UNIVERSIDAD AUTÓNOMA METROPOLITANA
Casa abierta al tiempo

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Ingeniería en Computación

Título:

Sistema de Clasificación de Información en Equipos de Trabajo.

Alumnos:

García Rios Salvador 204358596

11I

Asesorado por:

M. en C. Rafaela Blanca Silva López

Contenido

Descripción

Sistema de Clasificación de Información en Equipos de Trabajos

lo que hará el este sistema informático será recibir aportes de usuarios registrados a un tema o grupo al que fue asignado, también podrá leer y usar aportes que hayan hecho otros usuarios en el tema o grupo, para poder llegar a esa funcionalidad del sistema el usuario tendrá que hacer algunos pasos antes, el primer paso que tendrá que hacer el usuario del sistema será registrarse en el sistema, en este paso de registro el usuario tendrá que proporcionar información personal general , la información personal general proporcionada será nombre, apellidos, edad, genero, nombre de usuario y password que tendrá dentro del sistema, estos dos últimos datos serán con los que se autentificara para poder entrar al sistema.

E l siguiente paso será llenar un cuestionario en el cual se obtendrá información particular del usuario, esta información particular será intereses, actitudes, valores, conocimientos técnicos, esta información será proporcionada a base a una encuesta o cuestionario previamente hecho.

Con la recopilación de esta información el sistema informático evaluara la información proporcionada y asignara al usuario a un grupo o tema.

Una vez registrado el usuario, este podrá hacer cambios en la información proporcionada, podrá darse de baja del sistema de informático, también podrá unirse a otros grupos o temas, dar de baja algún grupo o tema que ya no le interese y también podrá hacer aportes o leer aportes sobre el grupo o tema que tiene asociado, los aportes podrán ser solo escribir información en el área asignada o subir archivos.

El sistema tendrá uno o varios administradores los cuales harán la labor de verificar su buen funcionamiento, además de que harán depuraciones en los usuarios y grupos o temas del sistema de información, esto se refiere a que harán las altas bajas y cambios de los usuarios y grupos o temas.

Toda la información proporcionada como los aportes serán almacenados en una base de datos.

Encuesta

Una encuesta es un conjunto de preguntas normalizadas dirigidas a una muestra representativa de la población o instituciones, con el fin de conocer estados de opinión o hechos específicos.

Tipos

Las encuestas tienen por objetivo obtener información estadística indefinida, mientras que los censos y registros vitales de población son de mayor alcance y extensión. Este tipo de estadísticas pocas veces otorga, en forma clara y precisa, la verdadera información que se requiere, de ahí que sea necesario realizar encuestas a esa población en estudio, para obtener los datos que se necesitan para un buen análisis. Este tipo de encuesta abarca generalmente el UNIVERSO de los individuos en cuestión.

Otro tipo de Encuestas es ***Encuestas por Muestreo*** en donde se elige una parte de la población que se estima representativa de la población total. Debe tener un diseño muestral, necesariamente debe tener un marco de donde extraerla y ese marco lo constituye el censo de población. La encuesta (muestra o total), es una investigación estadística en que la información se obtiene de una parte representativa de las unidades de información o de todas las unidades seleccionadas que componen el universo a investigar. La información se obtiene tal como se necesita para fines estadístico-demográficos.

Una forma reducida de una encuesta por muestreo es un "*sondeo de opinión*", esta forma de encuesta es similar a un muestreo, pero se caracteriza porque la muestra de la población elegida no es suficiente para que los resultados puedan aportar un informe confiable. Se utiliza solo para recolectar algunos datos sobre lo que piensa un número de individuos de un determinado grupo sobre un determinado tema.

Actualmente, existen sistemas de gestión de encuestas en internet, que están acercando su utilización a investigadores que hasta el momento no tenían acceso a los medios necesarios para ejecutarlas.

