

# Universidad Politécnica del Estado de Morelos



DESARROLLO DE UN SISTEMA ADMINISTRATIVO DE HORARIOS PARA LA  
FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA, UTILIZANDO UN  
MODELO DE TRES CAPAS DE CLIENTE, SERVICIO WEB y BD

T E S I N A

Que para obtener el título de:

**INGENIERO EN INFORMÁTICA**

Presentan

**CALDERÓN SEGURA YESSICA YAZMÍN  
MARTÍNEZ BAHENA BEATRIZ**

Directores de Tesina

**DRA. MARICELA CLAUDIA BRAVO**

**DR. MARCO ANTONIO CRUZ CHÁVEZ**

Co-Director de Tesina

**MC. EDGARDO GONZALEZ HERNANDEZ**

## **Dedicatoria**

A mis padres **Víctor y Francisca** por que ellos fueron la parte esencial para mi formación, por que gracias a su gran esfuerzo, apoyo y sacrificio lograr que se cumplieran mis metas y ser lo que soy ahora.

A mis hermanos **Héctor, Marcial, Gustavo, Christian y Elizabeth** por su gran apoyo y confianza para que este sueño se cumpliera.

A mis sobrinos **Yatziry Lizzet y Jesús Jhovany** por que gracias a sus lindas sonrisas, dulces besitos y pequeños abrazos con esas manos tan suaves y tiernas, por la ansiedad de verlos crecer siempre fue un gran motivo para seguir adelante.

## **Agradecimientos**

A **Dios** por que me ha permitido despertar cada mañana por que el es la persona más importante la que decide si te vas o continúas en está vida, gracias a dios por darme la oportunidad de cumplir con mis objetivos y metas planteadas en mi vida.

A mi **Familia** por que gracias a ellos puede lograr mi formación profesional y personal, por que con su sacrificio, apoyo y dedicación pude concluir una etapa muy importante en mi vida, por que me ayudaron a seguir adelante. Gracias por ser mi orgullo y confiar en mí.

A mis padrinos **Pablo Martínez y Constantina Bahena** por su gran apoyo por confiar en mí y por brindarme todo su apoyo incondicional en una etapa de mi vida muy importante para continuar con mi preparación.

A mis asesores **Dr. Marco A. Cruz y la Dra. Maricela Bravo** por el tiempo dedicado, por su paciencia y apoyo incondicional para seguir con este proyecto. Gracias a sus conocimientos y sabios comentarios dados con la finalidad de obtener el mejor resultado.

A mi co-asesor **Edgardo González Hernández** por su gran apoyo y tiempo dedicado y sobre todo por sus comentarios brindados para obtener un buen resultado.

A mis **Profesores** por que gracias a su gran apoyo, dedicación y paciencia que tuvieron para mí preparación, que con sus conocimientos brindados puede lograr mi formación.

Beatriz Martínez Bahena

## Agradecimientos

Antes que nada agradezco a **Dios** por permitirme lograr una desarrollarme como una futura profesional en esta etapa de mi vida, porque aun que decaí en mis estudios nunca deserte de mi carrera, por tener a una hermosa familia quien siempre me apoyo y por darme la oportunidad de conocer a tantas personas que me brindaron su apoyo incondicional, sus conocimientos y sobre todo su tiempo.

Agradezco a mi Mama **Norma Neli Segura Beltrán** quien con dulces palabras me abrazaba y me impulsaba a seguir a delante y con gran sacrificio, esfuerzo me ayudo a lograr mi carrera universitaria y es por todos estos bellos momentos en donde mis llanto se convertía en risa gracias a sus sabias palabra, pido a dios me de licencia poder brindarle mi apoyo y recordarle que ningún esfuerzo es vano y que sobre todas las cosas lo más hermoso de mi vida es mi mami.

Agradezco a mi Papá **José Calderón Vargas** quien me permitió continuar estudiando, por ser paciente y creer en mí, por ser un hombre trabajador y demostrarme que la vida no es tan fácil.

Agradezco a mi Hermano **Erik José Calderón Segura** por su forma de expresarse.

Agradezco a mis **profesores** que a lo largo de mi carrera por ayudarme a cumplir mi sueño.

Agradezco a **Edmundo Fuentes Zuñiga** por que siempre tuve su apoyo incondicional tanto material como espiritual por ser una persona noble que confía en mí, por su forma de ser, por todo el amor que me ha brindado y en verdad es una persona que hay que admirar.

Agradezco la **Dra. Maricela Bravo** por hacerme saber que no estaba sola por su paciencia, conocimientos brindados y sobre todo por su forma de expresarse.

Agradezco el **Dr. Marco Cruz Chávez** por confiar en mi palabra y darme la oportunidad de desarrollar este proyecto y sobre todo que siempre estuvo pendiente de mostrarnos el camino correcto para lograr el objetivo claro con la respectiva paciencia, comprensión y amabilidad.

Agradezco al **profesor Edgardo González Hernández** por dirigir nuestra tesina correctamente por su paciencia y tiempo dedicado.

Agradezco **Omar** por ser una persona agradable, flexible por su tiempo por su forma de ser y sobre todo por su apoyo porque no ha olvidado lo que significa ser estudiante. En el poco tiempo que trate es un hombre con un gran conocimiento informático.

Agradezco **Erika** por su atención que me prestaba al preguntarle sobre alguna duda, Ella dejaba cualquier cosa que estuviese haciendo y prestaba atención a una simple estudiante de informática a quien reconozco intentaba por cualquier medio entender al tema. Gracias por tu amabilidad, sencillez y apoyo.

Agradezco a mi exprofesora **Ocotlan Díaz** porque ella fue el medio para conocer al Dr. Marco Cruz, porque gracias su gentileza, amabilidad, carácter, personalidad y su forma de expresión. En lo personal conocí muchas personas importantes para esta etapa de mi vida.

#### **Dedicatoria:**

Esta tesina la dedico a mis padres que con gran esfuerzo y sacrificio tuvieron paciencia para sacarme adelante a Edmundo Fuentes Zuñiga por ayudarme en el último momento de mi carrera Universitaria.

Yessica Yazmín Calderón Segura

## TABLA DE CONTENIDO

<b>LISTA DE FIGURAS .....</b>	<b>I</b>
<b>LISTA DE TABLAS .....</b>	<b>II</b>
<b>LISTA DE CÓDIGOS.....</b>	<b>II</b>
<b>RESUMEN.....</b>	<b>III</b>
<b>CAPÍTULO I.....</b>	<b>1</b>
INTRODUCCIÓN.....	1
1.1 Antecedentes del proyecto.....	1
1.2 Planteamiento del problema.....	1
1.3 Objetivo general .....	2
1.4 Justificación.....	2
1.5. Alcances y limitaciones .....	2
<b>CAPÍTULO II.....</b>	<b>4</b>
ESPECIFICACIÓN DE REQUERIMIENTOS .....	4
2.2 Funciones del producto.....	5
2.3. Análisis de requerimientos.....	6
2.4. Requerimientos funcionales .....	7
2.5. Requerimientos no funcionales.....	8
<b>CAPÍTULO III.....</b>	<b>8</b>
DISEÑO DEL SISTEMA .....	8
3.1. Identificación de actores .....	8
El sistema cuenta con los siguientes 5 actores. ....	8
3.2. Diagrama de caso de usos.....	9
3.3. Diagrama de secuencia.....	10
3.4. Diagrama de clases .....	11
3.6. Diseño de la base de datos.....	12
3.6.1. Análisis de requerimientos .....	13
3.6.2. Diseño conceptual.....	14
3.6.3. Diseño lógico.....	16
3.6.4. Diseño físico.....	18
<b>CAPÍTULO IV .....</b>	<b>21</b>
IMPLEMENTACIÓN.....	21
4.1 Implementación de las interfaces del software.....	21
4.2. Interfaz de inicio al sistema .....	21
4.3. Interfaz principal del sistema.....	22
4.4. Interfaz Registrar datos .....	24
4.5. Interfaz Eliminar datos .....	25
4.6. Interfaz Modificar datos .....	25
4.7. Pantalla Generar archivo.....	26
4.8. Presentación del archivo de entrada al calendarizador .....	27
4.9. Conexión a la base de datos.....	29
4.9.1. Código para hacer un registro .....	30

4.9.2. Código para eliminar un registro .....	31
4.9.3. Código para actualizar los datos .....	31
4.10. Lectura del archivo .....	31
4.11. Almacenamiento de datos en un arreglo.....	32
4.12. Las herramientas utilizadas para el desarrollo de esta aplicación son las siguientes: .....	33
4.12.1. Api's utilizadas.....	33
4.13. Implantación .....	35
<b>CAPÍTULO V</b> .....	<b>36</b>
EVALUACIÓN.....	36
4.1. Pruebas de caja blanca .....	36
4.2. Pruebas de caja negra.....	37
<b>CAPÍTULO VI</b> .....	<b>43</b>
CONCLUSIONES .....	43
6.1. Conclusiones .....	43
6.2. Aportaciones .....	43
6.3. Trabajos futuros .....	44
Anexo 1. Glosario.....	45
Anexo2. Acrónimos .....	47
Anexo 3. Código para hacer un registro, una eliminación y una actualización de los datos de la base de datos.....	49
Anexo 4. Manual de instalación y configuración de Apache Tomcat.....	52
Anexo 5. MANUAL DE INSTALACIÓN DE JAVA (JDK -1_5_0_16) .....	62
Anexo 6. Configuración de la CLASSPATH .....	65
BIBLIOGRAFÍA .....	67

## LISTA DE FIGURAS

Figura 2.1. Pasos para generar el archivo de entrada al calendarizador .....	7
Figura 3.1. Diagrama general de caso de usos del sistema .....	9
Figura 3.2. Diagrama general de secuencia.....	10
Figura 3.3. Diagrama de clases de servlet's .....	11
Figura 3.4. Arcquitectura de tres capas: cliente, servidor Web y servidor BD .....	12
Figura 3.5. Modelo Entidad Relación.....	15
Figura 3.6.Modelo ELKA.....	17
Figura 4.1. Pantalla de inicio .....	21
Figura4.1.2 Mensaje de error.....	21
Figura 4.3 Validación de campos .....	22
Figura 4.4. Pantalla principal.....	22
Figura 4.5. Pantalla de registro de datos .....	24
Figura 4.6. Mensaje de error enviado por el sistema .....	24
Figura 4.7 Pantalla para eliminar datos .....	25
Figura 4.8. Pantalla para visualizar datos .....	26
Figura 4.9. Pantalla para modificar datos .....	26
Figura 4.10. Pantalla para generar archivo de texto .....	27
Figura 4.11. Archivo de entrada al calendarizador.....	28
Figura 4.12. Representación de eventos por alumno .....	29
Figura 5.1. Pantalla de inicio .....	38
Figura 5.1. Verificación de campos.....	38
Figura 5.2. Validación de usuarios.....	39
Figura 5.3. Página principal del sistema .....	40
Figura 5.4. Validación de matrícula de alumnos.....	40
Figura 5.5. Pantalla de horario de alumnos .....	41
Figura 5.6. Validación de clave de profesores .....	42
Figura 5.7. Pantalla de horario de profesores.....	42
Figura 4.0.1.A. Setup de Apache Tomcat .....	52
Figura 4.2.A Licencia de Apache .....	53
Figura 4.3.A Componentes de Apache.....	53
Figura 4.4.A Dirección de la instalación Apache Tomcat .....	54
Figura 4.5.A Configuración de Apache Tomcat .....	54
Figura 4.6.A Dirección de Java Virtual .....	55
Figura 4.7.A Instalación en proceso.....	55
Figura 4.8.A Instalación completa de Apache Tomcat .....	56
Figura 4.9.A Probando que la instalación sea correcta.....	56
Figura 4.10.A Generar carpeta para Apache Tomcat .....	57
Figura 4.11.A Configurar carpetas para JSP .....	58
Figura 4.12.A Colocación de JSP.....	58
Figura 4.13.A Pantalla de ejemplo de un JSP .....	59
Figura 4.14.A Como ejecutar de JSP con un Servlets.....	59
Figura 4.15.A Archivos.class.....	59
Figura 4.16.A Colocación de Driver de MySQL .....	60



Figura 4.17.A Configuración de Web.xml.....	60
Figura 4.18.A Configuración del mapeo de los servlets .....	61
Figura 4.19.A Configurar el action del formulario .....	61
Figura 5.1.A Instalador del JDK .....	62
Figura 5.2.A Preparación para la instalación .....	62
Figura 5.3.A Aceptación de licencia .....	62
Figura 5.4.A Proceso de instalación.....	63
Figura 5.5.A Proceso de instalación.....	63
Figura 5.6.A Instalación completa.....	64
Figura 5.7.A Imagen de eclipse cargando .....	64
Figura 5.8.A Dirección donde se guarda el proyecto .....	65
Figura 6.9.A Propiedades del sistema .....	65
Figura 6.10 Variables de entorno.....	66
Figura 6.11 Modificación de la variable de usuario.....	66

## LISTA DE TABLAS

Tabla 3.1 Materias .....	18
Tabla 3.2 Profesores .....	18
Tabla 3.3. Profe materia .....	19
Tabla 3.4. Grupo.....	19
Tabla 3.5.Facilidades .....	19
Tabla 3.6. Tipo plaza .....	19
Tabla 3.7. Salones.....	19
Tabla 3.8. Eventos.....	20
Tabla 3.9. Periodos.....	20
Tabla 3.10. Salon facilidad .....	20
Tabla 3.11. Alumno materia .....	20
Tabla 3.12. Materia facilidad .....	20
Tabla 3.13. Alumnos .....	20
Tabla 4.1. Métodos y objetos de los servlet´s .....	304

## LISTA DE CÓDIGOS

Código 4.9.1. Código para hacer una conexión a la bd .....	30
Código 4.9.2. Operación para registrar información a la bd .....	30
Código 4.9.3. Operación para eliminar información a la bd .....	31
Código 4.9.4. Operación para actualizar información a la bd .....	31
Código 4.10.1. Lectura de un archivo de texto .....	32
Código 4.11.1. Almacenamiento de datos en un arreglo.....	32
Código 5.1. Bucle if y while .....	36
Código 5.2. Enviar información al servlet.....	37

## RESUMEN

En el presente proyecto se desarrolla un sistema que permite llevar la administración de horarios. Se utiliza una arquitectura de tres capas: cliente, servicio Web y servidor de base de datos. El proyecto se orienta a usuarios que serán capaces de realizar diferentes funciones. El administrador podrá registrar, eliminar, actualizar y consultar datos de la base de datos y otras funciones, los profesores y alumnos podrán consultar su horario.

Primera fase, el administrador deberá importar datos de Excel a la base de datos mediante el uso de la herramienta phpMyAdmin o desde la consola de MySQL. Segunda fase, generar un archivo de entrada en formato texto al calendarizador. Este archivo contendrá información necesaria como asignación de materias a profesores y alumnos, características de cada salón y de cada materia. Tercera fase, importar a la base de datos el archivo de salida del calendarizador. Este archivo contendrá una solución de la asignación de horarios para profesores y alumnos de la Facultad de Ciencias Químicas e Ingeniería (FCQeI). El calendarizador es un software que realizará la asignación de horarios de cada profesor y de cada estudiante, como también los horarios de los salones. El calendarizador no es parte del alcance del proyecto. En este proyecto también se muestra los horarios de profesores y alumnos a través de la Web. Además de un diseño de la base de datos con la finalidad de que tenga más seguridad, esto para evitar mal uso de la información.

## **CAPÍTULO I INTRODUCCIÓN**

### **1.1. Antecedentes del proyecto**

En la Facultad de Ciencias Químicas e Ingeniería (FCQeI) de la UAEM se realiza la creación de horarios. Para cada semestre los horarios se elaboran en conjunto con los jefes de carrera y también con el jefe de administración.

La tarea de realizar las asignaciones de horarios puede ser una tarea muy compleja considerando que se deben cumplir un gran número de restricciones para que las asignaciones sean válidas y eficientes. Se deben tomar en cuenta los datos de los profesores, alumnos, periodos, tamaño de los salones, seriación de materias, características de cada salón, número de días que se imparte clase y otros.

Con la información obtenida, el administrador puede realizar la calendarización sin embargo, hacer esto de forma manual representa una carga de trabajo muy laboriosa para los administrativos debido al proceso de asignación de salones por grupo. Además el proceso se realiza en un tiempo considerable debido a que en cada inicio de semestre se elaboran horarios nuevos para cada grupo.

Algo muy importante de tomar en cuenta es que el estudiante decide las materias que tomará. Así mismo algunos profesores deciden su horario.

### **1.2. Planteamiento del problema**

Se requiere la creación de una base de datos con una estructura segura para evitar inconsistencias en los datos ya que deberán ser administrados hacia un software de un calendarizador el cual de acuerdo a las restricciones del problema y a los datos de entrada dados por la base de datos, realizará la calendarización de horarios.

Este software es un algoritmo que dará una solución para generar los horarios de profesores y alumnos donde los profesores titulares puedan elegir su horario y los

alumnos sólo podrán consultar su respectivo horario de acuerdo a las materias elegidas por ellos mismos.

La base de datos tendrá la información necesaria que es requerida por el calendarizador. A continuación se enumera dicha información:

- 1.- Número de salones.
- 2.- Aforo por cada salón.
- 3.- Currículo de carreras.
- 4.- Matrícula de los alumnos.
- 5.- Materias que imparten los profesores.
- 6.- facilidades que requieren los salones.

### **1.3. Objetivo general**

Crear un sistema que permita llevar la administración de horarios de forma simple y segura a través de un modelo de tres capas, cliente-servidor Web - servidor de BD, estableciendo consultas en línea por parte de profesores y alumnos.

### **1.4. Justificación**

El desarrollo de este proyecto permitirá añadir mejoras al sistema de generación de horarios de profesores y alumnos para la FCQel de la UAEM. Una mejora importante es la incorporación de técnicas de seguridad en la base de datos evitará violaciones y pérdida de información. Asimismo el sistema contará con un diseño gráfico para que la interfaz sea agradable y fácil de manipular.

