Universidad Autónoma Metropolitana – Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Sistema de asignación automático de tutores para los becarios de la División de CBI en la UAM Azcapotzalco

Proyecto Tecnológico

Trimestre 2017-Otoño

Oliver Saucedo Vargas
2112043966
al2112043966@correo.azc.uam.mx

Asesor:

Dr. José Alejandro Reyes Ortiz

Profesor Titular
Departamento de Sistemas
jaro@correo.azc.uam.mx

5 de Enero de 2018

Yo, José Alejandro Reyes Ortiz, declaro que aprobé el contenido del presente Reporte de Proyecto de integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Firma.

Yo, Oliver Saucedo Vargas, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el repositorio Institucional de UAM Azcapotzalco.

Firma.

RESUMEN

El Sistema de asignación automático de tutores para los becarios de la División de CBI en la UAM Azcapotzalco es un proyecto el cual surge a raíz del largo proceso que realizan los alumnos de la institución cada año tanto para llegar a ser un beneficiario como para realizar la renovación del Programa Nacional de Becas y Financiamiento (PRONABES) ofrecido por el Gobierno Federal; igualmente pretende reducir el personal que se enfoca en este tipo de trámites.

A lo largo de 264 horas invertidas, respectivamente entre el trimestre de invierno y primavera se logró diseñar e implementar un sistema web para la administración de becarios de la división CBI y la asignación de sus respectivos tutores basado en ciertos parámetros, tales como nombre de la licenciatura, departamento del profesor, número de becarios asignados a cada tutor, entre otros; y con ello llegar a reducir el personal encargado de la asignación de tutores, cumpliendo con el enfoque principal del proyecto para la UAM, lo cual deja la tarea al becario de realizar su registro y atender a las notificaciones en dicho portal mediante una cuenta de usuario enlazada con su matrícula; por otra parte la coordinación contara con las credenciales necesarias para poder asignar tutores, llenar reportes de seguimiento y la actualización de datos de los becarios.

Todo el sistema se desarrolla en lenguaje Java utilizando su tecnología ServerPages (JSP) en servidor web Apache TomCat con la vista implementada en HTML 5 en la cual se incluyen cinco hojas de estilo en cascada (CSS) y una base de datos implementada en MySQL.

Contenido

1. Introducción

Desde marzo de 2001, alumnos de la UAM se han visto beneficiados por el Programa Nacional de Becas y Financiamiento (PRONABES), hasta el trimestre 2014 Otoño 34,954 fueron las becas que el Gobierno Federal entregó a alumnos de la UAM, de las cuales 10,472 becas fueron entregadas en el año 2014. 3,268 de esas becas pertenecen a alumnos de la unidad Azcapotzalco. **[1]**

Para completar el trámite, el alumno debe acudir al coordinador de su licenciatura, quien le generará su hoja de registro, le asignará un tutor académico que le dará seguimiento en sus estudios para que el alumno conserve su beca **[2].** Todos estos procesos se llevan a cabo de manera manual lo que requiere mucho tiempo y personal para poder atender la gran cantidad de becarios nuevos así como en estado de renovación.

Se propone un sistema de información web y bus de servicios de empresa (BSE) donde el becario, una vez aprobado, solamente se registre y esté al pendiente de las notificaciones que la coordinación divisional realice sobre el becario. La coordinación divisional, por su parte, debe asignar el tutor, enviar un reporte de seguimiento al becario, actualizar la información del becario.

2. Antecedentes

2.1 Tesis

- Análisis, diseño e implementación de un sistema de información para el control de becarios y ex becarios de una asociación educativa. **[3]**

  Las similitudes en esta tesis son muy significativas, lo que se propone es diseñar e implementar un sistema en línea de pago de becas, ya que hacerlo de manera manual como se había estado realizando el trámite es lento y requiere de mayor personal. A diferencia de este proyecto citado, el enfoque principal del proyecto para la UAM es la asignación automática de tutores en base a diferentes condiciones, como nombre de la carrera, promedio del becario, número de matrícula, entre otros.

- Relación de la asignación de personal de enfermería con indicadores de resultado de la calidad de la atención en unidades de cuidados intensivos de adultos. **[4]**

  En este trabajo se realiza un análisis en hospitales de Colombia para la asignación de enfermeros con las relaciones paciente/enfermero y paciente/auxiliar similar a la relación que habrá de becario/tutor. En el caso citado se basan en los turnos hospitalarios y en los eventos que ocurren a lo largo de esos turnos, el promedio de los pacientes que atiende cada enfermera en una jornada y el tiempo que requiere cada paciente dependiendo de su enfermedad, mientras que en el sistema de becas será debido a los eventos transcurridos a lo largo de todo un trimestre, el promedio obtenido en ese trimestre, el tiempo transcurrido con la beca, y número de matrícula.

2.2 Proyectos terminales o de integración

- Sistema Web para la identificación automática de aspectos académicos y de experiencia profesional en expedientes curriculares. **[5]**
  En éste proyecto se utilizan tres módulos: pre-procesamiento, extracción de información y validación. Los primeros dos módulos constan de un analizador sintáctico, y aplicación de patrones para la extracción de la información. La similitud se encuentra en el módulo de validación que es un Sistema web basado en el modelo vista controlador (MVC), utilizando HTML y DreamWorks como Servlets, donde se muestran los resultados de los módulos anteriores, al igual que el Sistema de asignación de tutores mostrará las búsquedas de tutores y becarios mediante filtros.

- Implementación de un sistema web para dar seguimiento a las visitas realizadas a entidades financieras por parte de la Comisión Nacional Bancaria y de Valores **[6]**
  En este proyecto se utiliza una arquitectura de tres niveles o capas, la capa de presentación, la capa de negocio y la capa de datos, similar al modelo vista controlador que se propone para el Sistema de asignación de tutores. También se requiere del manejo de una base de datos para almacenar la información que se va a manejar y el uso de Servlets en la etapa intermedia en ambos proyectos.

2.3 Artículos

- Asignación de horarios de clases universitarias mediante algoritmos evolutivos. **[7]**
  El problema al que se enfrentan es la asignación de horarios para las diferentes materias de la Facultad de Ingeniería de manera que no deba ser manualmente ya que requiere mucho esfuerzo de personal, se enfrentan a un desafío similar de asignación manual. La manera en proponen solucionarlo difiere de la propuesta solución para el sistema de asignación de tutores, debido a que su problema consta de más variables como la disponibilidad de salones, de profesores, evitar empalmes entre otras, la solución propuesta es un algoritmo evolutivo.

- Architectural Model for Interoperability between Healthcare Service Providers in Colombia. **[8]**

  En éste artículo se propone un sistema de intercambio de información clínica y administrativa basado en arquitecturas orientadas a servicios (SOA) similar al propuesto para la asignación de becarios. Con la diferencia de que en ese trabajo el enfoque principal es el tipo de arquitectura que se desea utilizar, mientras que en el trabajo propuesto para el sistema de becarios el factor importante es la asignación de becarios más que meramente el manejo de información.

3. Justificación

Hoy en día es fundamental tener servicios web integrales y distribuidos capaces de satisfacer altos estándares de control. El Bus de Servicios de Empresa, BSE, integra las mejores características de diferentes tecnologías de información. Permite un bajo

acoplamiento entre sus aplicaciones, una plataforma desarrollada bajo estándares, integra servicio de mensajes, servicio web, transformación de datos y un eficiente enrutamiento para conexiones **[9]**.
La importancia de éste sistema de información será el módulo de asignación de tutores que será basándose en los criterios ya existentes por la Coordinación, lo cual ayudará a la coordinadora a asignarle a cada alumno becado su tutor basándose en la carrera del alumno, año de ingreso y área del profesor.

4. Objetivos

4.1 Objetivo general:

Implementar el sistema de seguimiento de tutores para alumnos becados de la UAM Azcapotzalco, con arquitectura orientada a servicios para agilizar la asignación de tutores.

4.2 Objetivos específicos:

- Diseñar e implementar un módulo web para la gestión de datos de alumnos becados para que se les asigne un tutor y dar seguimiento a sus estudios.
- Diseñar e implementar un módulo para apoyar la selección y asignación de tutores a becarios.
- Diseñar e implementar un módulo de generación reportes.

5. Marco Teórico

5.1 Sistema Web

Los “sistemas Web” o también conocido como “aplicaciones Web” son aquellos que están creados e instalados no sobre una plataforma o sistemas operativos (Windows, Linux). Sino que se aloja en un servidor en Internet o sobre una intranet. Su aspecto es muy similar a páginas Web que vemos normalmente, pero en realidad los ‘sistemas Web’ tienen funcionalidades muy potentes que brindan respuestas a casos particulares.

Las aplicaciones Web trabajan con bases de datos que permite procesar y mostrar información de forma dinámica para el usuario.

5.2 HTML 5

Que significa Lenguaje de Marcado para Hipertextos (HyperText Markup Language) Es considerado el producto de la combinación de HTML, CSS y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5. HTML está a cargo de la estructura.

HTML5 provee básicamente tres características: estructura, estilo y funcionalidad.

Nos brinda varios elementos que perfeccionan esta estructuración estableciendo qué es cada sección, eliminando así DIV innecesarios. Este cambio en la semántica hace que la estructura de la web sea más coherente y fácil de entender por otras personas y los navegadores podrán darle más importancia a según qué secciones de la web

faciliten dándole además la tarea a los buscadores, así como cualquier otra aplicación que interprete sitios web. **[10]**

5.3 Framework

Es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación.

Una estructura en capas que indica qué tipo de programas pueden o deben ser construidos y cómo se interrelacionan. Algunos marcos de trabajo de sistemas informáticos también incluyen programas reales, especifican interfaces de programación u ofrecen herramientas de programación para usar los marcos.

Un *framework* puede servir para un conjunto de funciones dentro de un sistema y cómo se interrelacionan; las capas de un sistema operativo; las capas de un subsistema de aplicación; cómo debería normalizarse la comunicación en algún nivel de una red. Un marco de trabajo es generalmente más complejo que un protocolo y más prescriptivo que una estructura. **[11]**

5.4 Servlets

El servlet es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor. Aunque los servlets pueden responder a cualquier tipo de solicitudes, éstos son utilizados comúnmente para extender las aplicaciones alojadas por servidores web, de tal manera que pueden ser vistos como applets de Java que se ejecutan en servidores en vez de navegadores web.

Los servlets son mejores para manejar las funciones de control de una aplicación, como despachar solicitudes, y manejar datos que no sean texto.

5.5 JSP

Java Server Pages (JSP) es una tecnología que nos permite mezclar HTML estático con HTML generado dinámicamente.

Con JSP podemos crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. **[12]**

5.6 Modelo vista controlador

Es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario.

“Se preocupa por crear procesos que aseguren calidad en los programas que se realizan y esa calidad atiende a diversos parámetros que son deseables para desarrollo, como la

estructuración de los programas o reutilización del código, lo que debe influir positivamente en la facilidad de desarrollo y el mantenimiento." **[13]**

5.6.1 Modelo

Esta es la capa donde se trabaja con los datos, debido a eso contendrá mecanismo para ingresar a la información y también para actualizar su estado. Típicamente las clases del modelo contendrán funciones que nos ayudaran a mostrar, insertar, actualizar y eliminar información de la base de datos.

5.6.2 Vista

Es la información que se presenta al usuario. Una vista será normalmente una página web, pero también puede ser un fragmento de página como el encabezado o pie de página.

Visto de otra forma, es la parte donde se interactúa con el usuario, y donde se especifican cosas como formularios, posición de datos, y como estos se despegaran en la pantalla.

5.6.3 Controlador

Sirve como un intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para procesar la solicitud http y echar a andar una página web.

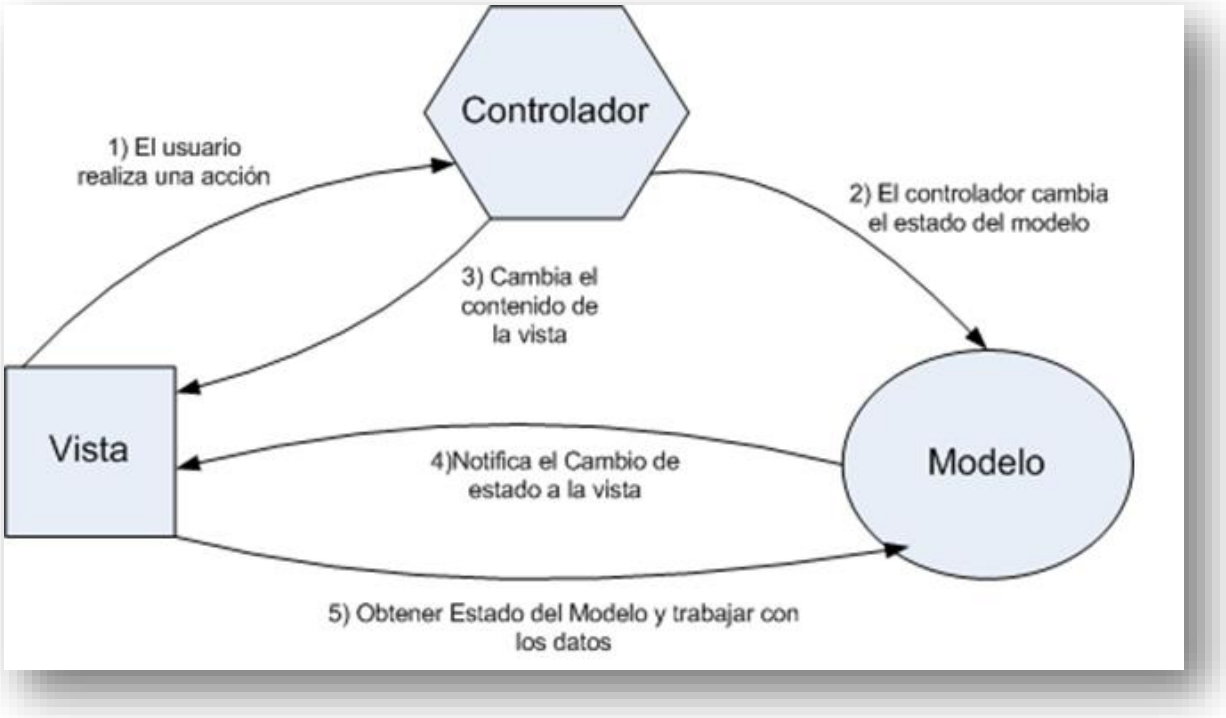


Figura 1. Ejemplo del funcionamiento del Modelo Vista-Controlador.

5.7 Arquitectura en 3 capas

Es un modelo de desarrollo software en que el objetivo primordial es la separación de las partes que componen un sistema software o también una arquitectura cliente-servidor.

5.7.1 Capa de presentación

Esta capa es el interface con el usuario y consiste en hardware como en un pc o una estación de trabajo y un navegador de red.

5.7.2 Negocio

En esta capa se proporciona la funcionalidad al extremo de los usuarios y contiene la lógica del negocio. (En donde se dice que se hace con los usuarios).

5.7.3 Datos

Esta capa incluye la base de datos, que tiene todos los datos de la organización. Su única función es pasarle las acciones que realice el usuario a la capa de negocio.

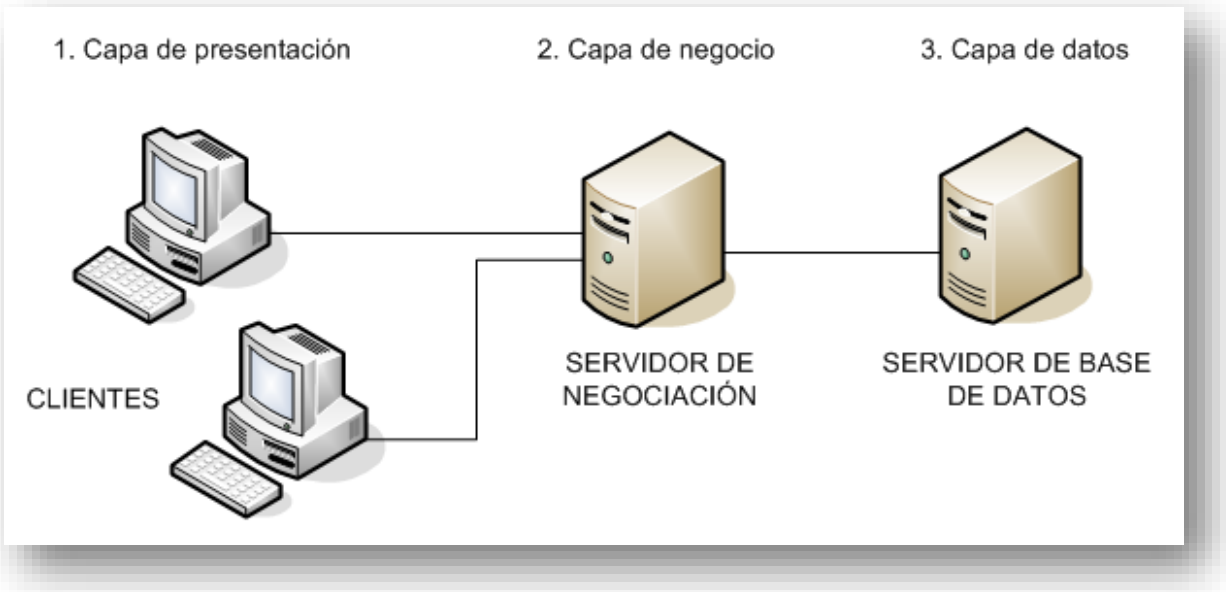


Figura 2. Diagrama de la Arquitectura en 3 capas.

5.8 Clases VO y DAO

Se utiliza para transferir varios atributos entre el cliente y el servidor o viceversa, básicamente consta de 2 clases:

5.8.1 Data Transfer Object/ Value Object (DTO/VO)

La primera es una clase java conocida como Value Object que únicamente contiene sus atributos, constructor, getters y setters, esta clase no tiene comportamiento.

5.8.2 Data Access Object (DAO)

La segunda es una clase del lado del servidor conocida como clase de negocio (en la implementación también se conoce como Business Object) Su función principal es el acceso a los datos desde la base de datos y llenar la clase Value Object y enviarla al cliente, o a su vez recibir la clase desde el cliente y enviar los datos al servidor, por lo general tiene todos los métodos CRUD (create, read, update y delete). Por lo general se tiene un DAO para cada tabla en la base de datos. **[15]**

6. Desarrollo del proyecto

El proyecto propuesto consta de varias etapas y el uso de diferentes tecnologías. Requiere la gestión de una base de datos para almacenar la información de los becarios y tutores. Basándonos en el modelo Vista Controlador se hará uso de JSPs, Servlets y de las clases VO y DAO, y hojas de estilo en cascada.

6.1 Diagrama de casos de uso

El diagrama de casos de uso para éste servicio web cuenta con la información del becario, para acceder primeramente deberá registrarse en la misma página del sistema, siendo su matrícula su nombre de usuario. Los módulos a los que pueda acceder un becario serán limitados en comparación a los de la coordinadora, eso se define dependiendo de quién ingrese al sistema, como se observa en la Figura 4.

