clientes, que es soportado por todas las versiones de los navegadores. Esto no se cumple para el código JavaScript u otro lenguaje Script o Tecnología web.

El diseño de la implemetación de los módulos con páginas JSP, radicó en cuatro puntos que debían implementarse en el sistema, estos puntos son:

- Tener de diferentes tipos de Búsqueda
- Vista de resultados encontrados
- Vista de detalles de un resultado
- Vista de medios de cultivo solicitado

Debido a esto, se optó por desarrollar por separado cada uno de estos módulos y conectarlos al contenedor para brindar nuevos servicios especificos en forma independiente, y lograr su funcionamiento el cual era complejo y difícil de programar.

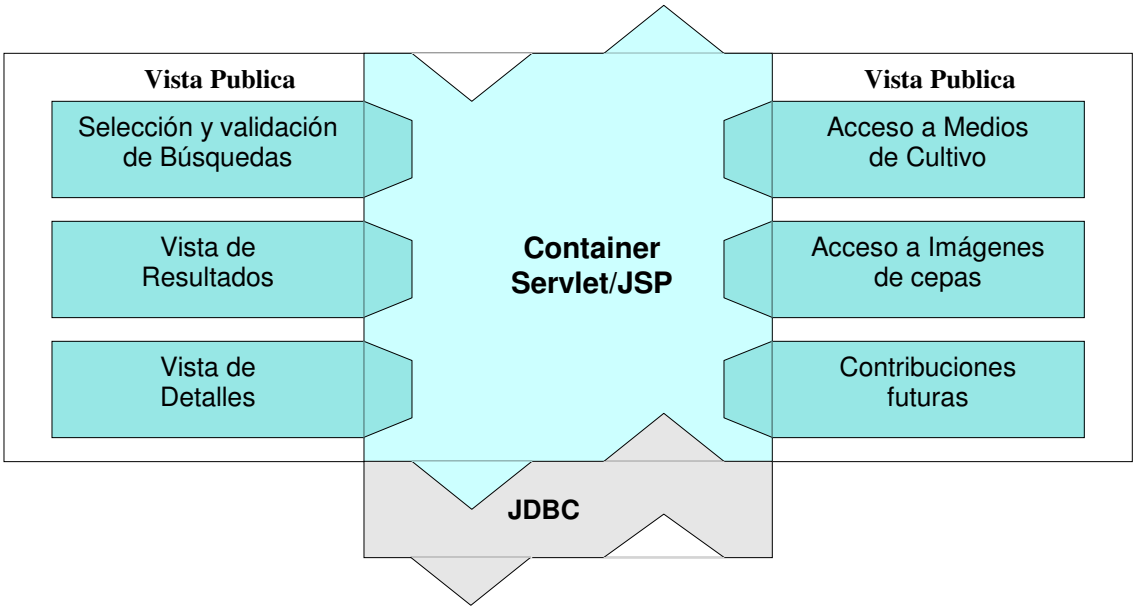


Figura 4.8 Módulos del sistema

El diseño modular mostrado en la figura 4.8, nos brinda mucha eficiencia en la codificación, reutilización de código, mantenimiento y extensión de capacidades del sistema, por medio de contribuciones posteriores o futuras de nuevos servicios. Este diseño también permite tener un control y facilita el rastreo de errores que pudieran encontrarse en la etapa de pruebas del sistema, yaque puede ser encontrado el lugar específico dónde se ejecuta dicho error. Esto gracias a la claridad y sencillez de programación de cada unos de los módulos. Estos módulos han sido diseñados de tal forma que el código encontrado en cada uno de ellos, es muy pequeño en comparación con la funcionalidad que proporcionan, gracias a la reutilización de código en todos los módulos. De esta forma, si se desea incorporar una nueva funcionalidad al sistema creando un nuevo módulo, sólo necesita concentrarse en unas cuantas líneas de código a programar, y si es necesario; incluir los mecanismos creados anteriormente. Este proceso se utilizó para crear los módulos de resultados, detalles y medios de cultivo.

Arquitectura Modular del Sistema Micro500

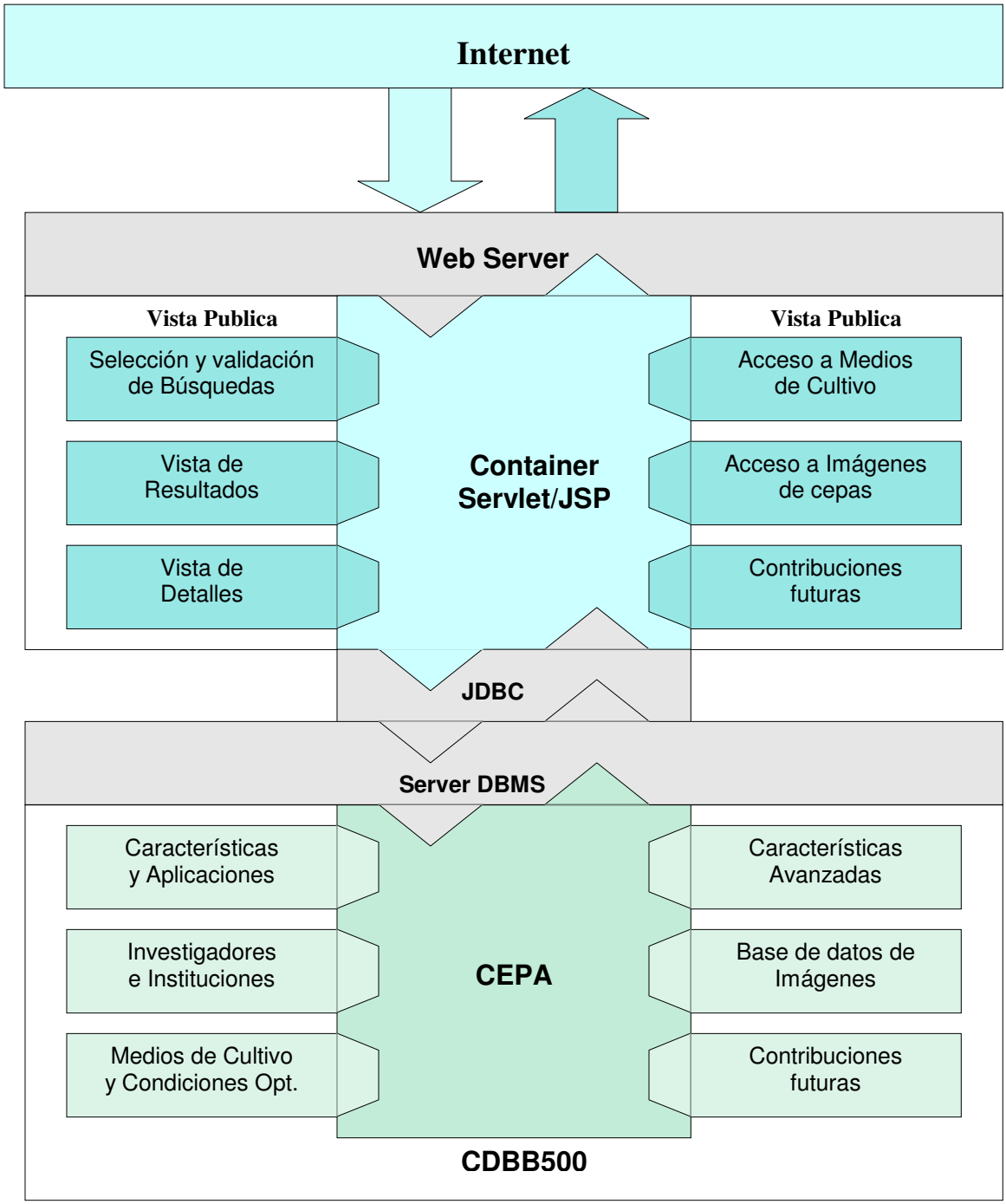


Figura 4.9 Arquitectura Modular de Micro500

4.3.2 Arquitectura modular de Micro500

La figura 4.9 nos muestra la arquitectura modular implementada en el sistema Micro500. En esta se observa en primera instancia, la petición de un cliente por medio de Internet, esta petición es recibida por el servidor web, que en este caso es Apache, una vez que el servidor web recibe la petición, se hace una petición al contenedor de servlets, que se encarga de procesar la información solicitada. Cada módulo está cargado en memoria a la espera de una solicitud y contiene todos los mecanismos utilizados por éste, el más común de ellos es la conexión a la base de datos CDBB500, la cual se logra por medio de JDBC. El cual su vez se conecta con el servidor del DBMS, para accesar a la base de datos y convertir la petición, al lenguaje propio del DBMS; para que pueda ser procesada la solicitud. Una vez que se obtiene la información esta es devuelta al JDBC, que se encarga nuevamente de traducirla, para que pueda ser presentada por el contenedor de servlets, y crear una página en tiempo de ejecución. Después de obtener esta página dinámica es transformada en una página estática, que puede ser reenviada con gran rapidez por el servidor web. Pudiera pensarse que este proceso hace lenta la devolución de la información, pero no es así; debido a la sofisticación y rendimiento de la tecnología implementada por el servidor dedicado del sistema Micro500.

4.3.3 Comentarios Finales

La arquitectura implementada nos facilita el posible cambio del DBMS, ya que podemos hacer uso del mismo JDBC para conectarnos, o en su defecto hacer uso de ODBC, si el DBMS así lo requiere. Del mismo modo si nosotros deseáramos cambiar el contenedor de servlets utilizado, sólo bastaría con cambiarlo por otro e incorporar los módulos, para obtener la misma funcionalidad. Ya que el código de los módulos no se modificaría en lo absoluto, porque ellos no están previamente compilados. Sino que están contenidos en archivos de texto que son compilados la primera vez que son solicitados y puestos en memoria, mientras el contenedor este en funcionamiento o se realicen cambios al código del módulo. En este caso el contenedor detecta que ha habido un cambio, y ante la primera petición, se recompilarán en tiempo de ejecución sin ningún problema. Claro si no se tiene un error en el código reescrito o modificado. La parte web server, también puede ser cambiada por algún otro, que soporte la conexión con el contenedor.
De esta forma no se esta sujeto a un software específico de ninguna índole, ni a un sistema operativo especifico, debido a la independencia de plataforma soportada por Java, el servidor web y el DBMS. Incluso si se desea, se pueden tener en sistemas operativos diferentes, o también quizás en computadoras diferentes.