Ejemplo de uso

1. Medir las relaciones entre variables demográficas, económicas y sociales.
2. Evaluar las estadísticas demográficas como errores, omisiones e inexactitudes.
3. Conocer profundamente patrones de las variables demográficas y sus factores asociados como fecundidad y migraciones determinantes.
4. Otorga información suplementaria en relación a la otorgada por los Censos.
5. Evaluar periódicamente los resultados de un programa en ejecución.
6. Probar la eficiencia de un método antes de aplicarlo al total de la población.
7. Saber la opinión del público acerca de un determinado tema.
8. Tener en cuenta el margen de error.

Ventajas y Desventajas

Ventajas

1. Bajo costo
2. Información más exacta (mejor calidad) que la del Censo debido al menor número de empadronadores permite capacitarlos mejor y más selectivamente.
3. Es posible introducir métodos científicos objetivos de medición para corregir errores.
4. Mayor rapidez en la obtención de resultados.
5. Técnica más utilizada y que permite obtener información de casi cualquier tipo de población.
6. Permite obtener información sobre hechos pasados de los encuestados.
7. Gran capacidad para estandarizar datos, lo que permite su tratamiento informático y el análisis estadístico.
8. Relativamente barata para la información que se obtiene con ello.
9. Te ayuda a conocer lo que quisieras conocer de la persona o personas encuestadas

Desventajas

El planeamiento y ejecución de la investigación suele ser más complejo que si se realizara por **censo**.

1. Requiere para su diseño de profesionales con buenos conocimientos de teoría y habilidad en su aplicación.

Hay un mayor riesgo de sesgo muestral.

Encuestas particulares

Encuesta piloto

Un tipo particular de encuesta, que tiene por objetivo preparar la verdadera encuesta. Se busca tener unos pocos criterios para diseñar o rediseñar las herramientas de trabajo, teniendo una idea previa de la población. Esta exploración es útil porque esta libre de conclusiones sobre el tema de estudio y sirve solo para mejorar la investigación; incluso restablecer un diagrama de flujo u otro tipo de planificación. Hay otras aplicaciones novedosas y son construir una muestra completamente estratificada y solo con los componentes de la población seleccionados para nuestro final interés; esta muestra no tiene valor predictor, pero sí puede utilizarse de una forma experimental, como grupo de control, y comparar sus resultados -parciales- con los que posteriormente hayamos obtenido en el muestreo probabilístico principal de toda la población y que así ya estaría estadísticamente bajo control. Ayudaría a la muestra completamente estratificada su uso en Investigación basada en la comunidad. Es de vital importancia en las organizaciones publicas y privadas.

Modelos de encuestas con resultados

Puede ayudar al estudiante el poder ver una encuesta sociológica, ya hecha, y también ver los resultados; para diseñar otra similar y para enterarse de las opiniones

al día sobre temas de su interés. Esto es posible y de forma gratuita. Los entes típicos que son profesionalmente conocidos son los organismos oficiales como el INE (Institutos Nacionales de Estadística), el CIS (Centros de Investigaciones Sociológicas), Cámaras de comercio e instituciones privadas como ASEP (Análisis Sociológicos, Económicos y Políticos); diseñan y realizan encuestas para variables estadísticas, sociológicas, políticas, en forma de índices, indicadores, escalas de actitud, barómetros de opinión y otros formatos, estudiados en la metodología social, y que puede conocer en su versión actual y real. Pueden encontrarse en Internet las encuestas completas y con resultados de tan solo semanas y de forma gratuita; por poner un ejemplo, se puede llegar al Instituto Nacional de Estadística de España comenzando en IPC (Índice de Precios al Consumo) o vía otras enciclopedias en español, es decir, utilizando enlaces partiendo de un tópico o materia.

Cuestionario

La encuesta se realiza siempre en función de un cuestionario, siendo éste por tanto, el documento básico para obtener la información en la gran mayoría de las investigaciones y estudios de mercado. El cuestionario es un documento formado por un conjunto de preguntas que deben estar redactadas de forma coherente, y organizadas, secuenciadas y estructuradas de acuerdo con una determinada planificación, con el fin de que sus respuestas nos puedan ofrecer toda la información que se precisa.

