

Universidad Autónoma Metropolitana Unidad Azcapotzalco  
División de Ciencias Básicas e Ingeniería  
Licenciatura en Ingeniería en Computación

Reporte  
Proyecto Integración

Sistema de Información para la Asignación de Tutores

Yeram Yulavi Esparza Rendón  
2123029641

Asesor:

Maricela Claudia Bravo Contreras  
Profesor Asociado  
Departamento de Sistemas

17 de julio de 2017

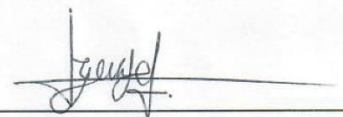
Yo, Maricela Claudia Bravo Contreras, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de la Universidad Autónoma Metropolitana Unidad Azcapotzalco



---

Maricela Claudia Bravo Contreras

Yo, Yeram Yulavi Esparza Rendón, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana Unidad Azcapotzalco para la publicación del Reporte de Proyecto de Integración en la Biblioteca Digital, así como en el Repositorio Institucional de la UAM Azcapotzalco



---

Yeram Yulavi Esparza Rendón

## Resumen

En este proyecto de integración se desarrolló un sistema de asignación de tutores que en parte automatizará el actual procedimiento.

El sistema también permite al coordinador visualizar estas asignaciones, aprobarlas y posteriormente enviar la carta correspondiente. De igual forma tiene acceso a los registros de los alumnos y los profesores y en caso de alguna modificación, realizarla sin tener que meterse a un programa de base de datos.

La asignación se hace en base a distintos criterios tales como la licenciatura, el género del alumno y también factores del tutor.

Se cuenta con un módulo para los alumnos y tres para el coordinador que consisten en visualizar asignaciones, profesores y alumnos. En estos módulos puede editar, eliminar y agregar. En el caso de las asignaciones también puede enviar la carta desde un botón y validar los datos.

## Tabla de Contenido

1. Introducción.....	1
2. Antecedentes.....	2
3. Justificación.....	4
4. Objetivo General.....	4
4.1 Objetivos Específicos.....	4
5. Marco Teórico.....	5
6. Desarrollo del Proyecto .....	7
6.1 funciones .....	10
6.1.1 La Clase ServletRegistro.....	11
6.1.2 Método AlumnosDAO.agregarAlumno .....	16
6.1.3 Método ProfesorDAO.selectProf .....	17
6.1.4 Enviar la carta .....	18
6.1.5 ServletEnvio .....	19
6.1.6 Carta Tutor (Ejemplo de un Método) .....	21
7. Resultados .....	29
8. Conclusiones .....	29
9. Bibliografía .....	30
10. Entregables .....	31

## 1. Introducción

Un tutor académico es el profesor que orienta y da seguimiento al desarrollo académico del alumno en el transcurso de su carrera. Se encarga de aconsejarlo mediante su experiencia profesional.

La tutoría es considerada como una herramienta de suma importancia para el estudiante, ya que a través de ella se puede lograr una vida académica con mayor rendimiento.

Actualmente la UAM<sup>1</sup> ofrece diferentes tipos de becas, entre estas se encuentra la de manutención y transporte cuyos recursos provienen del gobierno federal. Para que los alumnos de la UAM gocen de este recurso, la universidad se comprometió mediante un convenio con el gobierno federal, a que todo alumno que solicite la beca se le otorgará si cumple con la regularidad en el avance y promedio mínimo requerido. Así mismo se comprometió a que todo alumno becado le fuera asignado un tutor académico.

Actualmente la asignación de tutores se realiza de forma manual; es una labor ardua y propensa a errores.

El procedimiento para la asignación de tutor es el siguiente:

1. El alumno solicita la asignación de tutor al coordinador de desarrollo académico de cada división, proporcionando los datos necesarios (matrícula, carrera, nombre, etc.)
2. El coordinador de desarrollo académico busca al profesor que podría fungir como tutor mediante consultas a una tabla de Excel conforme a ciertas características (si el profesor es del campo de estudio de la carrera, si el profesor no está de sabático o licencia, etc.)
3. Se propone la asignación, y se elabora una carta que debe de ser firmada tanto por el tutor como por el alumno.
4. Ya que el alumno tiene la carta firmada por el tutor, el alumno regresa con el coordinador de desarrollo académico y la asignación propuesta se ratifica y se registra en la B.D.<sup>2</sup>

Un SI<sup>3</sup> es un conjunto de elementos que recolectan, procesan, almacenan, distribuyen y manejan grandes cantidades de información. Los SI logran importantes mejoras, automatizan y optimizan los procesos [1].

En este proyecto se diseñará e implementará un sistema de información para apoyar la asignación de tutor (en CBI) que realiza el coordinador de desarrollo académico.

---

<sup>1</sup> Universidad Autónoma Metropolitana

<sup>2</sup> Base de Datos

<sup>3</sup> Sistema de Información

## 2. Antecedentes

### Proyectos de integración o terminales

1. Sistema de asignación de ayudantes a profesores [2].

El sistema automatiza el procedimiento de la asignación de ayudantes a profesores del Departamento de Sistemas, mediante el uso de un algoritmo que evalúa diferentes criterios. Es similar a esta propuesta ya que este sistema también evaluará diferentes criterios, y es diferente ya que la asignación es de ayudantes y esta propuesta es para la asignación de tutores académicos.

2. Asignación óptima para el uso de laboratorios y talleres de la División de Ciencias Básicas e Ingeniería [3].

Este sistema automatiza el proceso que era manual de la asignación de laboratorios y talleres mediante un algoritmo. Es similar ya que es un sistema de asignación y agiliza el procedimiento que era manual, al igual que este, busca agilizar la asignación mediante un algoritmo. Es diferente ya que este sistema de asignación de laboratorios tiene un tema muy diferente al propuesto y automatiza el proceso por completo y el propuesto no, solo busca semiautomatizar.

3. Sistema de información para el seguimiento y monitoreo de actividades del alumno [4].

Este sistema busca personalizar las actividades para el alumno de acuerdo a su perfil de aprendizaje. Es similar ya que se busca arrojar un resultado mediante diferentes criterios. Es diferente porque los criterios que arroja el sistema son dados por el alumno y en el de asignación de tutor el criterio está dentro la misma información del sistema

## Tesis

4. Desarrollo e implementación del sistema para la asignación de la docencia en la Facultad de Estudios Superiores Cuautitlán [5].

Este sistema busca automatizar todos los procesos que permitan un mejor control y administración del proceso de asignación de la docencia. Es similar debido a que es un sistema de información que busca automatizar los procesos, al igual que la asignación de tutores académicos se busca semiautomatizar para un mejor control y administración. Es diferente porque aquí se busca automatizar completamente varios procesos y en el de asignación de tutores académicos solamente se busca semiautomatizar un proceso.

5. Herramienta de software para la asignación de las aulas y espacios físicos requeridos para la programación de asignaturas y grupos ofrecidos por cada una de las escuelas semestralmente en la universidad industrial de Santander [6].

Este sistema busca asignar de forma automática aulas para cada una de las asignaturas impartidas en la universidad. Es similar ya que este sistema de información busca asignar un aula de forma automática al igual que el sistema de asignación de tutor académico. Es diferente ya que el enfoque va dirigido a diferentes áreas cada uno, el de asignación de aulas no busca asignar dependiendo de diferentes criterios y el de asignación de tutores sí.