Capítulo 5

Capítulo 5

Implementación del Sistema Micro500

Una vez que se ha mostrado la arquitectura, con la cual se desarrolló el sistema y el concepto modular bajo las cual se incorporo la base de datos, a partir de un esquema conceptual complejo. Así como también las diferentes funcionalidades con las que contará el sistema. En este capítulo se mostrara como fueron implementados y programados los módulos, para poder llevar a cabo este concepto. También el diseño y funcionalidad de cada una de las vistas mostradas a los usuarios finales del sistema. Aquí se detalla parte de la construcción, en base a los requerimientos solicitados por parte del personal de la Colección Nacional de Cepas Microbianas y Cultivos Celulares del Cinvestav. Y así obtener un sistema que cumpla con los objetivos planteados inicialmente. Entre los más importantes se encuentra la posibilidad de expansión de capacidades, así como la incorporación de proyectos futuros relacionados con la base de datos CDBB500.

5.1 Paso de mensajes entre Vistas

Para poder codificar el sistema, fue necesario hacer un estudio de cómo estos módulos se comunicaran unos con otros, y que a su vez mostraran cierta independencia entre ellos. Para obtener este resultado, se utilizó como herramienta un diagrama de secuencia en UML, el cual se muestra en la figura 5.1. Este diagrama fue de mucha utilidad, ya que proporcionó mucha información para el diseño de la interfaz mostrada al usuario, para que el usuario pueda navegar entre los diferentes módulos y encontrar la información requerida. Si se observa el diagrama se puede ver que se tienen seis diferentes vistas, que son:

1.- Menú de Opciones.
2.- Solicitud de los Usuarios.
3.- Presentación de Resultados.
4.- Presentación de detalles de los resultados.
5.- Presentación de los medios de cultivo.
6.- Tratamiento de errores de usuario.

Estas vistas deben ser diseñadas a fin de que el usuario final, navegue entre ellas y se le facilite la búsqueda de la información de alguna cepa en especial. Una vez identificadas estas vistas; la relación entre ellas es por medio de diferentes tipos de mensajes, que servirán para la interacción entre los diferentes módulos implementados en el sistema. Como podemos ver, los mensajes son enviados de una vista a otra. La primera de ellas es por medio de un menú de búsqueda, este menú es diseñado a partir de los requerimientos de los usuarios y deberá permitir la selección de una búsqueda determinada. Después es introducida parte de la información solicitada; se muestran todos los resultados encontrados para que puedan ser analizados de manera independiente, por medio de la selección de cada uno de ellos. Así como datos especiales de los mismos, como puede ser su medio de cultivo y condiciones optimas de crecimiento.

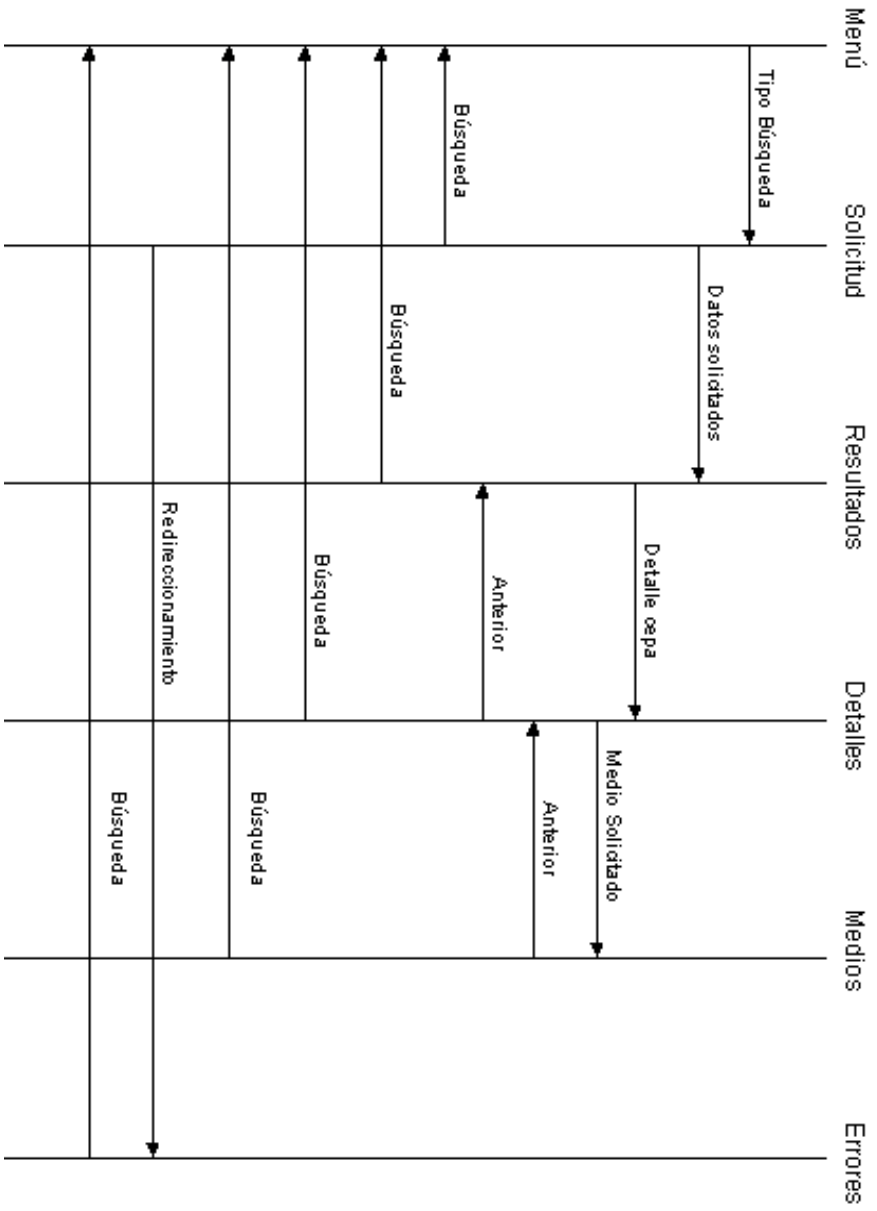


Figura 5.1 Diagrama de Secuencia de Micro500

El diagrama de secuencia también nos permite observar, que en todo momento se debe de poder acceder al menú de búsqueda, desde cualquier vista en la que nos encontremos. En la vista de medios y detalles, debemos de proveer un acceso a la vista anterior para comparar u obtener otro tipo de información. A continuación se mostrará la construcción de cada una de las vistas que serán presentadas a los usuarios a partir de las vistas presentadas por el diagrama de secuencia.

5.2 Implementación de la solicitud de búsquedas

Para la implementación de la vista de este módulo, se tuvo que estudiar y comprender cómo es que se conforma parte de la información de una cepa. Para ello se tratara de explicar parte de esta información. Todas las cepas tienen un género y una especie, los cuales sirven como identificación. Pero el nombre y especie no son únicos en la base de datos, ya que se tiene información de diferentes microorganismos con el mismo nombre y especie. Así que si realizamos una búsqueda de un Bacillus subtilis, al encontrar los resultados observaremos que existen varios microorganismos con idéntico género y especie, pero además algunos cuentan con un nivel rango infrasubespecífico y un nombre infrasubespecífico, los cuales son deseables que se muestren en caso de existir, ya que no todos los microorganismos cuentan con estas características.

Las cepas contenidas en la base de datos también cuentan con un número de registro en la base de datos CDBB, así que este número es mostrado como CDBB de la cepa, esto quiere decir su número único en la base de datos, este es importante porque nos permitirá distinguirla entre diferentes microorganismos que presentan idéntico género y especie.

Diversas colecciones en todo el mundo cuentan con estas mismas cepas y cada colección tiene un registro especial referenciado, por las iniciales de la colección. De esta manera cuando hablemos de la cepa registrada como CDBB 245, nos estaremos refiriendo a la cepa registrada en nuestra base de datos con el número 245, de la misma forma cuando encontremos un acrónimo como ATCC 102, nos referiremos a una cepa que pertenece a otra colección, en este caso a la American Type Culture Collections, y la cepa registrada con el número 102. Algunas colecciones en el mundo no tienen sus registros con sólo números, sino que éstos cuentan con caracteres y números. Parte de esta explicación radica en que nuestra base de datos, cuenta con microorganismos que también tienen registros en otras colecciones en el mundo. Sólo que cada colección otorga una identificación diferente, según su criterio y control de cada centro de investigación, institución u organización, que cuenta con estas colecciones. Por cada microorganismo registrado en nuestra base de datos, muchos de ellos tienen lo que denominamos acrónimos; estos acrónimos son identificaciones que tienen en otras colecciones internacionales afiliadas a la WFCC.

Existen diferentes tipos de cepas; la base de datos cuenta con información de 4 tipos como son: Bacterias, Hongos, Levaduras y Protozoarios. Obteniendo cada una aplicaciones y medios de cultivo diferentes. Cada uno de estos tipos de cepa cuentan con todas las características mencionadas anteriormente.