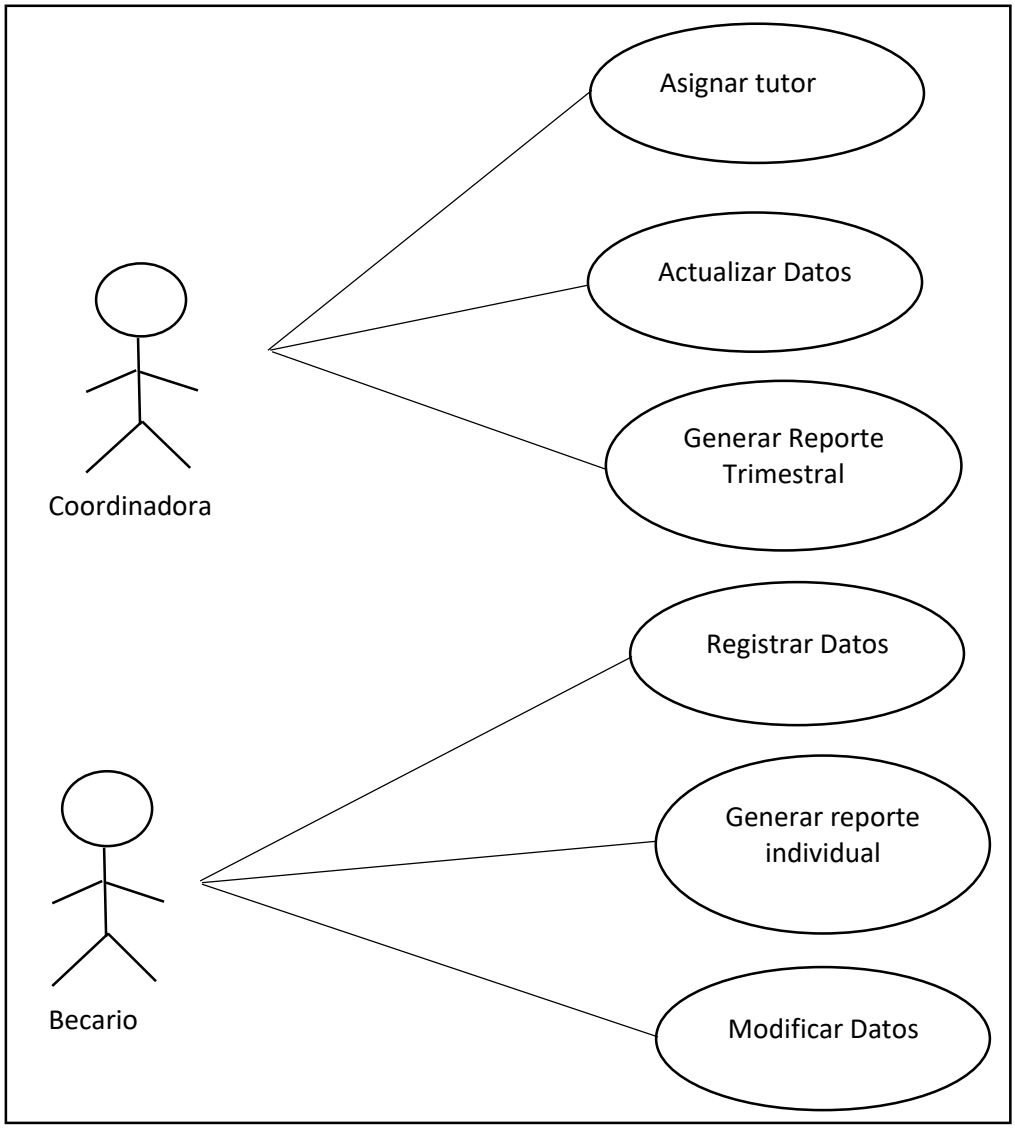


Figura 3. Diagrama de casos de uso.

6.2 Diseño de la base de datos

Debido a que esta aplicación web requiere el manejo de información, es necesario alojar dicha información en una base de datos, para este proyeco se diseñó la base de datos en MySQL Workbench 6.3, distribución gratuita. La estructura de la base de datos, cuyo nombre es 'becarios', consta de 5 tablas que de describen a continuación:

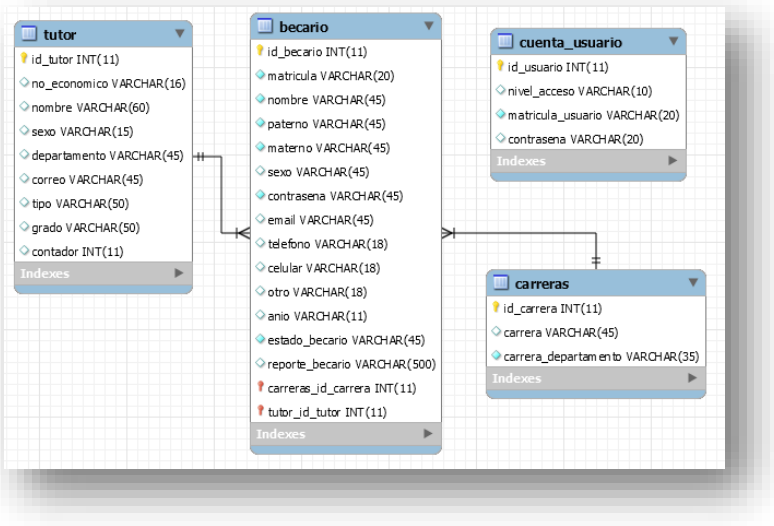


Figura 4. Diseño de la base de datos.

6.2.1 Tabla becario

Esta tabla es la más extensa, y podríamos decir la más importante, como su nombre lo indica contiene la información del becario, dicha información requerida es: matrícula, nombre(s), apellido paterno, apellido materno, sexo, correo electrónico, teléfono de casa, celular, teléfono alternativo, año de becario, estado de becario (éste campo no aparece visible a la hora de que se registra un nuevo becario, se autocompleta en la base de datos con el valor de 'activo'. La coordinadora es quien determina si es activo o se da de baja y puede modificarlo en el módulo "Actualizar Becario").

La llave primaria lleva como nombre 'id_becario', se propuso un id autoincrementable para no involucrar a la matrícula como llave primaria, lo cierto es que la matrícula sirve como identificador para cada alumno y es irrepetible, sin embargo, al ser llave primaria es más complicado la manipulación y uso de ese campo. Cabe recalcar que incluso el campo 'matrícula', junto con todos los demás campos de valor numérico como los teléfonos y año de becario, son almacenados de tipo VARCHAR, para que igual sea más fácil la manipulación de esos datos a la hora de hacer consultas mediante Servlets.

Cuenta con dos llaves foráneas, 'carreras_id_carrera' y 'tutor_id_tutor' para hacer la relación con esas tablas.

6.2.2 Tabla tutor

La tabla 'tutor' tiene los atributos 'id_tutor' como llave primaria, 'no_economico' que a pesar de ser un identificador para el tutor resulta más complicado utilizarlo como llave primaria, nombre (en este caso es el nombre completo empezando por apellido paterno), sexo, departamento (este campo resulta de gran utilidad a la hora de hacer la

asignación de tutores), correo, tipo, grado, contador (se agregó este campo para llevar un conteo de cuántos becarios le han sido asignados a cada tutor.

6.2.3 Tabla carrera

Nuevamente la llave primaria de esta tabla es un id autoincrementable llamado ‘id_carrera’, cuenta además con el campo ‘carrera’, donde se encuentran registradas las 10 carreras de ingeniería de la división de CBI, el campo ‘carrera_departamento’. Debido a que son 5 departamentos y 10 carreras en la división, se propuso asignar 2 carreras a cada departamento, la asignación propuesta fue la siguiente:

Departamento	Carrera
Ciencias Básicas	Ingeniería Física
	Ingeniería Química
Electrónica	Ingeniería Electrónica
	Ingeniería Eléctrica
Energía	Ingeniería Mecánica
	Ingeniería Metalúrgica
Materiales	Ingeniería Ambiental
	Ingeniería Civil
Sistemas	Ingeniería en Computación
	Ingeniería Industrial

Tabla 1. Relación Departamento-Carrera.

Esta relación se hizo con el propósito de facilitar la selección de tutores, así, dependiendo de la carrera del alumno, sólo le aparecerán a la coordinadora los tutores pertenecientes al departamento relacionado como posibles candidatos.

6.2.4 Tabla cuenta_usuario

Esta tabla se utiliza para realizar el inicio de sesión, ya sea como becario o como coordinador. Cuenta con un id autoincrementable llamado ‘id_usuario’ como llave primaria, ‘nivel_acceso’, ‘matricula_usuario’ y ‘contrasena’.

En el campo ‘nivel_acceso’ se define si se trata de un becario con el valor ‘USER’, éste valor es agregado por defecto para todos los que se registren por primera vez. El valor definido para la coordinadora es ‘ADMIN’, éste valor ha sido asignado con anterioridad directamente en la base de datos.

Figura 5. Recuadro de Inicio de seión.

Para acceder al sistema se requiere de un usuario y contraseña, al momento de registrarse, sólo se solicita asignar una contraseña pues el campo usuario es la matrícula de cada becario. En el caso de la coordinadora, su 'usuario' es su número económico, y la contraseña se ha generado y almacenado en la base de datos con anterioridad.

6.3 Modelo Vista Controlador

Una vez lista la base de datos, pasamos a la programación del sistema web basándonos en el modelo vista controlador logrando así tener una interfaz de usuario donde realice peticiones (vista), administrar las peticiones y ser el intermediario entre el usuario y los datos (controlador) y manipular los datos y realizar la conexión con la base de datos. El siguiente diagrama define la organización de este sistema de asignación de tutores:

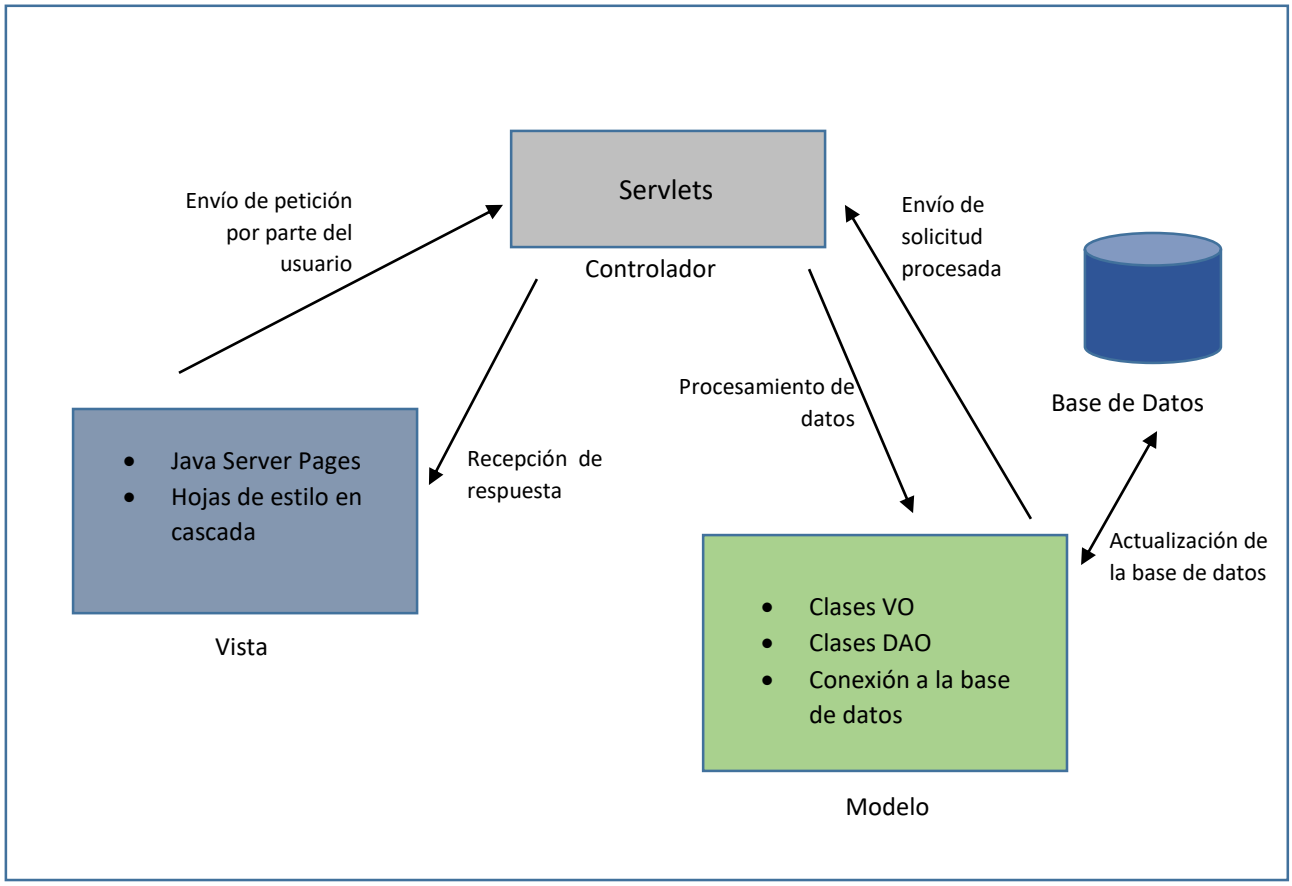


Figura 6. Diagrama de modelo vista controlador.

6.3.1 Modelo

En esta es la capa del modelo se trabaja con los datos, debido a eso contendrá mecanismo para ingresar a la información y también para actualizar su estado accediendo a la base de datos. Las clases del modelo contendrán funciones que nos ayudaran a mostrar, insertar, actualizar y eliminar información de la base de datos, eso se verá reflejado en las clases DAO y VO. Como se muestra en la figura 8.

Tanto los Servlets como los JSP son tecnologías de Java, por lo que el modelo mostrado con anterioridad, con sus respetivas clases, se programaron en el entorno de desarrollo NetBeans IDE 8.2 de distribución libre. Las librerías necesarias para el buen funcionamiento de la aplicación, igual de distribución libre, JDK 8.1 (que viene por defecto), Apache TomCat 8.0.27 (para el servidor web), para acceso y manipulación de la base de datos se requiere la librería MySQL JDBC Driver, y para la gestión de archivos, utilizados en los módulos de los reportes, se requiere de las librerías common-fileuploads y common-io.

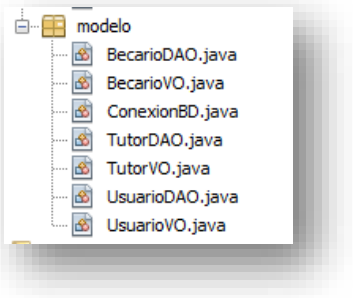


Figura 7. Clases VO y DAO, y de conexión.

6.3.1.1 Acceso a la base de datos

Esta clase perteneciente a la capa de modelo, que llamaremos “ConexionBD”, es la encargada de acceder a la base de datos para consulta, actualización, creación y eliminación de datos. A continuación se muestra parte del código de dicha clase, el resto del código se encuentra en el apéndice:

```
static public String bd = "becarios";
static public String login = "root";
static public String password = "";
static public String url = "jdbc:mysql://localhost:3306/"+bd;
```

En este fragmento se indica la base de datos, el usuario y contraseña de MySQL y la dirección que en este caso es localhost, pero podría ser la IP de la máquina donde se encuentre alojada la base de datos.

La aplicación al ser ejecutada con errores en alguno de estos campos mencionados puede mostrar las ventanas del servicio, pero no podrá acceder a la información de la base de datos.

6.3.1.2 Clases VO

En las clases Value Object (VO) se definen los atributos, el constructor y los métodos getter y setter.

Se ha creado una clase VO perteneciente a cada tabla de la base de datos, con excepción de la tabla ‘log_actividades’ puesto que su llenado depende de las acciones que se vayan realizando en el sistema, y la tabla ‘carreras’ a la cual no se le agregará información por parte del sistema. Dicho esto, las clases VO que contiene este proyecto son “BecarioVO”, “TutorVO” y “UsuarioVO”.

6.3.1.3 Clases DAO

La capa DAO contiene los métodos CRUD (create, read, update, delete), se tiene un DAO para cada clase VO lo que nos deja con las clases “BecarioDAO”,

“TutorDAO” y ”UsuarioDAO”. La estructura para cada método de estas clases es similar, tomemos como ejemlo el método *loginUsuario,* el resto del código se encuentra en el apéndice 11.2.3.1 Clase “BecarioDAO”:

```
try {
       ConexionBD c=new ConexionBD();
       Connection con=c.getConexion();

            Statement user;
            user = con.createStatement();
ResultSet log= user.executeQuery("select * from  cuenta_usuario where
matricula_usuario='"+unUsuario+"' and contrasena='"+contrasena+"';");

          while(log.next()){
          usuario=new UsuarioVO(log.getString("matricula_usuario") ,
log.getString("nivel_acceso"),log.getString("contrasena"));
                        }
                            user.close();
                               log.close();
                               con.close();
                               c.cerrarConexion();
                      c.cerrarConexion();
              } catch (SQLException e)
```

Primero realizamos la conexión a la base de datos haciendo uso de nuestra clase ConexionBD, una vez conectados creamos la sentencia de MySQL, Todo esto dentro de un try catch para caer en una excepción de SQL si es que no se logra establecer la conexión.

La clase “**BecarioDAO**” cuyos métodos son *agregarBecario, consultarBecario, recuperarContrasenia, actualizarBecario, actualizarTutor Becario, eliminarBecario,* 15 métodos de *consultaBecarioGeneral* debido a las 16 posibles combinaciones que se generan en el módulo de Consulta Becario, y *reporteBecario.*

La clase “**TutorDAO**” contiene los métodos *consultaTutorPorDepartamento* y el método *consultaTutores.*

Y la clase “**UsuarioDAO**” contiene el método *loginUsuario.*

6.3.2 Controlador

Debido a que esta es la capa intermedia carga el modelo de datos (las clases DAO.java) e invoca a la vista (JSP), esta capa está construida con Servlets, la palabra *servlet* deriva de otra anterior, applet, que se refiere a pequeños programas que se ejecutan en el contexto de un navegador web.

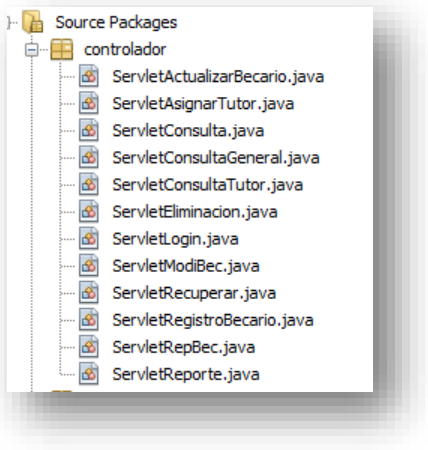


Figura 8. Controlador definido mediante Servlets.

6.3.2.1 Servlets

Este programa cuenta con 9 clases Servlet las cuales están mencionadas en la figura 9, el nombre que se les ha asignado nos ayuda a entender su funcionamiento. Debido a que son el paso intermedio entre la vista (HTML, JSP) y el modelo (Java) las clases Servlet requieren en su mayoría de las siguientes librerías, en el apéndice 11.3.1 Servlets se encuentra el resto del código:

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

Por lo general los Servlets procesan las peticiones GET

```
protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {

         response.setContentType("text/html;charset=UTF-8");
         request.setCharacterEncoding("UTF-8");

          String usuario=request.getParameter("_user");
          String contra=request.getParameter("_passw");

          UsuarioVO unUsuario=UsuarioDAO.loginUsuario(usuario,contra);
          BecarioVO unBecario=BecarioDAO.consultarBecario(usuario);
          TutorVO unTutor=TutorDAO.consultaTutores(usuario);
```

Aquí se puede ver que para la manipulación de datos requiere acceder a las clases DAO y VO, y para enviar la respuesta lo hace a través de HTML.