6. Sistema de información para la evaluación docente [7].

Este sistema busca proponer un modelo de evaluación de desempeño docente y su respectiva automatización del sistema. Es similar porque se busca automatizar la evaluación y en el sistema se evalúa a un docente a través de distintos criterios al igual que el presente se asignará al tutor por medio de ciertos criterios. Se diferencia del sistema de asignación de tutor académico ya que uno busca evaluar a los docentes y el otro solo asignar.

### 3. Justificación

Actualmente el proceso que se realiza para la asignación de tutor es tardado, tedioso y propenso a errores, ya que requiere que el alumno vaya a buscar a su cubículo tanto al coordinador desarrollo académico como al candidato a tutor, lo que se busca con el desarrollo del sistema de información, es que el alumno ingrese a una página web, proporcione sus datos y espere a recibir por correo su carta de asignación. Por su parte el coordinador de desarrollo académico recibe el listado de solicitudes de tutor y los tutores recomendados por el sistema para decidir la asignación.

Con este sistema de información tanto el alumno como el coordinador de desarrollo académico reducirán el tiempo y errores durante la asignación.

### 4. Objetivo General

Diseñar y desarrollar un sistema de información que permita realizar la asignación de tutor de manera semiautomática.

#### 4.1 Objetivos Específicos

1. Diseñar la B.D. de alumnos, profesores y tutores basado en el modelo entidad-relación para el óptimo manejo de información.
2. Diseñar e implementar el módulo de recomendación de tutor para automatizar el proceso actual.
3. Diseñar e implementar el módulo de Solicitud de Tutor a través de la cual el alumno podrá ingresar los datos para la solicitud.
4. Diseñar e implementar el módulo de Solicitudes Registradas a través del cual el coordinador visualice todas las solicitudes generadas de los alumnos y pueda editar, eliminar y enviar la carta.
5. Diseñar e implementar el del módulo de Tutores a través del cual el coordinador confirme la asignación.



## 5. Marco teórico

Una base de datos es una colección organizada de datos.

En la actualidad, los sistemas de bases de datos más populares son las bases de datos relacionales. Un lenguaje llamado SQL<sup>4</sup> es el lenguaje estándar internacional que se utiliza casi universalmente con las bases de datos relacionales para realizar consultas (es decir, solicitar información que satisfaga ciertos criterios) y para manipular datos.

Los programas en Java se comunican con las bases de datos y manipulan sus datos utilizando la API<sup>5</sup> JDBC<sup>6</sup>.

Una base de datos relacional es una representación lógica de datos que permite acceder a éstos sin necesidad de considerar la estructura física de estos datos.

Una base de datos relacional almacena los datos en tablas.

A continuación Se muestran las palabras clave a continuación del lenguaje SQL.

Palabra clave de SQL	Descripción
<b>SELECT</b>	Recupera datos de una o más tablas.
<b>FROM</b>	Las tablas involucradas en la consulta. Se requiere para cada <b>SELECT</b> .
<b>WHERE</b>	Los criterios de selección que determinan cuáles filas se van a recuperar, eliminar o actualizar.
<b>GROUP BY</b>	Criterios para agrupar filas.
<b>ORDER BY</b>	Criterios para ordenar filas.
<b>INNER JOIN</b>	Fusionar filas de varias tablas.
<b>INSERT</b>	Insertar filas en una tabla especificada.
<b>UPDATE</b>	Actualizar filas en una tabla especificada.
<b>DELETE</b>	Eliminar filas de una tabla especificada.

Figura 5.1

Los Servlets son módulos escritos en Java que se utilizan en un servidor, que puede ser o no ser servidor web, para extender sus capacidades de respuesta a los clientes al utilizar las potencialidades de Java.

Los Servlets basados en Web generalmente extienden a la clase HttpServlet. La clase HttpServlet redefine el método service para diferenciar entre las peticiones típicas recibidas de un navegador Web cliente.

<sup>4</sup> Structured Query Language

<sup>5</sup> Application Programming Interface (La interfaz de programación de aplicaciones)

<sup>6</sup> Java Database Connectivity (permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java)

Los dos tipos de peticiones de HTTP más comunes (también conocidos como métodos de petición) son get y post. Una petición get obtiene (o recupera) la información de un servidor. Las peticiones get se utilizan comúnmente para recuperar un documento de HTML o una imagen. Una petición post se utilizan comúnmente para enviar información, como la información de autenticación o los datos de un formulario que recopila la entrada del usuario a un servidor.

La clase HttpServlet define los métodos doGet y doPost para responder a las peticiones get y post de un cliente, respectivamente. [8]

En software de computadores, un Data Access Object (DAO, Objeto de Acceso a Datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo. El término se aplica frecuentemente al Patrón de diseño Object. [9]

En una aplicación, hay tantos DAOs como modelos. Es decir, en una base de datos relacional, por cada tabla, habría un DAO.

Los DTO (Data Transfer Object) o también denominados VO (Value Object). Son utilizados por DAO para transportar los datos desde la base de datos hacia la capa de lógica de negocio y viceversa. [10]