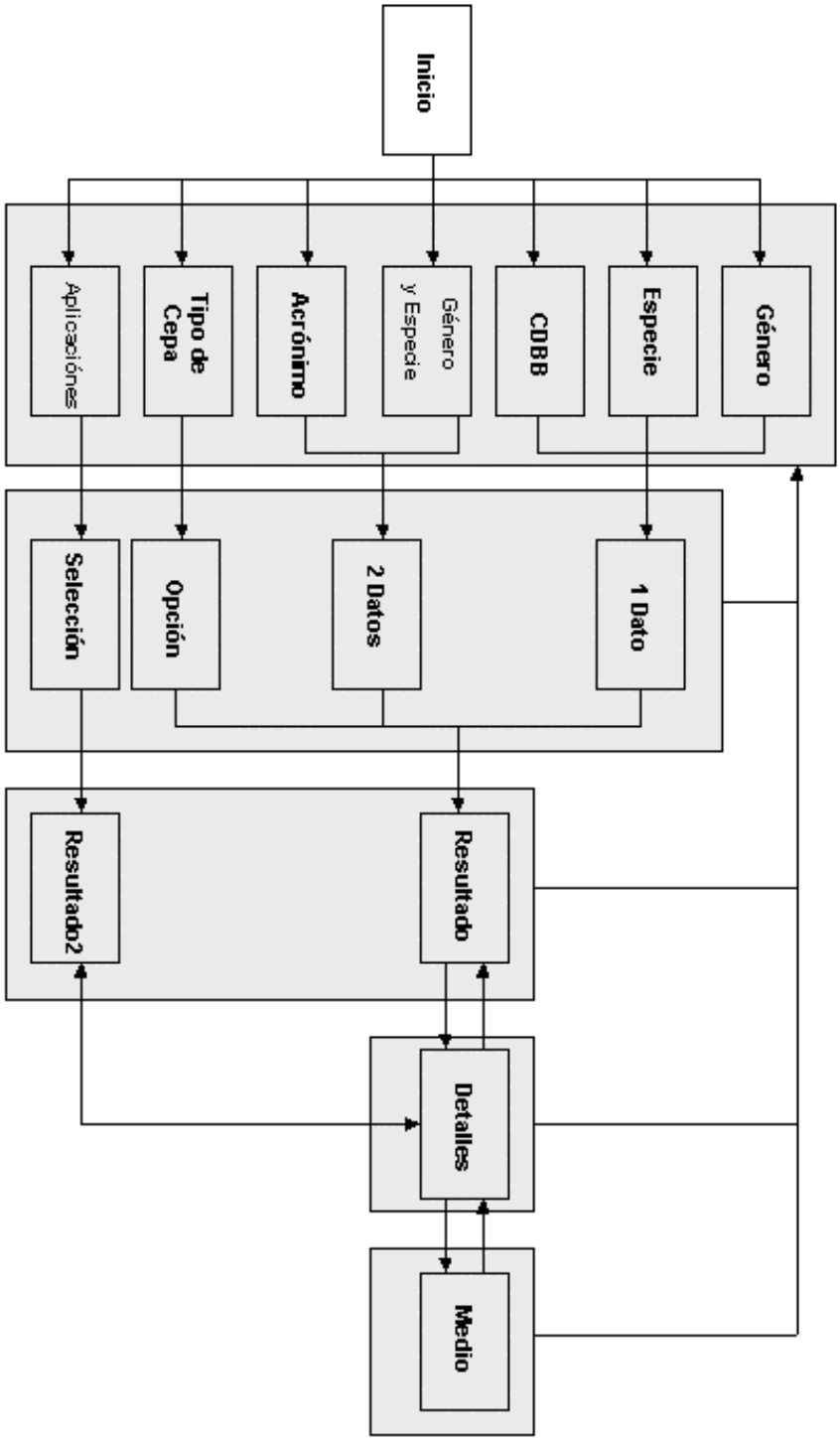


Figura 5.2 Diagrama de Diseño para Micro500

Para crear las diferentes vistas que presentara una búsqueda, se utilizó el diagrama presentado en la figura 5.2. En éste podemos observar que se desea tener una página de inicio, que a su vez nos conecte con las diferentes opciones de búsqueda. Cada búsqueda puede contener diferentes tipos de datos utilizados para su procesamiento. Una búsqueda por género y una búsqueda por número CDBB requieren de sólo un dato. Pero cada uno de ellos es distinto, ya que para género se requiere una cadena de caracteres y el número CDBB requiere de un número entero. Debido a esta similitud se han separado junto con la búsqueda de especie, para así ser procesadas por una página que pueda distinguir entre los diferentes datos que han sido enviados. Para el caso de acrónimo y género-especie, se deben tener en cuenta dos datos, estos datos contienen información que deberá ser procesada de manera independiente y conjuntada para encontrar los resultados solicitados. Cuando seleccionemos tipo de cepa, sólo se podrá seleccionar una opción de las mostradas como son: bacterias, algas, hongos y protozoarios. Por último en la opción de aplicación el motor redireccionará la petición a otra vista, que procesará la información de forma diferente a las opciones anteriores. A partir del diagrama de la figura 5.2, también se muestra cómo a partir de la vista de detalles, podemos obtener información de los medios de cultivo y las imágenes, pero para llegar a detalles siempre es a partir de los resultados que se presentan al hacer una consulta.

Figura 5.3 Diseño de presentación

La figura 5.3 muestra el diseño que fue implementado para lograr interactuar con los usuarios, y que en cierta medida les facilite la búsqueda de la información. Para ello se tiene disponible un menú de búsquedas que está siempre disponible, independientemente de la vista en la que nos encontremos. Esto es muy práctico, debido a que no se necesita salir de un anidamiento de vistas, ya que podemos cambiar nuestra opción tomada inicialmente desde cualquier parte donde nos encontremos en el sistema, sin ninguna afectación.

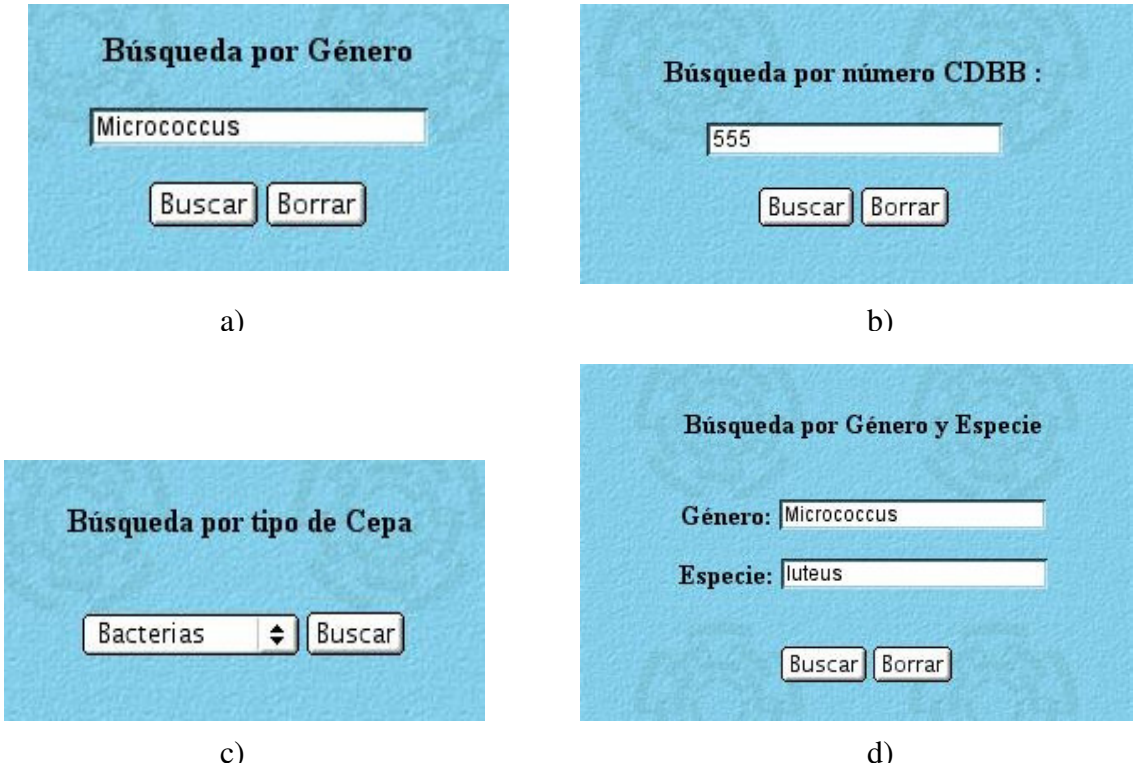


Figura 5.4 Diferentes vistas de una búsqueda

La figura 5.4 muestra el tipo de información que puede ser introducida para realizar una búsqueda. En la figura a) se introduce una cadena de caracteres; en la figura b) tenemos un número entero; en la figura c) se muestran sólo opciones y en la figura d) se solicitan dos datos para encontrar un resultado. Aunque si bien es posible solo contener una caja de entrada de texto y considerar todo este tipo de información, podemos no tener una búsqueda tan especializada y tardarnos más, realizando una búsqueda que de esta forma resulta muy sencilla. Por ejemplo que se solicita información sobre una cepa de número 245, y como resultado pueden aparecer quizás 30 cepas con parte de la referencia con número 245, quizás aplicaciones con este número y hasta el final, la cepa que se esta buscando. Debido a ello es más práctico para este tipo de información tener este tipo de búsquedas, ya que en determinado momento son más específicas y especializadas. También de esta forma si deseamos incorporar un nuevo tipo de búsqueda es muy práctico, ya que solo se añade como otra opción. Si por el contrario se desea cambiar en su totalidad estas formas de búsqueda, podemos obtener los resultados sólo transfiriendo la solicitud de la misma forma en que es soportada con las vistas actuales.

Una vez que se ha descrito parte de la implementación de los tipos de búsqueda, la siguiente parte muestra cómo a partir de los datos solicitados, la información es desplegada con un cierto formato para facilitar la selección de la información.