### **1.5. Alcances y limitaciones**

Alcances:

1. Desarrollar de un sistema administrativo de horarios para la FCQel, mediante el diseño de una arquitectura de tres capas: cliente – servidor Web – servidor de base de datos.
2. Diseñar una interfaz gráfica mediante un estándar Web (JSP), la cual se permitirá el acceso al sistema a través de un navegador. A través de esta interfaz, los usuarios podrán realizar consultas a la base de datos, utilizando una conectividad JDBC a una base de datos en MySQL.
3. Desarrollar un módulo que genere un archivo en formato de texto con extensión txt, en el cual se guarde la asignación de materias a profesores y alumnos, características de cada salón y de cada materia.
4. Desarrollar un módulo que lea la solución obtenida del algoritmo de calendarización de horarios para darle un formato especificado por la FCQel para su consulta en línea por parte de los alumnos y profesores.

Limitaciones:

1. Las materias tomadas por los alumnos e impartidas por los profesores son datos conocidos que se obtienen de la toma de materias realizada por los alumnos y profesores de acuerdo al concurso de meritos realizado por la facultad.

## **1.6. Organización de la tesina**

En esta sección se hace una breve descripción del contenido de los siguientes capítulos que forman parte de esta tesina.

**CAPÍTULO 1. INTRODUCCIÓN.** En este capítulo se presentan los antecedentes del proyecto, el planteamiento del problema, el objetivo general, y la justificación del mismo, y los alcances y las limitaciones del sistema.

**CAPÍTULO 2. ESPECIFICACIÓN DE REQUERIMIENTOS.** En este capítulo se presenta el análisis de requerimientos del sistema, así como los requerimientos funcionales y los requerimientos no funcionales del sistema.

**CAPÍTULO 3. DISEÑO DEL SISTEMA.** En este capítulo se especifica el diseño de la solución, se presentan también los casos de uso, el diagrama de secuencia, el diagrama de clases y la arquitectura de tres capas. También se muestra el diseño del modelo de la base de datos.

**CAPÍTULO 4. IMPLEMENTACIÓN.** En este capítulo se describirán las API's y herramientas utilizadas para el desarrollo del sistema, Así como las partes de programación más relevantes del sistema.

**CAPÍTULO 5. PRUEBAS.** En este capítulo se mostrarán evidencias gráficas de que el sistema cumplió con el objetivo y los requerimientos planteados inicialmente.

**CAPÍTULO 6. CONCLUSIONES.** En este capítulo se presentarán las lecciones aprendidas durante el desarrollo del sistema, las potencialidades de uso de la tecnología empleada para otros casos prácticos. Así como trabajos a futuro.

**ANEXOS.** Los anexos agregados hacen referencia al glosario de términos utilizados en la tesina.

**BIBLIOGRAFÍA.** Utilización de fuentes bibliográficas de libros utilizados para definir términos y conocimientos determinados. También se hace referencia a las páginas Web visitadas.

## **CAPÍTULO II ESPECIFICACIÓN DE REQUERIMIENTOS**

### **2.1. Perspectiva del producto**

#### **a) Interfaces de usuario**

- Las interfaces del usuario serán diseñadas mediante JSP (JavaServer Pages).Lo cual permitirá el acceso al sistema a través de un explorador como Internet Explorer, Mozilla y otros.
- El sistema podrá ser accesado en varios sistemas operativos que cuenten con un navegador Web. Estos sistemas pueden ser Microsoft Windows XP, Microsoft Windows Vista, Linux y otros.

#### **b) Requerimientos de adaptación del sitio**

- La base de datos debe ser desarrollada en el manejador de base de datos MYSQL.
- Las interfaces serán montadas en el servidor Apache Tomcat 5.5.

- El desarrollo de los servlets será mediante la utilización del lenguaje de programación java.
- Los servlets es el medio de comunicación entre la base de datos y la interfaz del sistema estos están montados en el servidor Apache Tomcat y son los que dan el control de entrada a la base de datos.

## 2.2 Funciones del producto.

- Se presentará una pantalla inicial que servirá de entrada a la aplicación, desde la cual se validen los usuarios que pueden acceder al sistema.
- La pantalla de horarios permitirá a los profesores y alumnos consultar su horario correspondiente al curso.
- El administrador del sistema podrá acceder a toda la información de profesores y alumnos. También podrá consultar el horario de salones para poder conocer la disponibilidad en un día o periodo determinado, considerando que existe la posibilidad de que un profesor solicite un cambio de horario en su materia.
- La pantalla de profesores titulares permitirá la selección de su horario a través del administrador del sistema.
- La base de datos almacena datos en un archivo de texto con extensión txt (archivo1.txt). Estos datos son los necesarios para que el Software de calendarización trabaje.
- El calendarizador es un Software externo que está ligado al sistema desarrollado en el presente trabajo. El calendarizador genera la calendarización de horarios de cada estudiante y de cada profesor .El resultado se guarda en un archivo (archivo2.txt).
- El archivo de texto definido como archivo2.txt generado por el algoritmo de calendarización será leído por el sistema administrador de horarios. Este archivo contendrá una solución para la calendarización de horarios de la FCQel.

### 2.3. Análisis de requerimientos

En la mayoría de las escuelas la asignación de horarios se hace de forma manual, de esta manera es complicado y laborioso para los administradores ya que se tienen que cumplir ciertas restricciones. Algunas de estas restricciones son:

- Un estudiante no debe tener clases en la última hora del día.
- Un estudiante no debe tener más de tres clases seguidas.
- Un estudiante no debe tener una sola clase al día.
- Un profesor no debe dar más de tres clases seguidas.
- Un profesor no debe dar una sola clase al día.
- Un estudiante no atenderá más de un evento a la vez.
- El salón tendrá el aforo suficiente para atender a todos los estudiantes de la clase.
- El salón debe satisfacer todas las necesidades requeridas por la clase del profesor.
- Un salón solo puede tener una clase (evento) a la vez.
- Todos y cada uno de los eventos deberán ser programados.

Las primeras cuatro restricciones se consideran como restricciones suaves debido a que pueden ser violadas. Para tener una muy buena solución al problema de calendarizador de horarios se requiere que el número de violaciones en las restricciones suaves sea el mínimo. Las restricciones restantes se consideran como restricciones duras, lo que significa que todas estas restricciones se deben cumplir para considerar que se tiene una solución al problema.

Otro aspecto importante para el desarrollo del sistema es la seguridad de la base de datos. Es un aspecto crítico que se debe considerar para evitar accesos no deseados a la información por partes de los usuarios del sistema.



En la Figura 2.1 muestra los pasos de cómo se obtiene la información de entrada al calendarizador.

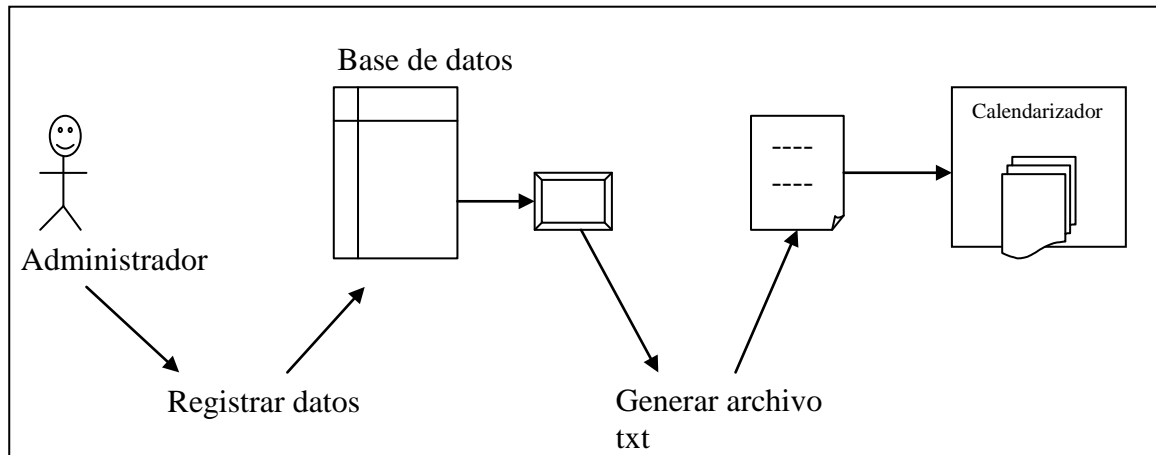


Figura 2.1. Pasos para generar el archivo de entrada al calendarizador

Explicación de la figura 2.1:

El administrador registra los datos, estos datos se guardan en la base de datos después el administrador genera un archivo de texto, este archivo es enviado al calendarizador, el calendarizador envía otro archivo con la asignación de horarios, este archivo es importado a la BD.

## 2.4. Requerimientos funcionales

Los siguientes puntos describen de forma detallada los requerimientos funcionales que definen las acciones que debe realizar el sistema.

1. Registrar datos. El administrador del sistema debe registrar los datos necesarios para la asignación de horarios de la FCQel.
2. Almacenar información. La información registrada será guardada en la base de datos.
3. Generar archivo. El administrador del sistema debe generar un archivo txt con la representación simbólica de los datos para que el calendarizador trabaje.

4. Guardar los datos generados por el calendarizador. La solución obtenida del algoritmo de calendarización de horarios es guardada en la base de datos.
5. Consultar información. Los usuarios podrán consultar en línea su horario correspondiente al curso.

## **2.5. Requerimientos no funcionales**

Los siguientes puntos describen los requerimientos no funcionales del sistema. Es decir, estos requerimientos no forman parte de la funcionalidad principal del sistema de tres capas.

- Entorno de desarrollo. La aplicación se ejecutará en un sistema operativo Windows con aplicaciones Java- Tomcat-MySQL.
- Portabilidad. El sistema podrá llevarse a diferentes sistemas operativos como Microsoft Windows XP, Microsoft Windows Vista y Linux.
- Seguridad. Este sistema tendrá los permisos de acceso necesarios. Se crearán servlets para la seguridad de la información registrada en la base de datos. Existirán cuatro tipos de usuarios con diferentes privilegios: Administrador, profesores titulares, profesores no titulares y alumnos.
- Interfaz. Se presentará una interfaz sencilla y fácil de utilizar.

## **CÁPÍTULO III DISEÑO DEL SISTEMA**

En esta sección se especifica el diseño de la solución, se hace uso de la notación de UML (Unified Modeling Language) para representar los diagramas de caso de usos, los diagramas de secuencia, los diagramas de clases y el diseño de la arquitectura de tres capas. Así como también el diseño del modelo de la base de datos.

### **3.1. Identificación de actores**

El sistema cuenta con los siguientes 5 actores.

- Alumno: El alumno podrá consultar su horario del curso.

- Administrador: El administrador podrá modificar, actualizar y consultar toda la información necesaria para la asignación de horarios. Matrícula de alumnos, profesores y materias, aforo de salones, características de cada salón.
- Profesor titular: El profesor titular podrá seleccionar y consultar su horario del curso.
- Profesor no titular: El profesor podrá consultar su horario del curso.
- Profesor temporal: El profesor podrá consultar su horario del curso.

### 3.2. Diagrama de caso de usos

En la figura 3.1 se muestra el diagrama general de caso de usos que describe lo que hace el usuario dentro del sistema.

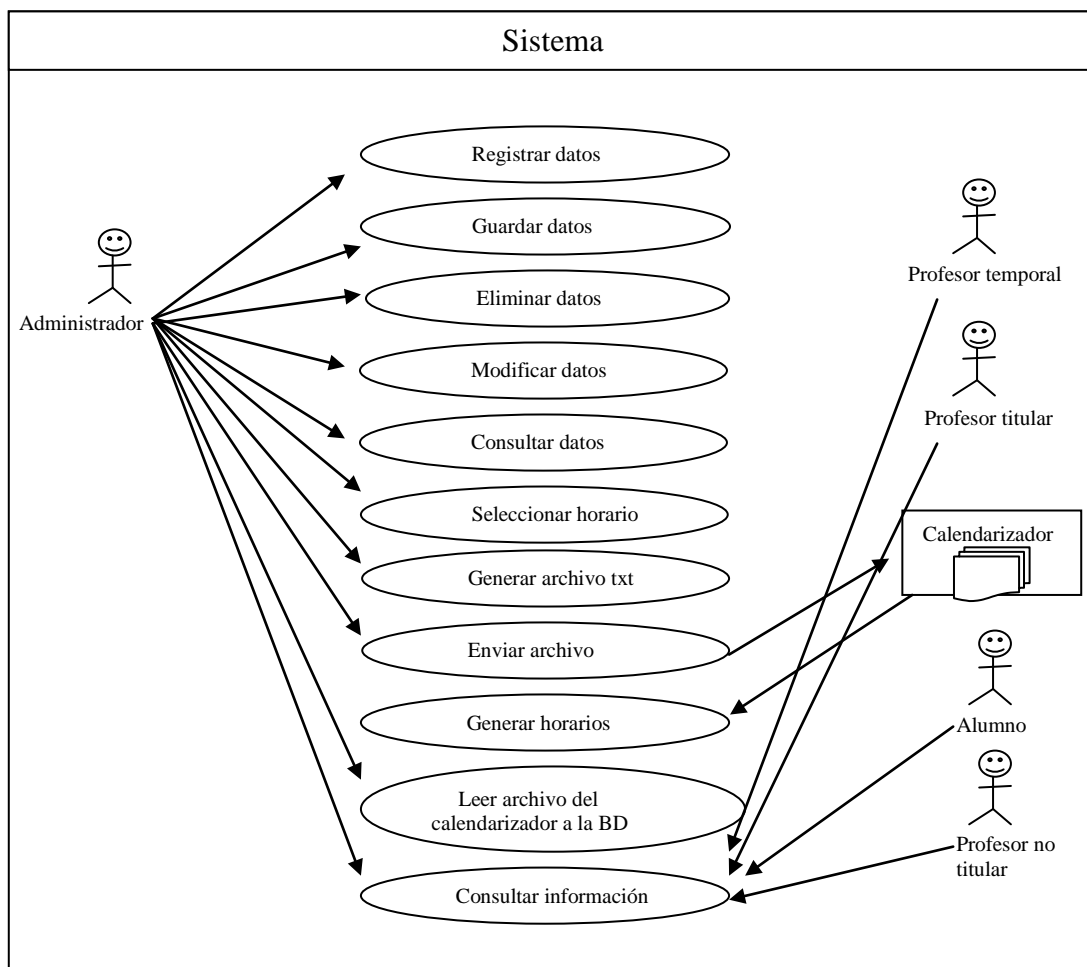


Figura 3.2. Diagrama general de caso de usos del sistema

### 3.3. Diagrama de secuencia

El diagrama de secuencia muestra las interacciones entre los actores y las operaciones. En la figura 3.2 se muestra el diagrama de secuencia general del sistema.

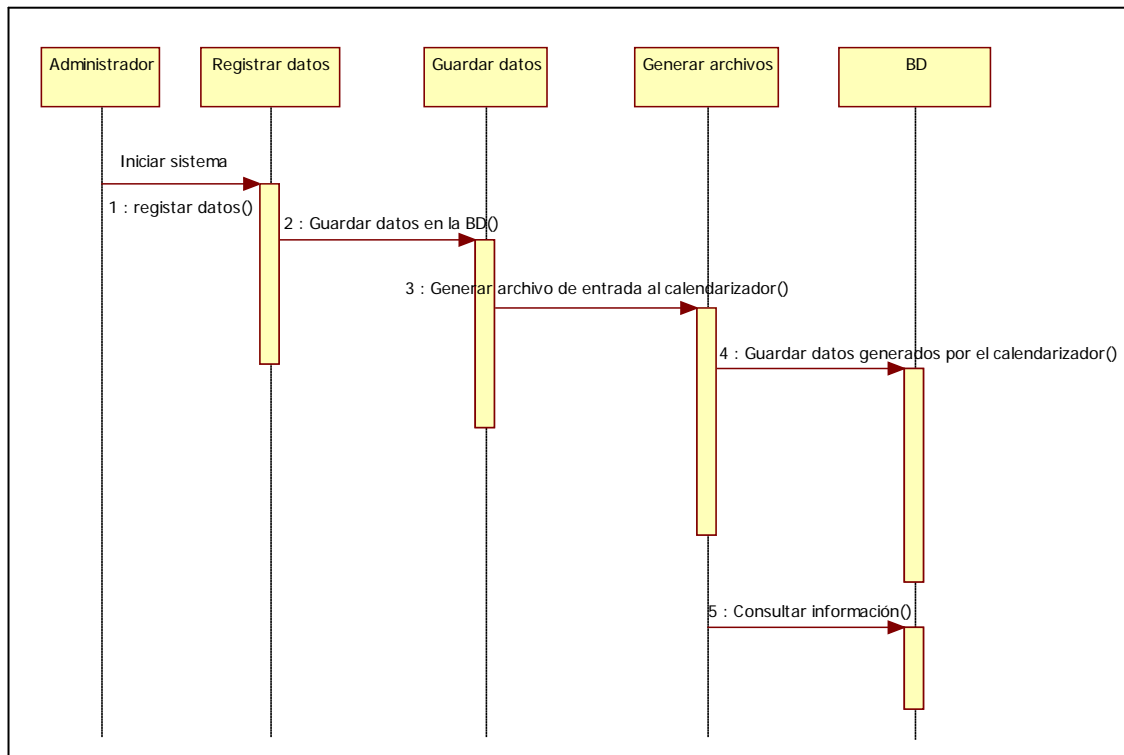


Figura 3.3. Diagrama general de secuencia

### 3.4. Diagrama de clases

Este diagrama de clases muestra las diferentes clases que componen el sistema y cómo se relacionan unas con otras. Se incluyen también los nombres de los métodos.