6.3.3 Vista

El usuario interactúa con esta capa, es donde se muestran los formularios y realiza las peticiones que ofrece el sistema, como herramienta en esta capa tenemos los Java Server Pages o JSP que en este proyecto se han diseñado 28 clases JSP, cada una de ellas representa una ventana, desde la página de inicio y cada actualización en las ventanas tras realizar una acción. La figura 9 muestra el listado de dichas clases.

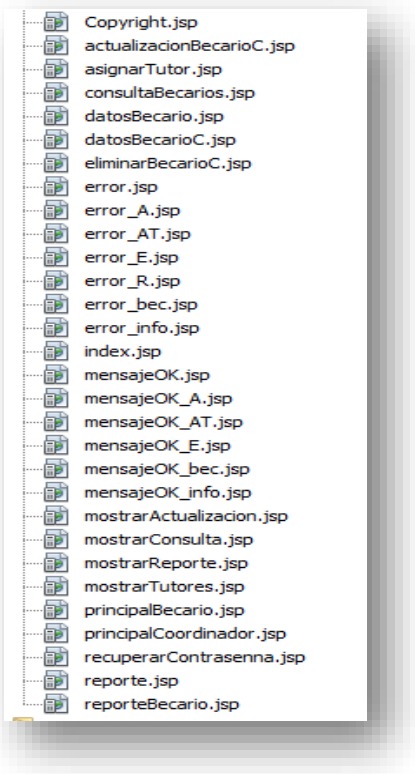


Figura 9. Clases JSP para la vista.

6.3.3.1 HTML 5 JSP

Las páginas estáticas están diseñadas en HTML, para este proyecto se utilizó HTML 5 que contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance, los servlets son utilizados para realizar páginas dinámicas. Ya que los Servlet utilizan el método GET para recibir la petición del usuario, en la capa de vista se utiliza el método POST. A continuación un fragmento de código donde se utiliza el método POST y se hace referencia al Servlet encargado de procesar la solicitud en la clase *index.html*:

```
<form action=”ServletLogin” method=”post>
```

6.3.3.2 CSS

Las Hojas de estilo en cascada (CSS) son las que le dan el formato (color, tamaño y tipo de fuente, estilo de las tablas, etc), se puede dar el formato dentro de la misma clase html, pero al ser tantas clases se tendría que repetir el formato en cada una, es por eso que se utilizan CSS y debido a que el formato es homogéneo resulta más útil y sencillo, sólo se requirió de 3 clases CSS: “general”, “menu” y “tablas”. A continuación se muestra un fragmento del estilo general, el resto del código se encuentra en el apéndice 11.4.2 general.css:

```
*{
color:#000000;
margin:0px;
font-family:Verdana;
}
body{
background-color:#d6d5d5;
text-align:center;
}
#envase{
max-width:1200px;
min-width:320px;
background-color:#FFFFFF;
text-align:left;
margin:auto;
```

El módulo de asignación de tutores se desarrolló en lenguaje Java con JSP/Servlet. La vista se implementó en HTML 5 con hojas de estilo en cascada CSS, la conexión a la base de datos se diseñó en MySQL, se utilizó servidor TomCat para alojar la aplicación web.

El navegador con el que se probó este sistema web es Google Chrome, de descarga gratuita. La venta principal se muestra de la siguiente manera:

Figura 10. Ventana principal.

6.4 Módulos para la coordinadora

La coordinadora cuenta con más privilegios que el becario, puede acceder y modificar la información del becario, revisar el reporte del becario, puede eliminar a algún becario, consultar los becarios mediante ciertos filtros como carrera, tutor, año de becario y estado del becario y el módulo de asignación de tutor. La siguiente pantalla muestra el inicio de sesión como coordinador.

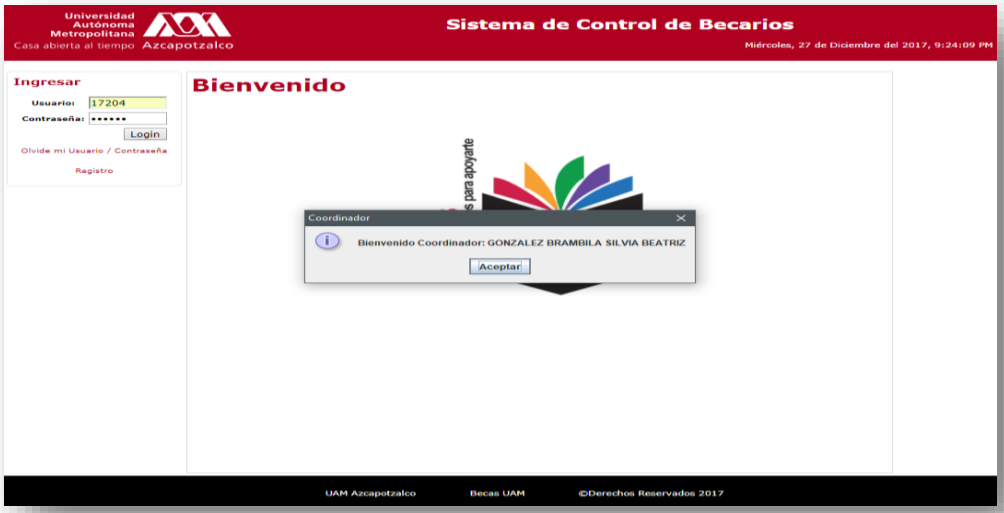


Figura 11. Inicio de sesión como coordinador.

Una vez iniciada la sesión como coordinador, se mostrará la ventana de inicio como coordinador, en lugar del recuadro de *Log in* se muestra un recuadro con la información del coordinador, como se muestra en la figura 12.

Figura 12. Ventana de inicio como coordinador.

6.4.1 Actualizar datos

La coordinadora puede acceder a la información con la que se registran los becarios y modificar ciertos campos como números de teléfono, año de becario y estado de becario. Para realizar dichas operaciones primero se debe ingresar la matrícula del alumno a modificar, como se muestra en la figura 13.

Figura 13. Pantalla de actualización de becario, búsqueda.

Una vez agregada la matricula, y si es una matrícula válida, se despliega la siguiente ventana de la figura 14, donde se pueden modificar todos los campos, con excepción de la matrícula.

Figura 14. Pantalla de actualización de becario, formulario.

6.4.2 Asignar tutor

En el módulo de asignación de tutor, se localiza al becario al que se le requiera asignar un tutor mediante su matrícula, como lo muestra la figura 15.

Figura 15. Pantalla de asignación de tutor, búsqueda.

Posteriormente se despliega una lista con los posibles tutores del departamento al que pertenece su carrera (en la tabla 1 se muestra la relación utilizada para la asignación de carreras por departamento), en la misma lista, aparece el número de becarios asignados a cada tutor para evitar una mala distribución de asignación, así, el tutor con menos becarios asignados tenga será un buen candidato para futuros becarios, como lo muestra la figura 16.

Figura 16. Pantalla de asignación de tutor, lista de tutores.

Éste módulo es exclusivo para la coordinadora, si desea hacer cambio de tutor sólo debe ingresar la matrícula del becario y reasignarle un tutor, las opciones que se desplieguen cumplirán con las mismas condiciones de un becario nuevo.

6.4.3 Generar reporte

Se despilega una tabla con la información del becario consultado, incluyendo su último reporte de acuerdo a los comentarios de su tutor, como lo muestra la figura 17.

Figura 17. Reporte del becario.

6.4.4 Consulta de becarios

En esta pantalla se puenden consultar todos los becarios mediante diferentes filtros de búsqueda: estado de becario (alta, baja), año de becario, nombre de tutor y carrera. Para mostrar resultados es necesario marcar por lo menos una opción, la búsqueda la puede realizar con uno o más campos seleccionados, como lo muestra la figura 18.

Figura 18. Consulta de becarios.

Los resultados de dicha búsqueda se muestran en una tabla, la cual puede ser ordenada de forma ascendente o descendente, esta opción está disponible en cada columna de la tabla, con la opción de imprimir la tabla o toda la página, la figura 19 muestra los resultados de la búsqueda anterior.

Figura 19. Pantalla de consulta de becarios, resultados.

6.4.5 Eliminación de becario

El último módulo para el coordinador es el de eliminación de becario, donde el coordinador agregará la matrícula del becario a eliminar y le aparecerá un mensaje de confirmación, ya que toda su informción será borrada de la base de datos permanentemente, como lo muesta la figura 20.

Figura 20. Pantalla de eliminación de becario.

6.5 Módulos disponibles para el becario

Como ya se ha mencionado, los privilegios del becario son menores a los de la coordinadora, el becario solo accede para revisar y modificar su información y para generar su erporte trimestral, la figura 21 muestra la pantalla de inicio para un becario.

Figura 21. Pantalla de inicio para un becario.

6.5.2 Modificar datos

Sección donde el becario realiza los cambios pertinentes a su información personal, como números de teléfono y correo electrónico, sin embargo no puede modificar su año de becario, tutor ni estado de becario, la figura 22 muestra la vista de dicho formulario.

Figura 22. Pantalla de modificación de datos del beacrio.

6.5.1 Generar reporte individual

Sección donde el becario generará un reporte de acuerdo a lo sucedido en sus tutorías, en esta sección puede escribir su reporte directo en un área de texto mientras que en otra sección le muestra su último reporte, una vez actualizando esta información se mostrará el último reporte enviado, como lo muestra la figura 23.

Figura 23. Pantalla de reporte de becario.

7. Resultados

Una vez concluido el desarrollo del servicio web, y la configuración y llenado de la base de datos, vienen las pruebas tanto de becario como de Coordinador, probando sus respectivos módulos. Empecemos con la pantalla de inicio:

Figura 24. Pantalla de inicio del Sistema de Control de Becarios.

Estando en ésta pantalla hay tres acciones posibles que se describen a continuación:

La primera opción es ingresar al sistema con un Usuario y contraseña, pero si se intenta ingresar, dando click en el botón Login sin haber llenado los campos, aparece el mensaje que se muestra en la figura 25.

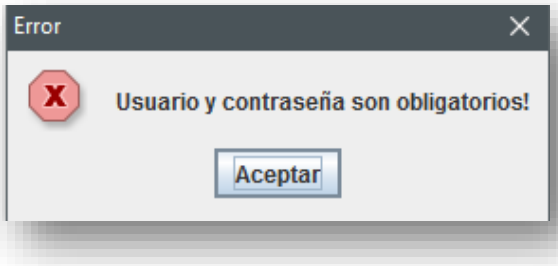


Figura 25. Mensaje de error debido a campos vacíos.

Si se intenta ingresar pero la matricula no coinside con la contraseña, o la matrícula no se encuentra registrada aparecerá el mensaje de la figura 26.

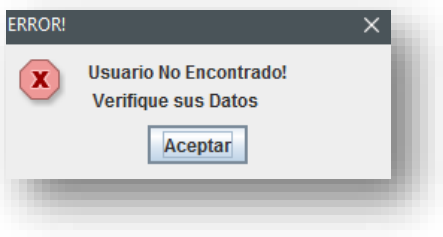


Figura 26. Mensaje de error debido a datos incorrectos.

7.1 Registro de un nuevo becario

Supongamos que la matrícula ingresada no se encuentra registrada en el sistema, entonces procedemos a registrar un nuevo usuario en el sistema. Registraremos al alumno Oliver Saucedo Vargas con matrícula 2112043966, con teléfono de casa 22070091, celular 5551589630 y otro teléfono 18586892, correo electrónico olivers1590@gmail.com como datos personales, en cuanto a información académica será la carrera de ingeniería en Computación, con beca desde el año 2015, estado activo y cuya contraseña será 1590, como lo muestra la figura 27.

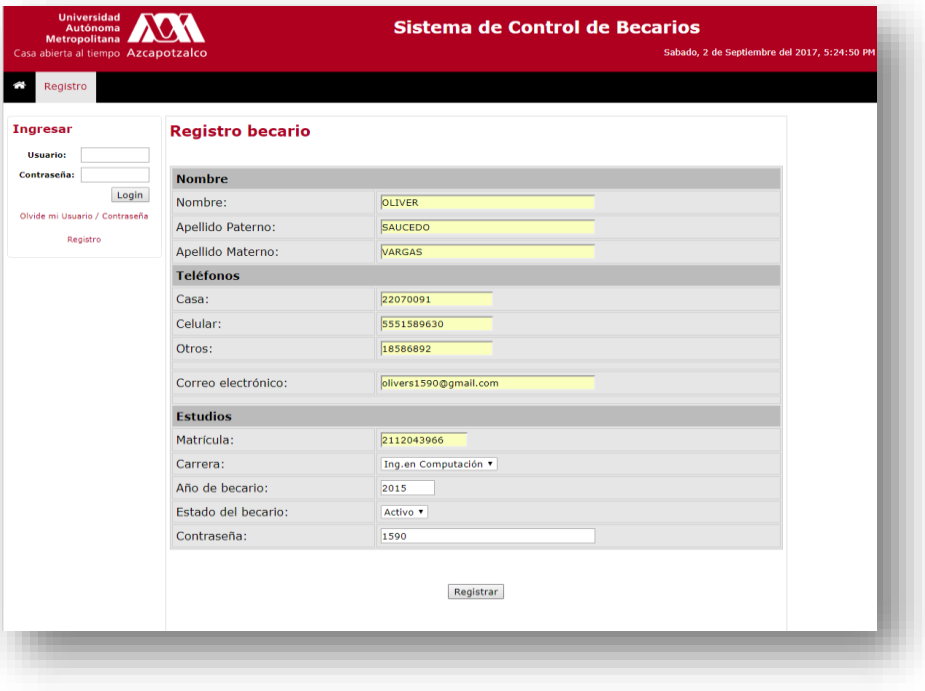


Figura 27. Pantalla de Registro de becario, Oliver Saucedo Vargas.

Si hemos llenado los campos correctamente, y de acuerdo con la información anterior podemos notar que el registro del alumno Oliver Saucedo Vargas fue exitoso, como lo muestra la figura 28.

Figura 28. Mensaje de registro exitoso.

Ahora intentaremos registrar a otro alumno, esta vez al alumno José Perez López con duplicidad de la matricula 2112043966 pero todos los demás campos diferentes, como lo muestra la figura 29.

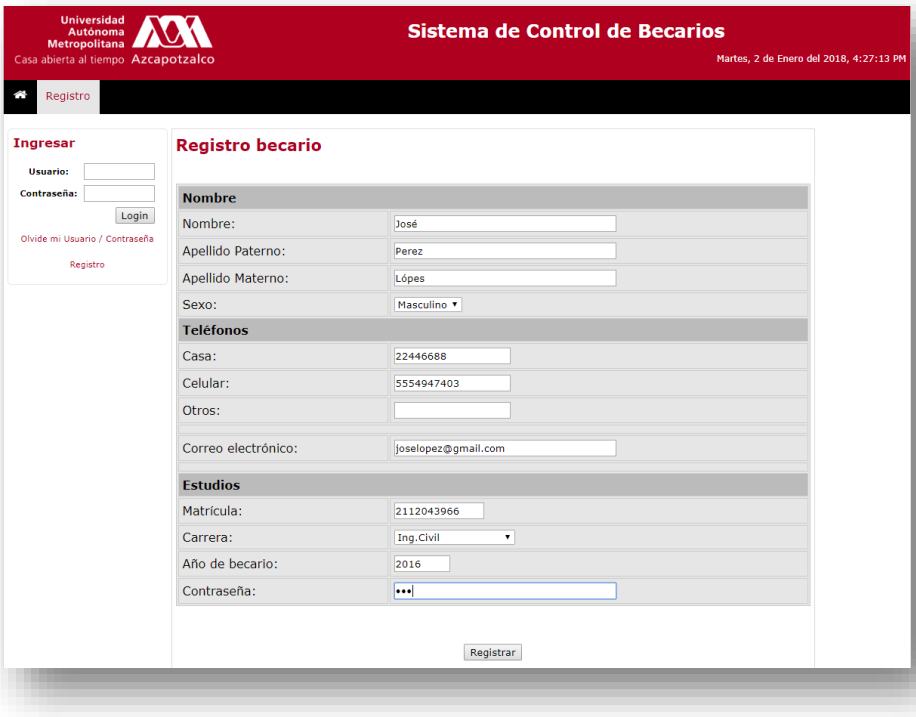


Figura 29. Pantalla de registro de becario, José Perez López.

Al intentar registrar otro alumno pero con una matrícula repetida nos apareció el mensaje de error de registro mostrado en la figura 30.

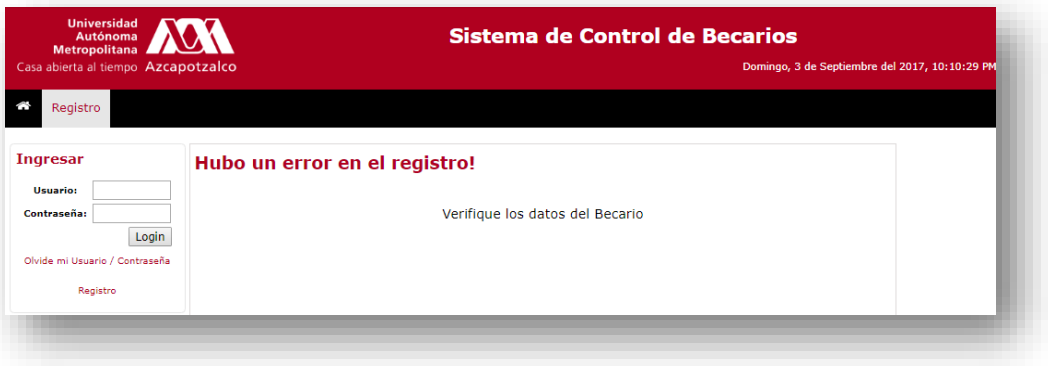


Figura 30. Mensaje de error de registro.

7.2 Sesión como becario

Una vez registrado en el sistema, ingresamos con la matrícula 2112043966 y la contraseña 1590 que definimos al llenar el registro para el alumno Oliver Saucedo Vargas y nos aparecerá un mensaje dándonos la bienvenida al sistema como lo muestra la figura 31.

Figura 31. Mensaje de Bienvenida al ingresar un becario.

Las opciones a las que tiene acceso un becario son “Datos Becario” y Reporte Becario”.

7.2.1 Datos Becario

Al ingresar al módulo de Datos Becario solo puede ver su información donde ya aparece su estado como becario, asignado por defecto con el valor “Alta” y modificado por la coordinadora, dependiendo de la situación de cada becario.

El becario solo puede modificar los campos de telefonos y correo electrónico, mientras que la coordinadora puede modificar el resto de los campos. En este ejemplo agregaremos otro telefono, el cual será 18586802, como lo muestra la figura 32.