5.3 Implementación de la Presentación de Resultados

Para que se puedan presentar los resultados obtenidos en una búsqueda, debemos recordar que se hace un acceso a la base de datos y la página es creada en forma dinámica, dependiendo de los resultados encontrados. De esta manera podemos obtener una cantidad variable de 1 a *N* resultados en algunas búsquedas, pero no en todas. Cuando buscamos por número CDBB o Acrónimo, solo podremos obtener uno y solo un resultado. Debido a que no tenemos dos cepas con registro CDBB duplicado y el acrónimo con que cuenta en otra colección, también es propio de una única cepa; aunque para otro tipo de búsqueda como género o especie podemos encontrar múltiples resultados.

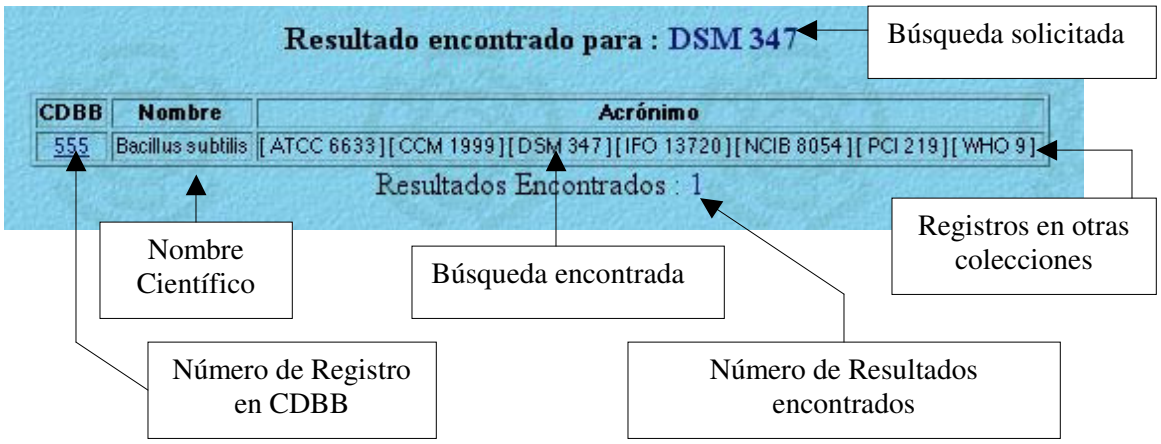
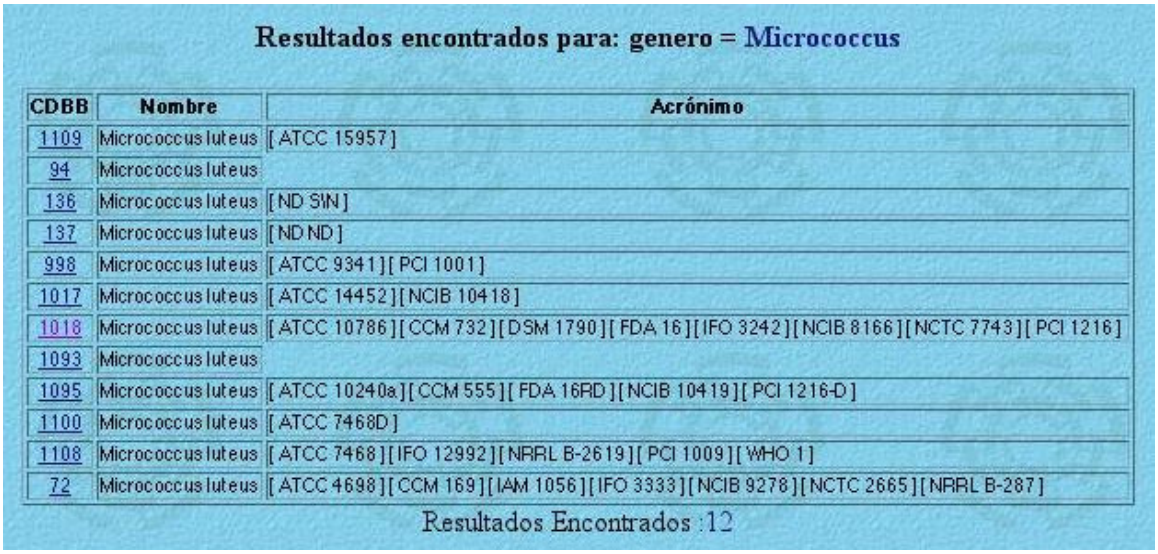


Figura 5.5 Diseño de Vista de Resultados

La presentación de los resultados se diseñó, con base a la información que el personal de la colección de cepas microbianas solicitó, según su propia experiencia con este tipo de información. En la figura 5.5 se puede observar cómo cada información de cepa encontrada muestra el número de registro en la base de datos, su nombre científico que comprende el género, especie y rango infrasubespecífico, junto con nombre infrasubespecífico, en caso de contener estos datos. Después se colocan todos los acrónimos con los que cuenta la cepa en otras colecciones, algunas de ellas no contienen ninguno; pero algunas más contienen varios. Como se muestra en la figura, en cada búsqueda realizada se muestra la misma información. El resultado de encontrar un acrónimo DSM 347 y este resultado coincide con un acrónimo contenido en la cepa, que como se ha dicho antes, sólo una cepa contiene este dato, por lo cual solo se muestra un resultado y aunque la cepa cuenta con varios acrónimos, el resultado nos muestra que se ha encontrado ese acrónimo como parte de la referencia de una cepa. De igual manera si se hiciera una consulta por medio de género que sea igual a micrococcus, se obtendría el resultado de la figura 5.6 que nos muestra doce resultados encontrados, todos ellos con el mismo nombre científico construido por el género y la especie. Pero cada uno es distinguido por su número de registro y los diferentes acrónimos para cada uno de los resultados encontrados.



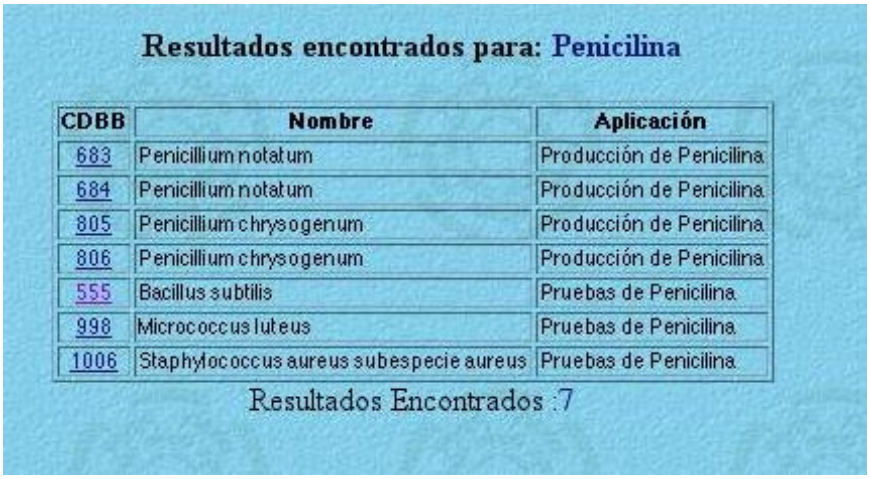
Resultados encontrados para: genero = Micrococcus

CDBB	Nombre	Acrónimo
1109	Micrococcus luteus	[ATCC 15957]
94	Micrococcus luteus	
136	Micrococcus luteus	[ND SIN]
137	Micrococcus luteus	[ND ND]
998	Micrococcus luteus	[ATCC 9341] [PCI 1001]
1017	Micrococcus luteus	[ATCC 14452] [NCIB 10418]
1018	Micrococcus luteus	[ATCC 10786] [CCM 732] [DSM 1790] [FDA 16] [IFO 3242] [NCIB 8166] [NCTC 7743] [PCI 1216]
1093	Micrococcus luteus	
1095	Micrococcus luteus	[ATCC 10240a] [CCM 555] [FDA 16RD] [NCIB 10419] [PCI 1216-D]
1100	Micrococcus luteus	[ATCC 7468D]
1108	Micrococcus luteus	[ATCC 7468] [IFO 12992] [NRRL B-2619] [PCI 1009] [WHO 1]
72	Micrococcus luteus	[ATCC 4698] [CCM 169] [IAM 1056] [IFO 3333] [NCIB 9278] [NCTC 2665] [NRRL B-287]

Resultados Encontrados :12

Figura 5.6 Vista Resultados

También se puede observar que el número CDBB, es utilizado para obtener información especifica de la cepa seleccionada. Ya que se hace una consulta dinámica cuando se presiona sobre el número. Cuando se hace esta consulta nuevamente se accede a la base de datos y se extrae información, pero sólo de esa cepa. Cada vez que nosotros seleccionamos una cepa diferente, se tiene el mismo proceso; así que cuantas veces seleccionemos un número distinto, tantas veces hacemos solicitudes a la base se datos, las respuestas que se obtienen de este proceso son muy rápidas, debido a la optimización del código y la devolución de código HTML puro a la máquina cliente. Además de que no se tiene algún otro lenguaje intermediario que no sea Java.



Resultados encontrados para: Penicilina

CDBB	Nombre	Aplicación
683	Penicillium notatum	Producción de Penicilina
684	Penicillium notatum	Producción de Penicilina
805	Penicillium chrysogenum	Producción de Penicilina
806	Penicillium chrysogenum	Producción de Penicilina
555	Bacillus subtilis	Pruebas de Penicilina
998	Micrococcus luteus	Pruebas de Penicilina
1006	Staphylococcus aureus subespecie aureus	Pruebas de Penicilina

Resultados Encontrados :7

Figura 5.6 Vista Resultados

En el menú de consultas existe una opción llamada aplicación, esta opción fue una capacidad que se incorporó al sistema en la etapa de pruebas finales debido a una última solicitud. Esta consulta fue orientada a las diferentes aplicaciones que puede realizar una cepa, pero estas aplicaciones pueden ser hechas también por diferentes cepas. Así que para explotar esta nueva característica se presenta la información, pero en vez de tener los acrónimos de esas cepas, se presentan las aplicaciones generales de ellas; por lo cual si buscamos por aplicaciones y solicitamos la información para penicilina, se mostrarán los resultados mostrados en la figura 5.6. Obsérvese que en aplicaciones, se muestran dos tipos de aplicaciones, una referida a producción de penicilina y otra referida a pruebas de penicilina. De esta forma podemos ver que diferentes cepas están relacionadas a la penicilina en diferentes maneras. De la misma forma que en los resultados mostrados anteriormente, también podemos acceder a información más específica a través del número CDBB, por medio de una vista llamada detalle de la cepa. A continuación se mostrará la implementación de la siguiente vista denominada detalle de la cepa.