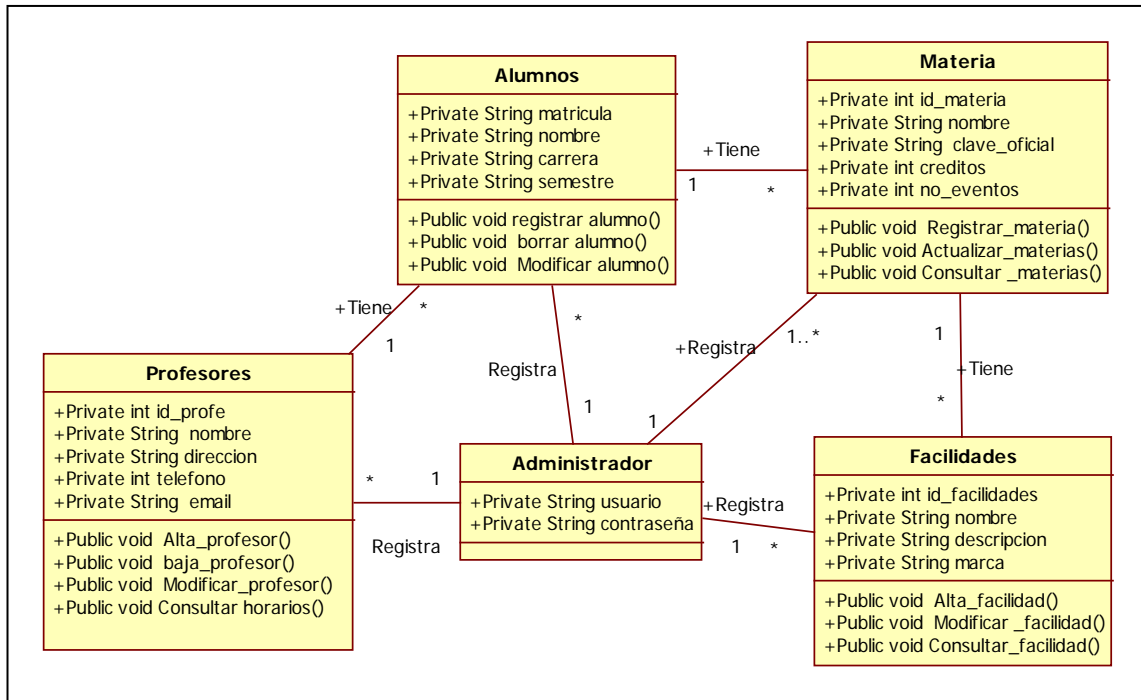


Figura 3.4. Diagrama de clases de servlet's

### 3.5. Arquitectura de tres capas

La arquitectura de tres capas nos ayuda a entender la comunicación entre componentes. Los clientes se comunican con el servidor Web mediante un protocolo de comunicación específico según el lenguaje de la aplicación, a su vez, el servidor Web se comunica con el servidor de base de datos mediante un controlador específico según el DBMS (Sistema de gestión de bases de datos) utilizado. En la figura 3.4 se muestra la arquitectura de tres capas.

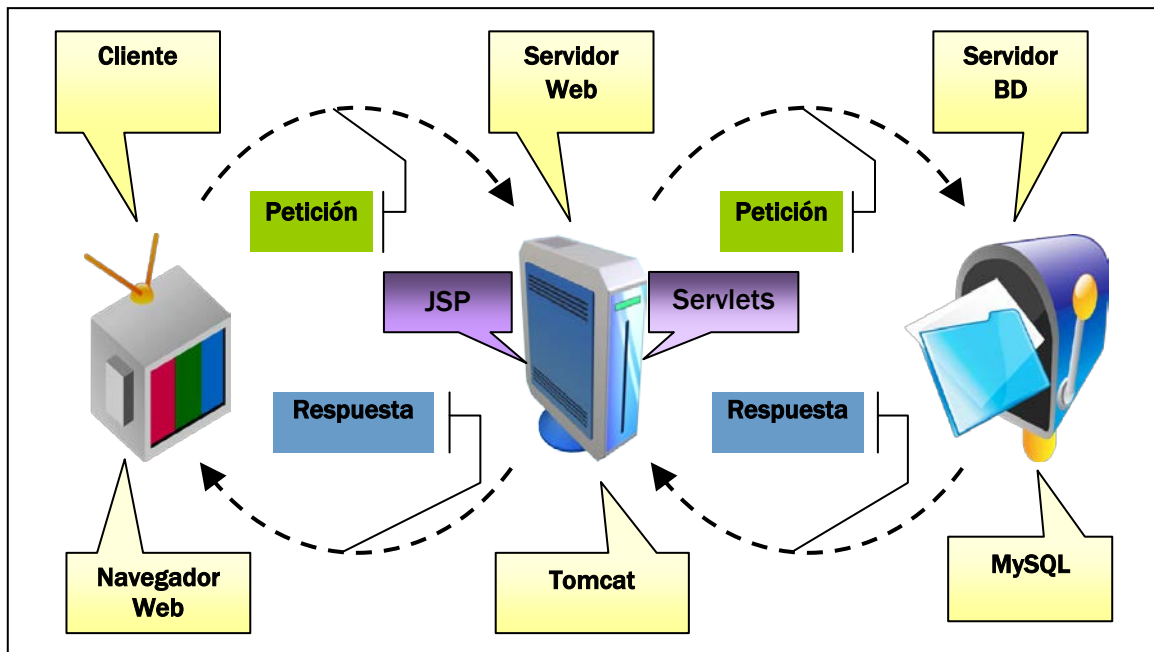


Figura 3.5. Arquitectura de tres capas: cliente, servidor Web y servidor BD

### 3.6. Diseño de la base de datos

En esta sección se presenta el diseño de la base de datos, el cual se desarrollo mediante 4 fases que son las siguientes:

- Análisis de requerimientos
- Diseño conceptual
- Diseño lógico
- Diseño físico

### 3.6.1. Análisis de requerimientos

El diseño de esta base de datos es para mantener un buen control y almacenamiento de datos necesarios para el algoritmo de calendarizador de horarios. Así como evitar intromisiones en la información.

El diseño de la base de datos debe considerar las siguientes restricciones necesarias para el control y almacenamiento de los datos:

Profesores:

- Un profesor puede impartir una o varias materias.
- Un profesor solo será asignado a un solo salón a la vez.
- Un profesor define cero, una o más facilidades por materia.
- Un profesor define su horario para materias titulares.

Materias:

- Una materia puede tener uno o varios eventos.
- Una materia puede impartirse en uno o varios salones.
- Un evento de una materia puede impartirse solo en un salón.

Salones:

- Un salón tendrá cero, una o muchas facilidades.
- Un salón puede tener cero, una o más materias.
- Un salón se le asigna en un periodo un solo evento en el mismo día.
- El salón tendrá el aforo suficiente para atender a todos los estudiantes.
- Un salón no debe ser asignada en horas en las cuales no está disponible.

Alumno:

- Un alumno asiste a uno o muchos eventos.
- Un alumno tiene una o muchas materias.

### 3.6.2. Diseño conceptual

#### Modelo Entidad-Relación

El modelo entidad-relación es el modelo conceptual más utilizado para el diseño conceptual de bases de datos. El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas. Las Entidades se identifican de acuerdo al análisis de requerimientos y a la necesidad del sistema. Una entidad es un objeto que existe y se distingue de otros objetos [10].

#### Tipos de relaciones del modelo entidad-relación

—————⊥ 1 aN (de uno a más)  
—————⊥ 1 aN (de uno a más)









### 3.6.3. Diseño lógico

El diseño lógico se acerca más a la implementación en el gestor de la base de datos que va ser utilizado DBMS. El objetivo es transformar el modelo entidad relación en tablas que podrán ser implementadas en un sistema manejador de base de datos (MySQL 5.0). En este caso se utilizó el modelo ELKA (Entity- Link -Key -Attribute) que es el más apropiado ya que elimina ciertas anomalías debidas a la redundancia [10].

Tipos de relaciones del modelo ELKA:

- a)  1 a 1 (de uno a uno)
- b)  0 a N (de cero a muchos)
- c)  1 a N (de uno a muchos)
- d)  N a M (de muchos a muchos)

## Modelo ELKA

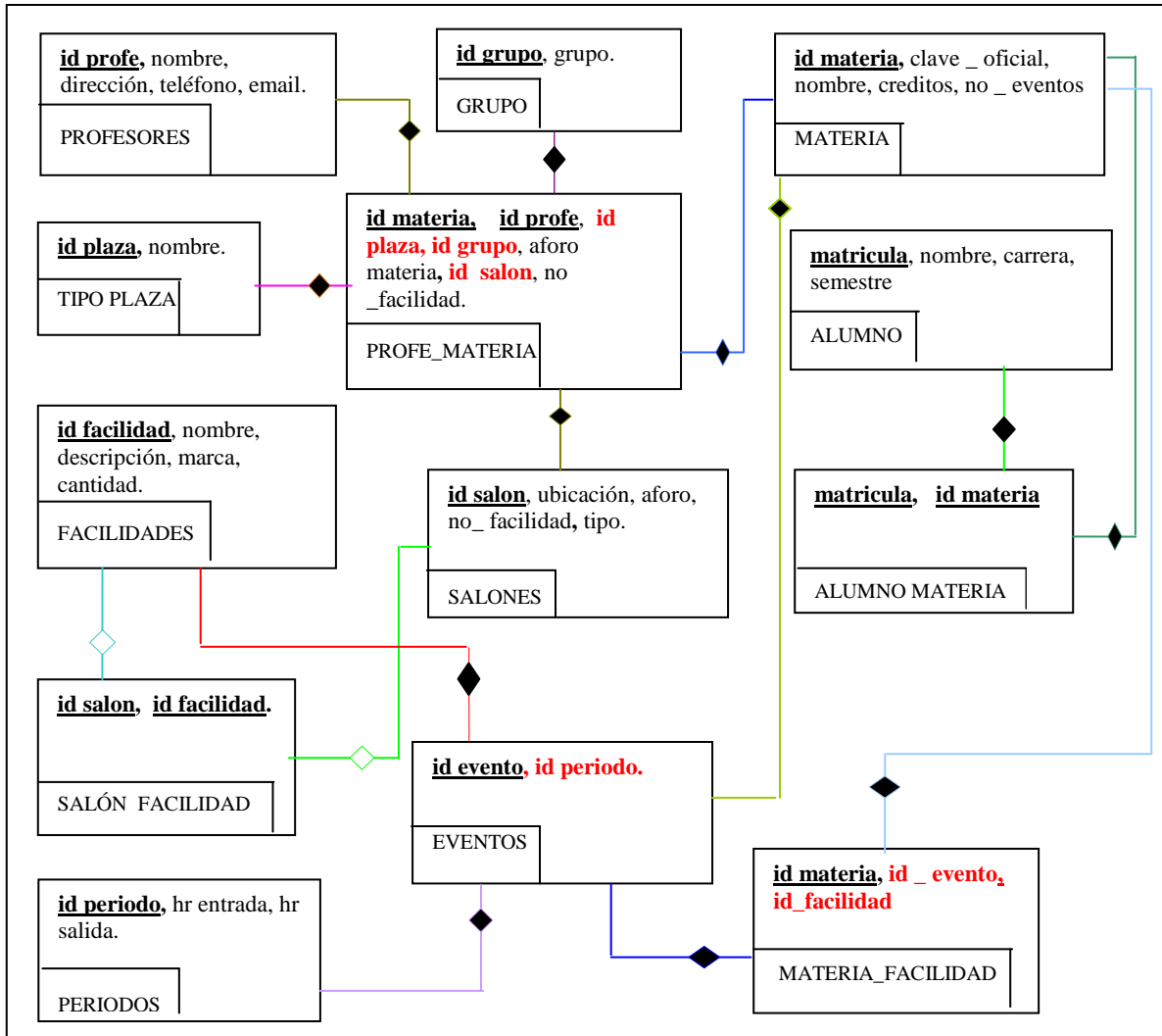


Figura 3.6. Modelo ELKA

### Descripción de la figura 3.6:

La relación Profesores hereda el atributo id\_profe a la entidad Profe\_materia se considera una restricción fuerte. La relación tipo\_plaza, grupo, materia, salones hereda el atributo Id\_plaza, Id\_grupo, Id\_salon, Id\_materia a la entidad Profe\_materia se consideran restricciones fuertes. La relación materia, alumnos hereda el atributo matricula, id\_materia a la entidad Alumno\_materia se consideran restricciones fuertes. La relación materia, evento, hereda el atributo id\_materia, id\_evento a la entidad Materia\_facilidad se consideran restricciones fuertes. La relación facilidades, salones hereda el atributo Id\_salon, Id\_facilidad a la entidad Salon\_facilidad se consideran restricciones suaves. La relación periodo, facilidad, materia\_facilidad hereda el atributo Id\_evento, id\_periodo a la entidad eventos consideran restricciones fuertes.

#### 3.6.4. Diseño físico

El diseño físico es parte del esquema lógico de bases de datos obtenido de la fase anterior. El esquema físico de una base de datos es una descripción de la implementación de una base de datos, describiendo las estructuras de almacenamiento y los métodos de acceso a esos datos [10].

Tabla 3.1 Materias

MATERIAS	
NOMBRE DEL CAMPO	TIPO DE CAMPO
id_materia	Int
Nombre	Varchar
Créditos	Int
Clave_oficial	Varchar
No_eventos	Int

Tabla 3. 2 Profesores

PROFESORES	
NOMBRE DEL CAMPO	TIPO DE CAMPO
id_profe	Varchar
Nombre	Varchar

Dirección	Varchar
Teléfono	Varchar
Email	Varchar

Tabla 3.3. Profe materia

PROFE MATERIA	
NOMBRE DEL CAMPO	TIPO DE CAMPO
id_profe	Varchar
id_materia	Int
id_grupo	Int
aforo_materia	Int
id_plaza	Int
no_facilidad	Int

Tabla 3. 4. Grupo

GRUPOS	
NOMBRE DEL CAMPO	TIPO DE CAMPO
Id_grupo	Int
Grupo	Varchar

Tabla 3.5.Facilidades

FACILIDADES	
NOMBRE DEL CAMPO	TIPO DE CAMPO
Id_facilidad	Int
Nombre	Varchar
Descripción	Varchar
Marca	Varchar
Cantidad	Int

Tabla 3.6. Tipo plaza

TIPO PLAZA	
NOMBRE DEL CAMPO	TIPO DE CAMPO
id_plaza	int
Nombre	varchar

Tabla 3. 7. Salones

SALONES	
NOMBRE DEL CAMPO	TIPO DE CAMPO
id_salon	int
Ubicación	varchar
Aforo	int
no_facilidades	int

Tabla 3.8. Eventos

EVENTOS	
NOMBRE DEL CAMPO	TIPO DE CAMPO
id_evento	int
id_periodo	int

Tabla 3.9. Periodos

PERIODOS	
NOMBRE DEL CAMPO	TIPO DE CAMPO
id_periodo	int
Hr_entrada	varchar
Hr_salida	varchar

Tabla 3.10. Salon facilidad

SALÓN FACILIDAD	
NOMBRE DEL CAMPO	TIPO DE CAMPO
id_salon	Int
id_facilidad	Int

Tabla 3.11. Alumno materia

ALUMNNO MATERIA	
NOMBRE DEL CAMPO	TIPO DE CAMPO
Matricula	varchar
id_materia	int

Tabla 3.12. Materia facilidad

MATERIA FACILIDAD	
NOMBRE DEL CAMPO	TIPO DE CAMPO
id_materia	int
id_evento	int
id_facilidad	int

Tabla 3.13. Alumnos

ALUMNOS	
NOMBRE DEL CAMPO	TIPO DE CAMPO
Matricula	varchar
Nombre	varchar
Carrera	varchar
Semestre	Int

## CAPÍTULO IV IMPLEMENTACIÓN

### 4.1 Implementación de las interfaces del software

En esta sección se describen las API'S (ver anexo 1) y herramientas utilizadas para el desarrollo del sistema. También se describen las pantallas de la aplicación y las partes de programación más relevantes del mismo.

### 4.2. Interfaz de inicio al sistema

La figura 4.1 muestra una pantalla en la que un usuario administrador intenta acceder al sistema. Sin embargo, al no registrarse correctamente, escribiendo una contraseña valida, o un nombre de usuario valido, aparece un mensaje de error que indica que no es posible dicho acceso y que se requiere una cuenta valida para ingresar al sistema como se muestra en la figura 5.1.2. Este proceso se repite hasta que el usuario se registre correctamente, o bien se cancela por el usuario si este decide ya no ingresar al sistema.



Figura 4.1. Pantalla de inicio

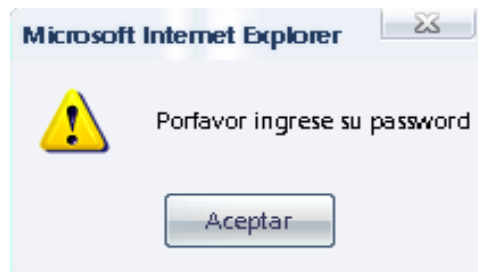


Figura4.1.2 Mensaje de error

Una vez que el usuario introduce una cuenta válida, figura 4.3, aparecerá un mensaje que indica al usuario que puede ingresar al sistema.

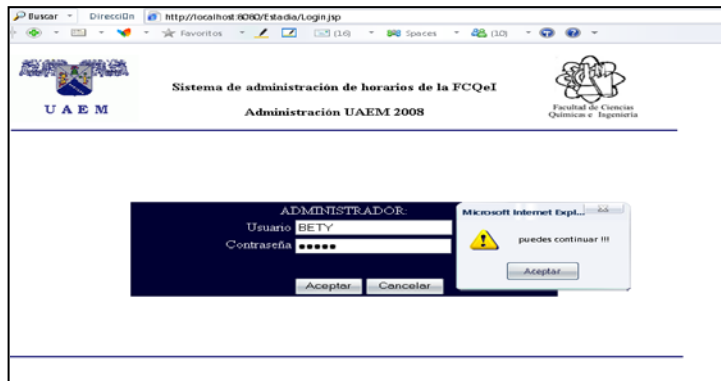


Figura 4.3 Validación de campos

### 4.3. Interfaz principal del sistema

Con la finalidad de hacer más práctica la manipulación de los datos, se presenta al usuario final la página principal del sistema figura 4.4. En esta sección el administrador podrá registrar, actualizar, eliminar y consultar datos de alumnos, profesores, materias y facilidades, la palabra “facilidades” se refiere a las herramientas que necesitan para dar una clase por ejemplo cañón, proyector de acetato, televisión, DVD y otros de la FCQeI de la UAEM.