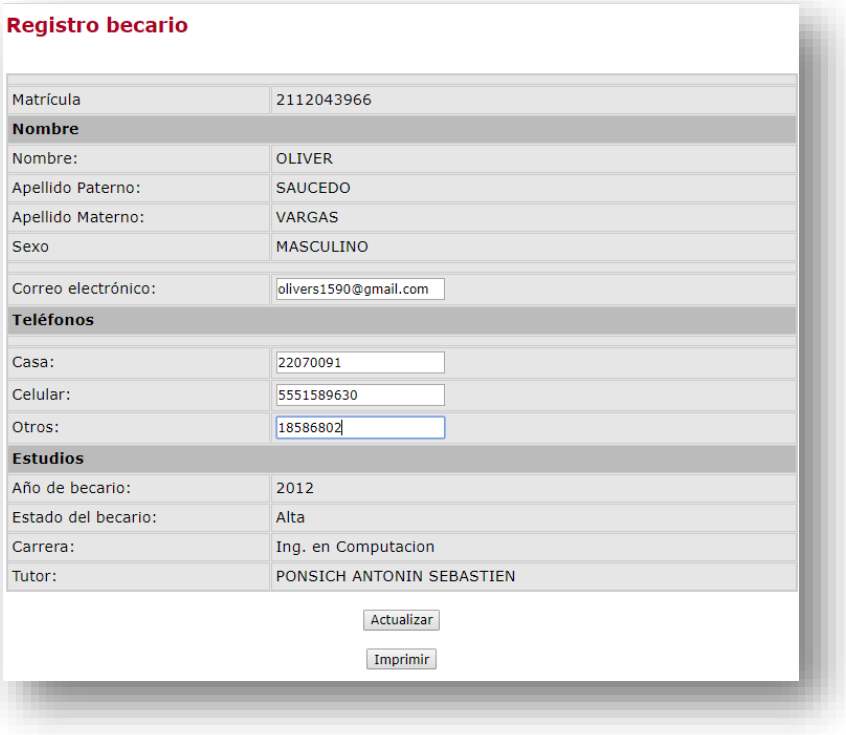


Figura 32. Consulta de datos mediante un usuario Becario.

7.2.2 Reporte Becario

Una de las responsabilidades de cada becario es generar un reporte sobre sus asesorías con su tutor, lo cual se realiza directamente en el área designada en este módulo, en un área de texto lo suficiente amplia, en este ejemplo escribiremos: “Reporte trimestral sobre asesorías, para la coordinadora.”, mientras que en un recuadro superior nos muestra nuestro último mensaje, por defecto muestra el mensaje: “El becario no ha realizado su reporte.”, como se muestra en la figura 33.

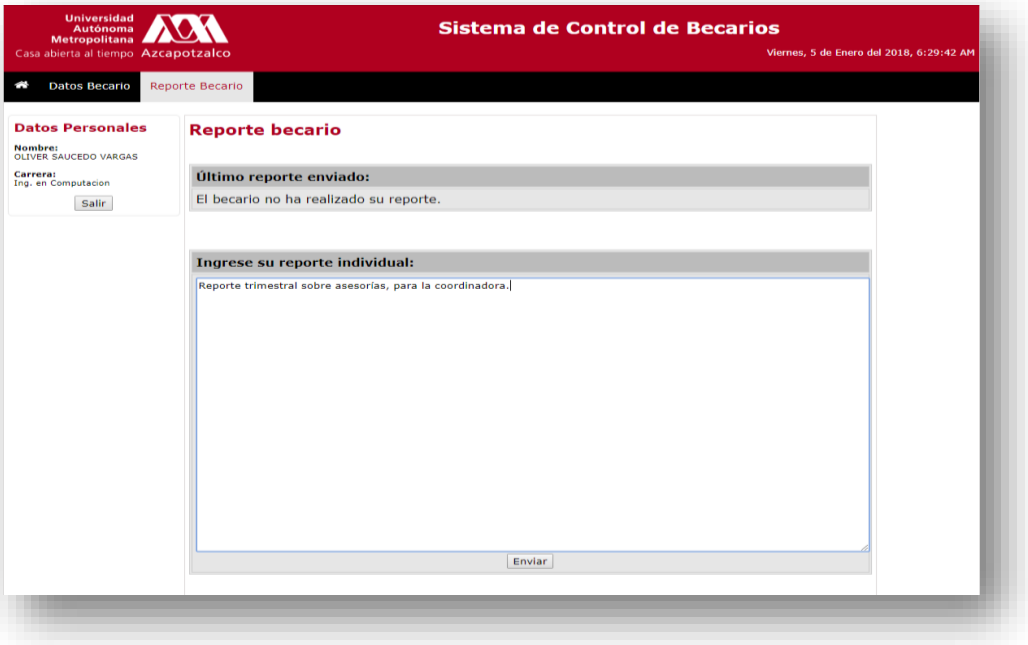


Figura 33. Pantalla de Reporte de becario.

7.3 Sesión como coordinador

La sesión como coordinador es más extensa y con más privileegios, la primer diferencia que puede no ser muy notoria es su registro, ya que no hay un formulario de registro dentro de esta aplicación, la gestión de datos del coordinador se hace directamente en la base de datos, más específicamente, dentro de la tabla 'cuenta_usuario' se agrega un nuevo usuario y en el campo 'nivel_acceso' se le asigna el valor 'ADMIN' como se muestra a continuación en la figura 34.

id_usuario	nivel_acceso	matricula_usuario	contrasena
1	ADMIN	17204	123456
2	USER	206202646	123456
4	USER	1111	1590
5	USER	2112043966	1590
6	USER	343434	123456
7	USER	8787878	1234567
8	USER	2211334466	ix
NULL	NULL	NULL	NULL

Figura 34. Tabla 'cuenta_usuario' vista desde MySQL Workbench.

La matrícula en el caso del coordinador se tomó su número económico* que en este caso es 17204 para la coordinadora Silvia Beatríz Gonzalez Brambila y la contraseña es proporsionada por el usuario, en este caso es 123456, el id es autoincrementable.

* La información de la tabla 'tutor' fue proporcionada por la coordinación para fines del buen funcionamiento de este proyecto.

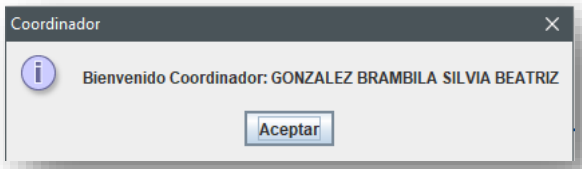


Figura 35. Mensaje de bienvenida al ingresar como coordinador.

Como se muestra en la figura 35, es muy similar al ingresar un becario, de hecho tanto becarios como coordinadores utilizan el mismo portal de *Log in,* lo que hace la diferencia es el nivel de acceso relacionado a cada usuario. Por lo tanto, si los datos son ingresados incorrectamente saldrá el mismo mensaje visto en la figura 26. De lo contrario, se habrá ingresado al módulo como coordinador, como se muestra en la figura 36.

Figura 36. Pantalla de Bienvenida, entrando como coordinador.

7.3.1 Actualización

Como coordinador, el primer paso para actualizar la información de algún becario es ingresar su matrícula como se muestra en la figura 37.

Figura 37. Plantilla de Actualización de becario.

Intentamos buscar al alumno cuy matrícula sea 222222222, y vemos que no existe ningún alumno registrado con dicha matrícula. Si la matrícula no coincide con ninguna registrada en la base de datos aparecerá el mensaje de error de la figura 38.

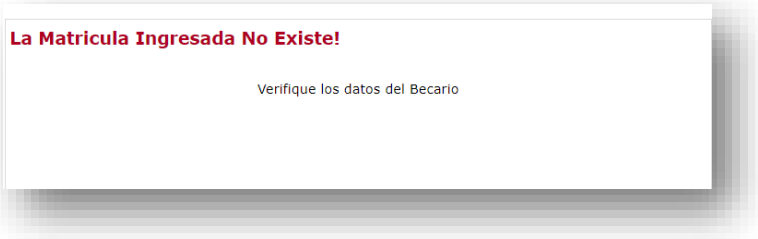


Figura 38. Mensaje de error debido a la matrícula buscada.

Ahora intentaremos buscar al alumno cuya matrícula sea 2112043966, debido a que es la matrícula del alumno previamente registrado, aparecerá el siguiente formulario y a diferencia del becario, el coordinador puede editar toda su información, incluyendo el estado de becario, con excepción de su matrícula, como se muestra en la figura 39.

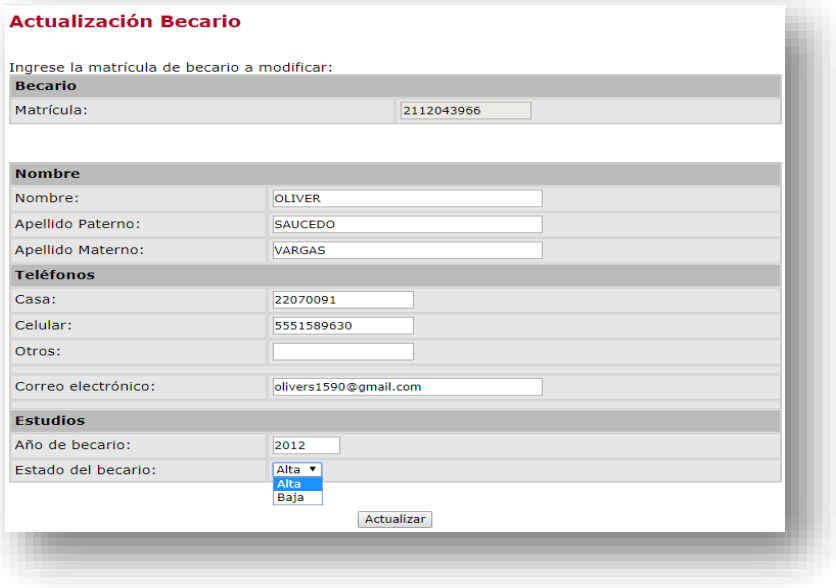


Figura 39. Pantalla de Actualización, formulario.

Una vez realizando los cambios deseados sobre la información del becario aparecerá el mensaje de la figura 40 de actualización exitosa, eso nos dice que la información ha sido modificada en la base de datos satisfactoriamente.

Figura 40. Mensaje de Actualización exitosa.

7.3.2 Asignación de tutor

El procedimiento es similar a la actualización de becario, primero debe introducirse la matrícula del becario al que se le asignará el tutor como se muestra en la figura 41, ya sea por primera vez o cambio de tutor. Si la matrícula no coincide se mostrará un mensaje de error como lo muestra la figura 38.

Figura 41. Pantalla de asignación de tutor.

Nuevamente ingresamos la matrícula 2112043966 para asignarle un tutor al alumno Oliver Saucedo Vargas. Una vez introducida una matrícula válida se mostrará la siguiente información, figura 42, una tabla con la información del becario seleccionado, que puede ya tener un tutor asignado, y otra con los profesores del departamento relacionado a su carrera.

Figura 42. Pantalla de asignación de tutor, con becario seleccionado.

Para esta prueba asignaremos al profesor “Ponsich Antonin Sebastien” que pertenece al departamento de Sistemas. La tabla muestra los tutores relacionados a la carrera del alumno, y en la columnoa derecha se muestran cuántos alumnos asignados tiene cada tutor, una vez que se reasigna un tutor, el contador se actualiza, esto sirve para que la asignación sea uniforme, como lo muestra la figura 43.

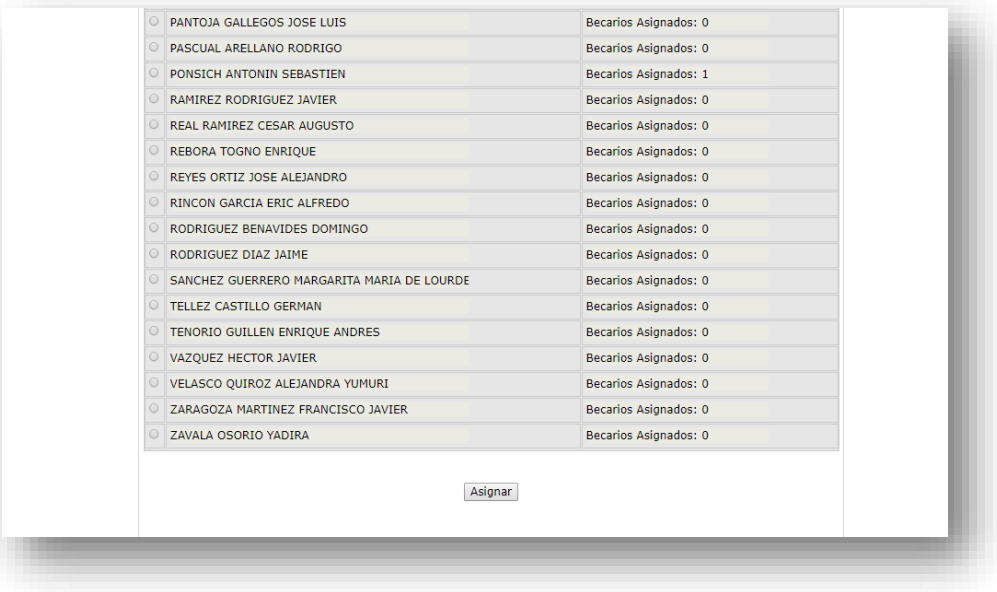


Figura 43. Selección de tutor en la pantalla de asignación de tutor.

Una vez asignado el tutor, aparecerá el mensaje de la figura 44, indicándonos que la información ha sido actualizada en la base de datos exitosamente.

Figura 44. Mensaje de asignación exitosa.

7.3.3 Reporte becario

En este módulo se despliega la información de cada becario, incluyendo su reporte, del mismo modo que en los demás módulos, el coordinador debe ingresar la matrícula del becario que desea consultar, como se muestra en la imagen 45, si la matrícula no coincide con la información de la base de datos aparecerá un mensaje de error como en la figura 38.

Figura 45. Pantalla de reporte de becario.

Al ingresar la matrícula de un becario se despliega una tabla con toda la información del becario incluido su reporte, también se tiene la opción de imprimir dicha información, en este ejemplo consultamos la información del alumno Oliver Saucedo Vargas, con matrícula 2112043966, y vemos que también se encuentra su reporte previamente cargado, como lo muestra la figura 46.

Figura 46. Información del becario.

7.3.4 Consulta de becarios

En este módulo de Consulta de Becarios se puede consultar los becarios por diferentes filtros, los cuales son: estado del becario, año de becario, nombre de tutor y carrera, como se muestra en la figura 47.

Figura 47. Pantalla de consulta de becarios.

La búsqueda se puede hacer seleccionando más de un campo como se muestra en la figura 48, esto nos da como resultado las 2^4 combinaciones binarias, eliminando solo la opción donde no se seleccionó ningun campo, en dado caso aparece el error de la figura 49.

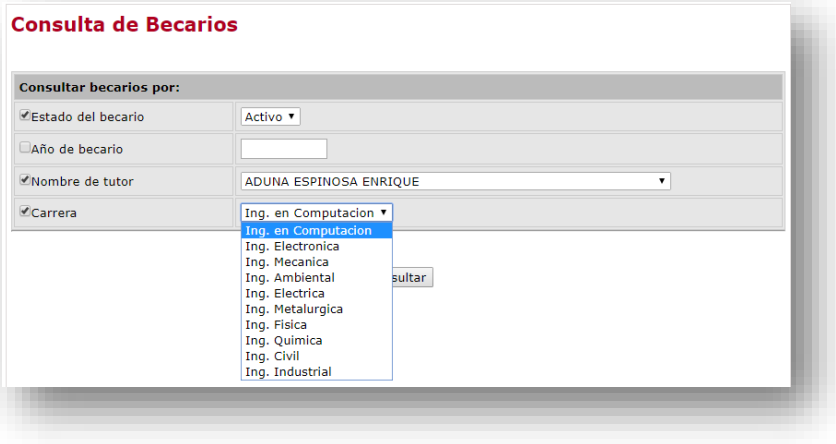


Figura 48. Selección múltiple en Consulta de Becarios.

Esto nos da como resultado 15 combinaciones en los campos de búsqueda, sin contar las opciones de campo. Lo cual hace esta búsqueda muy precisa al tener un gran número de filtros.

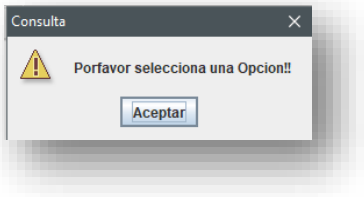


Figura 49. Mensaje de error al no seleccionar ningún campo.

Después de haber elegido nuestra combinación con sus respectivos valores aparece el siguiente mensaje indicando el número de becarios que coinciden con dicho filtrado.

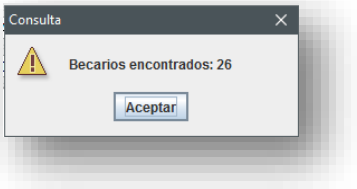
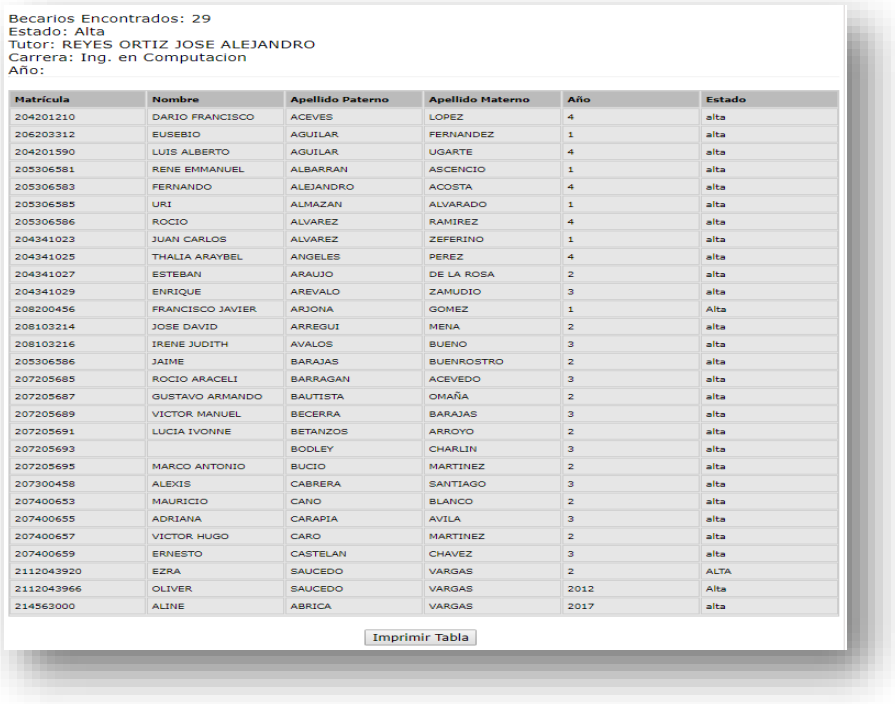


Figura 50. Mensaje de cantidad de becarios encontrados.

Haremos una prueba buscando todos los alumnos de la carrera “ingeniería en computación”, que estén dados de “alta” en el sistema y cuyo asesor sea el profesor “José Alejandro Reyes Ortíz”, podemos ver que hay 29 alumnos registrados que cumplen con dichos filtros. Cada columna de la tabla de resultados de la búsqueda puede ser ordenada e impresa para comodidad del coordinador como lo muestra la figura 51.