## 6. Desarrollo del Proyecto

En la figura 6.1 se muestra el modelo entidad relación

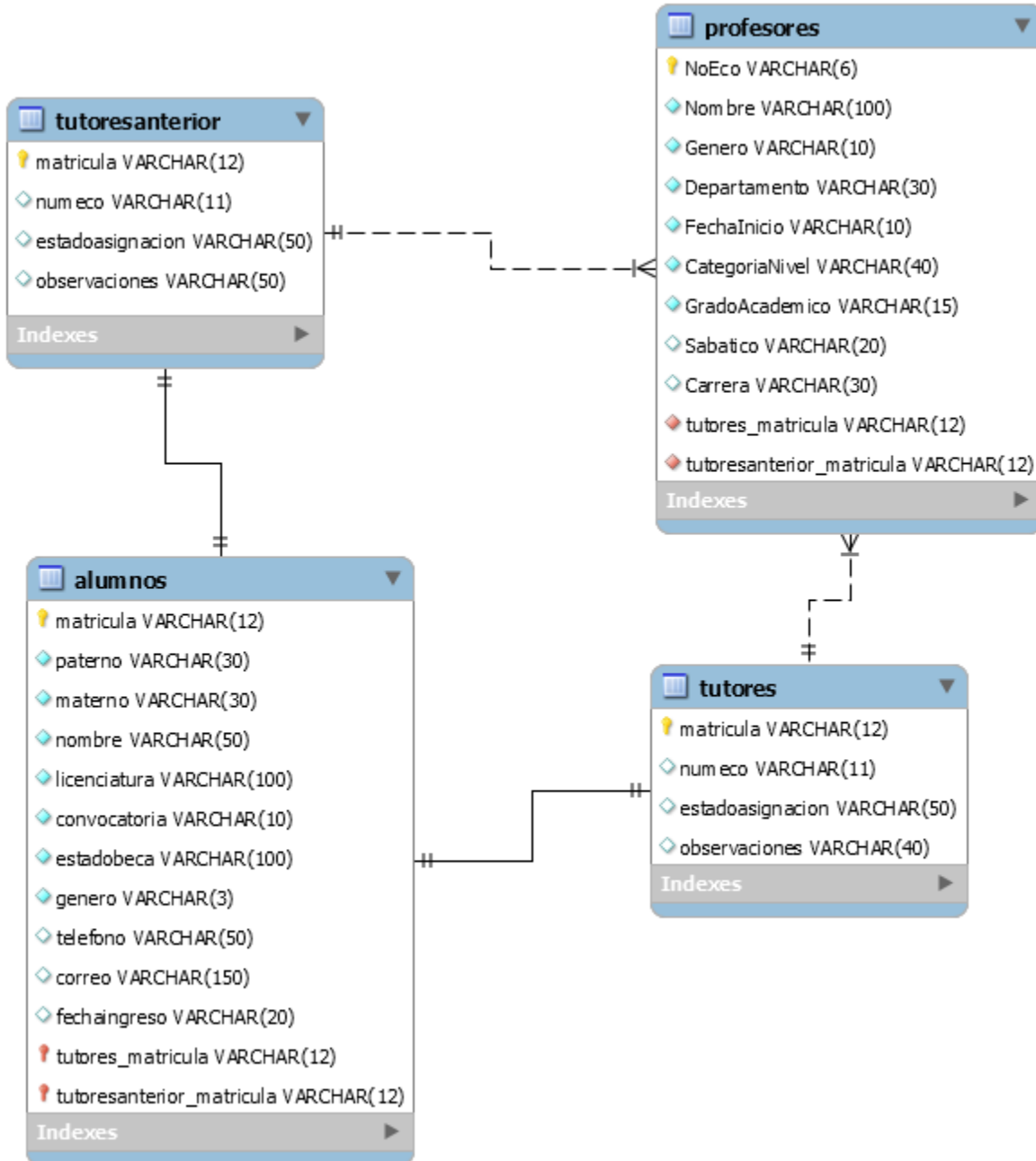


Figura 6.1

Con el modelo y la base de datos creada se procede a trabajar mediante la IDE<sup>7</sup> NetBeans  
 Primero estableciendo una conexión con la base de datos

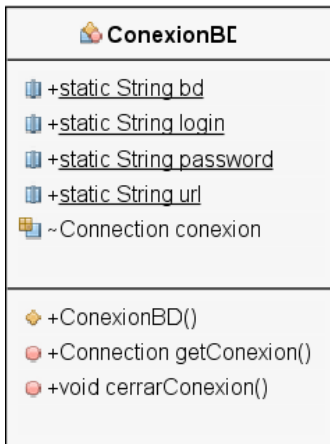


Figura 6.2

Se procede a crear los DAO y los VO de las tablas que se utilizaran

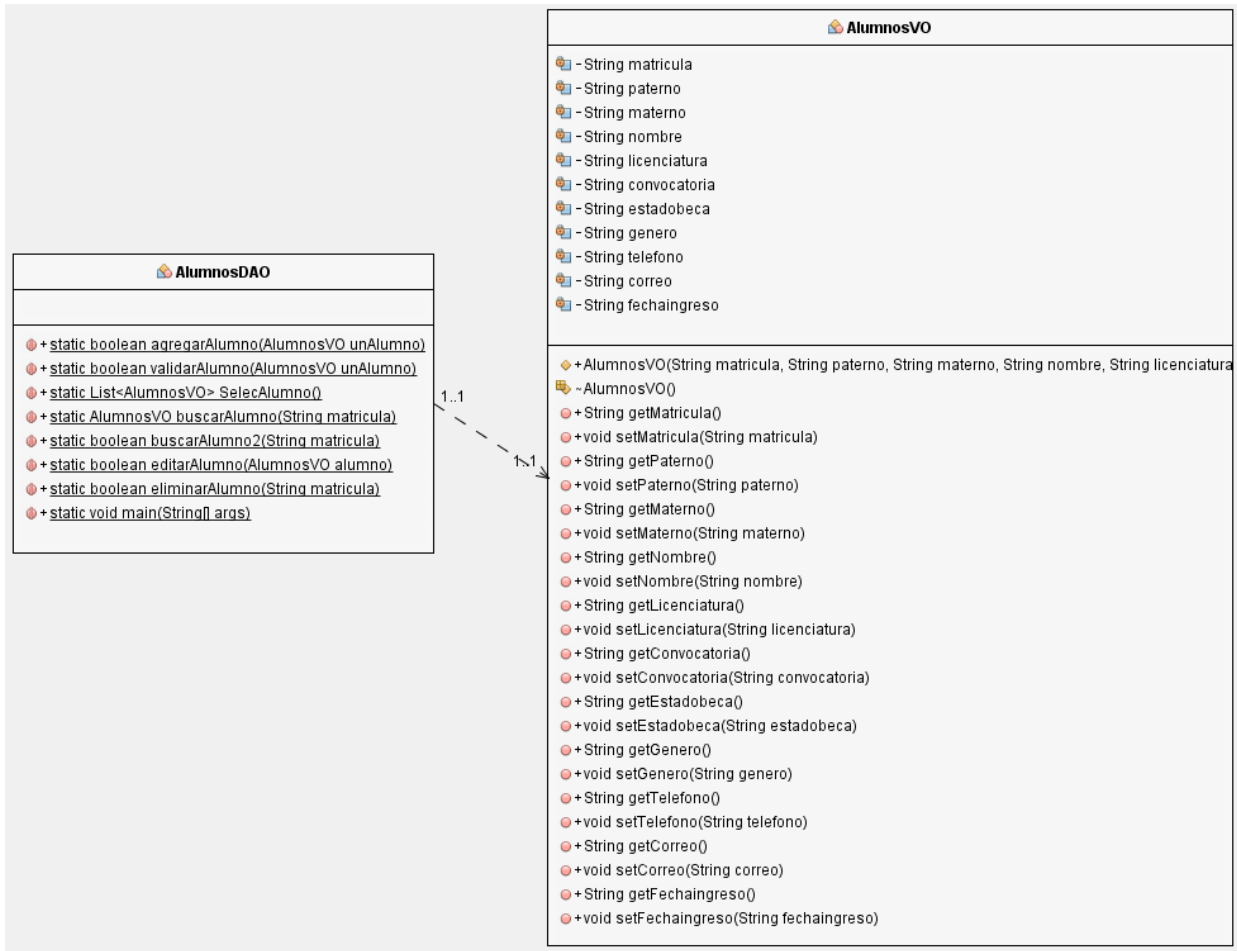


Figura 6.3

<sup>7</sup> Integrated Development Environment (entorno de desarrollo integrado) proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software