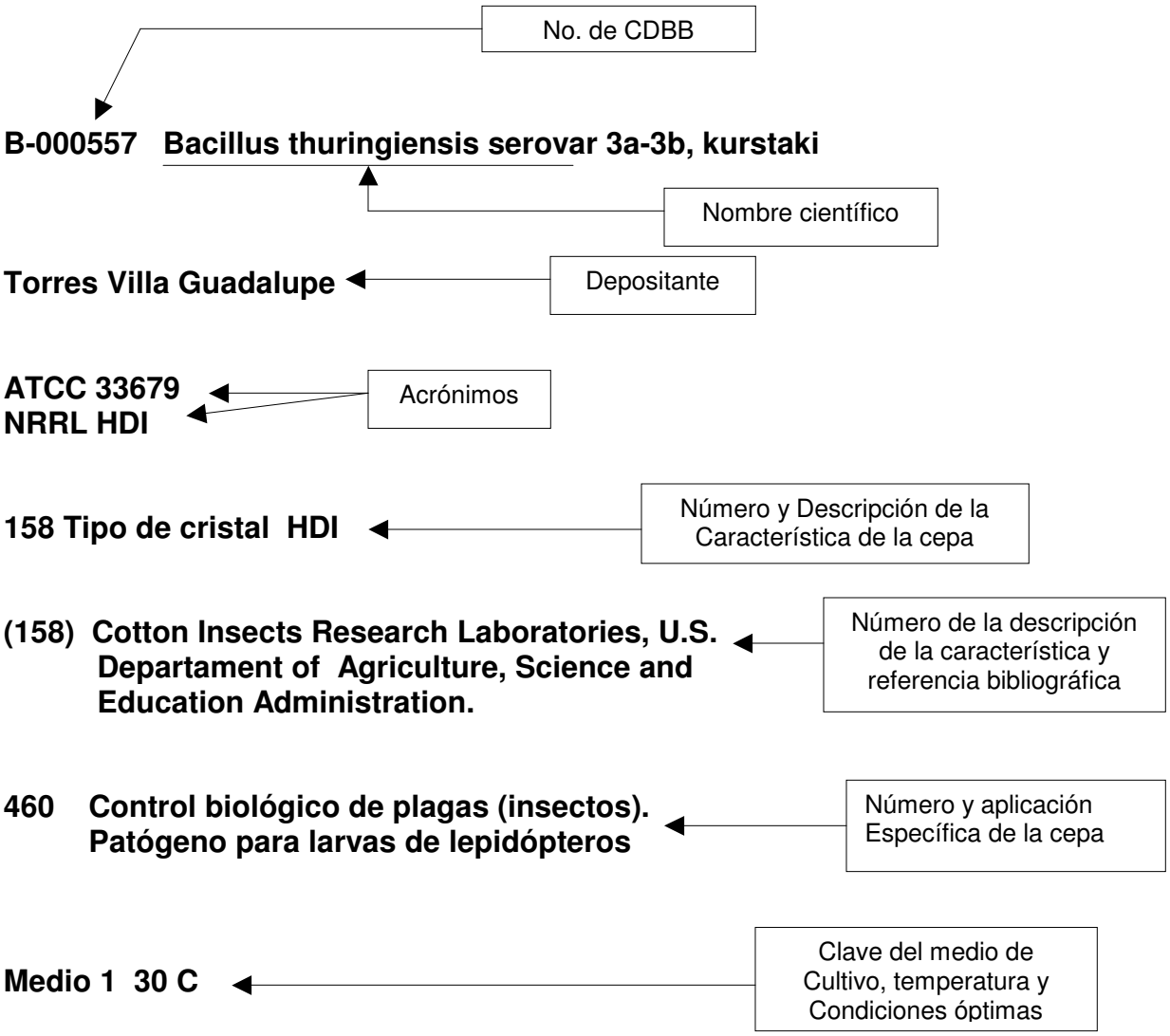
5.4 Implementación del detalle de cepa

Para el diseño de esta vista, el personal de la colección nacional de cepas microbianas; entregó un formato establecido, para visualizar la información de una manera específica y de especial interés, cuando se solicite la información de una cepa determinada. Este documento nos presenta el orden y la forma en que se desea el resultado. Para poder mostrar esta información, se requiere una serie de consultas a diversas tablas, así como a distintos catálogos con los que cuenta la base de datos.

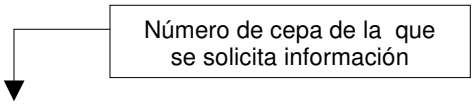
En el primer documento se puede observar que se tiene incluso el número CDBB, cómo anteriormente se encontraba antes de realizar las modificaciones a la base de datos. Este número CDBB contaba con caracteres y ceros como registro. El nombre de quien depositó la cepa, los acrónimos y descripción de la cepa junto con referencia de cada una de estas descripciones. También se encuentra el número de la aplicación y la aplicación misma así como el medio de cultivo utilizado para mantener esta cepa viva, junto con su temperatura óptima.

En un segundo documento, se presenta esta misma información con un nuevo formato utilizado para facilitar y mejorar parte de la información involucrada. Se muestra además el nombre de cada dato presentado para una mejor localización, con lo cual se tiene una mejora en cuanto a la presentación de datos. Un rasgo muy importante es que desde ahí mismo podemos consultar el medio de cultivo utilizado por dicha cepa, para obtener la preparación y cantidades utilizadas de cada elemento del medio de cultivo. Estas mejoras requirieron de un gran esfuerzo y fueron posibles, gracias a la manera en que fue implementado el sistema, y a algunas discusiones que permitieran llevar a cabo dichos cambios. A continuación se muestra el formato inicial y posteriormente el formato final, ya implementando en el sistema.

DISTRIBUCIÓN INICIAL DE LA INFORMACIÓN DE LAS CEPAS
EN CATÁLOGO



Distribución Final de la Información de las Cepas en Catálogo



Información de 555

Número CDBB: 555

Nombre: Bacillus subtilis

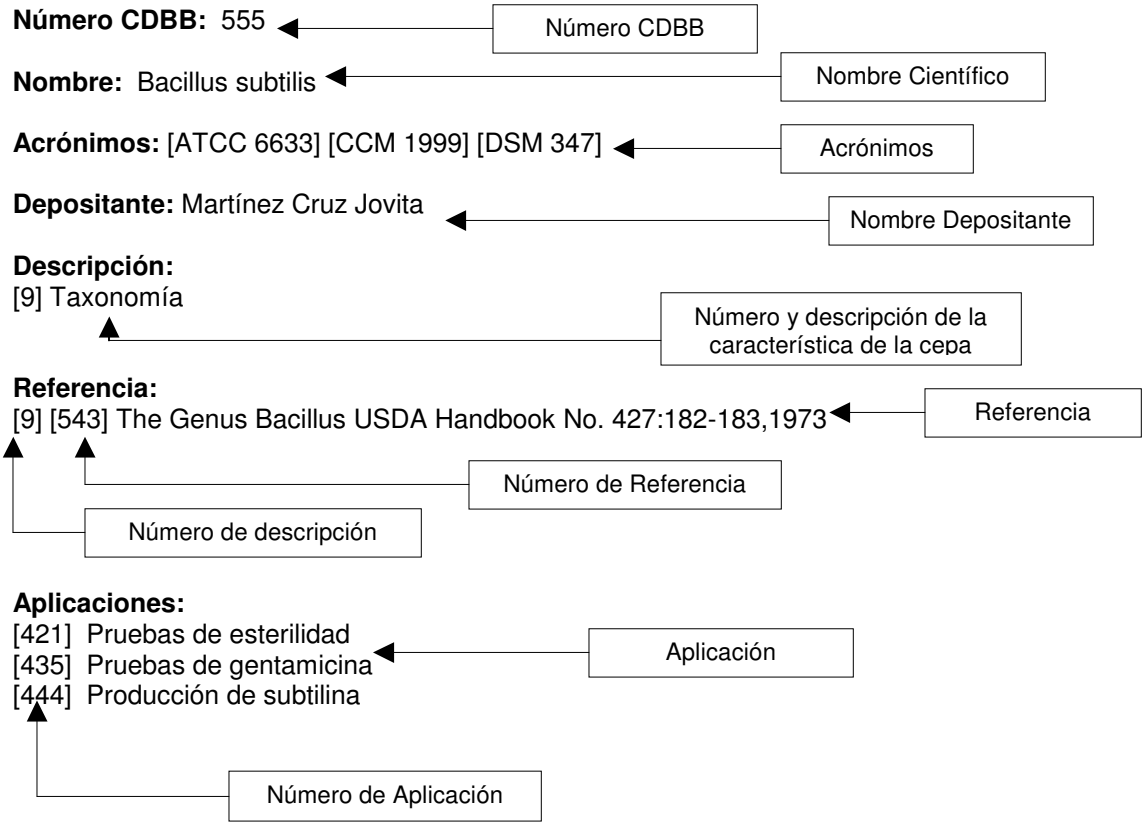
Acrónimos: [ATCC 6633] [CCM 1999] [DSM 347]