Becarios Encontrados: 29
Estado: Alta
Tutor: REYES ORTIZ JOSE ALEJANDRO
Carrera: Ing. en Computacion
Año:

Matrícula	Nombre	Apellido Paterno	Apellido Materno	Año	Estado
204201210	DARIO FRANCISCO	ACEVES	LOPEZ	4	alta
206203312	EUSEBIO	AGUILAR	FERNANDEZ	1	alta
204201590	LUIS ALBERTO	AGUILAR	UGARTE	4	alta
205306581	RENE EMMANUEL	ALBARRAN	ASCENCIO	1	alta
205306583	FERNANDO	ALEJANDRO	ACOSTA	4	alta
205306585	URI	ALMAZAN	ALVARADO	1	alta
205306586	ROCIO	ALVAREZ	RAMIREZ	4	alta
204341023	JUAN CARLOS	ALVAREZ	ZEFERINO	1	alta
204341025	THALIA ARAYBEL	ANGELES	PEREZ	4	alta
204341027	ESTEBAN	ARAUJO	DE LA ROSA	2	alta
204341029	ENRIQUE	AREVALO	ZAMUDIO	3	alta
208200456	FRANCISCO JAVIER	ARJONA	GOMEZ	1	Alta
208103214	JOSE DAVID	ARREGUI	MENA	2	alta
208103216	IRENE JUDITH	AVALOS	BUENO	3	alta
205306586	JAIME	BARAJAS	BUENROSTRO	2	alta
207205685	ROCIO ARACELI	BARRAGAN	ACEVEDO	3	alta
207205687	GUSTAVO ARMANDO	BAUTISTA	OMAÑA	2	alta
207205689	VICTOR MANUEL	BECERRA	BARAJAS	3	alta
207205691	LUCIA IVONNE	BETANZOS	ARROYO	2	alta
207205693		BODLEY	CHARLIN	3	alta
207205695	MARCO ANTONIO	BUCIO	MARTINEZ	2	alta
207300458	ALEXIS	CABRERA	SANTIAGO	3	alta
207400653	MAURICIO	CANO	BLANCO	2	alta
207400655	ADRIANA	CARAPIA	AVILA	3	alta
207400657	VICTOR HUGO	CARO	MARTINEZ	2	alta
207400659	ERNESTO	CASTELAN	CHAVEZ	3	alta
2112043920	EZRA	SAUCEDO	VARGAS	2	ALTA
2112043966	OLIVER	SAUCEDO	VARGAS	2012	Alta
214563000	ALINE	ABRICA	VARGAS	2017	alta

Imprimir Tabla

Figura 51. Lista de becarios encontrados en la consulta.

7.3.5 Eliminación de becario

El módulo de Eliminación de becario requiere de la matrícula del becario a eliminar, como se muestra en la figura 52.

Figura 52. Pantalla de eliminación de becario.

De no coincidir la matricula nos aparecerá el siguiente mensaje de error.

Figura 53. Mensaje de error, de eliminación de becario.

Como es bien sabido en bases de datos, una vez eliminando una tupla no se puede recuperar, es por eso que antes de eliminar a un becario aparece el siguiente mensaje de advertencia.

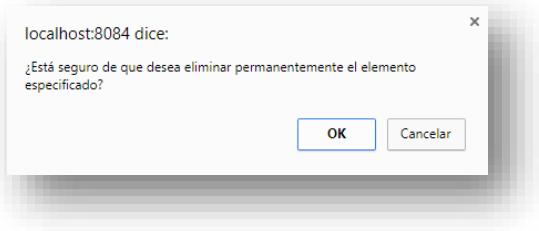


Figura 54. Mensaje emergente de confirmación.

Una vez confirmada la petición de eliminación de becario aparecerá el siguiente mensaje de operación exitosa, lo que nos indica que la base de datos se ha actualizado con dicha petición.

Figura 55. Mensaje de eliminación exitosa.

7.4 Recuperar contraseña

Ya que en la sección de actualización de la información por parte del becario no se puede cambiar la contraseña, si se le llega a olvidar debe ir a la página principal y acceder al enlace de “Olvidé mi usuario / contraseña” que se muestra a continuación.

Figura 56. Ventana de inicio.

Una vez accediendo a dicho enlace se mostrará la siguiente ventana, que requerirá la matrícula del becario, que para este ejemplo es “207400660” y su correo que utilizó al registrarse, el cual es “correo@uam.mx”, como se muestra en la figura 60. La coincidencia de ambos campos valida la recuperación de la contraseña, en caso contrario no habrá contraseña que mostrar.

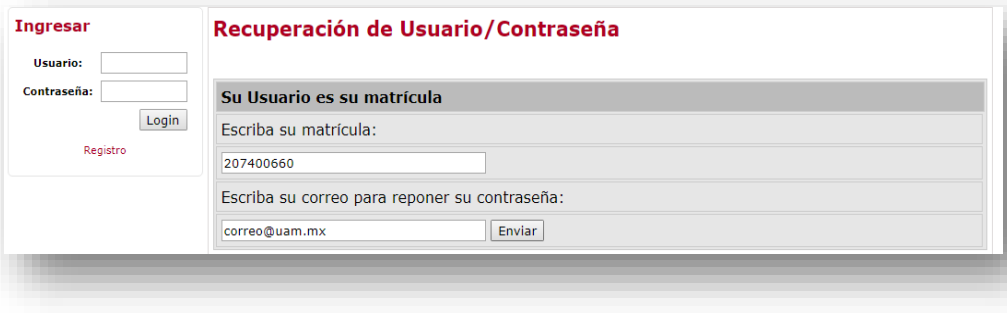


Figura 57. Pantalla de recuperación de Usuario/Contraseña.

Si se ingresan ambos campos correctamente aparecerá un campo emergente como el que muestra la figura 61, que contiene la contraseña del becario la contraseña correspondiente a los datos ingresados anteriormente es “39452”.

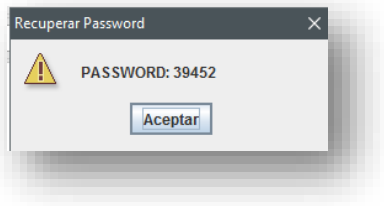


Figura 58. Mensaje de contraseña recuperada.

8. Análisis y discusión de resultados

Después de pruebas de asignación de tutores no se encontró ninguna falla, todos los registros fueron satisfactorios. El formulario le muestra a la coordinadora solo los profesores de la división de Ciencias Básicas e Ingeniería, y dentro de ese filtro, solamente propone como candidatos a ser tutor del becario a aquellos profesores que pertenezcan al departamento relacionado con la carrera de dicho becario. Una vez mostrando dichos profesores, se indica la cantidad de becarios asignados a cada profesor en el costado derecho del mismo formulario, para hacer una distribución uniforme de asignación de becarios por tutor.

La eliminación de becarios se vio reflejada en el sistema web y en la base de datos, al igual que la actualización de la información personal, y el registro de becarios.

La consulta resultó eficiente gracias a los filtros con los que cuenta pues puede ser por estado del becario (activo, baja), año de becario, tutor y carrera, dando 16 posibles combinaciones de búsqueda. Lo cual resulta muy útil para realizar búsquedas más específicas de los becarios que cumplas con ciertas condiciones específicas, por ejemplo: *Encontrar todos los becarios que han sido dados de baja del año 2015 de la carrera Ingeniería en Computación.*

9. Conclusiones

A la conclusión de este proyecto se puede apreciar la simplicidad con la que un becario puede registrarse en el sistema y recibir a su tutor, mejor aún, la coordinadora puede gestionar la información de los becarios y realizar la asignación de sus respectivos tutores de una manera más rápida y relacionada con cada becario en particular.

Se pretende agilizar el proceso de registro y disminuir los tiempos de asignación de tutor para así tener un mejor aprovechamiento de los recursos.

El manejo del sistema no tiene complejidad para ninguno de los usuarios ya sea becario, tutor o coordinador. Con esto se esquiva la necesidad de capacitaciones exhaustivas para los usuarios así como el gasto de recursos que se requiere para las mismas.

Se implantaron diferentes niveles de usuarios con permisos distintos para cada uno logrando así evitar que los usuarios, cuyos permisos más restringidos son los becarios, puedan manipular la información a la que solo tiene acceso el coordinador quien tiene el acceso directo a la lectura, escritura y manipulación de los datos.

Con esto puedo concluir diciendo que los objetivos que se perseguían al inicio del proyecto fueron cumplidos satisfactoriamente.

10. Referencias bibliográficas

[1] "Becas Manutención. Alcances. Universidad Autónoma Metropolitana. UAM.", Becas.uam.mx. [Online]. Available:

http://www.becas.uam.mx/sepuam/alcances.html. [Accessed: 06- Nov- 2016].

[2] "Becas Manutención. Información importante. Universidad Autónoma Metropolitana. UAM.", Becas.uam.mx. [Online]. Available:

http://www.becas.uam.mx/sepuam/infoimportante.html. [Accessed: 06- Nov- 2016].

[3] J. Gutiérrez Zúñiga and M. Chanca de la Cruz, "Análisis, diseño e implementación de un sistema de información para el control de becarios y exbecarios de una asociación educativa", Tesis.pucp.edu.pe, 2016. [Online]. Available: http://tesis.pucp.edu.pe/repositorio/handle/123456789/329?show=full. [Accessed: 10- Nov- 2016].

[4] G. Bayer, B. Riveros and Y. Vega, "Relación de la asignación de personal de enfermería con indicadores de resultado de la calidad de la atención en unidades de cuidados intensivos de adultos", Aquichan.unisabana.edu.co, 2016. [Online]. Available: http://aquichan.unisabana.edu.co/index.php/aquichan/article/view/3928/html. [Accessed: 10- Nov- 2016].

[5] I. Rosas Torres, "Sistema Web para la identificación automática de aspectos académicos y de experiencia profesional en expedientes curriculares", Licenciatura, Universidad Autónoma Metropolitana Unidad Azcapotzalco, 2015.

[6] N. Castillo Franco, "Implementación de un sistema web para dar seguimiento a las visitas realizadas a entidades financieras por parte de la Comisión Nacional Bancaria y de Valores", Licenciatura, Universidad Autónoma Metropolitana Unidad Azcapotzalco, 2014.

[7] J. Caballero and C. Arboleda, "Asignación de horarios de clases universitarias mediante algoritmos evolutivos", Educacioneningenieria.org, 2016. [Online]. Available:

http://www.educacioneningenieria.org/index.php/edi/article/view/15. [Accessed: 10- Nov- 2016].

[8] H. Castrillón, C. González and D. López, "ARCHITECTURAL MODEL FOR INTEROPERABILITY BETWEEN HEALTHCARE SERVICE PROVIDERS IN COLOMBIA", Scielo.org.co, 2012. [Online]. Available: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1909-97622012000200004&lng=en&nrm=iso&tlng=es. [Accessed: 11- Nov- 2016].

[9] S. Quishpe, D. Rivero and F. Rivas, "Diseño de un Sistema Web para Asignación de Becas con Integración e Interoperabilidad en Base a un Bus de Servicios", Revistapolitecnica.epn.edu.ec, 2016. [Online]. Available: http://revistapolitecnica.epn.edu.ec/ojs2/index.php/revista_politecnica2/article/view/750. [Accessed: 06- Nov- 2016].

[10] 2017. [Online]. Available: http://www.petersen.com.py/uploads/documentos/productos/18/archivo-de-educacion.pdf. [Accessed: 01- Sep- 2017].

[11] "¿Qué es Framework? - Definición en WhatIs.com", SearchDataCenter en Español, 2017. [Online]. Available: http://searchdatacenter.techtarget.com/es/definicion/Framework. [Accessed: 01- Sep- 2017].

[12] M. Alvarez, "Qué es JSP", DesarrolloWeb.com, 2017. [Online]. Available: https://desarrolloweb.com/articulos/831.php. [Accessed: 01- Sep- 2017].

[13] M. Alvarez, "Qué es MVC", DesarrolloWeb.com, 2017. [Online]. Available: https://desarrolloweb.com/articulos/que-es-mvc.html. [Accessed: 01- Sep- 2017].

[14] "Arquitectura Orientada a Servicios | SOA | Epicor", Epicor.com, 2017. [Online]. Available: http://www.epicor.com/lac/solutions/soa.aspx. [Accessed: 01- Sep- 2017].

[15] "Patrones de diseño en Java: MVC, DAO y DTO", Tutoriales de Programación Web con Java y PHP, 2017. [Online]. Available: http://www.ecodeup.com/patrones-de-diseno-en-java-mvc-dao-y-dto/. [Accessed: 01- Sep- 2017].

11. Apéndice

11.1 Script de la Base de Datos

```
        SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
```

11.1.1 Base de datos: `becarios`

```
--
DROP DATABASE IF EXISTS becarios;
CREATE DATABASE IF NOT EXISTS becarios
CHARACTER SET utf8 collate utf8_general_ci;
USE becarios;
--
```

11.1.2 Estructura de tabla para la tabla `becario`

```
--

CREATE TABLE IF NOT EXISTS `becario` (
  `id_becario` int(11) NOT NULL AUTO_INCREMENT,
  `matricula` varchar(20) NOT NULL,
  `nombre` varchar(45) NOT NULL,
  `paterno` varchar(45) NOT NULL,
  `materno` varchar(45) NOT NULL,
  `sexo` varchar(45) DEFAULT NULL,
  `contrasena` varchar(45) NOT NULL,
  `email` varchar(45) DEFAULT NULL,
  `telefono` varchar(18) DEFAULT NULL,
  `celular` varchar(18) DEFAULT NULL,
  `otro` varchar(18) DEFAULT NULL,
  `anio` varchar(11) DEFAULT NULL,
  `estado_becario` varchar(45) NOT NULL,
  `carreras_id_carrera` int(11) NOT NULL,
  `tutor_id_tutor` int(11) NOT NULL,
  PRIMARY KEY (`id_becario`,`carreras_id_carrera`,`tutor_id_tutor`),
  KEY `fk_becario_carreras_idx` (`carreras_id_carrera`),
  KEY `fk_becario_tutor1_idx` (`tutor_id_tutor`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=68 ;

--
```

11.1.3 Estructura de tabla para la tabla `carreras`

```
--

CREATE TABLE IF NOT EXISTS `carreras` (
  `id_carrera` int(11) NOT NULL AUTO_INCREMENT,
  `carrera` varchar(45) DEFAULT NULL,
  `carrera_departamento` varchar(35) NOT NULL,
  PRIMARY KEY (`id_carrera`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=11 ;
--
```

11.1.4 Estructura de tabla para la tabla `cuenta_usuario`

```
--

CREATE TABLE IF NOT EXISTS `cuenta_usuario` (
  `id_usuario` int(11) NOT NULL AUTO_INCREMENT,
  `nivel_acceso` varchar(10) DEFAULT NULL,
  `matricula_usuario` varchar(20) NOT NULL,
  `contrasena` varchar(20) DEFAULT NULL,
```

```
  PRIMARY KEY (`id_usuario`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=5 ;
----
```

11.1.5 Estructura de tabla para la tabla `tutor`

```
--
CREATE TABLE IF NOT EXISTS `tutor` (
  `id_tutor` int(11) NOT NULL AUTO_INCREMENT,
  `no_economico` varchar(16) DEFAULT NULL,
  `nombre` varchar(60) DEFAULT NULL,
  `sexo` varchar(15) DEFAULT NULL,
  `departamento` varchar(45) DEFAULT NULL,
  `correo` varchar(45) DEFAULT NULL,
  `tipo` varchar(50) DEFAULT NULL,
  `grado` varchar(50) DEFAULT NULL,
  `contador` int(11) DEFAULT NULL,
  PRIMARY KEY (`id_tutor`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=417 ;
--
-- Restricciones para tablas volcadas
--

--
-- Filtros para la tabla `becario`
--
ALTER TABLE `becario`
  ADD  CONSTRAINT  `fk_becario_carreras`  FOREIGN  KEY  (`carreras_id_carrera`)  REFERENCES
`carreras` (`id_carrera`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  ADD  CONSTRAINT  `fk_becario_tutor1`  FOREIGN  KEY  (`tutor_id_tutor`)  REFERENCES  `tutor`
(`id_tutor`) ON DELETE NO ACTION ON UPDATE NO ACTION;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

11.2 Modelo

11.2.1 Conexión a la base de datos

```
package modelo;
import java.sql.*;
import javax.swing.JOptionPane;
public class ConexionBD {
	static public String bd = "becarios";
	static public String login = "cliente";
	static public String password = "pass";
	static public String url = "jdbc:mysql://192.168.0.102:3306/"+bd;
   Connection conexion = null;
public ConexionBD() {
	try{
	    Class.forName("com.mysql.jdbc.Driver");
	    conexion = DriverManager.getConnection(url,login,password);
	      }catch(SQLException e){
	         e.printStackTrace();
	      }catch(ClassNotFoundException e){
	         e.printStackTrace();            }
	}
public Connection getConexion(){
	       return conexion;
	}
public void cerrarConexion(){
	try {
```

```
                conexion.close();
            } catch (SQLException e) {
                e.printStackTrace();} } }
```

11.2.2 Clases VO

11.2.2.1 Clase “BecarioVO”

```
package modelo;
public class BecarioVO {
    public BecarioVO(int matricula, String contrasena, String nombre,
String paterno, String materno, int telefono, int celular, int otro,
String email, String carrera, int anio, String estado_becario, TutorVO
tutor) {
        throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.
    }

    /**
     * @return the carrera
     */
    public String getCarrera() {
        return carrera;
    }

    /**
     * @param carrera the carrera to set
     */
    public void setCarrera(String carrera) {
        this.carrera = carrera;
    }

    /**
     * @return the nombre_tutor
     */
    public String getNombre_tutor() {
        return nombre_tutor;
    }

    /**
     * @param nombre_tutor the nombre_tutor to set
     */
    public void setNombre_tutor(String nombre_tutor) {
        this.nombre_tutor = nombre_tutor;
    }

    /**
     * @return the departamento_tutor
     */
    public String getDepartamento_tutor() {
        return departamento_tutor;
    }

    /**
     * @param departamento_tutor the departamento_tutor to set
     */
    public void setDepartamento_tutor(String departamento_tutor) {
        this.departamento_tutor = departamento_tutor;
    }

    /**
```

```
     * @return the correo_tutor
     */
    public String getCorreo_tutor() {
        return correo_tutor;
    }

    /**
     * @param correo_tutor the correo_tutor to set
     */
    public void setCorreo_tutor(String correo_tutor) {
        this.correo_tutor = correo_tutor;
    }

    public BecarioVO(String matricula, String contrasena, String nombre,
String paterno, String materno, String telefono, String celular, String
otro, String email, String carrera, String anio, String estado_becario,
TutorVO tutor) {
       // throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.
       this.matricula = matricula;
       this.contrasena = contrasena;
        this.nombre = nombre;
        this.paterno = paterno;
        this.materno = materno;
        this.telefono = telefono;
        this.celular = celular;
        this.otro = otro;
        this.email = email;
        this.carrera = carrera;
        this.anio = anio;
        this.estado_becario = estado_becario;
        this.tutor = tutor;

    }

    /**
     * @return the id_becario
     */
    public int getId_becario() {
        return id_becario;
    }

    /**
     * @param id_becario the id_becario to set
     */
    public void setId_becario(int id_becario) {
        this.id_becario = id_becario;
    }

    /**
     * @return the matricula
     */
    public String getMatricula() {
        return matricula;
    }

    /**
     * @param matricula the matricula to set
     */
    public void setMatricula(String matricula) {
        this.matricula = matricula;
```

```
}

/**
 * @return the nombre
 */
public String getNombre() {
    return nombre;
}

/**
 * @param nombre the nombre to set
 */
public void setNombre(String nombre) {
    this.nombre = nombre;
}

/**
 * @return the paterno
 */
public String getPaterno() {
    return paterno;
}

/**
 * @param paterno the paterno to set
 */
public void setPaterno(String paterno) {
    this.paterno = paterno;
}

/**
 * @return the materno
 */
public String getMaterno() {
    return materno;
}