Elaborar un cuestionario

A continuación figuran algunas reglas y consideraciones que es conveniente tener en cuenta a la hora de elaborar un cuestionario

Reglas de redacción del cuestionario

- El lenguaje utilizado debe estar acorde con el del sujeto al que se dirige la encuesta, utilizando el vocabulario y términos adecuados, y las preguntas redactadas de la forma mas corta posible, con el fin de facilitar su lectura y comprensión.
- Las preguntas deben plantearse con claridad y de forma inequívoca, un típico error de redacción consiste en incluir dos preguntas en una, lo que conduce a no poder concretar a cual corresponde la respuesta.
- Debe empezarse por las preguntas más fáciles o sencillas para pasar después a las más difíciles o complicadas
- Se debe tener un especial cuidado con la información de preguntas que puedan resultar delicadas o embarazosas para el encuestado, redactándolas de forma que pueda obtenerse la información sin provocar un rechazo o una falsa respuesta. Estas preguntas deben ir, además, al final del cuestionario.
- No se debe incluir en las preguntas juicios de valor ni afirmaciones que puedan condicionar las respuestas, ni que puedan verse afectadas por el orden en que figuren dentro del cuestionario.
- Las preguntas deben formularse de forma que faciliten tanto el esfuerzo de memoria que tenga que realizar el encuestado, como en su caso, la realización de cálculos, para evitar errores en las respuestas.

Tipos de preguntas

Un cuestionario deberá incluir preguntas de distintos tipos en función del planteamiento del mismo y del tema a investigar, así puede haber:

Preguntas abiertas: Son preguntas en las que se permite al encuestado cualquier respuesta. Con estas preguntas puede obtenerse una mayor riqueza de detalle en las contestaciones, pero tienen el inconveniente de ser difíciles de tabular las respuestas.

Preguntas cerradas: Son preguntas en las que sólo se permite contestar mediante una serie cerrada de alternativas. Con estas preguntas puede perderse riqueza en la información pero su cuantificación es fácil.

Preguntas semi-abiertas (o semi-cerradas): Son preguntas de características intermedias entre los dos tipos anteriores, que intentan no perder nunca mucha riqueza de información a costa de perder algo de facilidad en la tabulación de las respuestas.

Preguntas en batería: Son aquellas que se planifican para realizarlas secuencialmente en función de la respuesta dada a la pregunta de la secuencia anterior. Su objetivo es profundizar en una información siguiendo el hilo de las sucesivas repuestas.

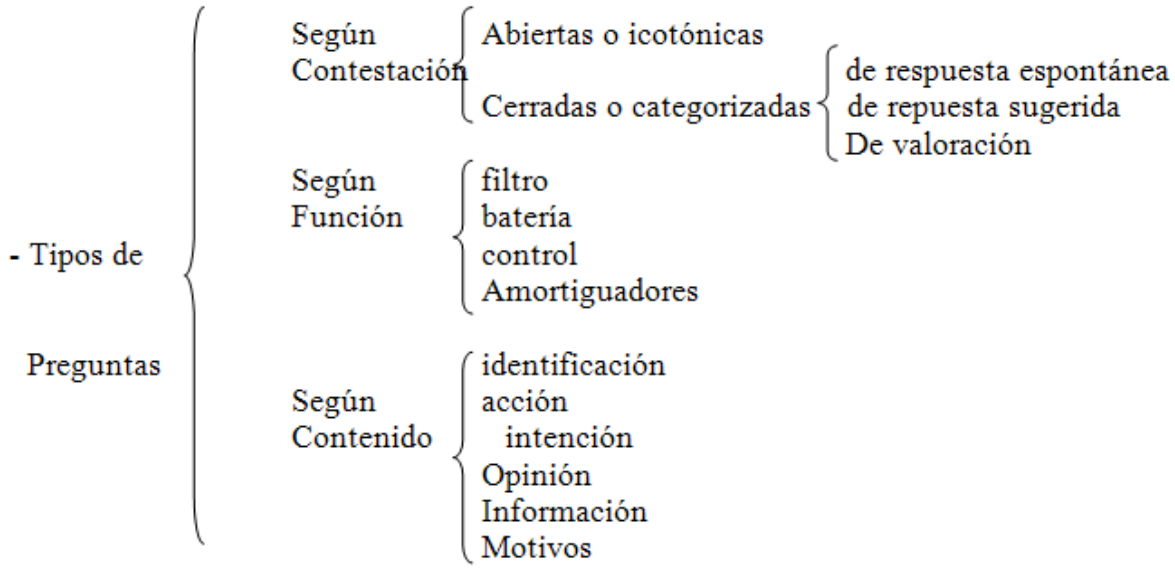
Preguntas de evaluación: Son preguntas dirigidas a obtener del entrevistado información sobre cómo valora una serie de cosas o aspectos. Pueden proporcionar una valoración de carácter numérico o una valoración de carácter cualitativo.