Figura 4.4. Pantalla principal



### **Descripción del menú principal:**

a) **Profesores:** Contiene las opciones de registrar los datos del profesor como clave del profesor, que es su número de control, nombre del profesor entre otros datos personales. También contiene las opciones de actualización de los datos o bien eliminar los datos.

b) **Alumnos:** Contiene las opciones de registrar los datos del alumno como matrícula del alumno que es su número de control, nombre del alumno, carrera y semestre en el que asiste. Así como las opciones de actualización de los datos del alumno y eliminación de datos.

c) **Materias:** Contiene las opciones de registrar los datos de las materias como por ejemplo clave de la materia, nombre de la materia así como su número de créditos que tiene esa materia, número de eventos(ver anexo.1) con el que cuenta la materia. También cuenta con las opciones de actualizar los datos y consultarlos si es necesario.

d) **Facilidades:** Contiene las opciones de registrar los datos de las facilidades con las que cuenta la FQCel. La palabra “facilidades” se refiere a las herramientas que se necesitan para dar una clase y las que son requeridas en un salón por ejemplo cañón, proyector de acetatos, televisión, DVD y otros. Cabe mencionar que el administrador podrá modificar los datos o bien consultarlos si es necesario.

e) **Otras actividades:** Contiene opciones de seleccionar horario de profesores titulares. También la opción de exportar datos donde el administrador podrá generar los archivos necesarios para poder generar el archivo de texto que servirá como entrada al calendarizador. Otra opción es consultar salones donde el administrador podrá consultar el horario de salones para poder conocer la disponibilidad en un día o periodo determinado. Otra opción consultar horarios donde el administrador podrá consultar los horarios de profesores y alumnos.

#### 4.4. Interfaz Registrar datos

En la figura 4.5 la interfaz permite al administrador registrar algunos datos personales de los profesores que laboran en la FCQeI de la UAEM, algunos datos a registrar son los siguientes: clave del profesor, que es su número de control, nombre del profesor, así como su dirección, teléfono y email del profesor.

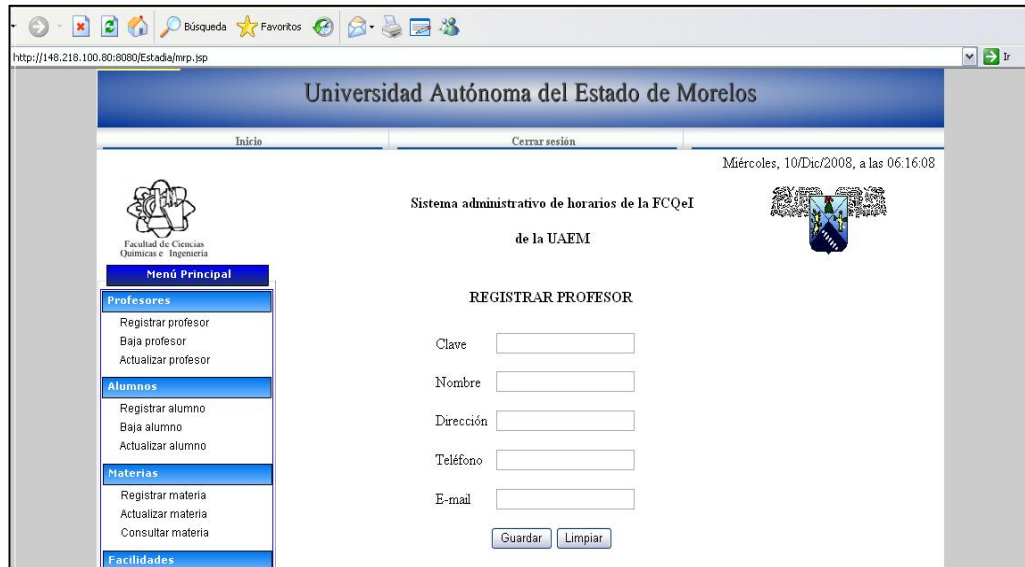


Figura 4.5. Pantalla de registro de datos

Si la información introducida por el usuario no es correcta, el sistema manda un mensaje de error indicando que no es posible guardar la información en la BD, por que algunos campos tienen un formato no valido, figura 4.6.

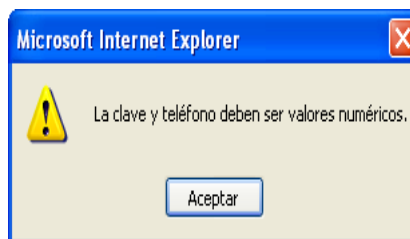


Figura 4.6. Mensaje de error enviado por el sistema

#### 4.5. Interfaz Eliminar datos

En la figura 4.7 se muestra la interfaz para que el administrador pueda eliminar los datos del profesor fue realizado con un pequeño menú desplegable donde muestra el número de control de los profesores que se encuentran registrados en la base de datos de la FCQel de la UAEM.

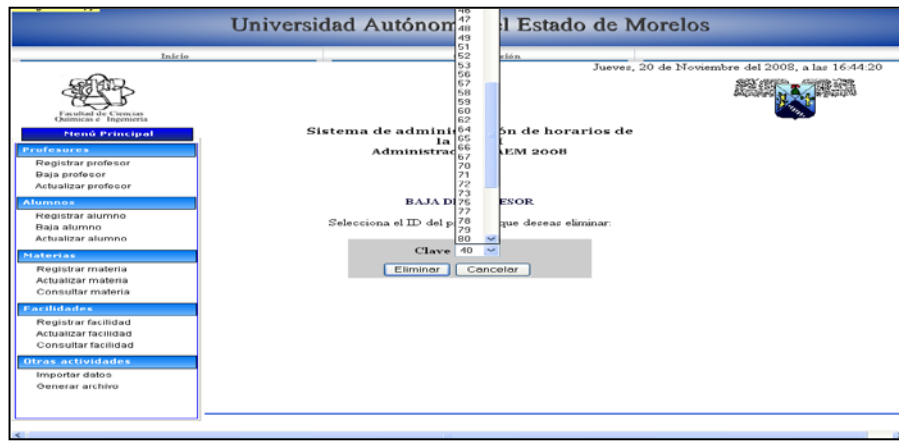


Figura 4.7 Pantalla para eliminar datos

#### 4.6. Interfaz Modificar datos

En esta sección el administrador, podrá modificar los datos personales del profesor que se encuentran registrados en la base de datos. También contiene un pequeño menú desplegable donde muestra la clave del profesor. Aquí el administrador puede seleccionar la clave del profesor del cual desea actualizar algunos datos. Como aparece en la figura 4.8.

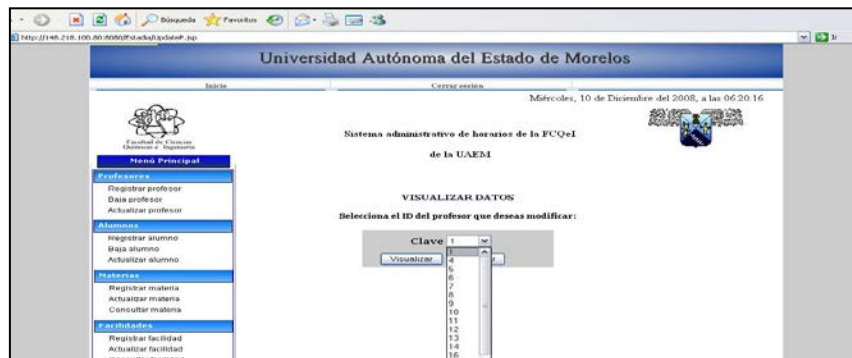


Figura 4.8. Pantalla para visualizar datos

Al pulsar el botón visualizar el sistema presenta esta pantalla que muestra los datos correspondientes al profesor para que puedan ser modificados como se muestra en la figura 4.9.

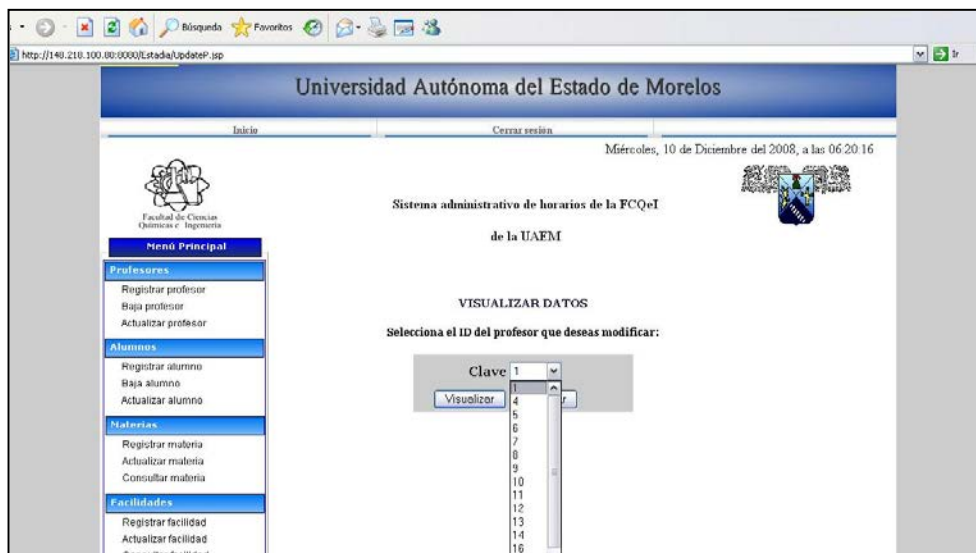


Figura 4.9. Pantalla para modificar datos

#### 4.7. Pantalla Generar archivo

En esta sección el administrador podrá generar un archivo de texto con la representación simbólica de los datos para que el algoritmo de calendarización trabaje correctamente, figura 4.10. El archivo generado es muy importante, ya que servirá para introducirlo posteriormente al sistema de asignación de horarios.

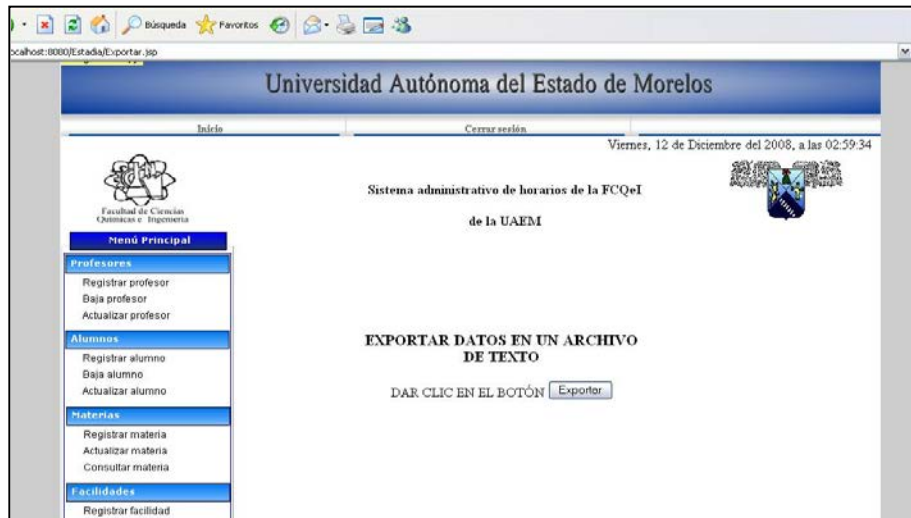


Figura 4.10. Pantalla para generar archivo de texto

#### 4.8. Presentación del archivo de entrada al calendarizador

A continuación se presenta una breve descripción del archivo de texto generado con el sistema desarrollado en este trabajo con la representación simbólica de los datos para que el calendarizador trabaje adecuadamente, como se muestra en la figura 4.11. El tipo de representación simbólica elegida se tomó del grupo de heurísticas Europeo para este tipo de problemas de calendarización, con el fin de tener una representación del problema de la FCQeI que pueda ser leída por cualquier investigador que trabaje ese tipo de problemas y pueda estudiarlo.

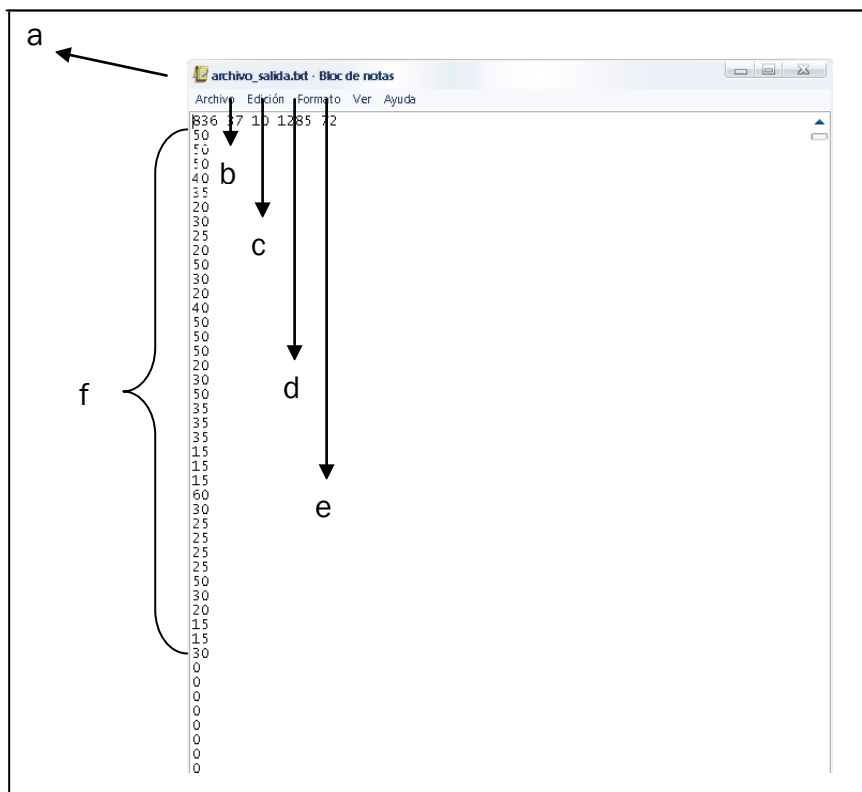


Figura 4.11. Archivo de entrada al calendarizador

**Descripción de la figura 4.11:**

- a) Este valor representa la suma del número de eventos de las materias.
- b) Este valor representa el número de salones con los que cuenta la FCQel.
- c) Este valor representa el número de facilidades es decir los aparatos con los que cuenta la FCQel por ejemplo: Cañón, DVD, Televisión etc.
- d) Este valor representa el número de alumnos inscritos en las 5 carreras con las que cuenta la FCQel.
- e) Este valor representa el número de profesores con los que cuenta la FCQel. Recordando que existen tres tipos de profesores, profesor titular, no titular y temporal.
- f) Estos valores representan la capacidad de cada salón por ejemplo el primer valor corresponde al primer salón que tiene un cupo de 50 alumnos y así sucesivamente para los demás salones.

De la línea 39 a la línea 1074299 se encuentra la multiplicación de la primera columna con cuarta columna del archivo. Esto quiere decir que cada alumno debe tener un total de 836 líneas con representación de ceros y unos para obtener la multiplicación, se utilizaron dos tablas de la base de datos: Alumno \_ materia. La tabla materia contiene todos los alumnos con sus respectivas materias y la tabla materia contiene todas las materias con su número de eventos. El uno representa que el alumno tomo el evento de la materia que se evalúa y el cero lo contrario.

La figura 4.12 muestra la representación de los eventos por cada estudiante como por ejemplo el primer estudiante de acuerdo con el registro de la base de datos sólo toma una materia. Cada materia contiene de 4 a 6 eventos dependiendo de la materia se hace la conversión de eventos a unos o ceros. Y así para los demás alumnos.

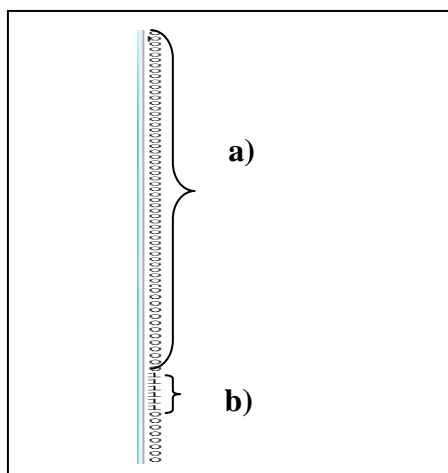


Figura 4.12. Representación de eventos por alumno

#### Descripción de la figura 4.12:

- a) Estos ceros significan que el alumno no toma esas materias.
- b) Estos unos significan que el alumno toma la materia.

#### 4.9. Conexión a la base de datos

Para crear una conexión desde java a la base de datos se utiliza el siguiente código

4.9.1:

```

//Para iniciar una conexión se debe de registrar el controlador
Class.forName("com.mysql.jdbc.Driver").newInstance();
//se hace la conexión. Aquí es necesario agregar el Server Host, el
nombre de la BD, el username y el password de MySQL.
Connection
conexion=DriverManager.getConnection("jdbc:mysql://localhost/bdname",
"usuario","password");
// aqui se obtiene la conexión
Statement st = conexion.createStatement ();
//se cierra la conexión
conexion.close ();

```

4.9.14. Código para hacer una conexión a la bd

#### 4.9.1. Código para hacer un registro

En las siguientes líneas de código 4.9.2 muestra como hacer un registro a la base de datos. El ejemplo completo se presenta en el anexo 3.

```

String query ="insert into alumnos values
("+matricula+", "+nombre+", "+carrera+", "+semestre+)";
// Ejecutar la consulta
st.executeUpdate (query);

```

Código. 4.9.2. Operación para registrar información a la base de datos

En el código 4.9.2 se muestra el registro de información a la base de datos, pero cabe mencionar que se puede hacer cualquier otra operación como las siguientes: (SELECT, INSERT, UPDATE o un DELETE).