/**
 * @param materno the materno to set
 */
public void setMaterno(String materno) {
    this.materno = materno;
}

/**
 * @return the sexo
 */
public String getSexo() {
    return sexo;
}

/**
 * @param sexo the sexo to set
 */
public void setSexo(String sexo) {
    this.sexo = sexo;
}

/**
 * @return the contrasena
 */
```

```
public String getContrasena() {
    return contrasena;
}

/**
 * @param contrasena the contrasena to set
 */
public void setContrasena(String contrasena) {
    this.contrasena = contrasena;
}

/**
 * @return the email
 */
public String getEmail() {
    return email;
}

/**
 * @param email the email to set
 */
public void setEmail(String email) {
    this.email = email;
}

/**
 * @return the telefono
 */
public String getTelefono() {
    return telefono;
}

/**
 * @param telefono the telefono to set
 */
public void setTelefono(String telefono) {
    this.telefono = telefono;
}

/**
 * @return the celular
 */
public String getCelular() {
    return celular;
}

/**
 * @param celular the celular to set
 */
public void setCelular(String celular) {
    this.celular = celular;
}

/**
 * @return the otro
 */
public String getOtro() {
    return otro;
}

/**
```

```
 * @param otro the otro to set
 */
public void setOtro(String otro) {
    this.otro = otro;
}

/**
 * @return the anio
 */
public String getAnio() {
    return anio;
}

/**
 * @param anio the anio to set
 */
public void setAnio(String anio) {
    this.anio = anio;
}

/**
 * @return the estado_becario
 */
public String getEstado_becario() {
    return estado_becario;
}

/**
 * @param estado_becario the estado_becario to set
 */
public void setEstado_becario(String estado_becario) {
    this.estado_becario = estado_becario;
}

/**
 * @return the carreras_id_carrera
 */
public int getCarreras_id_carrera() {
    return carreras_id_carrera;
}

/**
 * @param carreras_id_carrera the carreras_id_carrera to set
 */
public void setCarreras_id_carrera(int carreras_id_carrera) {
    this.carreras_id_carrera = carreras_id_carrera;
}

/**
 * @return the tutor
 */
public TutorVO getTutor() {
    return tutor;
}

/**
 * @param tutor the tutor to set
 */
public void setTutor(TutorVO tutor) {
    this.tutor = tutor;
}
```

```
    private int id_becario;
    private String matricula;
    private String nombre;
    private String paterno;
    private String materno;
    private String sexo;
    private String contrasena;
    private String email;
    private String telefono;
    private String celular;
    private String otro;
    private String anio;
    private String estado_becario;
    private int carreras_id_carrera;
    private int tutor_id_tutor;
    private TutorVO tutor;

     private String carrera;
     private String nombre_tutor;
     private String departamento_tutor;
     private String correo_tutor;
      private String carrera_departamento;
    //ActualizarBecario
   /* public BecarioVO(String matricula,String contrasena, String nombre,
String paterno, String materno,  String telefono, String celular, String
otro,String email,String carrera, String anio, String estado_becario,
TutorVO tutor) {

        this.matricula = matricula;
        this.contrasena = contrasena;
        this.nombre = nombre;
        this.paterno = paterno;
        this.materno = materno;
        this.telefono = telefono;
        this.celular = celular;
        this.otro = otro;
        this.email = email;
        this.carrera = carrera;
        this.anio = anio;
        this.estado_becario = estado_becario;

        this.tutor = tutor;

    }*/
    //Consulta Becario
     public BecarioVO(String matricula,String nombre,String paterno,String
materno,String sexo,String email,String telefono,String celular,String
otro,String anio,String estado_becario,String carrera,String
nombre_tutor,String departamento_tutor,String correo_tutor, String
carrera_departamento)
                        {

        this.matricula = matricula;
        this.nombre = nombre;
        this.paterno = paterno;
        this.materno = materno;
        this.sexo = sexo;
        this.email = email;
        this.telefono = telefono;
        this.celular = celular;
```

```
        this.otro = otro;
        this.anio = anio;
        this.estado_becario = estado_becario;
        this.carrera = carrera;
        this.nombre_tutor = nombre_tutor;
        this.departamento_tutor = departamento_tutor;
        this.correo_tutor = correo_tutor;
        this.carrera_departamento = carrera_departamento;

    }
     //Guardar Becario
    public BecarioVO(String matricula,String nombre,String paterno,String
materno,String sexo,String contrasena,String email,String telefono,String
celular,String otro,String anio,String estado_becario,int
carreras_id_carrera,int tutor_id_tutor) {

        this.matricula = matricula;
        this.nombre = nombre;
        this.paterno = paterno;
        this.materno = materno;
        this.sexo = sexo;
        this.contrasena = contrasena;
        this.email = email;
        this.telefono = telefono;
        this.celular = celular;
        this.otro = otro;
        this.anio = anio;
        this.estado_becario = estado_becario;
        this.carreras_id_carrera = carreras_id_carrera;
        this.tutor_id_tutor = tutor_id_tutor;


    }

     public BecarioVO(String matricula, String nombre, String paterno,
String materno, String sexo, String contrasena, String email, String
telefono, String celular, String otro, String anio, String estado_becario,
int carreras_id_carrera) {

        this.matricula = matricula;
        this.nombre = nombre;
        this.paterno = paterno;
        this.materno = materno;
        this.sexo = sexo;
        this.contrasena = contrasena;
        this.email = email;
        this.telefono = telefono;
        this.celular = celular;
        this.otro = otro;
        this.anio = anio;
        this.estado_becario = estado_becario;
        this.carreras_id_carrera = carreras_id_carrera;

    }


     public BecarioVO( String matricula, String nombre, String paterno,
String materno, String sexo, String contrasena, String email, String
telefono, String celular, String otro, String anio, String estado_becario)
{
```

```
        this.matricula = matricula;
        this.nombre = nombre;
        this.paterno = paterno;
        this.materno = materno;
        this.sexo = sexo;
        this.contrasena = contrasena;
        this.email = email;
        this.telefono = telefono;
        this.celular = celular;
        this.otro = otro;
        this.anio = anio;
        this.estado_becario = estado_becario;


    }

       public BecarioVO( String matricula,String email) {

       this.matricula = matricula;
       this.email = email;
       }

     public BecarioVO( String matricula, String nombre, String paterno,
String materno, String contrasena, String anio, String estado_becario) {

        this.matricula = matricula;
        this.nombre = nombre;
        this.paterno = paterno;
        this.materno = materno;
        this.contrasena = contrasena;
        this.anio = anio;
        this.estado_becario = estado_becario;


    }


     //Consulta General
      public BecarioVO( String matricula, String nombre, String paterno,
String materno, int carreras_id_carrera, String anio, String
estado_becario) {

        this.matricula = matricula;
        this.nombre = nombre;
        this.paterno = paterno;
        this.materno = materno;
        this.carreras_id_carrera = carreras_id_carrera;
        this.anio = anio;
        this.estado_becario = estado_becario;


    }

  //Actualizar Bercario


     public BecarioVO( String matricula, String nombre, String paterno,
String materno, String telefono, String celular,String otro, String email,
String anio) {

        this.matricula = matricula;
```

```
        this.nombre = nombre;
        this.paterno = paterno;
        this.materno = materno;
         this.telefono = telefono;
        this.celular = celular;
        this.otro = otro;
         this.email = email;
        this.anio = anio;

    }

    public BecarioVO(String matricula,TutorVO tutor) {
        this.matricula = matricula;

        this.tutor = tutor;
    }

    public BecarioVO(String matricula) {
        this.matricula = matricula;

    }

    public BecarioVO() {

    }


    @Override
    public String toString() {
        return "<td> " + " " + getMatricula() + " " + getNombre() + " " +
getPaterno() + " " + getMaterno() + " " + carreras_id_carrera + " " + anio
+ " " + estado_becario + " " + getTutor() + "</td>";
    }

    /**
     * @return the tutor_id_tutor
     */
    public int getTutor_id_tutor() {
        return tutor_id_tutor;
    }

    /**
     * @param tutor_id_tutor the tutor_id_tutor to set
     */
    public void setTutor_id_tutor(int tutor_id_tutor) {
        this.tutor_id_tutor = tutor_id_tutor;
    }

    /**
     * @return the carrera_departamento
     */
    public String getCarrera_departamento() {
        return carrera_departamento;
    }

    /**
     * @param carrera_departamento the carrera_departamento to set
     */
    public void setCarrera_departamento(String carrera_departamento) {
        this.carrera_departamento = carrera_departamento;
    }
```

```
}
```

11.2.2.2 Clase “TutorVO”

```
package modelo;
public class TutorVO {
    private int id_tutor;
    private String no_economico;
    private String nombre;
    private String sexo;
    private String departamento;
    private String correo;
    private String tipo;
    private String grado;
    private int contador;
    private int tota;

    public TutorVO(int id_tutor, String no_economico, String nombre,
String sexo, String departamento, String correo, String tipo, String
grado, int contador) {
        this.id_tutor= id_tutor;
        this.no_economico = no_economico;
        this.nombre = nombre;
        this.sexo = sexo;
        this.departamento = departamento;
        this.correo = correo;
        this.tipo = tipo;
        this.grado = grado;
        this.contador = contador;
    }

    public TutorVO(int id_tutor, String no_economico, String nombre,
String sexo, String departamento, String correo, String tipo, String
grado) {
        this.id_tutor= id_tutor;
        this.no_economico = no_economico;
        this.nombre = nombre;
        this.sexo = sexo;
        this.departamento = departamento;
        this.correo = correo;
        this.tipo = tipo;
        this.grado = grado;
    }

    public TutorVO(String no_economico, String nombre, String sexo, String
departamento, String correo, String tipo, String grado) {
        this.no_economico = no_economico;
        this.nombre = nombre;
        this.sexo = sexo;
        this.departamento = departamento;
        this.correo = correo;
        this.tipo = tipo;
        this.grado = grado;
    }

    public TutorVO(String nombre, int id_tutor, int tota) {
```

```
        this.nombre = nombre;
        this.id_tutor= id_tutor;
        this.tota= tota;
    }
 public TutorVO(String no_economico, String nombre, String departamento) {
        this.no_economico = no_economico;
        this.nombre = nombre;
        this.departamento = departamento;
    }
}
```

11.2..2.3 Clase “UsuarioVO”

```
package modelo;
public class UsuarioVO {
    private int id_usuario;
    private String nivel_acceso;
    private String matricula_usuario;
    private String contrasena;

    //loginUsuario
    public UsuarioVO(String matricula_usuario , String
nivel_acceso, String contrasena) {
        this.matricula_usuario = matricula_usuario;
        this.nivel_acceso = nivel_acceso;
        this.contrasena = contrasena;
    }

    public UsuarioVO(int id_usuario, String nivel_acceso, String
matricula_usuario, String contrasena) {
        this.id_usuario = id_usuario;
        this.nivel_acceso = nivel_acceso;
        this.matricula_usuario = matricula_usuario;
        this.contrasena = contrasena;
    }

    public UsuarioVO(int id_usuario, String contrasena) {
        this.id_usuario = id_usuario;
        this.contrasena = contrasena;
    }
    public UsuarioVO(int id_usuario, String nivel_acceso, String
contrasena) {
        this.id_usuario = id_usuario;
        this.nivel_acceso = nivel_acceso;
        this.contrasena = contrasena;
    }
}
```

11.2.3 Clases DAO

11.2.3.1 Clase “BecarioDAO”