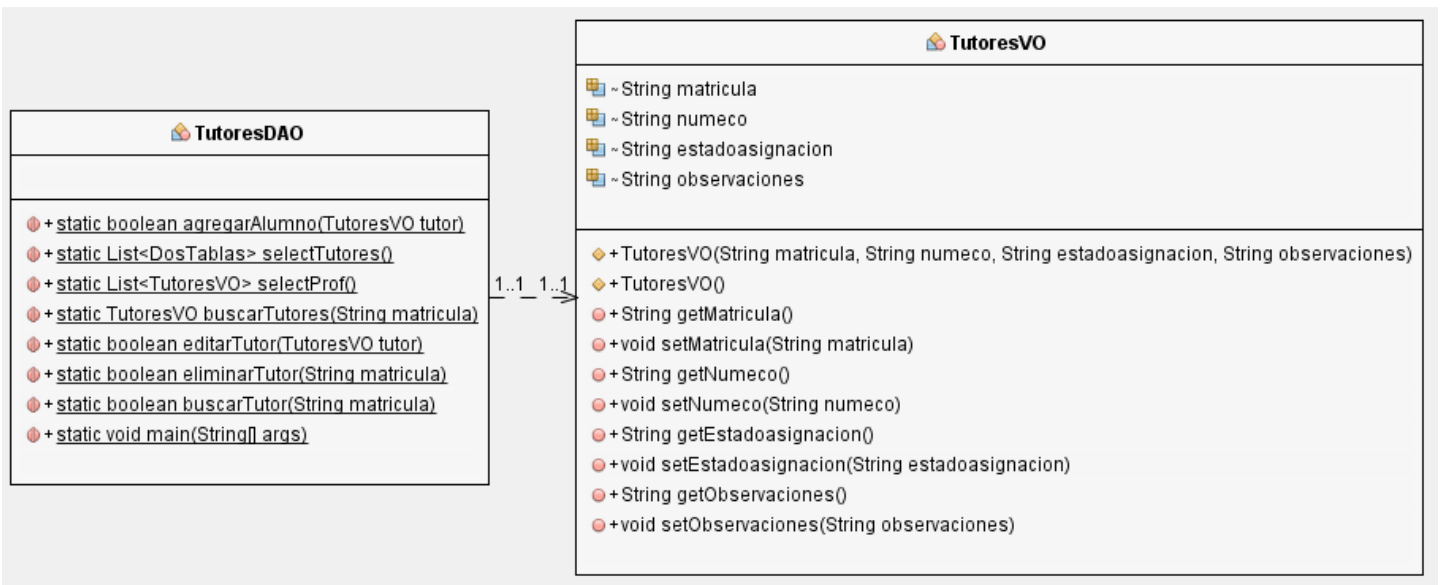


Figura 6.4

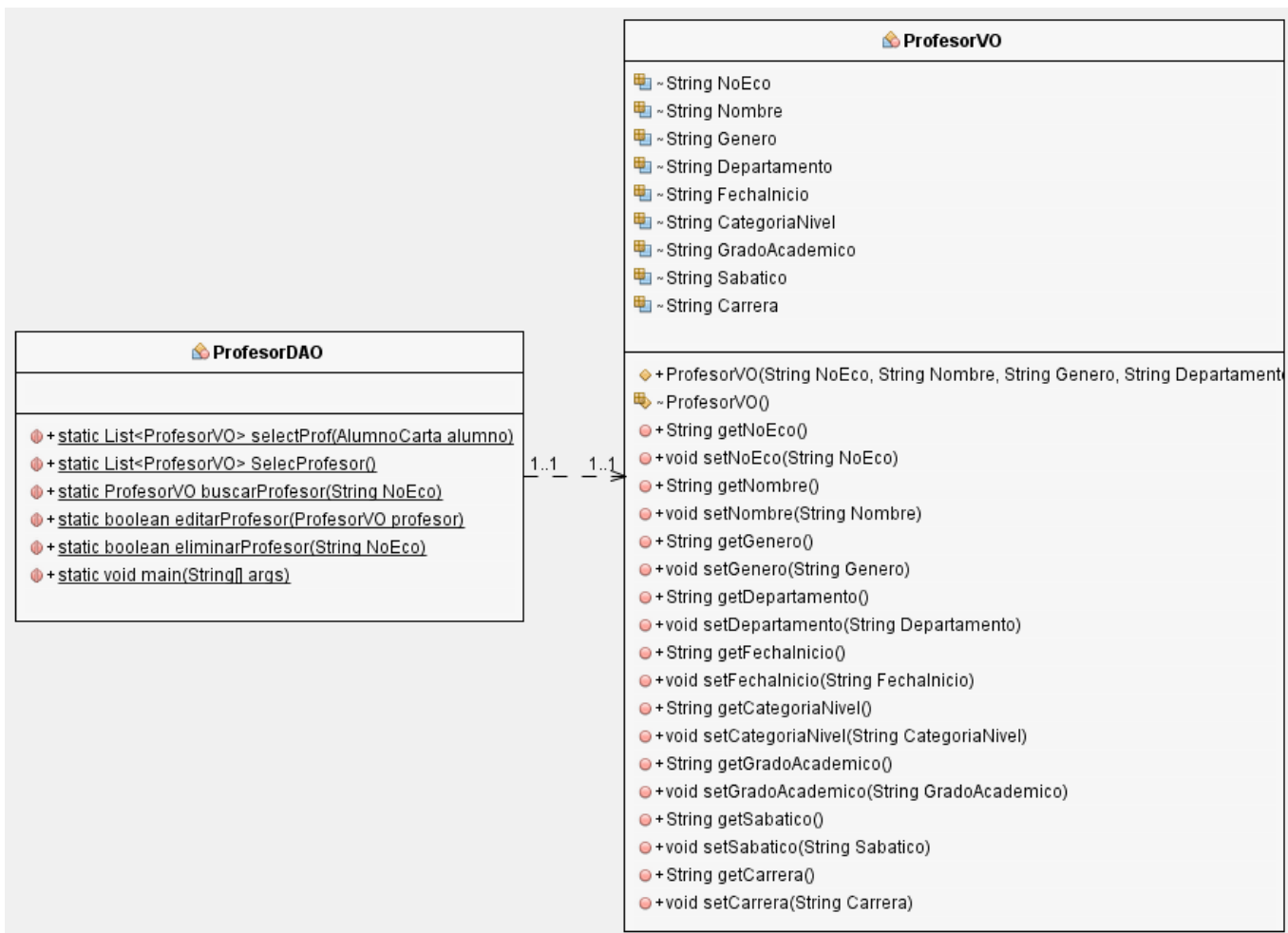


Figura 6.5

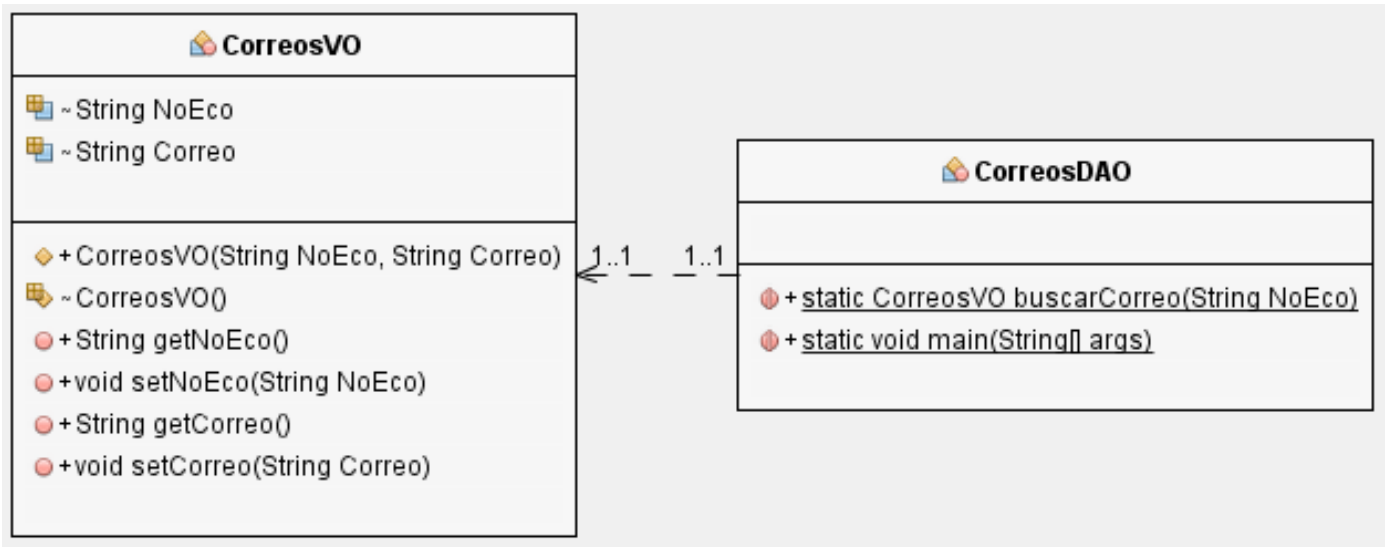


Figura 6.6

## 6.1 Funciones

localhost:8084/SistemaTutor/index.jsp

Buscar

Así es como se ve la JSP índice, donde recuperamos los datos y los pasamos a través de un formAction a un Servlet.

### Solicitud de Tutor

<input type="text" value="Matricula"/>	<input type="text" value="Nombre"/>
<input type="text" value="Apellido Paterno"/>	<input type="text" value="Apellido Materno"/>

Licenciatura

Convocatoria

Genero

Figura 6.7

 <b>ServletRegistro</b>
 <code>-final long serialVersionUID</code>
 <code>#void doPost(HttpServletRequest request, HttpServletResponse response)</code>

Figura 6.8

### 6.1.1 La Clase ServletRegistro

```
@WebServlet("/ServletRegistro")
public class ServletRegistro extends HttpServlet {

    private static final long serialVersionUID = 1L;
    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("matricula");
String paterno = request.getParameter("paterno");
String materno = request.getParameter("materno");
String nombre = request.getParameter("nombre");
String licenciatura = request.getParameter("licenciatura");
String convocatoria = request.getParameter("convocatoria");
String estadobeca = request.getParameter("estadobeca");
String genero = request.getParameter("genero");
String telefono = request.getParameter("telefono");
String correo = request.getParameter("correo");
String fechaingreso = request.getParameter("fechaingreso");
```

Aquí es donde obtengo los atributos que llegan desde el JSP

```
String m= matricula;
AlumnosVO unAlumno = new AlumnosVO(matricula, paterno, materno,
    nombre, licenciatura, convocatoria, estadobeca, genero,
    telefono, correo, fechaingreso);
```

Mando a llamar a Alumnos DAO que podemos ver el código más adelante, donde registra al alumno y a partir de ahí se procede a elegir al profesor.

```
boolean agregado = AlumnosDAO.agregarAlumno(unAlumno);
```

```
if (agregado) {
    if (genero.contains("F")) {
        String gen = "FEMENINO";
        if (licenciatura.contains("COMPUTACIÓN")) {
            String lic = "COMPUTACION";
            AlumnoCarta alumno = new AlumnoCarta(lic, gen);
            List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);
```

Se hace la separación dependiendo del género del alumno y se procede a llamar a ProfesorDAO

```

    boolean listaA = TutoresDAO.agregarAlumno(unTutor);
response.setContentType("text/html");
    request.getRequestDispatcher("exito.jsp").forward(request, response);
}
if (licenciatura.contains("AMBIENTAL")) {
    String lic = "AMBIENTAL";
    AlumnoCarta alumno = new AlumnoCarta(lic, gen);
    List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

    TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
    boolean listaA = TutoresDAO.agregarAlumno(unTutor);
    request.getRequestDispatcher("exito.jsp").forward(request, response);
}
if (licenciatura.contains("CIVIL")) {
    String lic = "CIVIL";
    AlumnoCarta alumno = new AlumnoCarta(lic, gen);
    List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

    TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
    boolean listaA = TutoresDAO.agregarAlumno(unTutor);
    request.getRequestDispatcher("exito.jsp").forward(request, response);