#### 4.9.2. Código para eliminar un registro

En las siguientes líneas de código 4.9.3 muestra como hacer una eliminación de los datos a la base de datos. El ejemplo completo se presenta en el anexo 3.

```
String query ="DELETE from alumnos WHERE matricula="
'+matricula+' ";
// Ejecutar la consulta
st.executeUpdate (query);
```

Código.4.9.3. Operación para eliminar información de la bd

#### 4.9.3. Código para actualizar los datos

En las siguientes líneas de código 4.9.4 muestra como hacer la actualización de los datos a la base de datos. El ejemplo completo se presenta en el anexo 3.

```
String query ="UPDATE alumnos SET nombre=" '+nombre+' "
WHERE matricula=" '+matricula+' ";
// Ejecutar la consulta
st.executeUpdate (query);
```

Código.4.9.4. Operación para actualizar información de la BD

#### 4.10. Lectura del archivo

El fragmento de código 4.10.1 muestra como hacer la lectura de un archivo de texto.

```

// Aquí se especifica la lectura del archivo
try{
    String linea;
    int f;
    BufferedReader entrada= new BufferedReader(new FileReader
("C://archivos//materias.txt"));
    while (null!=(linea=entrada.readLine())){ //lectura de archivo linea
por linea para leer
        la basura
        nrenMat++;
    }//while
    mat = new String[nrenMat][2];
    f=0;
    entrada= new BufferedReader(new FileReader
("C://archivos//materias.txt"));
    while (null!=(linea=entrada.readLine())){ //lectura de archivo para
obtener los datos dentro del bucle
        mat[f] = linea.split(",");// se especifica que el archivo esta
separado por ','
        f++;
    }//while
    entrada.close(); //cerrar el archivo

```

Código.4.10.1. Lectura de un archivo de texto

#### 4.11. Almacenamiento de datos en un arreglo

El código 4.11.1 muestra cómo se llena el arreglo con la información leída por medio de un BufferedReader (Instrucción java que se utiliza para lectura de archivos txt).

```

mat = new String[nrenMat][2];
f=0;
entrada2= new BufferedReader(new FileReader
("C://archivos//materias.txt"));
while (null!=(linea=entrada2.readLine())){ //lectura de archivo para
obtenerlos datos dentro del bucle se obtendranlos datos
mat[f] = linea.split(",");//se especifica que el archivo esta separaso por “ , ”
f++;
}//while

```

Código.4.11.1. Almacenamiento de datos en un arreglo

#### 4.12. Las herramientas utilizadas para el desarrollo de esta aplicación son las siguientes:

- Plataforma de desarrollo J2EE (Java Platform, Enterprise Edition): Es una plataforma de programación para ejecutar y desarrollar software de aplicaciones en lenguajes de programación java.
- JavaServer Web Development Kit: El JSWDK sirve para la implementación de las especificaciones de Servlets y JSP. Es utilizado como servidor para hacer pruebas de los servlets y las páginas JSP [1].
- Apache Tomcat: Es la implementación de las especificaciones de servlets y JSP. Al igual que el JSWDK puede ser usado como un servidor que funciona para probar los servlets y las páginas JSP [1].
- MySQL: Es un sistema de gestión de la base de datos relacional.
- Macromedia Dreamweaver: Es un programa muy utilizado en el sector de diseño y aplicaciones Web.
- 

##### 4.12.1. Api's utilizadas

Para la implementación de los módulos se hace uso de las siguientes API'S:

- a) JDBC (Java Database Connectivity). Es un API para trabajar con bases de datos desde la plataforma java independientemente de la base de datos. En este caso se utilizó para la conexión a la base de datos MySQL.
- b) JDK (Java Development Kit). Es un software que provee herramientas de desarrollo para la creación de programas en java. Puede instalarse en una computadora local o en una unidad de red.

La tabla 4.1 muestra algunos métodos y objetos utilizados en la implementación de servlets.

Tabla 4.1. Métodos y objetos de los servlets

Métodos	Descripción	Ejemplo
doGet y doPost	Reciben como parámetros objetos de tipo HttpServletRequest y HttpServletResponse.	DoGet: realiza la operación GET de http. DoPost: No devuelve datos acerca del cliente, sino tan solo las cabeceras.
getParameter()	Devuelve una cadena que contiene el valor simple del parámetro especificado, o null si el parámetro no existe en la petición.	String nombre = (String) req.getParameter("txtNombre");
Request	Es la petición del cliente. Es normalmente una subclase de la clase HttpServletRequest.	protected void service (ServletRequest req, ServletResponse res)
Response	Es la página JSP de respuesta y es una subclase de HttpServletResponse	protected void service (ServletRequest req, ServletResponse res)

### 4.13. Implantación

#### HARDWARE:

- Computadora personal: DELL
- Procesador: 3.000 GHz
- RAM: 1024 MG
- Disco Duro: 120 GB

#### SOFTWARE:

- Sistema Operativo: Windows XP
- Servidor Web: Apache Tomcat 5.5
- Programación: Java (JDK 1.5.0\_16)
- Estándares Web: JSP
- Servidor de BD: MySQL 5.0

#### SOFTWARE extra:

- Macromedia Dreamweaver MX 2004
- 123 Flash Banner
- 123 Flash Menú
- MySQL Administrator
- MySQL Query Browser

Para el desarrollo de este sistema fue necesario instalar y configurar el software antes mencionado. Para consultar los manuales de instalación (ver anexo 4).

## CAPÍTULO V EVALUACIÓN

Al haber terminado la implementación del sistema es necesario realizar una etapa de pruebas, esta hace posible verificar el correcto funcionamiento del mismo. En este apartado se muestran las pruebas realizadas al sistema.

### 4.1. Pruebas de caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Con este método se determina cuáles son los casos de prueba a partir del código fuente del software y se utilizan las especificaciones para determinar el resultado esperado del caso [20].

El código 5.2 muestra una solución para una prueba de caja blanca en la cual, en un principio no guardaba correctamente en el archivo de texto debido que la consulta no se guardaba sólo se ejecutaba, por lo tanto, por medio de un bucle if para recuperar sólo un dato y el bucle while para recuperar varios datos de la tabla, para verificar la existencia de ese campo y así mismo recuperar ese dato guardándolo en una variable para después almacenarlo en el archivo, en el siguiente código 5.2 muestra la solución del problema.

```
ResultSet rs = stmt.executeQuery("select sum(no_eventos) from materias");
    if (rs.next()){
        salida = rs.getInt(1) + " ";
    }

    rs = stmt.executeQuery("select count(id_profe) from
profesores");
    if (rs.next()){
        salida += rs.getInt(1) + " ";
    }
    rs = stmt.executeQuery("select * from salones");
    while (rs.next()){
        salida += "\r\n"+rs.getInt("aforo");
    }
}
```

Código.5.2. Bucle if y while

Se realizarón varias pruebas para el almacenamiento de la información en la base de datos en ocasiones guardaba correctamente y otras veces no guardaba satisfactoriamente, para ello acudimos ha revisar el código del Servlet y el formulario de JSP, se comprobo que los parámetros enviados al servlet no coincidían con el nombre del campo de texto del formulario. En el siguiente código 5.3 muestra la solución al problema.

```
String matricula = request.getParameter ("Id_alumno");  
String nombre = request.getParameter ("Nombre");  
String carrera = request.getParameter ("Carrera");  
int semestre = Integer.parseInt (request.getParameter  
("Semestre"));
```

Código.5.3. Enviar información al servlet

## 4.2. Pruebas de caja negra

Con este método los casos de prueba y los resultados se determinan a partir de la especificación funcional del método de una clase. Es decir, la prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Una prueba de caja negra examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software [20].

### **Acceso al sistema**

En la figura 5.1 se muestra la primera pantalla principal en la cual el administrador tendrá acceso al sistema, para realizar sus actividades, así mismo muestra la dirección del servidor donde se encuentra el sistema como se observa en la siguiente figura.

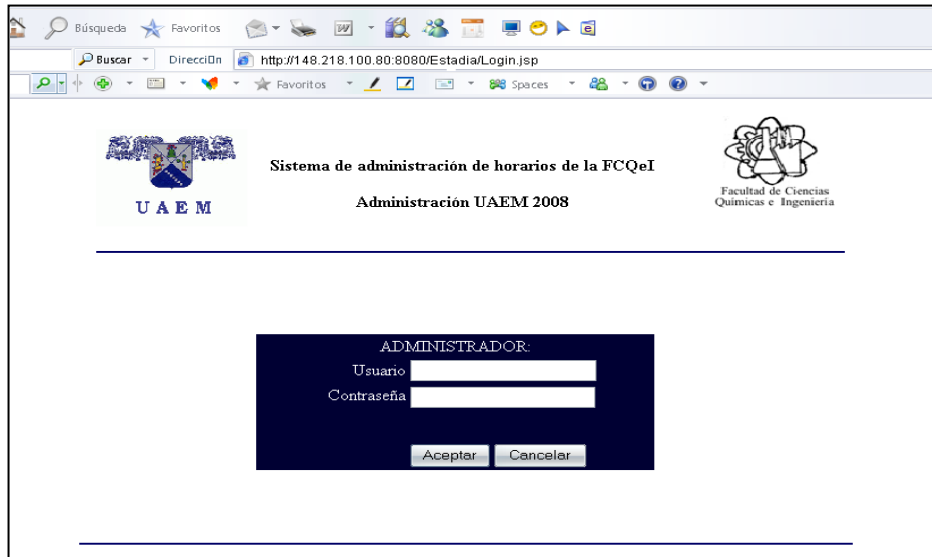


Figura 5.1. Pantalla de inicio

### Interfaz Registrar alumno

El administrador podrá crear, eliminar y modificar datos del alumno, validando el llenado correcto de todos los campos como se muestra en la figura 5.0.1.

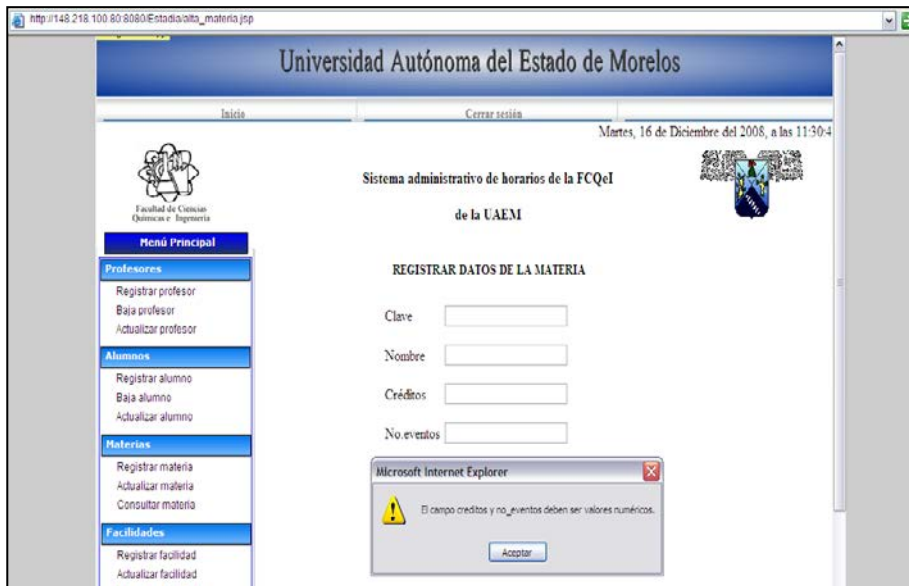


Figura 5.1.1. Verificación de campos



### 4.3. Pruebas de seguridad

Nivel de Seguridad del Sistema: Verificar los actores que pueden ingresar al sistema para que sean capaces de realizar las funciones del sistema.

Nombre de la prueba: Validación de usuarios

Objetivo: Mostar la validación de usuarios y contraseña de usuarios.

Procedimiento: 1.-Introduce los siguientes datos:

Usuario: ADMINISTRADOR

Contraseña: ADMIN

Resultados esperados:

En la figura 5.2 muestra los datos que el usuario ha introducido para poder ingresar al sistema.



Figura 5.2. Validación de usuarios

En la figura 5.3 se muestra la pantalla principal del sistema.



Figura 5.3. Página principal del sistema

Nombre de la prueba: Validación de matrículas de los alumnos que tienen acceso al sistema.

Objetivo: Mostar el horario de los alumnos.

Procedimiento:

1.-Introduce el siguiente dato:

Matricula: 006108521

En la figura 5.4 muestra los datos que el usuario ha introducido para poder ingresar al sistema.

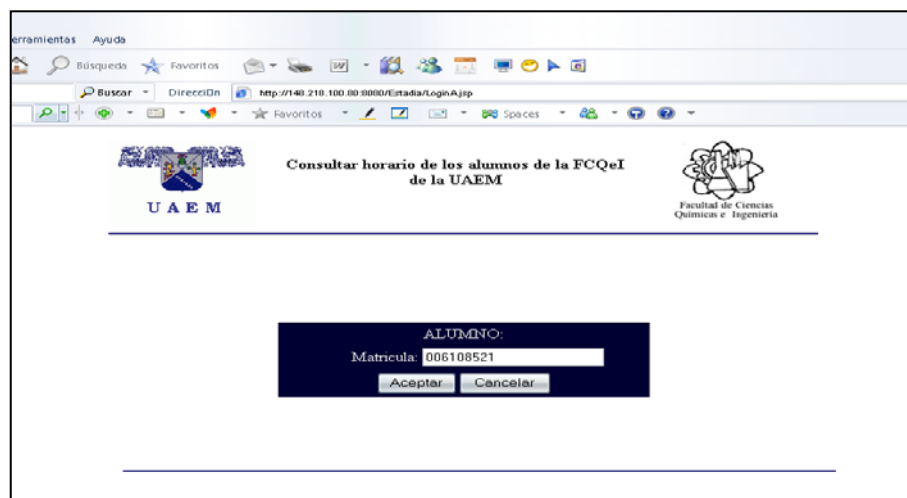
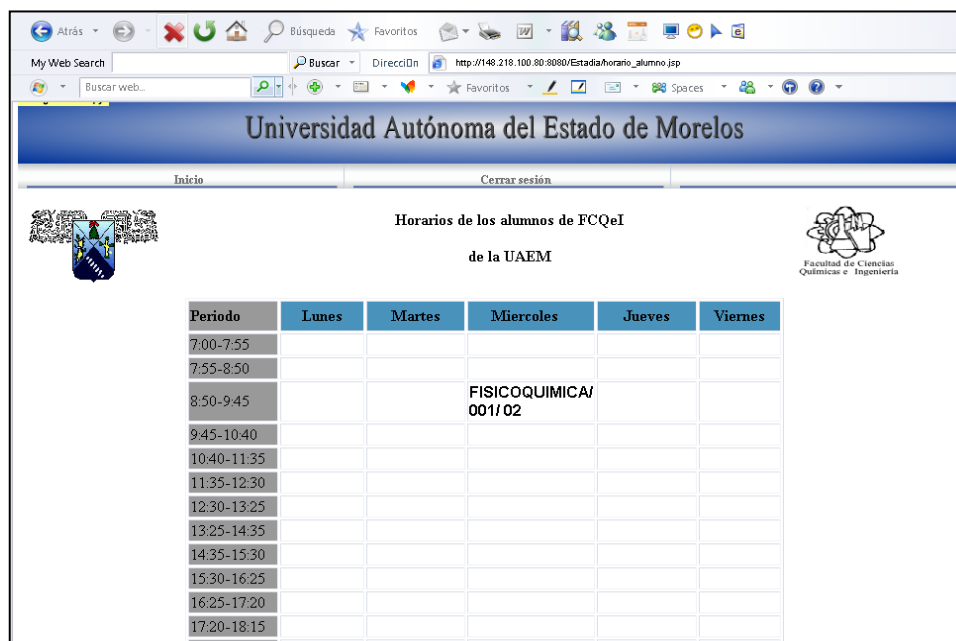


Figura 5.4. Validación de matrícula de alumnos

En la figura 5.5 se muestra el horario del alumno de la matricula introducida en la pantalla anterior.



Universidad Autónoma del Estado de Morelos

Inicio Cerrar sesión

Horarios de los alumnos de FCQeI  
de la UAEM

Facultad de Ciencias Químicas e Ingeniería

Periodo	Lunes	Martes	Miércoles	Jueves	Viernes
7:00-7:55					
7:55-8:50					
8:50-9:45			FISICOQUIMICA/ 001/02		
9:45-10:40					
10:40-11:35					
11:35-12:30					
12:30-13:25					
13:25-14:35					
14:35-15:30					
15:30-16:25					
16:25-17:20					
17:20-18:15					

Figura 5.5. Pantalla de horario de alumnos

Nombre de la prueba: Validación de claves de los profesores que tienen acceso al sistema.

Objetivo: Mostrar el horario de los profesores.

Procedimiento:

1.-Introduce el siguiente dato:

Clave: 001

En la figura 5.6 se muestra los datos que el usuario ha introducido para poder ingresar al sistema.

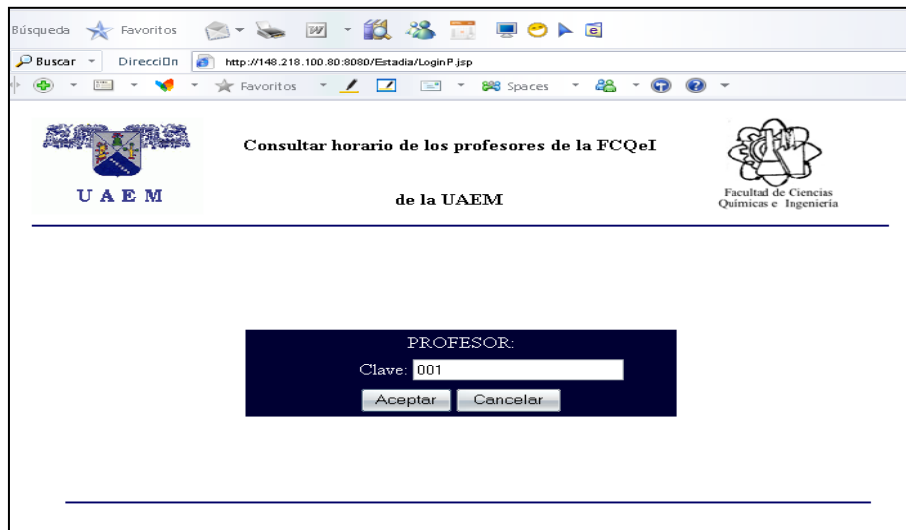


Figura 5.6. Validación de clave de profesores

En la figura 5.7 se muestra el horario del profesor de la clave introducida en la pantalla anterior.