```
package modelo;
```

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import static java.sql.Types.NULL;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import modelo.BecarioVO;
import modelo.TutorVO;

public class BecarioDAO {

public static boolean agregarBecario(BecarioVO unBecario){

                boolean agregado=false;
                String matricula = "";
                try {


                        ConexionBD c=new ConexionBD();
                        Connection con=c.getConexion();
                        Statement st=con.createStatement();

                        ResultSet rs=st.executeQuery("select matricula from
becario where matricula = '" + unBecario.getMatricula() + "' ");

                        while(rs.next()){
                            matricula = rs.getString("busqueda");
                            }


                        if(matricula.equals(unBecario.getMatricula())){

                        st.close();
                        con.close();
                        c.cerrarConexion();

                        }

                        else{

                        st.executeUpdate("INSERT INTO
becario(matricula,nombre,paterno,materno,sexo,contrasena,email,telefono,celu
lar,otro,anio,estado_becario,carreras_id_carrera,tutor_id_tutor) VALUES (
"+unBecario.getMatricula()+"
,'"+unBecario.getNombre()+"','"+unBecario.getPaterno()+"','"+unBecario.getMa
terno()+"','"+unBecario.getSexo()+"','"+unBecario.getContrasena()+"','"+unBe
cario.getEmail()+"',"+unBecario.getTelefono()+","+unBecario.getCelular()+","
+unBecario.getOtro()+","+unBecario.getAnio()+",'"+unBecario.getEstado_becari
o()+"',"+unBecario.getCarreras_id_carrera()+", '417' );");
                        st.executeUpdate("INSERT INTO
cuenta_usuario(matricula_usuario,nivel_acceso,contrasena) VALUES
("+unBecario.getMatricula()+",'USER','"+unBecario.getContrasena()+"');");
                        agregado=true;

                        st.close();
                        con.close();
                        c.cerrarConexion();
                        }

                } catch (SQLException e) {
                        agregado=false;
```

```
                        e.printStackTrace();
                }
                return agregado;
        }

public static BecarioVO consultarBecario(String matricula){
         BecarioVO becario=null;

         int id_becario = 0;
         String nombre = "";
         String mat = "";
         String paterno = "";
         String materno = "";
         String sexo = "";
         String contrasena = "";
         String email = "";
         String telefono = "";
         String celular = "";
         String otro = "";
         String anio = "";
         String estado_becario = "";
         int carreras_id_carrera = 0;
         int tutor_id_tutor = 0;

         String nombre_tutor = "";
         String departamento_tutor = "";
         String correo_tutor = "";

         String carrera = "";
         String carrera_departamento = "";


                try{
                        ConexionBD c=new ConexionBD();
                        Connection con=c.getConexion();
                        Statement st=con.createStatement();
                        ResultSet rs=st.executeQuery("select * from becario
where matricula = '" + matricula + "' ");
                        while(rs.next()){

               id_becario = rs.getInt("id_becario");
               nombre = rs.getString("nombre");
                mat = rs.getString("matricula");
               paterno = rs.getString("paterno");
               materno = rs.getString("materno");
               sexo =  rs.getString("sexo");
               contrasena = rs.getString("contrasena");
               anio = rs.getString("anio");
               estado_becario = rs.getString("estado_becario");
               carreras_id_carrera = rs.getInt("carreras_id_carrera");
               tutor_id_tutor = rs.getInt("tutor_id_tutor");



                           if(rs.getString("telefono")!= null){
                           telefono=rs.getString("telefono");
                           }

                           if(rs.getString("celular")!= null ){
                           celular=rs.getString("celular");
                           }

                           if(rs.getString("otro")!= null ){
```

```
                            otro=rs.getString("otro");
                            }

                            if(rs.getString("email")!= null){
                            email=rs.getString("email");
                            }

                        }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();


                        ConexionBD c2=new ConexionBD();
                        Connection con2=c2.getConexion();
                        Statement st2=con2.createStatement();
                        ResultSet rs2=st2.executeQuery("select * from tutor
where id_tutor = '" + tutor_id_tutor + "' ");

                        while(rs2.next()){
                            nombre_tutor = rs2.getString("nombre");
                            departamento_tutor =
rs2.getString("departamento");
                            correo_tutor = rs2.getString("correo");
                          }

                        rs2.close();
                        st2.close();
                        con2.close();
                        c2.cerrarConexion();

                         ConexionBD c3=new ConexionBD();
                        Connection con3=c3.getConexion();
                        Statement st3=con3.createStatement();
                        ResultSet rs3=st3.executeQuery("select * from
carreras where id_carrera = '" + carreras_id_carrera + "' ");

                        while(rs3.next()){
                            carrera = rs3.getString("carrera");
                            carrera_departamento =
rs3.getString("carrera_departamento");
                            }

                        rs3.close();
                        st3.close();
                        con3.close();
                        c3.cerrarConexion();


                        becario=new
BecarioVO(mat,nombre,paterno,materno,sexo,email,telefono,celular,otro,anio,e
stado_becario,carrera,nombre_tutor,departamento_tutor,correo_tutor,carrera_d
epartamento );
                        //becario=new
BecarioVO(rs.getString("matricula"),rs.getString("contrasena"),
rs.getString("nombre"), rs.getString("paterno"),
rs.getString("materno"),telefono,celular,otro,email,rs.getString("carrera"),
rs.getString("anio"), rs.getString("estado_becario"), "NULL");


                }catch(SQLException se){
                        se.printStackTrace();
```

```
                }
                return becario;
        }

public static BecarioVO recuperarContrasenia(String matricula, String
email){
         BecarioVO becario=null;


         String   contrasena = null;


                try{
                        ConexionBD c=new ConexionBD();
                        Connection con=c.getConexion();
                        Statement st=con.createStatement();
                        ResultSet rs=st.executeQuery("select * from becario
where matricula = '" + matricula + "' and email = '"+email+"' ; ");


                        while(rs.next()){


           contrasena = rs.getString("contrasena");

                        }
                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();


                        becario=new BecarioVO(contrasena);
                        //becario=new
BecarioVO(rs.getString("matricula"),rs.getString("contrasena"),
rs.getString("nombre"), rs.getString("paterno"),
rs.getString("materno"),telefono,celular,otro,email,rs.getString("carrera"),
rs.getString("anio"), rs.getString("estado_becario"), "NULL");


                }catch(SQLException se){
                        se.printStackTrace();
                }
                return becario;
        }

public static boolean actualizarBecario(BecarioVO unBecario,String usuario){
                boolean agregado=false;
                //tring tutor="";
                //String usuario="2123032184";

                try {
                        ConexionBD c=new ConexionBD();
                        Connection con=c.getConexion();
                        if(con!=null){

                                Statement st;
                                st = con.createStatement();
                                //st.executeUpdate("UPDATE becario SET
nombre='"+unBecario.getNombre()+"',paterno='"+unBecario.getPaterno()+"',mate
rno='"+unBecario.getMaterno()+"',
contrasena='"+unBecario.getContrasena()+"',carrera='"+unBecario.getCarrera()
+"',email='"+unBecario.getCorreo()+"',telefono='"+unBecario.getNcasa()+"',ce
lular='"+unBecario.getNcelular()+"',"
```

```
                                //      +
"otro='"+unBecario.getNotros()+"',anio='"+unBecario.getAnnoBecario()+"',esta
do_becario='"+unBecario.getEdoBecario()+"' where matricula='"+usuario+"'");
                                //st.executeUpdate("UPDATE becario SET
nombre='"+unBecario.getNombre()+"',paterno='"+unBecario.getPaterno()+"',mate
rno='"+unBecario.getMaterno()+"',
contrasena='"+unBecario.getContrasena()+"',carrera='"+unBecario.getCarrera()
+"',email='"+unBecario.getCorreo()+"',anio='"+unBecario.getAnnoBecario()+"',
estado_becario='"+unBecario.getEdoBecario()+"',telefono='"+unBecario.getNcas
a()+"',celular='"+unBecario.getNcelular()+"',otro='"+unBecario.getNotros()+"
' where matricula='"+usuario+"';");
                                st.executeUpdate("UPDATE becario SET
nombre='"+unBecario.getNombre()+"',paterno='"+unBecario.getPaterno()+"',mate
rno='"+unBecario.getMaterno()+"',email='"+unBecario.getEmail()+"',anio='"+un
Becario.getAnio()+"',telefono='"+unBecario.getTelefono()+"',celular='"+unBec
ario.getCelular()+"',otro='"+unBecario.getOtro()+"' where
matricula='"+usuario+"';");

                                agregado=true;
                                st.close();

                        }

                        con.close();
                        c.cerrarConexion();
                } catch (SQLException e) {
                        agregado=false;
                        e.printStackTrace();
                }
                return agregado;
        }

public static boolean actualizarTutorBecario(String usuario,String
id_tutor){
                boolean agregado=false;
                //tring tutor="";
                //String usuario="2123032184";

                try {
                        ConexionBD c=new ConexionBD();
                        Connection con=c.getConexion();

                        Statement st=con.createStatement();

                        if(con!=null){


                                st = con.createStatement();
                                //st.executeUpdate("UPDATE becario SET
nombre='"+unBecario.getNombre()+"',paterno='"+unBecario.getPaterno()+"',mate
rno='"+unBecario.getMaterno()+"',
contrasena='"+unBecario.getContrasena()+"',carrera='"+unBecario.getCarrera()
+"',email='"+unBecario.getCorreo()+"',telefono='"+unBecario.getNcasa()+"',ce
lular='"+unBecario.getNcelular()+"',"
                                //      +
"otro='"+unBecario.getNotros()+"',anio='"+unBecario.getAnnoBecario()+"',esta
do_becario='"+unBecario.getEdoBecario()+"' where matricula='"+usuario+"'");
                                //st.executeUpdate("UPDATE becario SET
nombre='"+unBecario.getNombre()+"',paterno='"+unBecario.getPaterno()+"',mate
rno='"+unBecario.getMaterno()+"',
contrasena='"+unBecario.getContrasena()+"',carrera='"+unBecario.getCarrera()
+"',email='"+unBecario.getCorreo()+"',anio='"+unBecario.getAnnoBecario()+"',
estado_becario='"+unBecario.getEdoBecario()+"',telefono='"+unBecario.getNcas
```

```
a()+"',celular='"+unBecario.getNcelular()+"',otro='"+unBecario.getNotros()+"
' where matricula='"+usuario+"';");
                                st.executeUpdate("UPDATE becario SET
tutor_id_tutor='"+id_tutor+"' where matricula='"+usuario+"';");

                                agregado=true;
                                st.close();

                        }

                        con.close();
                        c.cerrarConexion();
                } catch (SQLException e) {
                        agregado=false;
                        e.printStackTrace();
                }
                return agregado;
        }

public static boolean eliminarBecario(String usuario){
                boolean agregado=false;

                try {
                        ConexionBD c=new ConexionBD();
                        Connection con=c.getConexion();
                        if(con!=null){

                                Statement st;
                                st = con.createStatement();
                                //st.executeUpdate("SET SQL_SAFE_UPDATES =
0; Delete from becario where matricula='"+usuario+"';");
                                ResultSet rs=st.executeQuery("select * from
becario where matricula='"+usuario+"'");


                        if (rs != null && rs.next() ) {
                        st.executeUpdate("Delete from becario where
matricula='"+usuario+"';");
                        st.executeUpdate("Delete from cuenta_usuario where
matricula_usuario='"+usuario+"';");
                        agregado=true;
                        } else
                        { agregado=false;}

                        st.close();

                        }

                        con.close();
                        c.cerrarConexion();
                } catch (SQLException e) {
                        agregado=false;
                        e.printStackTrace();
                }
                return agregado;
        }


//Todos los campos
public static List consultaBecarioGeneral(String estado_becario,String anio,
int tutor,int carrera){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
```

```
                Statement st=con.createStatement();
                ResultSet rs;
                rs=st.executeQuery("select * from becario where
estado_becario='"+estado_becario+"' and anio = '"+anio+"' and tutor_id_tutor
= '"+tutor+"' and carreras_id_carrera = '"+carrera+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }


//solo Estado
public static List consultaBecarioGeneral_01(String estado_becario){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
                rs=st.executeQuery("select * from becario where
estado_becario='"+estado_becario+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }

//solo Anio
public static List consultaBecarioGeneral_02(String anio){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
```

```
                Statement st=con.createStatement();
                ResultSet rs;
                rs=st.executeQuery("select * from becario where
anio='"+anio+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }

//solo tutor
public static List consultaBecarioGeneral_03(String tutor){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
                rs=st.executeQuery("select * from becario where
tutor_id_tutor='"+tutor+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }

//solo carrera
public static List consultaBecarioGeneral_04(String carrera){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
```

```
                rs=st.executeQuery("select * from becario where
carreras_id_carrera = '"+carrera+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }

//solo estado y anio
public static List consultaBecarioGeneral_05(String estado_becario,String
anio){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
                rs=st.executeQuery("select * from becario where
estado_becario = '"+estado_becario+"' and anio = '"+anio+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }
//solo estado y tutor
public static List consultaBecarioGeneral_06(String estado_becario,String
tutor){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
```

```
                rs=st.executeQuery("select * from becario where
estado_becario = '"+estado_becario+"' and tutor_id_tutor = '"+tutor+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }

//solo estado y carrera
public static List consultaBecarioGeneral_07(String estado_becario,String
carrera){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
                rs=st.executeQuery("select * from becario where
estado_becario = '"+estado_becario+"' and carreras_id_carrera =
'"+carrera+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }

//solo anio y carrera
public static List consultaBecarioGeneral_08(String anio,String carrera){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
```

```
                rs=st.executeQuery("select * from becario where anio =
'"+anio+"' and carreras_id_carrera = '"+carrera+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }

//solo anio y tutor
public static List consultaBecarioGeneral_09(String anio,String tutor){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
                rs=st.executeQuery("select * from becario where anio =
'"+anio+"' and tutor_id_tutor = '"+tutor+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }

//solo tutor y carrera
public static List consultaBecarioGeneral_10(String tutor,String carrera){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
                rs=st.executeQuery("select * from becario where
tutor_id_tutor = '"+tutor+"' and carreras_id_carrera = '"+carrera+"';");
```

```
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }

//solo edo, anio y tutor
public static List consultaBecarioGeneral_11(String edo,String anio,String
tutor){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
                rs=st.executeQuery("select * from becario where
estado_becario = '"+edo+"' and anio = '"+anio+"'and tutor_id_tutor =
'"+tutor+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }


//solo edo, carrera y tutor
public static List consultaBecarioGeneral_12(String edo,String
carrera,String tutor){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
```

```
                rs=st.executeQuery("select * from becario where
estado_becario = '"+edo+"' and carreras_id_carrera = '"+carrera+"'and
tutor_id_tutor = '"+tutor+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }

//solo edo, carrera y anio
public static List consultaBecarioGeneral_13(String edo,String
carrera,String anio){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
                rs=st.executeQuery("select * from becario where
estado_becario = '"+edo+"' and carreras_id_carrera = '"+carrera+"'and anio =
'"+anio+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }


//solo tutor, carrera y anio
public static List consultaBecarioGeneral_14(String tutor,String
carrera,String anio){
       List<BecarioVO> listaBecarios=new ArrayList<BecarioVO>();
        try{    ConexionBD c=new ConexionBD();
```

```
                Connection con=c.getConexion();
                Statement st=con.createStatement();
                ResultSet rs;
                rs=st.executeQuery("select * from becario where
tutor_id_tutor = '"+tutor+"' and carreras_id_carrera = '"+carrera+"'and anio
= '"+anio+"';");
                while(rs.next()){
                BecarioVO becario=new BecarioVO();
                becario.setMatricula(rs.getString("matricula"));
                becario.setNombre(rs.getString("nombre"));
                becario.setPaterno(rs.getString("paterno"));
                becario.setMaterno(rs.getString("materno"));

becario.setCarreras_id_carrera(rs.getInt("carreras_id_carrera"));
                becario.setAnio(rs.getString("anio"));
                becario.setEstado_becario(rs.getString("estado_becario"));
                listaBecarios.add(becario); }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaBecarios;
    }




public static BecarioVO reporteBecario(String matricula){
  BecarioVO becario=null;

         int id_becario = 0;
         String nombre = "";
         String paterno = "";
         String materno = "";
         String sexo = "";
         String contrasena = "";
         String email = "";
         String telefono = "";
         String celular = "";
         String otro = "";
         String anio = "";
         String estado_becario = "";
         int carreras_id_carrera = 0;
         int tutor_id_tutor = 0;

         String nombre_tutor = "";
         String departamento_tutor = "";
         String correo_tutor = "";

         String carrera = "";
         String carrera_departamento="";


                try{
                        ConexionBD c=new ConexionBD();
                        Connection con=c.getConexion();

                        //Statement st=con.createStatement();
```

```
                        //ResultSet rs=st.executeQuery("select
becario.matricula,becario.contrasena,becario.nombre,becario.paterno,becario.
materno,becario.telefono,becario.celular,becario.otro,becario.email,becario.
carrera,becario.anio,becario.estado_becario,tutor.nombre from becario,tutor
where matricula='"+usuario+"' and becario.tutor_id_tutor=tutor.id_tutor;");

                        Statement st=con.createStatement();
                        ResultSet rs=st.executeQuery("select * from becario
where matricula = '" + matricula + "' ");


                        while(rs.next()){

                id_becario = rs.getInt("id_becario");
                nombre = rs.getString("nombre");
                paterno = rs.getString("paterno");
                materno = rs.getString("materno");
                sexo =  rs.getString("sexo");
                contrasena = rs.getString("contrasena");
                anio = rs.getString("anio");
                estado_becario = rs.getString("estado_becario");
                carreras_id_carrera = rs.getInt("carreras_id_carrera");
                tutor_id_tutor = rs.getInt("tutor_id_tutor");



                            if(rs.getString("telefono")!= null){
                            telefono=rs.getString("telefono");
                            }

                            if(rs.getString("celular")!= null ){
                            celular=rs.getString("celular");
                            }

                            if(rs.getString("otro")!= null ){
                            otro=rs.getString("otro");
                            }

                            if(rs.getString("email")!= null){
                            email=rs.getString("email");
                            }

                        }

                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();


                        ConexionBD c2=new ConexionBD();
                        Connection con2=c2.getConexion();
                        Statement st2=con2.createStatement();
                        ResultSet rs2=st2.executeQuery("select * from tutor
where id_tutor = '" + tutor_id_tutor + "' ");

                        while(rs2.next()){
                            nombre_tutor = rs2.getString("nombre");
                            departamento_tutor =
rs2.getString("departamento");
                            correo_tutor = rs2.getString("correo");
                          }
```

```
                        rs2.close();
                        st2.close();
                        con2.close();
                        c2.cerrarConexion();

                         ConexionBD c3=new ConexionBD();
                        Connection con3=c3.getConexion();
                        Statement st3=con3.createStatement();
                        ResultSet rs3=st3.executeQuery("select * from
carreras where id_carrera = '" + carreras_id_carrera + "' ");

                        while(rs3.next()){
                            carrera = rs3.getString("carrera");
                            carrera_departamento =
rs3.getString("carrera_departamento");
                                }

                        rs3.close();
                        st3.close();
                        con3.close();
                        c3.cerrarConexion();


                        becario = new
BecarioVO(matricula,nombre,paterno,materno,sexo,email,telefono,celular,otro,
anio,estado_becario,carrera,nombre_tutor,departamento_tutor,correo_tutor,car
rera_departamento );
                        //becario=new
BecarioVO(rs.getString("matricula"),rs.getString("contrasena"),
rs.getString("nombre"), rs.getString("paterno"),
rs.getString("materno"),telefono,celular,otro,email,rs.getString("carrera"),
rs.getString("anio"), rs.getString("estado_becario"), "NULL");


                }catch(SQLException se){
                        se.printStackTrace();
                }
                return becario;
    }
}
```

11.2.3.2 Clase “TutorDAO”

```
package modelo;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class TutorDAO {

     public static List consultaTutorPorDepartamento(String
carrera_departamento){
        List<TutorVO> listaTutor=new ArrayList<TutorVO>();


                try{
                       // TutorVO tutor=new TutorVO();
                        ConexionBD c=new ConexionBD();
```

```
                        Connection con=c.getConexion();

                        Statement st=con.createStatement();
                        ResultSet rs=st.executeQuery("select * from tutor
where departamento='"+carrera_departamento+"';");

                        while(rs.next()){
                        ConexionBD c2=new ConexionBD();
                        Connection con2=c2.getConexion();
                        Statement st2=con2.createStatement();
                        ResultSet rs2=st2.executeQuery("select count(*) AS
total  from becario where tutor_id_tutor = '" +rs.getInt("id_tutor")+ "' ");
                          while(rs2.next()){
                         int cont = rs2.getInt("total");

                        TutorVO tutor=new TutorVO();
                        tutor.setNombre(rs.getString("nombre"));
                        tutor.setId_tutor(rs.getInt("id_tutor"));
                        tutor.setTota(cont);
                        listaTutor.add(tutor);
                          }
                        rs2.close();
                        st2.close();
                        con2.close();
                        c2.cerrarConexion();

                        }
                        rs.close();

                        st.close();
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaTutor;
        }

      public static TutorVO consultaTutores(String economico){
        TutorVO tutor=new TutorVO();


                try{
                       // TutorVO tutor=new TutorVO();
                        ConexionBD c=new ConexionBD();
                        Connection con=c.getConexion();

                        Statement st=con.createStatement();
                        ResultSet rs=st.executeQuery("select * from tutor
where no_economico='"+economico+"';");

                        while(rs.next()){

                        tutor.setNo_economico(rs.getString("no_economico"));
                        tutor.setNombre(rs.getString("nombre"));
                        tutor.setDepartamento(rs.getString("departamento"));

                        }




                        rs.close();
```

```
                        st.close();
                        con.close();
                        c.cerrarConexion();


                }catch(SQLException se){
                        se.printStackTrace();
                }
                return tutor;
        }
}
```

11.2.3.3 Clase “UsuarioDAO”

```
package modelo;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class TutorDAO {

     public static List consultaTutorPorDepartamento(String
carrera_departamento){
        List<TutorVO> listaTutor=new ArrayList<TutorVO>();


                try{
                       // TutorVO tutor=new TutorVO();
                        ConexionBD c=new ConexionBD();
                        Connection con=c.getConexion();

                        Statement st=con.createStatement();
                        ResultSet rs=st.executeQuery("select * from tutor
where departamento='"+carrera_departamento+"';");

                        while(rs.next()){
                        ConexionBD c2=new ConexionBD();
                        Connection con2=c2.getConexion();
                        Statement st2=con2.createStatement();
                        ResultSet rs2=st2.executeQuery("select count(*) AS
total  from becario where tutor_id_tutor = '" +rs.getInt("id_tutor")+ "' ");
                          while(rs2.next()){
                         int cont = rs2.getInt("total");

                        TutorVO tutor=new TutorVO();
                        tutor.setNombre(rs.getString("nombre"));
                        tutor.setId_tutor(rs.getInt("id_tutor"));
                        tutor.setTota(cont);
                        listaTutor.add(tutor);
                          }
                        rs2.close();
                        st2.close();
                        con2.close();
                        c2.cerrarConexion();

                        }
                        rs.close();

                        st.close();
```

```
                        con.close();
                        c.cerrarConexion();
                }catch(SQLException se){
                        se.printStackTrace();
                }
                return listaTutor;
        }

      public static TutorVO consultaTutores(String economico){
        TutorVO tutor=new TutorVO();


                try{
                       // TutorVO tutor=new TutorVO();
                        ConexionBD c=new ConexionBD();
                        Connection con=c.getConexion();

                        Statement st=con.createStatement();
                        ResultSet rs=st.executeQuery("select * from tutor
where no_economico='"+economico+"';");

                        while(rs.next()){

                        tutor.setNo_economico(rs.getString("no_economico"));
                        tutor.setNombre(rs.getString("nombre"));
                        tutor.setDepartamento(rs.getString("departamento"));

                        }




                        rs.close();
                        st.close();
                        con.close();
                        c.cerrarConexion();


                }catch(SQLException se){
                        se.printStackTrace();
                }
                return tutor;
        }
}
```

11.3 Controlador

11.3.1 Servlets

11.3.1.1 “ServletActualizarBecario”

```
package controlador;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.swing.JOptionPane;
import modelo.BecarioDAO;
import modelo.BecarioVO;
```

```
import modelo.TutorVO;

@WebServlet(name = "ServletActualizarBecario", urlPatterns =
{"/ServletActualizarBecario"})
public class ServletActualizarBecario extends HttpServlet {

    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException
{
                response.setContentType("text/html;charset=UTF-8");
                 request.setCharacterEncoding("UTF-8");
                 String matricula=request.getParameter("busqueda");

                 String nombre=request.getParameter("nombre");
               String paterno=request.getParameter("paterno");
                 String materno=request.getParameter("materno");
                 String anio=request.getParameter("anio");
                 String telefono=request.getParameter("telefono");
                 String celular=request.getParameter("celular");
                 String otro=request.getParameter("otro");
                 String email=request.getParameter("email");

                 TutorVO tutor = null;
                 String usuario=request.getParameter("busqueda");
                BecarioVO unBecario=new BecarioVO(matricula,nombre,
paterno,materno , telefono, celular, otro, email,anio);
              boolean
respuesta=BecarioDAO.actualizarBecario(unBecario,usuario);
                        if(matricula != null){
                             if(respuesta){

request.getRequestDispatcher("mensajeOK_A.jsp").forward(request,
response);

                            }else{

request.getRequestDispatcher("error_A.jsp").forward(request,
response);
                            }
                        }else
                        {
                            JOptionPane.showMessageDialog(null, "La
matrícula es obligatoria", "Error", JOptionPane.ERROR_MESSAGE);

request.getRequestDispatcher("index.jsp").forward(request,
response);
                        }
       }
}
```

11.3.1.2 “ServletAsignarTutor”

```
package controlador;
import java.io.IOException;
import java.io.PrintWriter;
```

```
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.swing.JOptionPane;
import modelo.BecarioDAO;
import modelo.BecarioVO;
import modelo.TutorVO;

@WebServlet(name = "ServletAsignarTutor", urlPatterns =
{"/ServletAsignarTutor"})
public class ServletAsignarTutor extends HttpServlet {

    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException
{
                response.setContentType("text/html;charset=UTF-8");
                 request.setCharacterEncoding("UTF-8");
                 String usuario=request.getParameter("busqueda");
                 int matricula = Integer.parseInt(usuario);
                 String id_tutor=request.getParameter("id_tutor");

               boolean
respuesta=BecarioDAO.actualizarTutorBecario(usuario, id_tutor);

                             if(respuesta){

request.getRequestDispatcher("mensajeOK_AT.jsp").forward(request,
response);

                            }else{

request.getRequestDispatcher("error_AT.jsp").forward(request,
response);
                            }
       }
}
```

11.3.1.3 “ServletConsulta”

```
package controlador;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import modelo.BecarioDAO;
import modelo.BecarioVO;

@WebServlet(name = "ServletConsulta", urlPatterns = {"/ServletConsulta"})
public class ServletConsulta extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
```

```
        response.setContentType("text/html;charset=UTF-8");
                request.setCharacterEncoding("UTF-8");
        String usuario=request.getParameter("busqueda");

                BecarioVO becario=BecarioDAO.consultarBecario(usuario);
                if(becario.getMatricula().equals(usuario)){
                        request.setAttribute("becario", becario);

        request.getRequestDispatcher("mostrarActualizacion.jsp").forward(req
uest, response);
                }else{

        request.getRequestDispatcher("error_A.jsp").forward(request,
response);
                }
        }
}
```