}
if (licenciatura.contains("ELECTRICA")) {
    String lic = "ELECTRICA";
    AlumnoCarta alumno = new AlumnoCarta(lic, gen);
    List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

    TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
    boolean listaA = TutoresDAO.agregarAlumno(unTutor);
    request.getRequestDispatcher("exito.jsp").forward(request, response);
}
if (licenciatura.contains("ELECTRÓNICA")) {
    String lic = "ELECTRONICA";
    AlumnoCarta alumno = new AlumnoCarta(lic, gen);
    List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

    TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
    boolean listaA = TutoresDAO.agregarAlumno(unTutor);
    request.getRequestDispatcher("exito.jsp").forward(request, response);
}
if (licenciatura.contains("FÍSICA")) {
    String lic = "FISICA";
    AlumnoCarta alumno = new AlumnoCarta(lic, gen);
    List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);
    TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
    boolean listaA = TutoresDAO.agregarAlumno(unTutor);
    request.getRequestDispatcher("exito.jsp").forward(request, response);
}

```



```

}
    if (licenciatura.contains("INDUSTRIAL")) {
        String lic = "INDUSTRIAL";
        AlumnoCarta alumno = new AlumnoCarta(lic, gen);
        List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);
        TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
        boolean listaA = TutoresDAO.agregarAlumno(unTutor);
        request.getRequestDispatcher("exito.jsp").forward(request, response);
    }

    if (licenciatura.contains("MECÁNICA")) {
        String lic = "MECANICA";
        AlumnoCarta alumno = new AlumnoCarta(lic, gen);
        List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);
        TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
        boolean listaA = TutoresDAO.agregarAlumno(unTutor);
        request.getRequestDispatcher("exito.jsp").forward(request, response);
    }

    if (licenciatura.contains("QUÍMICA")) {
        String lic = "QUIMICA";
        AlumnoCarta alumno = new AlumnoCarta(lic, gen);
        List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);
        TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
        boolean listaA = TutoresDAO.agregarAlumno(unTutor);
        request.getRequestDispatcher("exito.jsp").forward(request, response);
    }

    if (licenciatura.contains("METALÚRGICA")) {
        String lic = "METALURGICA";
        AlumnoCarta alumno = new AlumnoCarta(lic, gen);
        List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

        TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
        boolean listaA = TutoresDAO.agregarAlumno(unTutor);
        request.getRequestDispatcher("exito.jsp").forward(request, response);
    }

}

    if (genero.contains("M")) {
        String gen = "MASCULINO";

        if (licenciatura.contains("COMPUTACIÓN")) {
            String lic = "COMPUTACION";
            AlumnoCarta alumno = new AlumnoCarta(lic, gen);
            List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

            TutoresVO unTutor = new TutoresVO(m, lista.get(0).getNoEco(), "", "pendiente");
            boolean listaA = TutoresDAO.agregarAlumno(unTutor);
            response.setContentType("text/html");