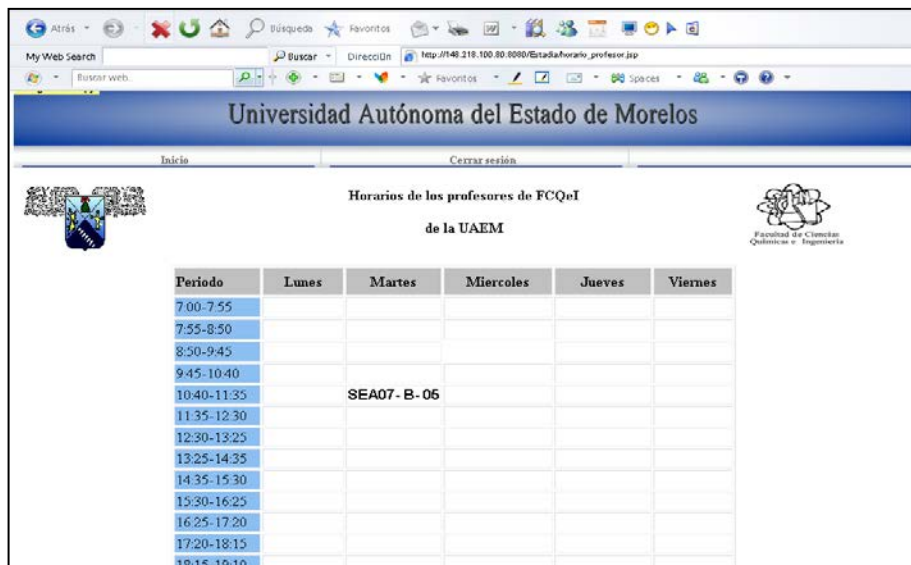


Figura 5.7. Pantalla de horario de profesores

## **CAPÍTULO VI CONCLUSIONES**

### **6.1. Conclusiones**

El problema tratado en esta investigación deriva de la necesidad de los administradores de contar con un sistema integral que les permita realizar el análisis de la asignación de horarios para profesores y alumnos de manera más rápida y sencilla. Este sistema es un modelo aproximado al existente en la Facultad de Ciencias Químicas e Ingeniería (FCQel) de la UAEM. Lo cual indica que actualmente sirve como una parte en el apoyo para la asignación de horarios de esa escuela. Este modelo se pretende que en futuro se pueda adaptarlo a otras escuelas con la finalidad de que para los administradores sea más práctica y rápida la asignación de horarios.

Para la elaboración de este proyecto se utilizó la tecnología de java server pages (JSP) y Servlet's para presentarlo en ambientes Web, MySQL para el desarrollo de la base de datos y el servidor de aplicaciones Apache Tomcat para montar los JSP's y Servlet's. Así mismo se desarrolló desde el diseño de la base de datos utilizando el modelo ELKA para relacionar las entidades.

Algunos de los retos enfrentados y logrados fueron adquirir conocimientos académicos, sobre tecnologías nuevas y herramientas, con los cuales no contábamos al iniciar el proyecto. Pero finalmente se logró el objetivo de concluir el desarrollo del proyecto exitosamente.

### **6.2. Aportaciones**

1.-Un Módulo donde alumnos y profesores pueden consultar su horario correspondiente a cada semestre.

2.- La creación de un módulo que ayude al administrador del sistema a generar un archivo de texto con la representación simbólica de la información de la base de datos que servirá como entrada al software de calendarización.

3.-Módulo de lectura de un archivo generado por el calendarizador el cual contiene una solución de asignación de horarios de profesores y alumnos de la FCQel.

4.- Desarrollo de un sistema para la administración de horarios donde el administrador podrá registrar, eliminar, actualizar y consultar todos los datos necesarios para la asignación de horarios de la FCQel de la UAEM. Para la creación de este sistema se utilizó una arquitectura de tres capas: cliente, servidor Web y servidor de base de datos.

### **6.3. Trabajos futuros**

1.- Realizar un Módulo donde los alumnos de las 5 carreras con las que cuenta la FQCel puedan hacer la toma de materias correspondientes a cada semestre. Esto se hace actualmente de forma independiente al sistema.

2.- Realizar un Módulo donde todos los profesores de la FQCel puedan seleccionar su horario conforme a sus necesidades, esto con la finalidad de hacer más rápido el proceso de asignaciones de horario. Actualmente esto se hace pero la entrada de la información la realiza el administrador.

3.- Trabajar con el calendarizador para adaptarlo a un modelo más real que represente a la asignación de horarios en la FCQel.

4.-Continuar con el sistema de administración de horarios y con el problema de calendarización de horarios para que en un futuro sea fácil de mapear a modelos de otras universidades.

## Anexo 1. Glosario

**Actor.** Es algo que es externo a la organización pero que interactúa con él.

**Algoritmo.** Es una lista bien definida, ordenada y finita de operaciones que permite hallar la solución a un problema [18].

**Apache Tomcat.** Es la implementación de las especificaciones de servlets y JSP. Al igual que el JSWDK puede ser usado como un pequeño servidor que funciona para probar los servlets y las páginas JSP o puede estar integrado dentro del servidor Web Apache [1].

**Application Programming Interface (API).** Es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. [13]

**Arreglo.** Un arreglo es una colección objetos numerados del mismo tipo, en donde cada variable o celda en el arreglo tiene un índice. Las celdas están enumeradas del 0 al N-1, donde N es el n° de celdas del arreglo es decir su capacidad [19].

Para inicializar un array existen 2 maneras:

- `int[] arreglo={100,200,302,40`
- `int[] arreglo=new int[4]`

**Atributo.** Se conoce como un campo de un registro.

**Calendarizador.** Es un algoritmo que dará una solución efectiva para generar los horarios de profesores y alumnos.

**Código.** Es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está descrito por completo su funcionamiento [16].

**Enlace.** Un enlace es la relación o forma en que se relacionan las entidades.

**Entidad.** Una entidad es cualquier objeto del cual se desean almacenar datos dentro de una base de datos.

**Evento.** Son definidos por el salón en donde se impartirán las clases aplicadas por los profesores, Se caracterizan por contener alumnos, periodos, profesores con una específica materia impartida por los profesores.

**HyperText Markup Language (HTML).** Es el lenguaje de marcado predominante para la construcción de páginas Web.

**Hypertext Transfer Protocol (HTTP).** Protocolo de nivel de aplicación usado extensivamente en Internet para el acceso a documentos [15].

**Java.** Es un lenguaje de programación que ofrece la potencia del diseño orientado a objetos con una sintaxis fácilmente accesible y un entorno robusto y agradable.

Java aporta a la Web una interactividad que se había buscado durante mucho tiempo entre usuario y aplicación [18].

**JDBC (Database Connectivity).** Es un API para trabajar con bases de datos desde Java, independientemente de la base de datos a la que accedemos.

**JSP (JavaServer Pages).** Es una tecnología Java que permite generar contenido dinámico para Web, en forma de documentos HTML, XML o de otro tipo [12].

**Llave.** Es un atributo o conjunto de atributos que permite identificar unívocamente a un elemento de una entidad.

**Modelo.** En diseño orientado a objetos, una representación del mundo real en unas abstracciones de software denominadas clases y la relación entre ellas.

**MySQL.** Es un sistema de gestión de base de datos relacional.

**Navegador Web.** Software que permite al usuario conectarse a un servidor de Web utilizando Hypertext Transfer Protocol (HTTP). Microsoft Internet Explorer, Netscape Navigator, HotJava de Sun, son populares navegadores de Web.

**Periodo.** Son las horas de las clases impartidas por los profesores por ejemplo: Dr. Marco Cruz Chávez imparte clase de Informática de 7:00 a.m. -9:00 a.m.

- Pone al alcance de cualquiera la utilización de aplicaciones que se pueden incluir directamente en páginas Web (aplicaciones denominadas applets).
- Proporciona un conjunto de clases potente y flexible.



**Servidor.** Es un programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes.

**Servlets.** Son programas que se ejecutan en un servidor Web, que fungen como una capa intermedia entre una petición proveniente de un navegador Web u otro cliente HTTP, y las bases de datos [1].

**Unified Modeling Language (UML).** Es un conjunto de herramientas, que permite modelar, analizar y diseñar sistemas orientados a objetos [14].

## Anexo2. Acrónimos

Acrónimo	Definición
----------	------------

API	Application Programming Interface (Interfaz de Programación de Aplicaciones)
DBMS	Sistema de gestión de bases de datos
ELKA	Entity Link Key Attribute
FCQeI	Facultad de Ciencias Químicas e Ingeniería
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol (protocolo de transferencia de hipertexto)
JDBC	Java DataBase Connectivity
JDK	Java Development Kit
JSP	JavaSeverPages
UAEM	Universidad Autónoma del Estado de Morelos
UML	Unified Modeling Language (Lenguaje Unificado de Modelado)

### Anexo 3. Código para hacer un registro, una eliminación y una actualización de los datos de la base de datos.

Código para hacer un registro en la base de datos

```
import javax.servlet.http.*;
import java.io.*;
import java.rmi.*;
import java.util.*;
import java.sql.*;

public class Alta_alumno extends HttpServlet{

    public void doPost(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException{
    PrintWriter web = response.getWriter();
    try{
        response.setContentType("text/html");
        HttpSession sesion = request.getSession(true);
        //se reciben los valores del formulario
        String matricula = request.getParameter("Matricula");
        String nombre = request.getParameter("Nombre");
        String carrera = request.getParameter("Carrera");
        int semestre = Integer.parseInt(request.getParameter("Semestre"));
        //paso 1 registro de drive
        web.println("Inicio <br>");
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        web.println("Driver registrado<br>");
        //paso 2 tenemos conexion
        Connection conexion =
        DriverManager.getConnection("jdbc:mysql://localhost/nameBD","usuario","password");
        web.println("<br>Conexion (sesion)lista");
        //paso 3 construimos el query
        String query = "insert into alumnos
        values("+matricula+", "+nombre+", "+carrera+", "+semestre+)";
        web.println("<br>"+query);
        //paso 4 obtenemos un statement
        Statement st = conexion.createStatement();
        //paso 5 ejecutamos el query
        st.executeUpdate(query);
        web.println("insert exitoso");
        //paso 6 Siempre hay que cerrar la coneccion
        st.close();
        conexion.close();
    }catch(SQLException error){
        System.out.print("ERROR SQL ->"+error);
    }
}
```

```

}catch(Exception error){
    System.out.print("ERROR GENERAL ->" + error);
    }

}
}

```

Código para eliminar un registro de la base de datos

```

import javax.servlet.http.*;
import java.io.*;
import java.rmi.*;
import java.util.*;
import java.sql.*;

public class baja_alumno extends HttpServlet{

    public void doPost(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException{
        PrintWriter web = response.getWriter();
        try{
            response.setContentType("text/html");
            HttpSession sesion = request.getSession(true);
            //se reciben los valores del formulario 1
            String matricula = request.getParameter("matricula");
            web.println(matricula);
            //paso 1 registro de drive
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            web.println("<br>Driver registrado");
            //paso 2 tenemos conexion
            Connection conexion =
DriverManager.getConnection("jdbc:mysql://localhost/nameBD","usuario","passwor
d");
            web.println("<br>Conexion lista");
            //paso 3 construimos el query
            String query= "delete from alumnos where
matricula='"+matricula+"'";
            web.println("<br>" + query);
            //paso 4 obtenemos un statement
            Statement st = conexion.createStatement();
            //paso 5 ejecutamos el query
            st.executeUpdate(query);
            web.println("<br>Delete exitoso");
            //paso 6 Siempre hay que cerrar la coneccion
            st.close();
            conexion.close();

```

```

        }catch(SQLException error){
            System.out.print("ERROR SQL ->" + error);
        }catch(Exception error){
            System.out.print("ERROR GENERAL ->" + error);
        }
    }
}

```

Código para actualizar los datos de un registro de la base de datos

```

import javax.servlet.http.*;
import java.io.*;
import java.rmi.*;
import java.util.*;
import java.sql.*;

public class Modificar_alumno extends HttpServlet{
    public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException{
        PrintWriter web = response.getWriter();
    try{
        response.setContentType("text/html");
        HttpSession sesion = request.getSession(true);
        //se reciben los valores del formulario
        String matricula = request.getParameter("matricula");
        String nombre = request.getParameter("nombre");
        String carrera = request.getParameter("carrera");
        int semestre = Integer.parseInt(request.getParameter("semestre"));
        Class.forName("com.mysql.jdbc.Driver").newInstance(); //paso 1 registro de drive
        web.println("<br>Driver registrado");//paso 2 tenemos conexion
        Connection conexion = DriverManager.getConnection("jdbc:mysql://localhost/nameb
d","usuario","password");
        //paso 3 construimos el query
        String query = "update alumnos set
nombre='"+nombre+"',carrera='"+carrera+"',semestre='"+semestre+"
matricula='"+matricula+"'";
        web.println("<br>update exitoso");//paso 4 obtenemos un statement
        Statement stmt = conexion.createStatement();
        stmt.executeUpdate(query); //paso 5 ejecutamos el query
        stmt.close(); //paso 6 Siempre hay que cerrar la coneccion
        conexion.close(); }catch(SQLException error){
            System.out.print("ERROR SQL ->" + error);
        }catch(Exception error){ System.out.print("ERROR GENERAL -
>" + error);
    }
}

```

## Anexo 4. Manual de instalación y configuración de Apache Tomcat.

### INSTALACIÓN DE APACHE TOMCAT

- 1.- Descargar una versión de Apache Tomcat lo puedes hacer en la siguiente página <http://jakarta.apache.org>.
2. Descomprimir el archivo, por ejemplo en: c:\apache-tomcat-5.5.17i.exe (Windows) Haz doble clic en el programa de instalación de apache.
- 3.- Aparece la siguiente ventana para iniciar la instalación. Dar clic en Next para continuar.



Figura 4.0.1.A. Setup de Apache Tomcat

4.- Aquí debes de seleccionar la opción de I Agree para aceptar la licencia.

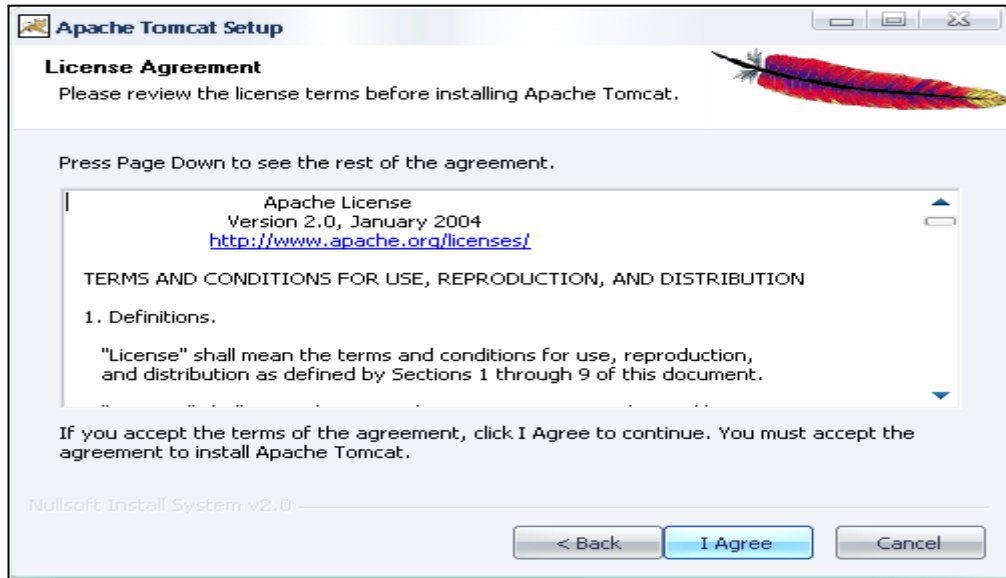


Figura 4.0.2.A Licencia de Apache

5.- Marcar los componentes que quieres instalar y haz clic en Next para continuar.

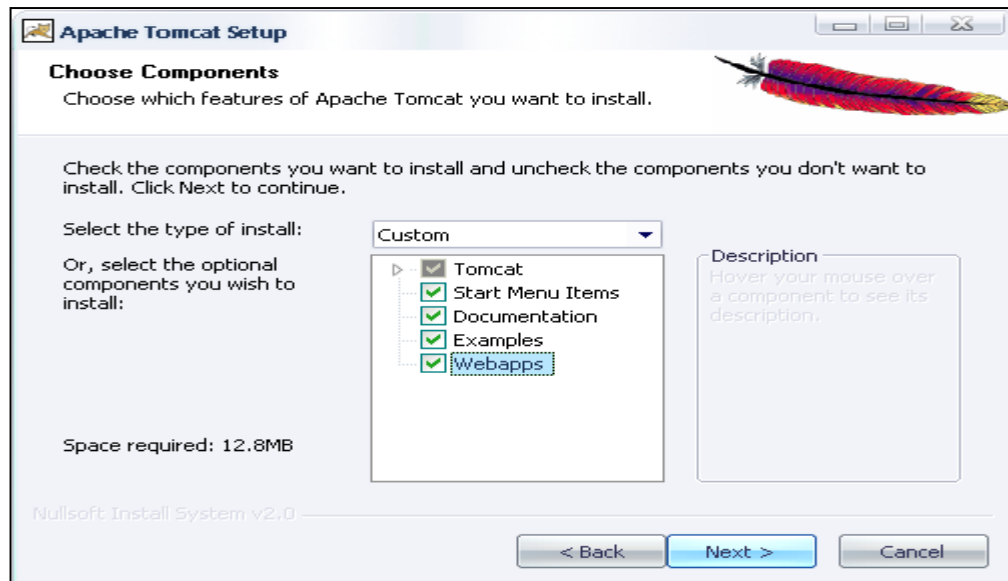


Figura 4.0.3.A Componentes de Apache

6.- En este paso se pone la dirección donde va a ser instalado, pero puedes cambiar de dirección pulsando Browse y eligiendo la dirección donde quieres la instalación. Dar clic en Next para continuar.