Depositante: Martínez Cruz Jovita

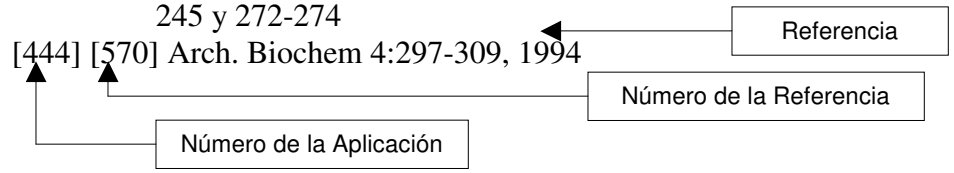
Descripción:
[9] Taxonomía

Referencia:
[9] [543] The Genus Bacillus USDA Handbook No. 427:182-183,1973

Aplicaciones:
[421] Pruebas de esterilidad
[435] Pruebas de gentamicina
[444] Producción de subtilina

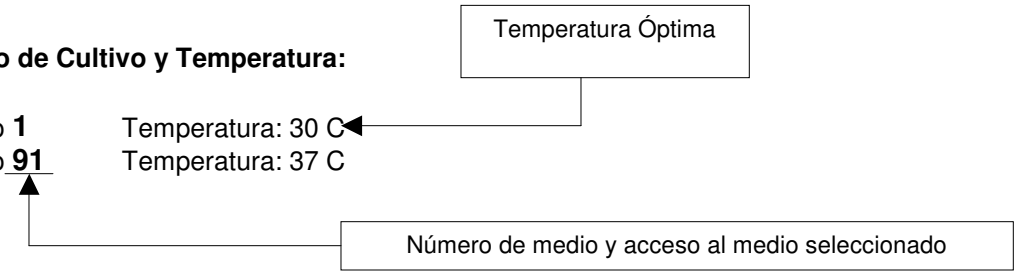


Referencias:
[421] [544] Pharmacopeia, 22nd. Rev. pp. 1483-1488,1990
[435] [560] Kavanagh, F. ed. Analytical microbiology. New York: Academic Press, 1963, pp. 237-
245 y 272-274
[444] [570] Arch. Biochem 4:297-309, 1994



Medio de Cultivo y Temperatura:

Medio **1** Temperatura: 30 C
Medio **91** Temperatura: 37 C



Una vez presentado el formato final sobre el cual se llevó la implementación de la vista de detalle de una cepa, se muestra la presentación final de cuando se hace una solicitud de este tipo en el sistema, esto se muestra en la figura 5.7

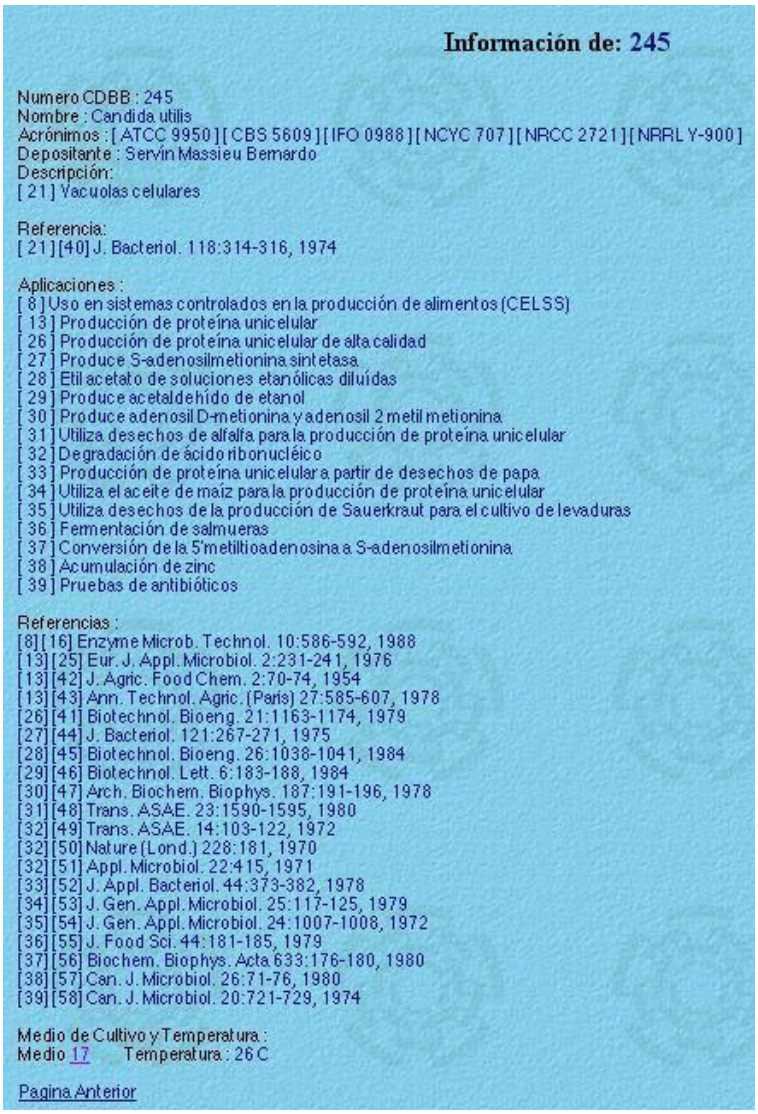


Figura 5.7 Vista de detalle de una cepa

El resultado obtenido es muy útil, porque tenemos una separación visible de toda la información, ya que se han adicionado títulos a cada parte de la información. En la vista de ejemplo vemos que esta cepa cuenta con múltiples aplicaciones. Cada una de estas aplicaciones tiene una referencia y para saber qué referencia pertenece a determinada aplicación, la referencia va acompañada de un número inicial que representa la aplicación. De esta manera si una aplicación contiene varias referencias distintas, estas son mostradas con el mismo número de aplicación y diferente referencia, de esta manera se evitan confusiones con este tipo de información.

5.5 Implementación de medios de cultivo

La implementación de esta vista, requirió de un análisis profundo debido a la complejidad que muestran los datos para ser incorporados, y que esta información fuera disponible desde el web. Existen distintos medios de cultivo y cada uno de ellos contiene información sobre la forma en que deben ser preparados, así como todos los ingredientes utilizados y las cantidades a usar, como se muestra en la figura 5.8

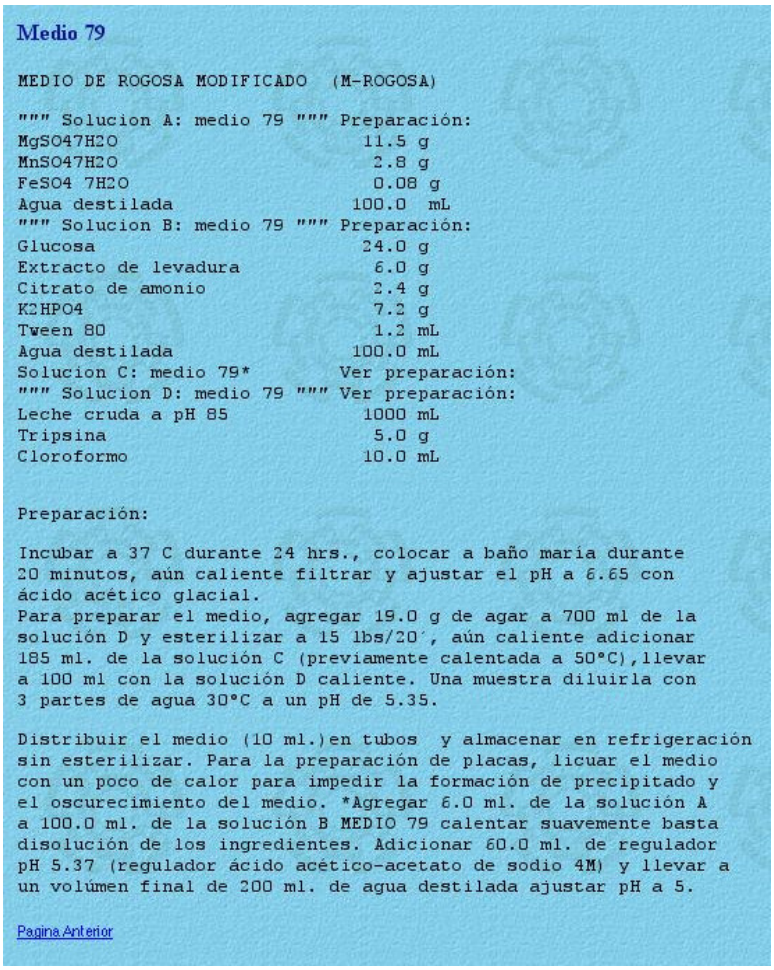


Figura 5.8 Medio de Cultivo 79

En las diferentes colecciones existentes, se observó que estos medios eran proporcionados en archivos pdf, que necesitaban ser transferidos a la máquina del cliente para poder ser abiertos desde un lector de archivos de este tipo. Algunas otras colecciones envían el archivo que contiene dicha información con una previa solicitud del mismo. Se planteó como objetivo que esta información pudiera estar disponible y con un formato establecido para los ingredientes y cantidades. Se tuvo la necesidad de adicionar una nueva tabla que contenía información de estos medios de cultivo y se vació información en archivos para

que por medio de un mecanismo creado con JSP, se incorporaran estos medios de cultivo en forma dinámica, cuando se solicitan a través de la vista de detalle de cepa.