```

```

    request.getRequestDispatcher("exito.jsp").forward(request, response);
}
if (licenciatura.contains("AMBIENTAL")) {
    String lic = "AMBIENTAL";
    AlumnoCarta alumno = new AlumnoCarta(lic, gen);
    List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

    TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
    boolean listaA = TutoresDAO.agregarAlumno(unTutor);
    request.getRequestDispatcher("exito.jsp").forward(request, response);
}
if (licenciatura.contains("CIVIL")) {
    String lic = "CIVIL";
    AlumnoCarta alumno = new AlumnoCarta(lic, gen);
    List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

    TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
    boolean listaA = TutoresDAO.agregarAlumno(unTutor);
    request.getRequestDispatcher("exito.jsp").forward(request, response);
}
if (licenciatura.contains("ELECTRICA")) {
    String lic = "ELECTRICA";
    AlumnoCarta alumno = new AlumnoCarta(lic, gen);
    List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

    TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
    boolean listaA = TutoresDAO.agregarAlumno(unTutor);
    request.getRequestDispatcher("exito.jsp").forward(request, response);
}
if (licenciatura.contains("ELECTRÓNICA")) {
    String lic = "ELECTRONICA";
    AlumnoCarta alumno = new AlumnoCarta(lic, gen);
    List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

    TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
    boolean listaA = TutoresDAO.agregarAlumno(unTutor);
    request.getRequestDispatcher("exito.jsp").forward(request, response);
}
if (licenciatura.contains("FÍSICA")) {
    String lic = "FISICA";
    AlumnoCarta alumno = new AlumnoCarta(lic, gen);
    List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

    TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
    boolean listaA = TutoresDAO.agregarAlumno(unTutor);
    request.getRequestDispatcher("exito.jsp").forward(request, response);
}

```

```

}
    if (licenciatura.contains("INDUSTRIAL")) {
        String lic = "INDUSTRIAL";
        AlumnoCarta alumno = new AlumnoCarta(lic, gen);
        List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

        TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
        boolean listaA = TutoresDAO.agregarAlumno(unTutor);
        request.getRequestDispatcher("exito.jsp").forward(request, response);
    }

    if (licenciatura.contains("MECÁNICA")) {
        String lic = "MECANICA";
        AlumnoCarta alumno = new AlumnoCarta(lic, gen);
        List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

        TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
        boolean listaA = TutoresDAO.agregarAlumno(unTutor);
        request.getRequestDispatcher("exito.jsp").forward(request, response);
    }

    if (licenciatura.contains("QUÍMICA")) {
        String lic = "QUIMICA";
        AlumnoCarta alumno = new AlumnoCarta(lic, gen);
        List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

        TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
        boolean listaA = TutoresDAO.agregarAlumno(unTutor);
        request.getRequestDispatcher("exito.jsp").forward(request, response);
    }

    if (licenciatura.contains("METALÚRGICA")) {
        String lic = "METALURGICA";
        AlumnoCarta alumno = new AlumnoCarta(lic, gen);
        List<ProfesorVO> lista = ProfesorDAO.selectProf(alumno);

        TutoresVO unTutor = new TutoresVO(matricula, lista.get(0).getNoEco(), "", "pendiente");
        boolean listaA = TutoresDAO.agregarAlumno(unTutor);
        request.getRequestDispatcher("exito.jsp").forward(request, response);
    }

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

```

## 6.1.2 Método AlumnosDAO.agregarAlumno

```
public static boolean agregarAlumno(AlumnosVO unAlumno){
    boolean agregado = false;
    try{
        ConexionBD c = new ConexionBD();
        Connection con= c.getConexion();
        if(con!=null){
            Statement st;
            st = con.createStatement();
            st.executeUpdate("INSERT INTO alumnos (`matricula`, `paterno`, `materno`, `nombre`, `licenciatura`, `convocatoria`,
`estadobeca`,`"
                + " `genero`, `telefono`, `correo`, `fechaingreso`) VALUES
("+unAlumno.getMatricula()+",""+unAlumno.getPaterno()+",""+
                +
                """+unAlumno.getMaterno()+",""+unAlumno.getNombre()+",""+unAlumno.getLicenciatura()+",""+unAlumno.getConvocatori
a()+",""+
                +
                """+unAlumno.getEstadobeca()+",""+unAlumno.getGenero()+",""+unAlumno.getTelefono()+",""+unAlumno.getCorreo()+"""+
                + ",""+unAlumno.getFechaingreso()+""");
            agregado=true;
            st.close();
        }
        c.cerrarConexion();
    }
    catch(SQLException e){
        agregado=false;
        e.printStackTrace();
    }
    return agregado;
}
```

### 6.1.3 Método ProfesorDAO.selectProf

```
public static List<ProfesorVO> selectProf(AlumnoCarta alumno) {
    List<ProfesorVO> lista = new ArrayList<ProfesorVO>();
    try {
        ConexionBD c = new ConexionBD();
        Connection con = c.getConnection();
        if (con != null) {
            Statement st;
            st = con.createStatement();
            ResultSet rs = st.executeQuery("SELECT * FROM profesores WHERE Carrera ="
                + alumno.getCarrera()
                + " AND Genero=" + alumno.getGenero() + "\n"
                + "ORDER BY RAND()\n"
                + "LIMIT 1");
            while (rs.next()) {
                ProfesorVO cm = new ProfesorVO(rs.getString("NoEco"), rs.getString("Nombre"),
                    rs.getString("Genero"), rs.getString("Departamento"),
                    rs.getString("FechaInicio"), rs.getString("CategoriaNivel"),
                    rs.getString("GradoAcademico"), rs.getString("Sabatico"),
                    rs.getString("Carrera"));
                lista.add(cm);
            }
            st.close();
        }
        c.cerrarConexion();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return lista;
}
```

## 6.1.4 Enviar la carta

Asignaciones						
Mostrar <input type="text" value="10"/> registros por pagina			Buscar: <input type="text" value="Buscar registros"/>			
Matricula	Numero Eco	Estado	Convocatoria	Editar	Eliminar	Enviar
			17-I			
		Carta firmada entregada	15/O			
		Carta firmada entregada	15/O			
		Carta firmada entregada	16/P			
		Elaborada y no entregada	16/P			
		Carta firmada entregada	15/O			
		Carta firmada entregada	15/O			
		Carta firmada entregada	15/O			
		Carta firmada entregada	15/O			
		Carta firmada entregada	16/I			

Mostrando registros de 1 al 10 de un total de 324 registros

Anterior **1** 2 3 4 5 ... 33 Siguiente

Figura 6.9

Como se puede apreciar hay un botón de enviar carta que nos dirige por medio de un Servlet a otra página que mostrará los datos completos para confirmar el envío tal como se muestra en la figura 6.10

The image shows a web form titled "Enviar Carta" with the following fields and values:

- Nombre: Yeram Yulavi Esparza Rendon
- Matricula: 2123029641
- E - mail: zokye@hotmail.com
- Teléfono: 26310575
- Licenciatura: INGENIERÍA EN COMPUTACIÓN
- Trimestre Actual: Trimestre
- Nombre: CHAVEZ LOMELI LAURA ELENA

Below the fields is a large blue rectangular area, and at the bottom is a red button labeled "Generar".