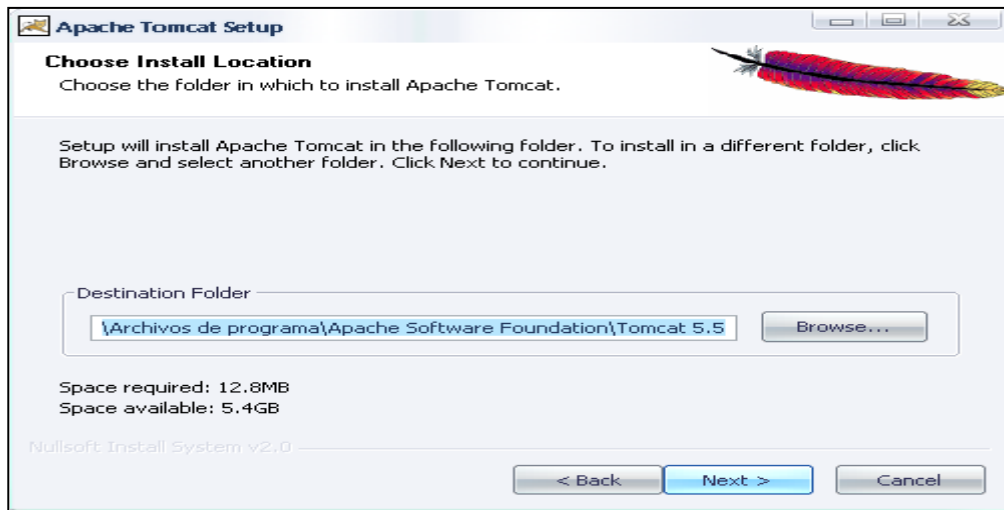


Figura 4.0.4.A Dirección de la instalación Apache Tomcat

7.- En esta pantalla muestra algunas opciones de configuración puedes poner usuario y contraseña o dejarlo así y dar clic en Next para continuar

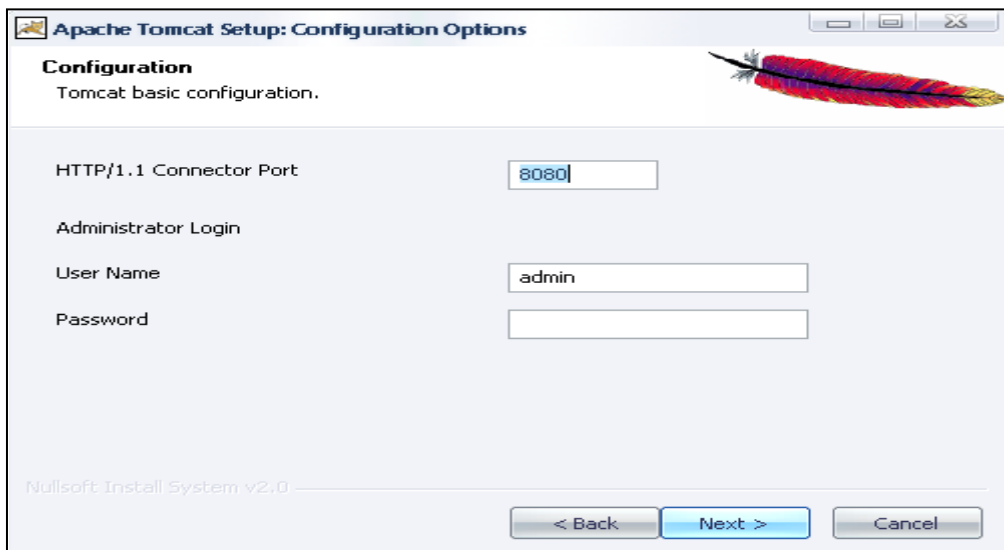


Figura 4.0.5.A Configuración de Apache Tomcat



8.-Como ya antes mencionado para poder instalar Apache Tomcat debes ya tener instalado el JDK. En esta pantalla te pone la dirección de la PATH del J2SE 5.0 JRE instalado en tu sistema. Haz clic en Install.

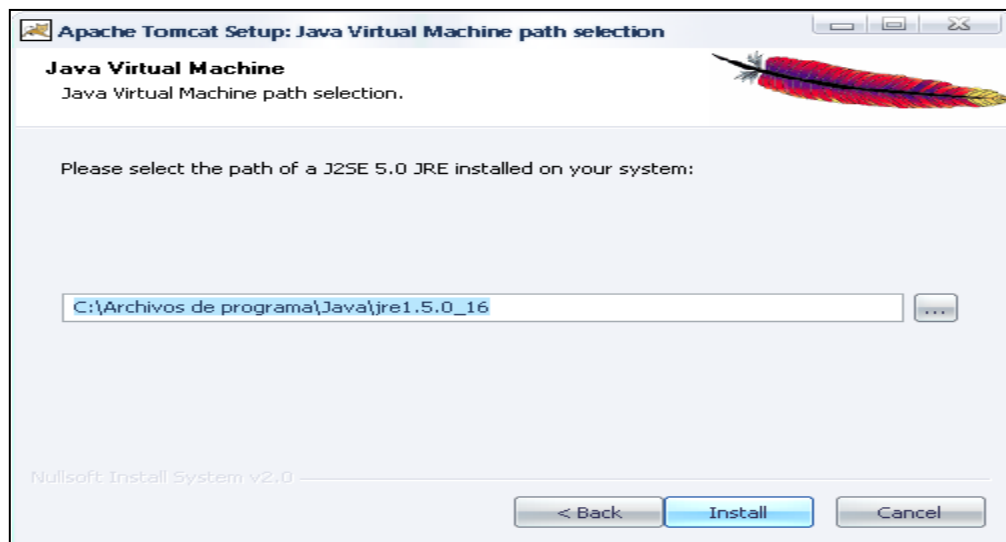


Figura 4.0.6.A Dirección de Java Virtual

9.- Esperar mientras Apache Tomcat se termina de instalar.

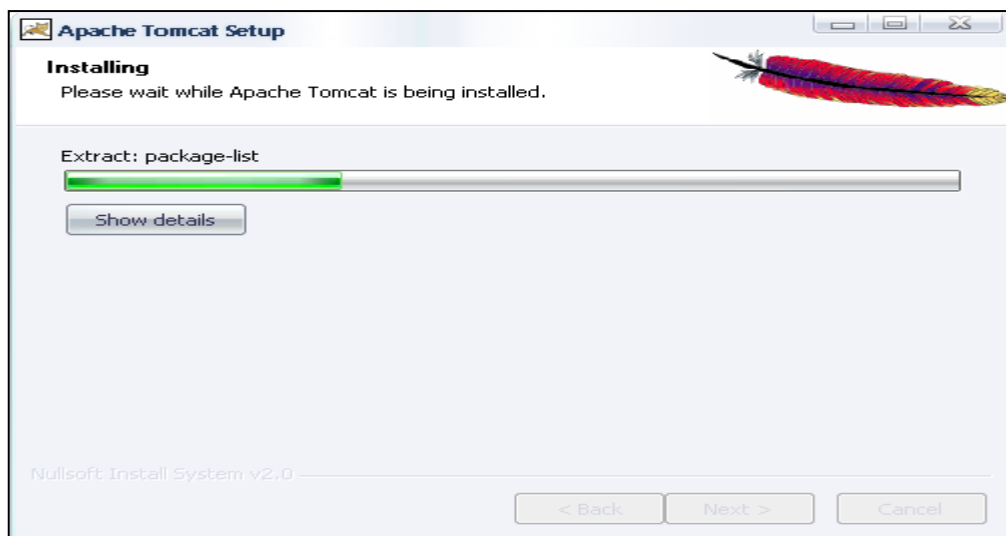


Figura 4.0.7.A Instalación en proceso

10.- Al terminar, pulsa Finish. Y automáticamente se inicia el servidor.

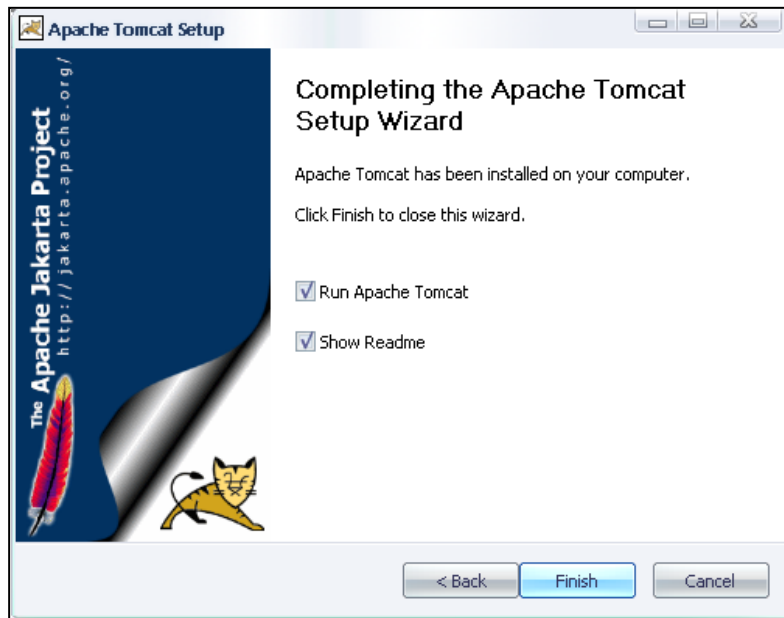


Figura 4.0.8.A Instalación completa de Apache Tomcat

11.-Para probar que se haya instalado correctamente en un navegador Web coloca la siguiente dirección `http://localhost:8080`.



Figura 4.0.9.A Probando que la instalación sea correcta

## PASOS PARA USAR EL SERVIDOR APACHE TOMCAT

Una vez ya instalado el servidor Apache Tomcat en la dirección C:\Archivos de programa \ apache Software Foundation\Tomcat 5.5, genera las siguientes carpetas:

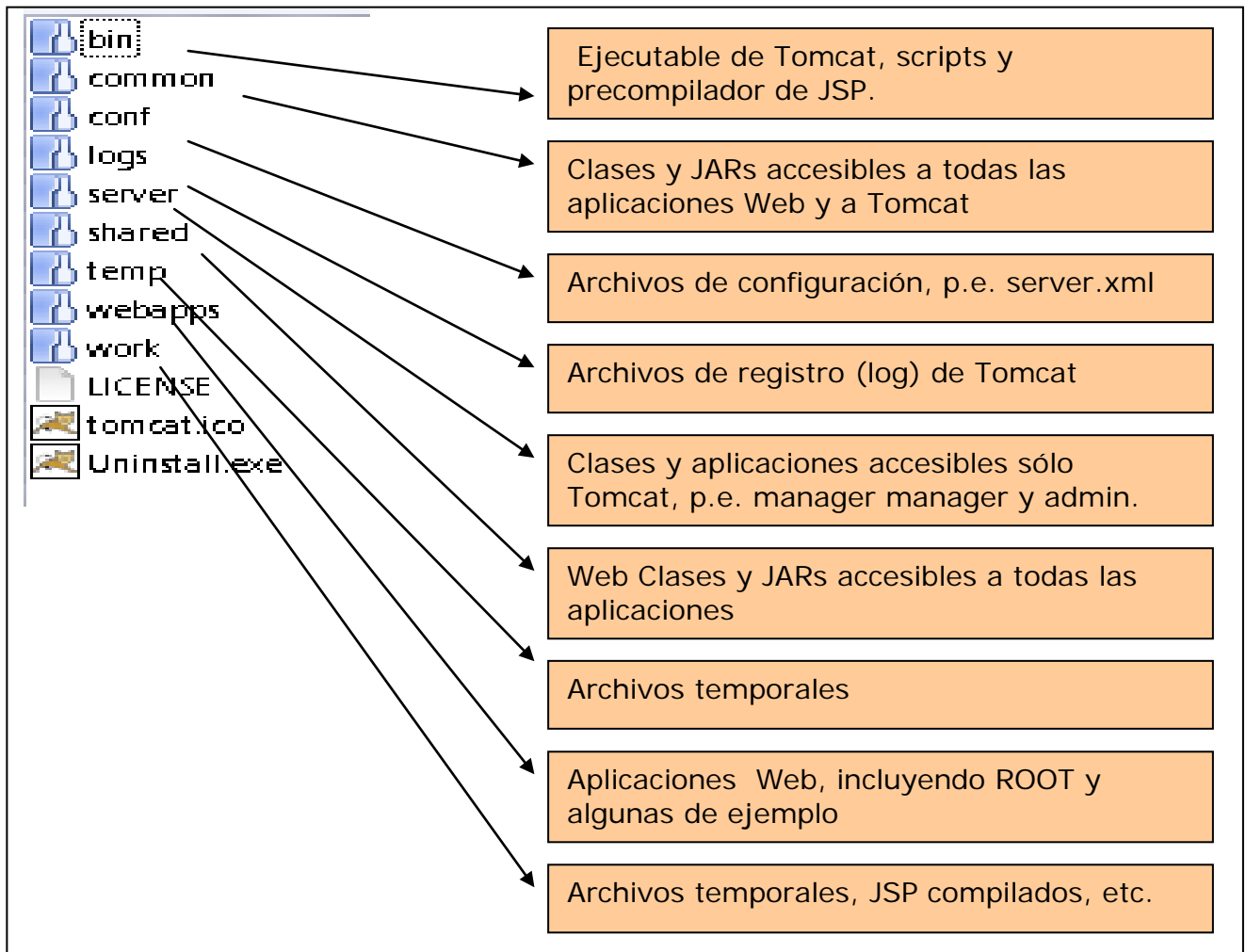


Figura 4.0.10.A Generar carpeta para Apache Tomcat

Para la configuración de ejemplos de JSP's y SERVLET's debes acceder al siguiente directorio C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\webapps donde se encuentran los siguientes directorios que muestra la estructura de la aplicación Web.

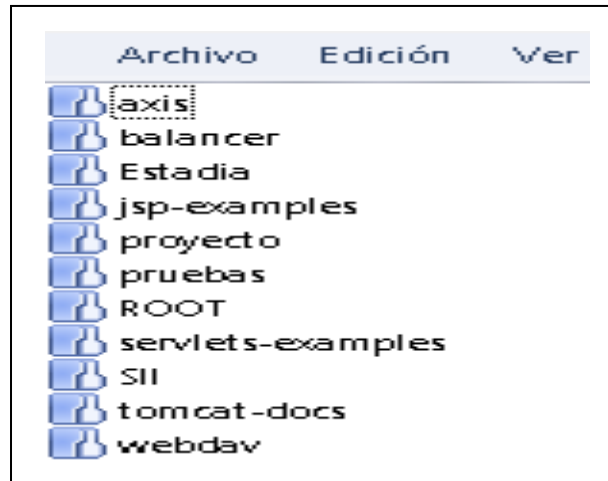


Figura 4.0.11.A Configurar carpetas para JSP

Por ejemplo en el siguiente directorio C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\webapps\Estadia se encuentra las siguientes carpetas para poder ejecutar los JSP solo con colocarlos en esta dirección y funcionaran correctamente.



Figura 4.0.12.A Colocación de JSP

Para ejecutar un JSP debes colocar en el navegador Web la siguiente dirección [http://localhost:8080/Estadia/alta\\_alum.jsp](http://localhost:8080/Estadia/alta_alum.jsp).



Figura 4.0.13.A Pantalla de ejemplo de un JSP

Ahora bien si quieres ejecutar JSP con Servlets en la siguiente dirección C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\webapps\Estadia\WEB-INF.

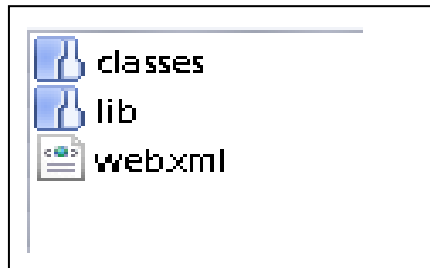


Figura 4.0.14.A Como ejecutar de JSP con un Servlets

En la carpeta clases coloca los archivos .class generados por ejemplo en la figura 4.16.A muestra los .CLASS.



Figura 4.0.15.A Archivos.class

En la carpeta lib puedes colocar algunas librerías o algunos driver por ejemplo para utilizar conexiones a una BD se coloca el driver de MySQL como se muestra en la figura 4.0.16.A.

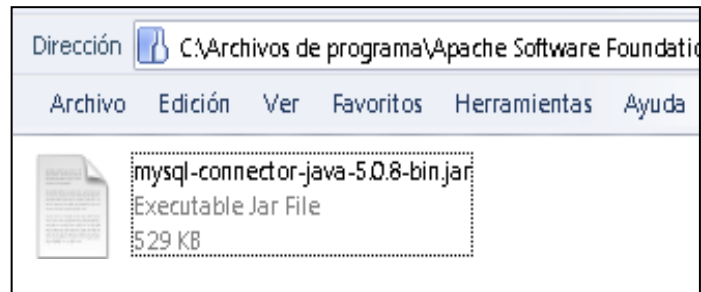


Figura 4.0.16.A Colocación de Driver de MySQL

Un punto muy importante es la configuración del archivo llamado web.xml como se muestra en la s figura 4.0.17.A.