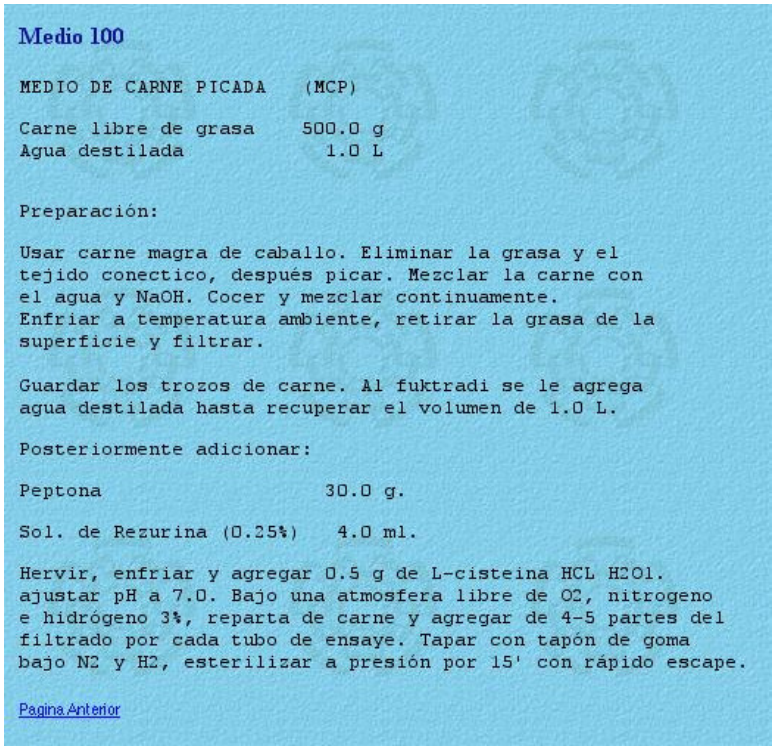


Figura 5.9 Medio de cultivo 100

Algunas cepas cuentan con diversos medios de cultivo, para ello se utiliza la liga a una página anterior y así poder consultar la información de otro medio de cultivo. Para que los datos mantuvieran el formato establecido mostrado en la figura 5.9, fue necesario realizar una serie de procesamientos de estos archivos, entre los cuales se encuentran la verificación de suprimir caracteres especiales debido al sistema operativo utilizado, y la verificación de la integridad de los datos contenidos en cada uno de ellos.

5.5.1 Comentarios Finales

En este capítulo se ha mostrado el diseño de la presentación de los datos. Mucho de este diseño fue implementado bajo solicitud del personal de la colección nacional de cepas microbianas CDBB. Algunas otras mejoras fueron incorporadas por la experiencia e inquietud del mismo personal. El sistema desarrollado ha captado mucho interés, ya que su diseño permite incorporar otros proyectos futuros, parte de ello a sido comprobado ya que al finalizar el proyecto fueron incorporados dos servicios adicionales, como búsqueda por aplicación y una vista de los medios de cultivo, mostrados en este mismo capítulo.

Capítulo 6

Capítulo 6

Conclusiones y Perspectivas

En esta parte se final muestran algunos sistemas que existen en otras partes del mundo. Cada uno de ellos contiene diferentes características y maneras de proporcionar búsquedas de los microorganismos. Diversos sistemas fueron analizados para el desarrollo de Micro500. En la parte intermedia de este capítulo, se dan las conclusiones finales de esta tesis, y en la parte final se muestran las perspectivas a futuro del sistema Micro500.

6.1 Otros Sistemas de Colecciones Microbianas

En este punto analizaremos otros sistemas de colecciones microbianas existentes en diversas partes del mundo, resaltando algunas características que se consideraron importantes en la búsqueda de microorganismos. El primero de ellos es el de Agricultural Research Service (ARS) que se muestra a continuación en la figura 6.1.

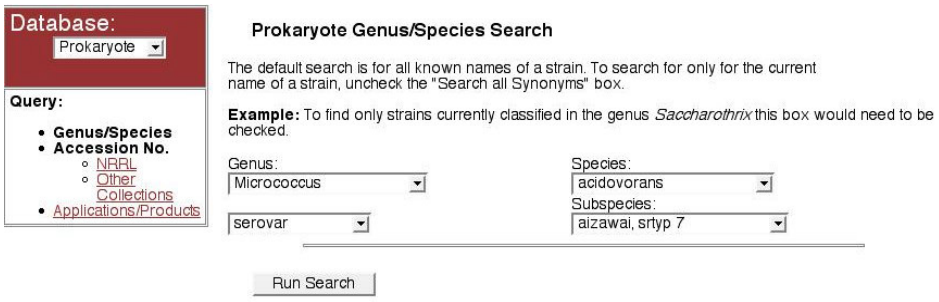


Figura 6.1 Agricultural Research Service Culture Collection ARS

En la figura 6.1 se observa que las búsquedas se realizan por medio de cajas de selección que contienen los microorganismos conocidos por la base de datos, en donde podemos seleccionar el género, especie, y subespecie. Las ventajas que se observan aquí es que no es necesario escribir el nombre de la cepa, aunque esto de primera vista parece muy atractivo y funcional no es así, ya que se encuentran algunas desventajas notorias en este tipo de búsqueda. por que si no se selecciona el género y especie respectivo, la búsqueda se dificulta, ya que podemos crear combinaciones de género y especie no existentes, y aún cuando estas pudieran existir, surge el mismo problema con las subespecies si no se conocen. Además de que la selección en las cajas respectivas no es de un número pequeño. También se dificulta la selección respectiva en computadoras de baja resolución, donde en algún momento la selección puede dificultarse en pantalla. Las ligas contenidas hacia otras páginas, nos muestran la consulta en otras colecciones que en algún momento pueden desviar el uso en esta colección, debido a las desventajas de su tipo de consulta y utilizar la de otras colecciones, no siendo conveniente para un sitio web que trata de atraer más visitantes.

La figura 6.2 nos muestra una colección de Bélgica, esta incluye una búsqueda más sofisticada que incluye diversos campos y que en determinado momento requiere el uso de ayuda y de consultar un manual de uso, lo que puede resultar tedioso para algunas personas y en determinado momento ocasionar confusión al tener varias cajas de texto a llenar.

Aunque esto no le quita merito, debemos recordar que un aspecto a tomar en cuenta es la facilidad de uso intuitivo en este tipo de sistemas. Otra desventaja que se encontró es que al navegar por los resultados y solicitar otra consulta, debemos regresarnos a todas las páginas mostradas para ver el menú nuevamente, o en su caso, escribir otra vez la dirección URL en la barra de direcciones del navegador.

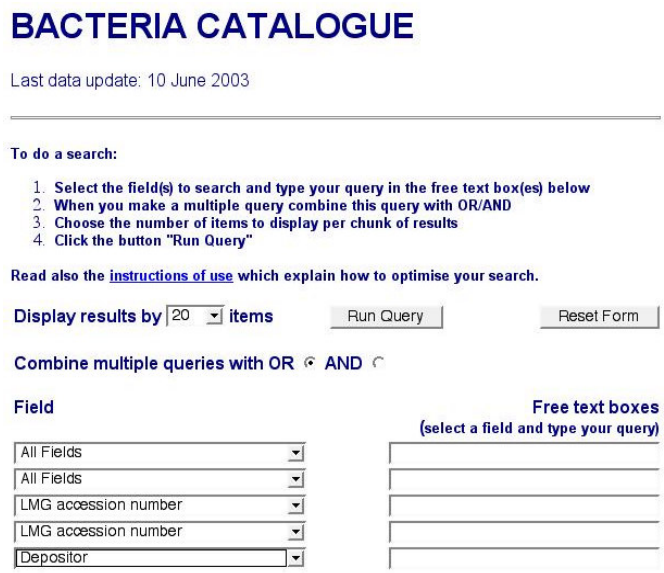


Figura 6.2 Belgian Coordinated Collections of Micro-organisms BCCM

Figura 6.3 Collection of Institut Pasteur, Paris-France

La figura 6.3 nos muestra la colección del Instituto Pasteur, ésta contiene consultas mucho más específicas. Aunque cuenta con diversos campos a ser llenados, podemos elegir de entre ellos el más adecuado. El único inconveniente que se encontró es que al igual que en

el anterior, al mostrarse los resultados se pierde el menú de búsqueda y hay que regresar entre páginas para encontrarlo. Además de que las páginas interiores muestran diversa información que puede no ser de interés, saturando la presentación. Gran parte de este diseño es muy bueno y fue de interés para nuestro desarrollo.

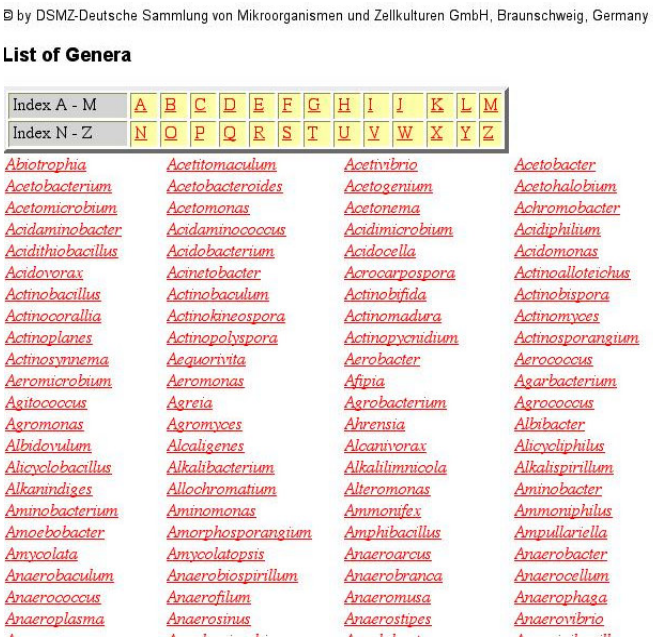


Figura 6.4 Deutsche Sammlung von Mikroorganismen und Zellkulturen DSMZ

La colección de Alemania DSMZ nos muestra una búsqueda muy peculiar ya que la búsqueda de microorganismos se realiza por medio del género. Para encontrar alguno de ellos es necesario sólo pulsar con el ratón en el menú su primera letra. La figura 6.4 muestra una búsqueda para la letra 'A', en donde podemos observar que todos los géneros comienzan con dicha letra. Sin embargo encontrar una cepa en particular se dificulta porque hay que leer diferentes nombres antes de encontrar alguna en especial. Para esta búsqueda se mostraron al menos ciento resultados y entre todos ellos se debe de seleccionar la que se desea consultar y así obtener información particular de la cepa. Se pierde parte de la funcionalidad debido a que si se desea hacer una consulta de un género en especial, por el diseño mismo, se mostrarán diversas cepas que no son solicitadas ya que tienen la misma letra inicial.