Figura 6.10

### 6.1.5 ServletEnvio

```
@WebServlet("/ServletEnvio")
public class ServletEnvio extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        String nombre = request.getParameter("nombre");
        String numero = request.getParameter("numero");
        String correo = request.getParameter("correo");
        String trimestre = request.getParameter("trimestre");
```

```
String nombrebecario= request.getParameter("nombrebecario");
String matricula = request.getParameter("matricula");
String email= request.getParameter("email");
String telefono= request.getParameter("telefono");
String licenciatura = request.getParameter("licenciatura");
String departamento = request.getParameter("departamento");
```

```
String genero = request.getParameter("genero");
String grado = request.getParameter("grado");
boolean envio = false;
```

```
if(email.contains("null")){
    email = " ";
}
if(genero.contains("FEMENINO")){
    CartaTutor ct = new CartaTutor(nombre, numero, correo, trimestre,
        nombrebecario, matricula, email, telefono,
        licenciatura, departamento);

    if(grado.contains("DOCTORADO")){
        envio = ct.cartaDra();
    }
    if(grado.contains("MAESTRIA")){
        envio = ct.cartaMtra();
    }
    if(grado.contains("LICENCIATURA")){
        envio = ct.cartaProfa();
    }
}
else if(genero.contains("MASCULINO")){
    CartaTutor ct = new CartaTutor(nombre, numero, correo, trimestre,
        nombrebecario, matricula, email, telefono, licenciatura, departamento);
    if(grado.contains("DOCTORADO")){
        envio = ct.cartaDr();
    }
    if(grado.contains("MAESTRIA")){
        envio = ct.cartaMtro();
    }
    if(grado.contains("LICENCIATURA")){
        envio = ct.cartaProf();
    }
}
```

Se manda llamar al método para generar la carta dependiendo el grado académico del profesor que le toco la tutoría correspondiente



```

    }    if(envio){
request.getRequestDispatcher("ExitoEnvio.jsp").forward(request, response);
    }

}
}

```

### 6.1.6 CartaTutor Ejemplo de un Metodo

```

public boolean cartaDr() throws IOException {
    Calendar fecha = new GregorianCalendar();

    int dia = fecha.get(Calendar.DATE);
    int mes = fecha.get(Calendar.MONTH) + 1;
    String mess = "";
    int año = fecha.get(Calendar.YEAR);

    switch (mes) {
        case 1:
            mess = "Enero";
            break;
        case 2:
            mess = "Febrero";
            break;
        case 3:
            mess = "Marzo";
            break;
        case 4:
            mess = "Abril";
            break;
        case 5:
            mess = "Mayo";
            break;
        case 6:
            mess = "Junio";
            break;
        case 7:
            mess = "Julio";
            break;
        case 8:
            mess = "Agosto";
            break;
    }
}

```

```

case 9:
    mess = "Septiembre";
    break;
case 10:
    mess = "Octubre";
    break;
case 11:
    mess = "Noviembre";
    break;
case 12:
    mess = "Diciembre";
    break;
default:
    break;
}

```

```
String fecha_final = "México, Ciudad de México, " + dia + " de " + mess + " de " + año;
```

```
String rutaArchivo = System.getProperty("user.home") + "/CARTATUTOR.docx";
```

```
File archivoXLS = new File(rutaArchivo);
```

```
/*Si el archivo existe se elimina*/
```

```
if (archivoXLS.exists()) {
```

```
    archivoXLS.delete();
```

```
}
```

```
/*Se crea el archivo*/
```

```
archivoXLS.createNewFile();
```

```
XWPFDocument documento = new XWPFDocument();
```

```
XWPFParagraph fecha_formato = documento.createParagraph();
```

```
fecha_formato.setAlignment(ParagraphAlignment.RIGHT);
```

```
XWPFParagraph asunto = documento.createParagraph();
```

```
asunto.setAlignment(ParagraphAlignment.RIGHT);
```

```
XWPFParagraph espacio1 = documento.createParagraph();
```

```
espacio1.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFParagraph espacio2 = documento.createParagraph();
```

```
espacio2.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFParagraph espacio3 = documento.createParagraph();
```

```
espacio3.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph espacio4 = documento.createParagraph();  
espacio4.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph nombreCoordinadora = documento.createParagraph();  
nombreCoordinadora.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph cargo = documento.createParagraph();  
cargo.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph presente = documento.createParagraph();  
presente.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph espacio5 = documento.createParagraph();  
espacio5.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph carta = documento.createParagraph();  
carta.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph espacio6 = documento.createParagraph();  
espacio6.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph carta2 = documento.createParagraph();  
carta2.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph espacio7 = documento.createParagraph();  
espacio7.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph nombrefirma = documento.createParagraph();  
nombrefirma.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph espacio8 = documento.createParagraph();  
espacio8.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph espacio9 = documento.createParagraph();  
espacio9.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFFParagraph datos = documento.createParagraph();  
datos.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFParagraph datos2 = documento.createParagraph();
datos2.setAlignment(ParagraphAlignment.LEFT);
```

```
XWPFRun r1 = fecha_formato.createRun();
r1.setText(fecha_final);
```

```
XWPFRun r2 = asunto.createRun();
r2.setText("Asunto: Asignación de tutor");
```

```
XWPFRun r3 = nombreCoordinadora.createRun();
r3.setText("Dra. Maricela Claudia Bravo Contreras,");
r3.setBold(true);
```

```
XWPFRun r4 = cargo.createRun();
r4.setText("Coordinadora de Desarrollo Académico");
```

```
XWPFRun r5 = presente.createRun();
r5.setText("Presente.");
```

```
XWPFRun r6 = carta.createRun();
    r6.setText("Me permito comunicar a usted que el Dr. " + nombre + " con número económico " + numero
    + "," + " ha aceptado ser "
    + "mi tutor y dar seguimiento a mi calidad de becario, a partir del trimestre " + trimestre + ".");
```

```
XWPFRun r7 = carta2.createRun();
r7.setText("Agradezco su atención y adjunto datos de contacto");
```

```
XWPFRun r8 = nombrefirma.createRun();
r8.setText("Nombre y firma del becario");
r8.setText(" ");
r8.setText("Nombre y firma del tutor");
```

```
XWPFRun r9 = datos.createRun();
r9.setText("_____");
r9.setText(" ");
r9.setText("_____");
```

```
XWPFRun r10 = datos.createRun();
r10.setText(" ");
```

```
//CREAMOS LA TABLA
```

```

XWPFTable table = documento.createTable();

//PRIMERA FILA
XWPFTableRow tableRowOne = table.getRow(0);
// COLUMNNA 1 FILA 1
XWPFFParagraph one = tableRowOne.getCell(0).addParagraph();
one.setSpacingAfter(0);

XWPFRun r11 = one.createRun();
r11.setText(nombrebecario + " ");
r11.setBold(true);

// COLUMNNA 2 FILA 1
XWPFFParagraph two = tableRowOne.addNewTableCell().addParagraph();
two.setSpacingAfter(0);

XWPFRun r12 = two.createRun();
r12.setText("Dr. " + nombre );
r12.setBold(true);

//SEGUNDA FILA
XWPFTableRow tableRowTwo = table.createRow();

//COLUMNNA 1 FILA 2
XWPFFParagraph three = tableRowTwo.getCell(0).addParagraph();
three.setSpacingAfter(0);

XWPFRun r13 = three.createRun();
r13.setText("Matrícula: " + matricula + " ");

//COLUMNNA 2 FILA 2
XWPFFParagraph four = tableRowTwo.getCell(1).addParagraph();
four.setSpacingAfter(0);

XWPFRun r14 = four.createRun();
r14.setText("No Económico: " + numero );

//TERCERA FILA
XWPFTableRow tableRowThree = table.createRow();