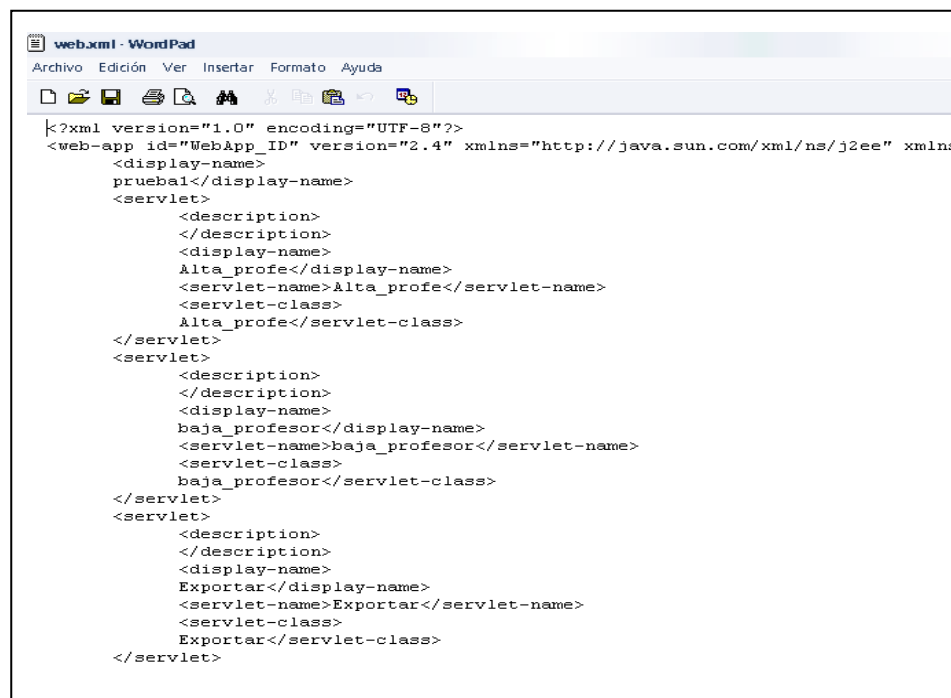


Figura 4.0.17.A Configuración de Web.xml

En este archivo debes de colocar el nombre del servlet así como el nombre de la clase, otro paso importante es el mapeo como se muestra en la figura 4.0.18.A

```
<servlet-mapping>
  <servlet-name>Modificar_alumno</servlet-name>
  <url-pattern>/Modificar_alumno</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>Modificar_materia</servlet-name>
  <url-pattern>/Modificar_materia</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>consultar_materia</servlet-name>
  <url-pattern>/consultar_materia</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>Modificar_facilidad</servlet-name>
  <url-pattern>/Modificar_facilidad</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>consultar_facilidad</servlet-name>
  <url-pattern>/consultar_facilidad</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>LoginA</servlet-name>
  <url-pattern>/LoginA</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>LoginP</servlet-name>
  <url-pattern>/LoginP</url-pattern>
</servlet-mapping>
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>default.html</welcome-file>
  <welcome-file>default.htm</welcome-file>
  <welcome-file>default.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

Figura 4.0.18.A Configuración del mapeo de los servlets

La figura 4.0.19.A muestra la manera de cómo mandar llamar un servlet, solo coloca el nombre de la clase en el action del formulario.

```
<form id="form1" name="form1" method="post" action="Alta_profe"
onSubmit="return validar(this)">
```

Figura 4.0.19.A Configurar el action del formulario

## Anexo 5. MANUAL DE INSTALACIÓN DE JAVA (JDK -1\_5\_0\_16)

Para poder instalar Apache Tomcat y cualquier versión de Eclipse es necesario instalar el JDK.

- 1.- Descargar una versión de JDK lo puedes hacer en la siguiente página <http://java.sun.com/javase/downloads/index.jsp>.
- 2.- Dar doble clic sobre el siguiente ejecutable que se encuentra en la dirección donde se descargo (Mostrado en la figura 5.0.1.A).

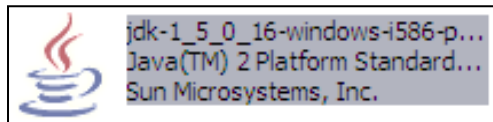


Figura 5.0.1.A Instalador del JDK

- 3.- En seguida se inicia el proceso de instalación como se muestra en la figura 5.0.2.A

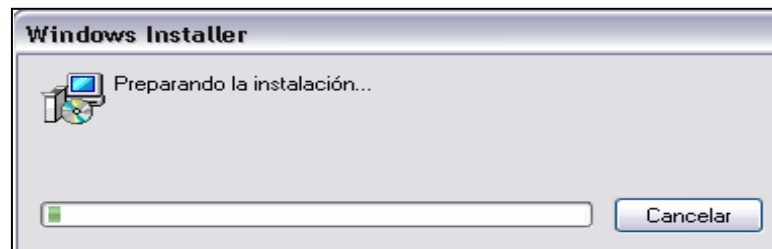


Figura 5.0.2.A Preparación para la instalación

- 4.- En la siguiente pantalla se acepta la licencia y dar clic en el botón Next para continuar el proceso de instalación.

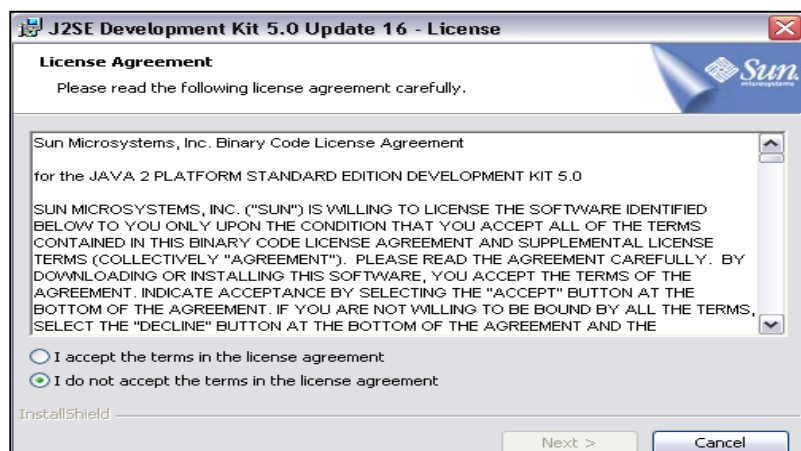


Figura 5.0.3.A Aceptación de licencia

- 5.- Esperar mientras se está instalando.



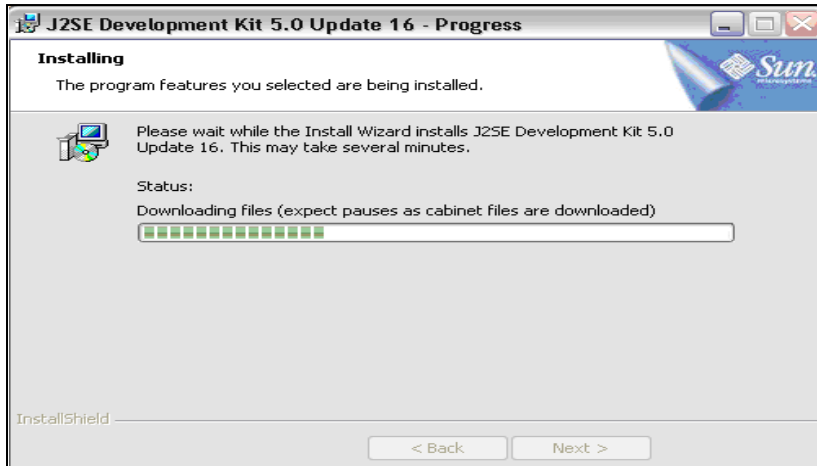


Figura 5.0.4.A Proceso de instalación

6.-La figura 5.0.5.A muestra en pantalla los programas que se desean instalar. Dar clic Next para continuar con la instalación.

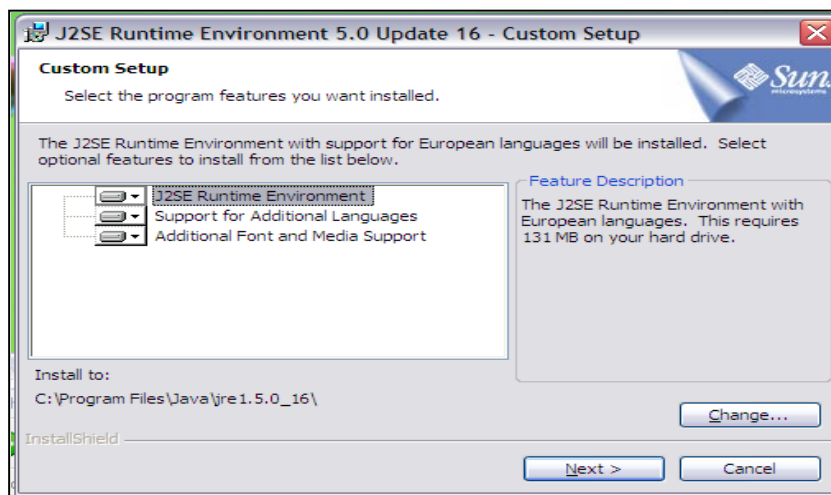


Figura 5.0.5.A Proceso de instalación

7.-En esta pantalla muestra que la instalación fue completada. Dar clic en Finish.

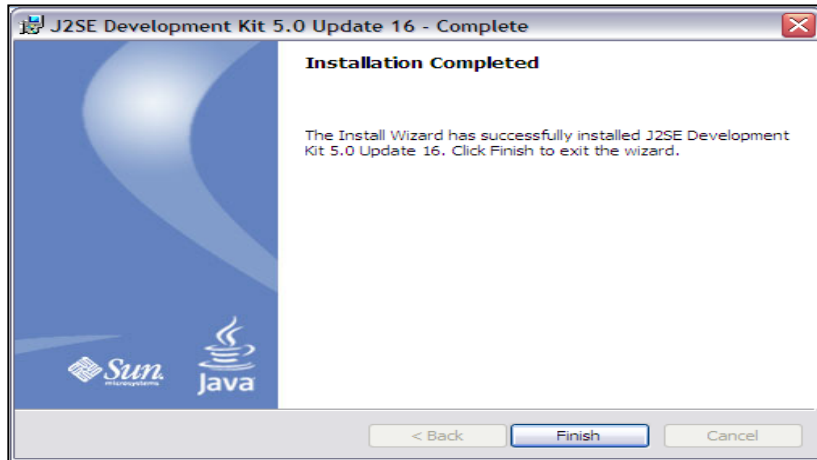


Figura 5.0.6.A Instalación completa

9.- para probar que se instalo bien el JDK. Ir a la página <http://java.sun.com/javase/downloads/index.jsp>.

10.- Descarga el archivo WinZip: eclipse-jee-ganymede-win32.

11.- Descomprimir el archivo y en la carpeta se encuentra un icono de eclipse. Dar en doble clic sobre el.



Figura 5.0.7.A Imagen de eclipse cargando

12.- Después del proceso de la figura 5.0.8.A se muestra la dirección en donde se guardarán los archivos con los que trabajaremos

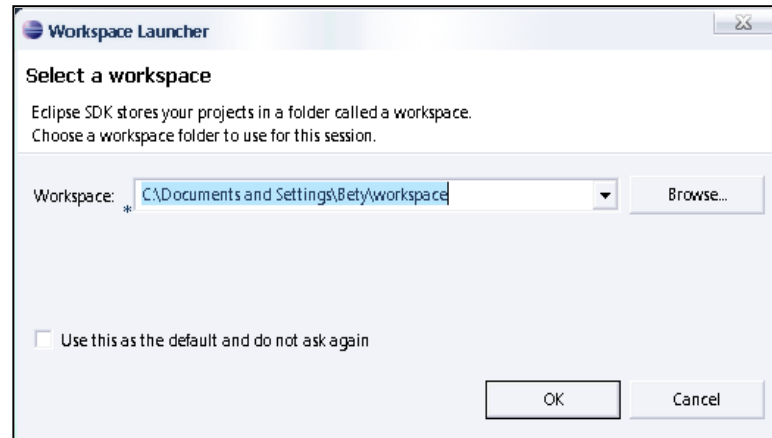


Figura 5.0.8.A Dirección donde se guarda el proyecto

## Anexo 6. Configuración de la CLASSPATH

- 1.- Ir a inicio dar clic derecho sobre el icono de Mi PC.
- 2.- Dar clic sobre Propiedades y te muestra la figura 6.0.9.A Selecciona Opciones avanzadas y dar un clic sobre variables de entorno.

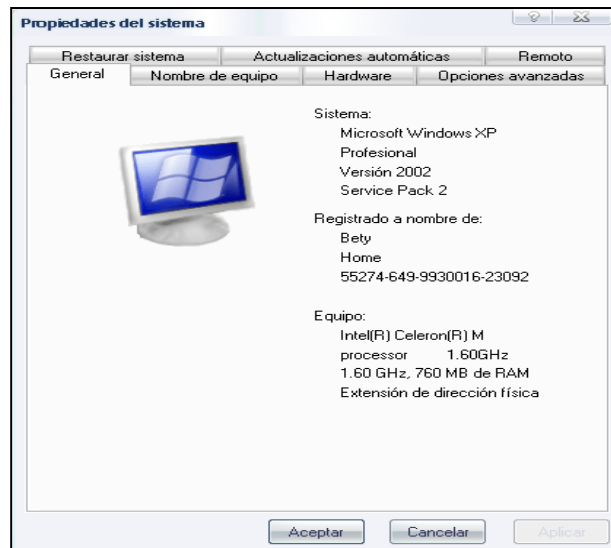


Figura 6.0.9.A Propiedades del sistema

3.- En la figura 6.10.A se muestra varias opciones elegir la opción de CLASSPATH

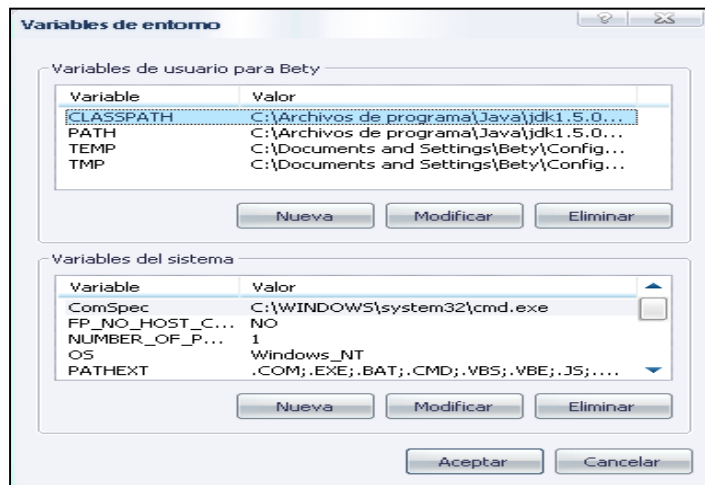


Figura 6.0.10.A Variables de entorno

4.- En este paso colocar la siguiente dirección C:\Archivos de programa\Java\jdk1.5.0\_16\bin; como se muestra en la figura 6.11.A y dar clic en aceptar.

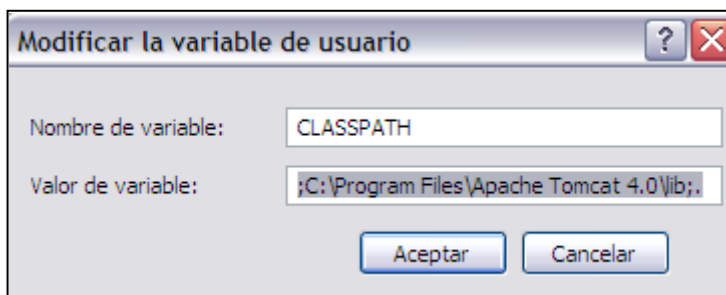


Figura 6.0.11 .A Modificación de la variable de usuario.

## BIBLIOGRAFÍA

Libro:

[1] Hall, Marty. (2001) Servlets y JavaServer Pages. Guía Práctica. México: Pearson Educación

[2] Joyanes Aguilar, Luís y Zahonero Martínez Ignacio. (2001) Programación en Java 2.  
Monterrey Nuevo León: Mc Graw Hill.

[3] kurniawan, Budi.(2002) Java for the web with servlet JSP. Richmond Hill Toronto:New Riders.

[4] Bruegger, Bernd y Dut Oit, Allen H. (2000) Ingeniería de software Orientado a Objeto. México: Prentice Hall

[5] Garza Mora, David. (2001) Aprendiendo MYSQL en 21 días. México: Pearson Educación.

[6] Hullman, Larry. (2003) Guía de aprendizaje MySQL. Madrid: Pearson educación, S.A.

[7] Castro, Elizabeth .(2002) Html for world wide web. corbetl califonia:Peacht Press.

[8] Lan, Sommerville. (2000) Ingeniería de Software. México: addison Wesley.

[9] M.Deitel, Harvey. (1999) Como Programar en Java. México: Pearson.

[10] Hugh E.Williams (2000) Php and Mysql.San Antonio California:Lane

## Sitios de Internet:

[11] Wikimedia Foundation, Inc. Java Development Kit. México. 20 de Noviembre de 2008. <http://es.wikipedia.org/wiki/JDK>

[12] Wikimedia Foundation, Inc. JavaServer Pages. México. 20 de Noviembre de 2008. [http://es.wikipedia.org/wiki/Java\\_Server\\_Pages](http://es.wikipedia.org/wiki/Java_Server_Pages)

[13] Wikimedia Foundation, Inc. Application Programming Interface. México. 20 de Noviembre de 2008. [http://es.wikipedia.org/wiki/Application\\_Programming\\_Interface](http://es.wikipedia.org/wiki/Application_Programming_Interface)

[14] *ingenierosoftware.com. Desarrollo de Software Orientado a Objeto. Joaquín Gracia. (2003). México. 20 de Noviembre de 2008. <http://www.ingenierosoftware.com/analisisydiseno/uml.php>*

[15] Wikimedia Foundation, Inc. HTML. México. 20 de Noviembre de 2008. [http://es.wikipedia.org/wiki/C%C3%B3digo\\_HTML](http://es.wikipedia.org/wiki/C%C3%B3digo_HTML)

[16] Wikimedia Foundation, Inc. Código Fuente. México. 01 de Diciembre de 2008. [http://es.wikipedia.org/wiki/C%C3%B3digo\\_fuente](http://es.wikipedia.org/wiki/C%C3%B3digo_fuente)

[17] Guía de Iniciación al Lenguaje JAVA. México. 01 de Diciembre de 2008. [http://pisuerga.inf.ubu.es/lisi/Invest/Java/Tuto/I\\_3.htm](http://pisuerga.inf.ubu.es/lisi/Invest/Java/Tuto/I_3.htm)

[18] Wikimedia Foundation, Inc. Algoritmo. México. 01 de Diciembre de 2008. <http://es.wikipedia.org/wiki/Algoritmo>

[19] Arreglo. México. 01 de Diciembre de 2008. <http://delfosis.uam.mx/~sgb/Java/Arreglos.html>

[20] Ingenieria de software, trackback. Sabueso Web al servicio de la Ingenieria de Sistemas e Informatica16 de diciembre de 2008. <http://sabuesoweb.subdominius.com/2007/07/18/metodos-de-caja-blanca-y-caja-negra/>