11.3.1.4 “ServletConsultaGeneral”

```
package controlador;
import java.io.IOException;
import java.io.PrintWriter;
import static java.lang.System.out;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import modelo.BecarioDAO;
import modelo.BecarioVO;
import javax.swing.JOptionPane;

public class ServletConsultaGeneral extends HttpServlet {

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {

        String t_edo="null";
        String t_anno="null";
        String t_tutor="null";
        String t_carrera="null";


                List<BecarioVO> lista=new ArrayList<BecarioVO>();


if(request.getParameter("edo")== null && request.getParameter("anno")== null
&& request.getParameter("tutor")== null && request.getParameter("carrera")==
null){ JOptionPane.showMessageDialog(null, "Porfavor selecciona una
Opcion!!", "Consulta", JOptionPane.WARNING_MESSAGE);}

else if(request.getParameter("edo")!= null && request.getParameter("anno")!=
null && request.getParameter("tutor")!= null &&
request.getParameter("carrera")!= null){
                lista =
BecarioDAO.consultaBecarioGeneral(request.getParameter("t_edo"),request.getP
arameter("t_anno"),Integer.parseInt(request.getParameter("t_tutor")),Integer
.parseInt(request.getParameter("t_carrera")));
                }
```

```
else if(request.getParameter("edo")!= null && request.getParameter("anno")==
null && request.getParameter("tutor")== null &&
request.getParameter("carrera")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_01(request.getParameter("t_edo"));}

else if(request.getParameter("anno")!= null && request.getParameter("edo")==
null && request.getParameter("tutor")== null &&
request.getParameter("carrera")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_02(request.getParameter("t_anno"));}

else if(request.getParameter("tutor")!= null &&
request.getParameter("edo")== null && request.getParameter("anno")== null &&
request.getParameter("carrera")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_03(request.getParameter("t_tutor"));}

else if(request.getParameter("carrera")!= null &&
request.getParameter("edo")== null && request.getParameter("anno")== null &&
request.getParameter("tutor")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_04(request.getParameter("t_carrera"));}

else if(request.getParameter("edo")!= null && request.getParameter("anno")!=
null && request.getParameter("carrera")== null &&
request.getParameter("tutor")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_05(request.getParameter("t_edo"),request.g
etParameter("t_anno") );}

else if(request.getParameter("edo")!= null &&
request.getParameter("tutor")!= null && request.getParameter("carrera")==
null && request.getParameter("anno")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_06(request.getParameter("t_edo"),request.g
etParameter("t_tutor") );}

else if(request.getParameter("edo")!= null &&
request.getParameter("carrera")!= null && request.getParameter("tutor")==
null && request.getParameter("anno")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_07(request.getParameter("t_edo"),request.g
etParameter("t_carrera") );}

else if(request.getParameter("anno")!= null &&
request.getParameter("carrera")!= null && request.getParameter("tutor")==
null && request.getParameter("edo")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_08(request.getParameter("t_anno"),request.
getParameter("t_carrera") );}

else if(request.getParameter("anno")!= null &&
request.getParameter("tutor")!= null && request.getParameter("carrera")==
null && request.getParameter("edo")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_09(request.getParameter("t_anno"),request.
getParameter("t_tutor") );}

else if(request.getParameter("tutor")!= null &&
request.getParameter("carrera")!= null && request.getParameter("anno")==
null && request.getParameter("edo")== null){
```

```
                lista =
BecarioDAO.consultaBecarioGeneral_10(request.getParameter("t_tutor"),request
.getParameter("t_carrera") );}

else if(request.getParameter("edo")!= null && request.getParameter("anno")!=
null && request.getParameter("tutor")!= null &&
request.getParameter("carrera")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_11(request.getParameter("t_edo"),request.g
etParameter("t_anno"),request.getParameter("t_tutor")  );}

else if(request.getParameter("edo")!= null &&
request.getParameter("carrera")!= null && request.getParameter("tutor")!=
null && request.getParameter("anno")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_12(request.getParameter("t_edo"),request.g
etParameter("t_carrera"),request.getParameter("t_tutor")  );}

else if(request.getParameter("edo")!= null &&
request.getParameter("carrera")!= null && request.getParameter("anno")!=
null && request.getParameter("tutor")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_13(request.getParameter("t_edo"),request.g
etParameter("t_carrera"),request.getParameter("t_anno")  );}

else if(request.getParameter("tutor")!= null &&
request.getParameter("carrera")!= null && request.getParameter("anno")!=
null && request.getParameter("edo")== null){
                lista =
BecarioDAO.consultaBecarioGeneral_14(request.getParameter("t_tutor"),request
.getParameter("t_carrera"),request.getParameter("t_anno")  );}

request.setAttribute("list", lista);

                for (int i=0; i<lista.size();i++){
                    out.print(" "+lista.toString()+" ");
                }

                if(lista.size()==0){
                JOptionPane.showMessageDialog(null, "Becarios encontrados:
"+lista.size(), "Consulta", JOptionPane.WARNING_MESSAGE);

request.getRequestDispatcher("consultaBecarios.jsp").forward(request,
response);
                }else{
                JOptionPane.showMessageDialog(null, "Becarios encontrados:
"+lista.size(), "Consulta", JOptionPane.WARNING_MESSAGE);

request.getRequestDispatcher("mostrarConsulta.jsp").forward(request,
response);
                }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods.
Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

11.3.1.5 “ServletConsultaTutor”

```
package controlador;
import java.io.IOException;
import java.io.PrintWriter;
import static java.lang.System.out;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.swing.JOptionPane;
import modelo.BecarioDAO;
import modelo.BecarioVO;
import modelo.TutorDAO;
import modelo.TutorVO;

@WebServlet(name = "ServletConsultaTutor", urlPatterns =
{"/ServletConsultaTutor"})
public class ServletConsultaTutor extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

            String usuario=request.getParameter("busqueda");
            response.setContentType("text/html;charset=UTF-8");
            request.setCharacterEncoding("UTF-8");
```

```
            BecarioVO becario=BecarioDAO.consultarBecario(usuario);
            String matricula = usuario;

                if(matricula!=null){
                if(becario.getMatricula().equals(usuario)){
                       String carrera=request.getParameter("carrera");
                        List<TutorVO> listaTutor=new ArrayList<TutorVO>();

listaTutor=TutorDAO.consultaTutorPorDepartamento(becario.getCarrera_departam
ento());
                        request.setAttribute("list", listaTutor);
                        request.setAttribute("becario", becario);

        request.getRequestDispatcher("mostrarTutores.jsp").forward(request,
response);
                }else{

        request.getRequestDispatcher("error_AT.jsp").forward(request,
response);
                }
                }else
                        {

request.getRequestDispatcher("error_AT.jsp").forward(request, response);
                        }
    }
}
```

11.3.1.6 “ServletEliminacion”

```
package controlador;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.swing.JOptionPane;
import modelo.BecarioDAO;
import modelo.BecarioVO;

@WebServlet(name = "ServletEliminacion", urlPatterns =
{"/ServletEliminacion"})
public class ServletEliminacion extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
                response.setContentType("text/html;charset=UTF-8");
                request.setCharacterEncoding("UTF-8");

                int
matricula=Integer.parseInt(request.getParameter("busqueda"));

                String usuario=request.getParameter("busqueda");

                boolean respuesta=BecarioDAO.eliminarBecario(usuario);

                            if(matricula> 0 ){
                           if(respuesta==true){

request.getRequestDispatcher("mensajeOK_E.jsp").forward(request, response);
```

```
                            }else{
                                    PrintWriter out = response.getWriter();

response.setContentType("text/html;charset=UTF-8");
                                    out.println("<script
type=\"text/javascript\">");
                                    out.println("alert(\"¡No existe
becario!\");");
                                    out.println("</script>");

request.getRequestDispatcher("error_E.jsp").forward(request, response);

                            }
                        }else{

                         JOptionPane.showMessageDialog(null, "La matrícula es
obligatoria", "Error", JOptionPane.ERROR_MESSAGE);

request.getRequestDispatcher("index.jsp").forward(request, response);
                        }
        }
}
```

11.3.1.7 “ServletLogin”

```
package controlador;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.swing.JOptionPane;
import modelo.BecarioDAO;
import modelo.BecarioVO;
import modelo.TutorDAO;
import modelo.TutorVO;
import modelo.UsuarioDAO;
import modelo.UsuarioVO;

@WebServlet(name = "ServletLogin", urlPatterns = {"/ServletLogin"})
public class ServletLogin extends HttpServlet {

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

         String usuario=request.getParameter("_user");
         String contra=request.getParameter("_passw");


         UsuarioVO unUsuario=UsuarioDAO.loginUsuario(usuario,contra);
         BecarioVO unBecario=BecarioDAO.consultarBecario(usuario);
         TutorVO unTutor=TutorDAO.consultaTutores(usuario);


                        if(!"".equals(usuario)){
```

```
                            if(unUsuario!=null){

                                HttpSession sess = request.getSession(true);


if("ADMIN".equals(unUsuario.getNivel_acceso())){


sess.setAttribute("USER",unTutor.getNombre());

sess.setAttribute("CARRERA",unTutor.getDepartamento());
                                        sess.setAttribute("ID",
unTutor.getNo_economico());


                                    JOptionPane.showMessageDialog(null,
"Bienvenido Coordinador: "+unTutor.getNombre(), "Coordinador",
JOptionPane.INFORMATION_MESSAGE);

request.getRequestDispatcher("principalCoordinador.jsp").forward(request,
response);
                                    }
                                    else{


sess.setAttribute("USER",unBecario.getNombre()+" "+unBecario.getPaterno()+"
"+unBecario.getMaterno());

sess.setAttribute("CARRERA",unBecario.getCarrera());
                                        sess.setAttribute("ID",
unBecario.getMatricula());

                                    JOptionPane.showMessageDialog(null,
"Bienvenido: "+unBecario.getNombre()+" "+unBecario.getPaterno()+"
"+unBecario.getMaterno(), "Becario", JOptionPane.INFORMATION_MESSAGE);

request.getRequestDispatcher("principalBecario.jsp").forward(request,
response);


                                }
                            }else{
                                    JOptionPane.showMessageDialog(null,
"Usuario No Encontrado!\n Verifique sus Datos", "ERROR!",
JOptionPane.ERROR_MESSAGE);

request.getRequestDispatcher("index.html").forward(request, response);
                            }
                        }else
                        {
                            JOptionPane.showMessageDialog(null, "Usuario y
contraseña son obligatorios!", "Error", JOptionPane.ERROR_MESSAGE);

request.getRequestDispatcher("index.html").forward(request, response);
                        }
    }

    private UsuarioVO getCurrentUser(HttpServletRequest req) {
        HttpSession sess = req.getSession(false);
             if (sess == null) return null;
    return (UsuarioVO) sess.getAttribute("USER");
}
```

```
    private void doLogout(HttpServletRequest req) {
        HttpSession sess = req.getSession(false);
        if (sess != null) {
         sess.invalidate();
        }
}

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods.
Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

11.3.1.8 “ServletRecuperar”

```
@WebServlet("/ServletRecuperar")
public class ServletRecuperar extends HttpServlet {
        private static final long serialVersionUID = 1L;


    public ServletRecuperar() {
        // TODO Auto-generated constructor stub
    }
        /**
```

```
         * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
         */
        protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
                response.setContentType("text/html;charset=UTF-8");
                 request.setCharacterEncoding("UTF-8");

                 String matricula=request.getParameter("busqueda");
                 String email = request.getParameter("email");

                 BecarioVO becario =
BecarioDAO.recuperarContrasenia(matricula,email);
                        if(matricula!= null){
                            if(becario !=null){
                                    JOptionPane.showMessageDialog(null,
"PASSWORD: " +becario.getMatricula(), "Recuperar Password",
JOptionPane.WARNING_MESSAGE);

request.getRequestDispatcher("index.jsp").forward(request, response);

                            }else{

request.getRequestDispatcher("error_info.jsp").forward(request, response);
                            }
                        }else
                        {
                            JOptionPane.showMessageDialog(null, "La
matrícula es obligatoria", "Error", JOptionPane.ERROR_MESSAGE);

request.getRequestDispatcher("index.jsp").forward(request, response);
                        }
        }
}
```

11.3.1.9 “ServletRegistroBecario”

```
@WebServlet("/ServletRegistroBecario")
public class ServletRegistroBecario extends HttpServlet {
        private static final long serialVersionUID = 1L;


    public ServletRegistroBecario() {
        // TODO Auto-generated constructor stub
    }
        /**
         * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
         */
        protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
                response.setContentType("text/html;charset=UTF-8");
                 request.setCharacterEncoding("UTF-8");

                 String matricula=request.getParameter("busqueda");
                 String nombre =request.getParameter("nombre");
                 String paterno=request.getParameter("paterno");
                 String materno =request.getParameter("materno");
                 String sexo =request.getParameter("sexo");
                 String contrasena =request.getParameter("_passw");
                 String email = request.getParameter("email");
                 String telefono=request.getParameter("telefono");
```

```
                String celular=request.getParameter("celular");
                String otro= request.getParameter("otro");
                String estado_becario
=request.getParameter("estado_becario");
                String anio= request.getParameter("anio");
                int carreras_id_carrera
=Integer.parseInt(request.getParameter("carrera"));
                int tutor_id_tutor = 0;

                BecarioVO unBecario=new BecarioVO(matricula,nombre,
paterno,materno,sexo,contrasena,email, telefono, celular, otro, anio,
estado_becario,carreras_id_carrera,tutor_id_tutor);
                        boolean
respuesta=BecarioDAO.agregarBecario(unBecario);
                        if(matricula!= null){
                            if(respuesta){

request.getRequestDispatcher("mensajeOK.jsp").forward(request, response);

                            }else{

request.getRequestDispatcher("error.jsp").forward(request, response);
                            }
                        }else
                        {
                            JOptionPane.showMessageDialog(null, "La
matrícula es obligatoria", "Error", JOptionPane.ERROR_MESSAGE);

request.getRequestDispatcher("index.jsp").forward(request, response);
                        }
        }
}
```

11.4 Vista

11.4.1 index.jsp

```
<!DOCTYPE html>
<%
    HttpSession sess = request.getSession(true);
    sess.setAttribute("USER","null");%>

<html>
<head>
    <meta charset = "utf-8"/>

    <title>Ingresar-Sistema de Control de Becarios</title>
    <link rel="stylesheet" href="css/general.css">
</head>

<body>

    <div id="envase">
    <header>
    <a href="http://www.uam.azc.mx"><img src="imgs/uam1.png" width="290" height="92"
style="float: left;" />  </a>
    <h1>Sistema de Control de Becarios</h1><br>


      <script type="text/javascript">
tday=new Array("Domingo","Lunes","Martes","Miércoles","Jueves","Viernes","Sabado");
```

```
tmonth=new
Array("Enero","Febrero","Marzo","Abril","Mayo","Junio","Julio","Agosto","Septiembre","Oct
ubre","Noviembre","Diciembre");

function GetClock(){
var d=new Date();
var nday=d.getDay(),nmonth=d.getMonth(),ndate=d.getDate(),nyear=d.getYear();
if(nyear<1000) nyear+=1900;
var d=new Date();
var nhour=d.getHours(),nmin=d.getMinutes(),nsec=d.getSeconds(),ap;

if(nhour==0){ap=" AM";nhour=12;}
else if(nhour<12){ap=" AM";}
else if(nhour==12){ap=" PM";}
else if(nhour>12){ap=" PM";nhour-=12;}

if(nmin<=9) nmin="0"+nmin;
if(nsec<=9) nsec="0"+nsec;

document.getElementById('clockbox').innerHTML=""+tday[nday]+", "+ndate+" de
"+tmonth[nmonth]+" del "+nyear+", "+nhour+":"+nmin+":"+nsec+ap+"";
}

window.onload=function(){
GetClock();
setInterval(GetClock,1000);
}
</script>
<div id="clockbox" style="float: right; margin-right: 5px;"></div>
</header>
    <br>
    <section id="menu_ingresar">
      <ul class="ingresar">
       <h1>Ingresar</h1> <br>
       <form action="ServletLogin" method="post">
  <center>
      <table>

      <tr><th>Usuario:</th><th><input type="text" name="_user"
style="width:90px;"></th></tr>
      <tr><th>Contraseña:</th><th> <input type="password" name="_passw"
style="width:90px;"></th></tr>
       <tr><th> </th><th>   <input type="submit" value="Login" style="float: right;">
</th></tr>
      </table>
   </center>

      </form>
        <center>
         <li><a href="recuperarContrasenna.jsp">Olvide mi Usuario /
Contraseña</a></li><br>
                  <li><a href="datosBecarioC.jsp">Registro</a></li><br>
        </center>
       </ul>
    </section>
    <section id="cuerpo_principal">
    <h3><b>Bienvenido</b></h3>
```

```
    <br><br>
    <br><br>

<center>
    <IMG SRC="imgs/beca.png" WIDTH=250 HEIGHT=280 BORDER=0>
</center>

 <div class="estira"></div>
  <div class="estira"></div>
      <div class="espacio_mediano"></div>
    </section>
    <div class="estira"></div>
    <footer>
      <a href="http://www.azc.uam.mx/">UAM Azcapotzalco</a>
        <a href="http://www.becas.uam.mx/">Becas UAM </a>
      <a href="Copyright.jsp">©Derechos Reservados 2017</a>
    </footer>
    </div>
  </body>
</html>
```

11.4.2 general.css

```
*{
color:#000000;
margin:0px;
font-family:Verdana;
}

body{
background-color:#d6d5d5;
text-align:center;
}

#envase{
max-width:1200px;
min-width:320px;
background-color:#FFFFFF;
text-align:left;
margin:auto;
}


header{
background-color:#b0001f;
height:100px;
width:100%;
font-size:75%;
}


header h1{
color:#FFFFFF;
margin:0px;
text-align:center;
padding-top:20px;
```

```
font-size:200%;
}

footer{
background-color:black;
height:40px;
text-align:center;
padding-top:20px;
margin-top:5px;
width:100%;
}

footer a{
display:inline;
padding-left:60px;
text-decoration:none;
color:#FFFFFF;
font-size:75%;
}

footer a:hover{
color:#FFFFFF;
}

#clockbox{
color:#FFFFFF;
}

#menu_ingresar{
font-size:70%;
width:200px;
margin-left:5px;
padding-bottom:1px;
float:left;
border: 1px solid #E6E6E6;
border-radius:5px;
padding:5px;
}

#cuerpo_principal{
width:70%;
margin-left:5px;
float:left;
border: 1px solid #ddd9d9;
padding:5px;
min-height:700px;
}


#cuerpo_principal h2 b{
font-size:21px;
color:#b0001f;
}

#cuerpo_principal h3 b{
font-size:30px;
color:#b0001f;
}
```

12. Entregables

En el presente proyecto tecnológico se propuso entregar a la conclusión lo siguiente:

- Manual de usuario
- Software
- Reporte final

Estos 3 materiales se encuentran en la raíz del disco compacto de donde se tomó este reporte final, el manual de usuario en formato PDF y el software desarrollado en compresión ZIP, que contiene los Scripts para generar la base de datos ‘becario’ y su llenado (por lo menos la tabla ‘tutor’), y la aplicación en Java.