```

```
//COLUMNA 1 FILA 3
XWPFParagraph five = tableRowThree.getCell(0).addParagraph();
five.setSpacingAfter(0);

XWPFRun r15 = five.createRun();
r15.setText("E-mail: "+email +" ");

//COLUMNA 2 FILA 3
XWPFParagraph six = tableRowThree.getCell(1).addParagraph();
six.setSpacingAfter(0);

XWPFRun r16 = six.createRun();
r16.setText("E-mail: "+correo);

//CUARTA FILA
XWPFTableRow tableRowFour = table.createRow();

//COLUMNA 1 FILA 4
XWPFParagraph seven = tableRowFour.getCell(0).addParagraph();
seven.setSpacingAfter(0);

XWPFRun r17 = seven.createRun();
r17.setText("Teléfono: "+telefono +" ");

//COLUMNA 2 FILA 4
XWPFParagraph eight = tableRowFour.getCell(1).addParagraph();
eight.setSpacingAfter(0);

XWPFRun r18 = eight.createRun();
r18.setText("Departamento: " +departamento );

//QUINTA FILA
XWPFTableRow tableRowFive = table.createRow();

//COLUMNA 1 FILA 5
XWPFParagraph nine = tableRowFive.getCell(0).addParagraph();
nine.setSpacingAfter(0);

XWPFRun r19 = nine.createRun();
r19.setText("Licenciatura: " +licenciatura+" ");
```

```

//COLUMNA 2 FILA 5
XWPFPParagraph ten = tableRowFive.getCell(1).addParagraph();
ten.setSpacingAfter(0);

XWPFRun r20 = ten.createRun();
r20.setText("");

//QUITAR PARRAFO QUE SE CREA POR DEFAULT
tableRowOne.getCell(0).removeParagraph();
tableRowOne.getCell(1).removeParagraph();
tableRowTwo.getCell(0).removeParagraph();
tableRowTwo.getCell(1).removeParagraph();
tableRowThree.getCell(0).removeParagraph();
tableRowThree.getCell(1).removeParagraph();
tableRowFour.getCell(0).removeParagraph();
tableRowFour.getCell(1).removeParagraph();
tableRowFive.getCell(0).removeParagraph();
tableRowFive.getCell(1).removeParagraph();

//SIN BORDES
table.getCTTbl().getTblPr().unsetTblBorders();

//ANCHO
CTTblWidth width = table.getCTTbl().addNewTblPr().addNewTblW();
width.setType(STTblWidth.DXA);
width.setW(BigInteger.valueOf(9072));

FileOutputStream word = new FileOutputStream(rutaArchivo);
documento.write(word);
word.close();

EnvioCartaMail ecm = new EnvioCartaMail();
ecm.EnvioCorreo(rutaArchivo, email);
return true;
}

```

## 6.1.7 Envío de Correo con JavaMail

```
{

public boolean EnvioCorreo (String ruta, String correo){

try
{
// se obtiene el objeto Session. La configuración es para
// una cuenta de gmail.
Properties props = new Properties();
props.put("mail.smtp.host", "smtp.gmail.com");
props.setProperty("mail.smtp.starttls.enable", "true");
props.setProperty("mail.smtp.port", "587");
props.setProperty("mail.smtp.user", "*****");
props.setProperty("mail.smtp.auth", "true");

Session session = Session.getDefaultInstance(props, null);
// session.setDebug(true);

// Se compone la parte del texto
BodyPart texto = new MimeBodyPart();
texto.setText("Texto del mensaje");

// Se compone el adjunto con la imagen
BodyPart adjunto = new MimeBodyPart();
adjunto.setDataHandler(
new DataHandler(new FileDataSource(ruta)));
adjunto.setFileName("cartatutor.docx");

// Una MultiParte para agrupar texto e imagen.
MimeMultipart multiParte = new MimeMultipart();
multiParte.addBodyPart(texto);
multiParte.addBodyPart(adjunto);

// Se compone el correo, dando to, from, subject y el
// contenido.
MimeMessage message = new MimeMessage(session);
message.setFrom(new InternetAddress(correo));
message.addRecipient(
Message.RecipientType.TO,
new InternetAddress(correo));
```



```

message.setSubject("Carta Tutor");
message.setContent(multiParte);

// Se envia el correo.
Transport t = session.getTransport("smtp");
t.connect("*****", "*****");
t.sendMessage(message, message.getAllRecipients());
t.close();
}
catch (Exception e)
{
    e.printStackTrace();
    return false;
}
return true;
}
}

```

Lo que está en gris se sustituye el correo al cual será enviado y el correo donde será enviado y la contraseña del mismo.

## 7. Resultados

Gracias al desarrollo de este sistema se reducirá el tiempo en que se hace una asignación, también se eliminan las variantes, por ejemplo, cuando el coordinador está o no disponible para hacer la asignación, podrá hacerse el registro desde cualquier lugar y habrá una mejor organización de la base de datos.

## 8. Conclusiones

Con la creación de este sistema logramos semiautomatizar el proceso actual de la asignación de tutor y brindarle al alumno poder registrarse en cualquier parte y esperar el correo de la carta que le tendrá que llevar al tutor asignado.

También se economiza el tiempo que el coordinador tomaba al hacer la asignación dándole ese tiempo para realizar otras distintas actividades.

## 9. Bibliografía

- [1] GestioPolis Conocimiento en Negocios, "Sistemas de información y su importancia para la empresa", <http://www.gestiopolis.com/sistemas-informacion-importancia-empresa/>, 2017.
- [2] K. A. Fuentes Mayén, "Sistema de asignación de ayudantes a profesores", proyecto tecnológico, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2014.
- [3] A. E. Villarruel Barajas, "Asignación óptima para el uso de laboratorios y talleres de la División de la División de Ciencias Básicas e Ingeniería", proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2012.
- [4] A. Palacios González, "Sistema de información para el seguimiento y monitoreo de actividades del alumno", proyecto tecnológico, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México.
- [5] I. Núñez Consuelos, "Desarrollo e implementación del sistema para la asignación de la docencia en la Facultad de Estudios Superiores Cuautitlán", Facultad de Estudios Superiores Cuautitlán, Universidad Autónoma de México, México, 2011.
- [6] J. A. Moreno González, "Herramienta de software para la asignación de las aulas y espacios físicos requeridos para la programación de asignaturas y grupos ofrecidos por cada una de las escuelas semestralmente en la universidad industrial de Santander", Escuela de Ingeniería de Sistemas e Informática, Universidad Industrial de Santander, Colombia, 2011.
- [7] J. N. Gutiérrez Villegas, "Sistema de información para la evaluación docente", Escuela Superior De Ingeniería Mecánica Y Eléctrica Sección De Estudios De Posgrado e Investigación Programa De Posgrados En Ingeniería De Sistema, Instituto Politécnico Nacional, 2006.
- [8], [9], [10] P. J. Deitel, H. M. Deitel, Como Programar en Java, Editorial PEARSON, 2008.

## 10. Entregables

CD con el proyecto nombrado como SistemaTutor.zip

Las librerías necesarias.

Reporte con el P.I.