La colección de NCBI de la figura 6.5, muestra un diseño bastante similar al de la figura 6.4, pero si observamos bien, cuando solicitamos alguna letra en especial se muestran todas las cepas con su género y especie listas para ser seleccionadas, pero el problema es que si la cepa que buscamos se encuentra en la última posición, obtenemos una gran cantidad de resultados que no tiene sentido mostrar; para lo que debemos de utilizar las barras de desplazamiento hasta encontrar la cepa deseada. Lo que resta parte de la funcionalidad, como se mostró anteriormente con este tipo de búsqueda.

Figura 6.5 National Collections of Industrial Marine and food Bacteria NCIMB

6.2 Conclusiones Finales

El sistema Micro500 es parte de un desarrollo que ha constado de diversas etapas. A través de los años, se han implementado algunos sistemas que en algún momento permitieron brindar ciertos servicios conforme los tiempos y la tecnología lo permitían. Este nuevo desarrollo se ha llevado a cabo; de acuerdo a la visión a futuro que se tiene sobre diversos temas de investigación que pueden ser incorporados a corto plazo, y con miras a perfeccionar el sistema en muchos aspectos. También, como parte de un proyecto ambicioso, fue que el sistema ofreciera la información concerniente a las cepas a través de Internet.

Para llevar a cabo este proyecto se analizaron diversos sistemas de colecciones internacionales, y así poder obtener algunas características de ellas. Para llevar a cabo la implementación de estas características en nuestro sistema se requirió de un diseño nuevo. Para lo que se hizo uso de diagramas en UML, y obtener el comportamiento deseado del sistema. Para alcanzar este diseño se analizaron diferentes tipos de manejadores de bases de datos que cumplieran con las expectativas, exigencias, rendimiento y calidad que el sistema requería. También se vislumbró la idea de incorporar la reutilización de código para la implementación del sistema. La orientación a objetos fue el camino para lograr esta reutilización y se combinó con un concepto de módulos; que permitieran una alta independencia entre diferentes partes del sistema como: el sistema operativo, servidor web, contenedor de servlets, manejador de la base de datos, y la implementación de otros módulos, que permitan la extensión de diferentes servicios a los que brinda actualmente. La forma en que fue construido, facilita en gran medida el mantenimiento y algunos posibles cambios que pudieran hacerse en algún momento.

Debido a que existen diversas formas adoptadas por otras colecciones en el mundo para realizar búsquedas, se realizo un estudio para obtener un diseño que proporcione más facilidad de uso y una mejor presentación de la información para Micro500. Sin olvidar la rapidez y la compatibilidad con todos los navegadores existentes en Internet, la presentación de la información y la interacción con los usuarios del sistema, diseñada de manera que exista una forma intuitiva de obtener la información solicitada.

En esta parte final del documento se mostraron las características y desventajas que tienen algunos sistemas de búsqueda en las diferentes colecciones. Quizás este no es el mejor sistema implementado hasta el momento. Pero sí ofrece una nueva propuesta y algunas características que otros no ofrecen como: independencia, facilidad, rapidez, presentación de datos, presentación de medios de cultivo, ya que la mayoría de los otros sistemas no la proporcionan o la brindan en archivos que separados. Aunque las características mencionadas no lo hagan el mejor, si lo mantiene al nivel de entre los mejores en su tipo en todo el mundo y el único en México desarrollado e implementado hasta hoy.

6.3 Perspectivas

El sistema implementado hasta el momento es parte de un proyecto que pretende incorporar diversos temas de investigación entre los que se encuentran:

Bases de datos activas.- Este proyecto deriva de poder obtener una base de datos que mantenga una alta integridad de los datos, esta base de datos deberá reaccionar ante ciertos eventos y ejecutar ciertas acciones, dependiendo de las condiciones encontradas en los datos.

Base de datos de imágenes.- Incorporar a corto plazo una base de datos con imágenes de cada una de las cepas, imágenes que podrán ser mostradas para obtener los detalles particulares de cada cepa, completando la información obtenida de cada una de ellas cuya imagen muestre sus principales características.

Análisis estadístico.- Obtener diferentes tipos de análisis estadísticos a partir de la información contenida en la base de datos y recuperar resultados a partir de diferentes técnicas de análisis que obtengan un alto desempeño.

Autómatas celulares.- Implementar simulación y experimentación con autómatas celulares, y obtener comportamientos globales colectivos bajo la determinación de reglas locales de interacción, y obtener laboratorios virtuales en lugar de experimentar con los diferentes fenómenos directamente.

Interoperabilidad de bases de datos.- Aquí se pretende alcanzar una interoperabilidad con diferentes bases de datos ajenas e independientes del sistema en el web, que pueden complementar la información solicitada en forma transparente y ser mostradas al usuario.

Cada uno de los campos de investigación mostrados, pueden por si mismos, tener diversas líneas de investigación con mucho campo de ampliación.

Bibliografía

[1] Jesús Manuel Olivares Ceja *"Construcción de una base de datos pictográficos con aplicación a la colección de microorganismos CDBB-500"* Tesis de Maestría, Secc. de Computación, Depto. Ingeniería Eléctrica CINVESTAV. México 1996.

[2] Jovita Martínez Cruz y Sergio V. Chapa Vergara Informe Final: *"Proyecto Colección de Cultivos Microbianos del CINVESTAV-IPN"* Base de Datos Convenio FB122/EO16/94 México D.F. Abril de 1996.

[3] Sergio V. Chapa Vergara ***"Programación Automática a partir de descriptores de Flujo"*** Tesis Doctorado, Secc. de Computación, Depto. Ingeniería Eléctrica CINVESTAV. México 1991.

[4] Roberto Tecla Parra *"Diseño e Implementación de un sistema para edición de flujogramas"* Tesis Maestría, Secc. de Computación, Depto. Ingeniería Eléctrica CINVESTAV. México 2001.

[5] Peter Pin-Shan Chen *"The Entity-Relationship Model Toward a Unified View of Data"* ACM Transactions on Database System, Vol 1, No. 1, pp 9-36, March 1976, Massachusetts Institute of Technology.

[6] Norman Murray, Carole Goble y Norman W. P A *"Framework for Describing Visual Interfaces to Databases"* Journal of Visual Languages and Computing (1998) 429-456 Article No. V1970060 Departament of Computer Science, University of Manchester, Oxford.

[7] Francesca Benzi, Dario Maio y Stefano Rizzi Visionary *"Viewpoint Based Visual Language for Querying Relational Databases"* Journal of Visual Language and Computing (1999) 117-145, DEIS Universitá di Bologna, Viale Risorgimento 2, Bologna Italy.

[8] Bahram Parvin, Qing Yang Gerald Fontenay, Mary Helen Barcellos Hoff. *"BioSig: An Imaging Bioinformatic System for Studying Phenomics Computer"* IEEE Bioinformatics,July(2002) pp 65-71.

[9] José Hernández Orallo *"La Disciplina de los Sistemas de Bases de Datos. Historia, Situación Actual y Perspectivas"* Dep. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Mayo 2002.

[10] Gary Periman *"Achieving Universal Usability by Designing for Change"* IEEE Internet Computing March-April 2002, pp 46-55.

[11] Eric Altendort, Moses Hohman, Roman Zabicki *"Using J2EE on a Large, Web-Based Project"* IEEE Software, March-April 2002, pp 81-89.

[12] Aaron E. Walsh *"JavaServer Pages 2.0"* Dr. Dobb´s Journal, pp 48-55, July 2003, Boston College.

[13] Pedro Enrique Alday Echavarría *"Diseño de base de datos con EVEX Entidad Vínculo Extendido para Xwindows"* Tesis Maestría Secc. de Computación, Depto. Ingeniería Eléctrica CINVESTAV. México 1997.

[14] Laura Méndez Seguro ***"DALI Herramienta para la representación gráfica de Información"*** Tesis Maestría Secc. de Computación, Depto. Ingeniería Eléctrica CINVESTAV. México 1998.

[15] Sergio V. Chapa Vergara y Noe Sierra Romero *"Panorama general de los lenguajes Visuales"* Febrero 2001.

[16] Agustín Froufe Quintas *"JavaServer Pages Manual de Usuario y Tutorial"* Ed. Alfaomega RA-MA Editorial, 2002.

[17] Marty May *"Core Servlets and JavaServer Pages"* Sun microsystem Ed. Prentice Hall, 2000.

[18] Mark C. Chan, Steven W. Griffith y Anthony F. Iasi *"1001 Tips para pr ogramar con Java"* Ed. McGraw-Hill, 1997.

[19] Mario Piattini, Adoración de Miguel *"Fundamentos y Modelos de Bases de Datos"* Ed. Alfaomega RA-MA Editorial, 2ª. Edición 1999.

[20] Arman Danesh *"HTML 4 Manual de Referencia"* Ed. Prentice Hall Hispanoamericana, 1997.

[21] Ramón Soria *"HTML Diseño y creación de páginas Web"* Alfaomega Grupo Editor, 1998.

[22] David A. Ruble *"Análisis y Diseño Práctico de Sistemas Cliente -Servidor con GUI"* Prentice Hall, Hispanoamericana, 1998.

[23] Bob Reselman *"Active Server Pages 3.0"* Ed. Pearson Education, S.A. 2000.

[24] Paul S. Wang *"Java con Programación Orientada a Objetos y Aplicaciones en la WWW"* Ed. International Thomson Editores S.A. de C.V. 2000.

[25] Jim Conallen *"Building Web Applications with UML"* Ed. The Addison-Wesley Object Technology Series, 2000.

[26] John C. Worsley, Joshua D. Drake *"Practical PostgreSQL"* Ed. O´Reilly & Associates, Inc. 2002.

[27] http://www.apache.org

[28] http://www.postgresql.org

[29] http://www.java.sun.com

[30] http://www.apple.com

[31] http://www.mac.com