Preguntas introductoras o motivadoras: Son las que se realizan al principio de la entrevista y que tienen como objetivo despertar el interés de la persona que se va a entrevistar, intentando motivarle y predisponerle favorablemente para la realización del cuestionario. Las repuestas a estas preguntas generalmente, no se tienen en cuenta ya que en la mayoría de los casos su único objetivo es facilitar la entrevista.

Pruebas preliminares del cuestionario

El cuestionario debe elaborarse teniendo en cuenta que las contestaciones de sus preguntas tienen que suministrar toda la información de que se precisa, y además, debe de ser fácil y posible realizar la cuantificación y el análisis de los datos que se obtengan, y que estos permitan obtener, a su vez las conclusiones adecuadas. Una vez elaborado el cuestionario, y debido a su importancia para el análisis del mercado, es conveniente realizar una prueba del mismo a un numero reducido de personas, antes de realizar la encuesta. Esta prueba sirve para descubrir los posible defectos del cuestionario, las dificultades que pudiera presentar, pudiendo así efectuar las correcciones necesarias. De esta forma se obtienen mejores garantías sobre la efectividad del cuestionario definitivo.

TIPOS DE ENCUESTA



- Reglas para la formulación de preguntas

- Organización y preparación del cuestionario

1. Definición

Técnica cuantitativa que consiste en una investigación realizada sobre una muestra de sujetos, representativa de un colectivo más amplio que se lleva a cabo en el contexto de la vida cotidiana, utilizando procedimientos estandarizados de interrogación con el fin de conseguir mediciones cuantitativas sobre una gran cantidad de características objetivas y subjetivas de la población.

Ventajas:

- Técnica más utilizada y que permite obtener información de casi cualquier tipo de población.
- Permite obtener información sobre hechos pasados de los encuestados.
- Gran capacidad para estandarizar datos, lo que permite su tratamiento informático y el análisis estadístico.
- Relativamente barata para la información que se obtiene con ello.

Inconvenientes:

- No permite analizar con profundidad temas complejos (recurrir a grupos de discusión).

El Cuestionario es el instrumento de la encuesta y es un instrumento de recogida de datos rigurosamente estandarizado que operacionaliza las variables objeto de observación e investigación, por ello las preguntas de un cuestionario son los indicadores.

2. Tipos de cuestionarios.

a) entrevista personal→ hacen uso de encuestadores
b) por correo→ envío por correo de un cuestionario, es + barata, pero tienen el inconveniente de un índice de respuesta no elevado, por lo que hay que hacer sucesivas oleadas, lo que puede hacer que nuestra muestra no sea representativa.
c) Cuestionarios telefónicos→ no controlamos a la persona que responde, son baratas.
d) Cuestionarios auto-adictos→ se realizan a una población cautiva.

3. Tipos de preguntas:

a) Según la contestación que admitan:

- *abiertas* (preguntas que sólo formulan las pregunta, sin establecer categorías de respuesta)→ Se deben utilizar muy poco en las encuestas porque después de la encuesta hay que cerrarlas y luego estandarizarlas.
- *Cerradas*: Dicotónicas (establecen sólo 2 alternativas de respuesta, “Si o No” y a veces Ns/Nc)→Se deben utilizar sólo para temas muy bien definidos que admiten estas 2 alternativas como respuesta.

  Categorizadas (además de la pregunta, establecen las categorias de respuesta)→a su vez se subdividen en:

  De respuesta espontánea→el encuestador no debe leerle la respuesta al encuestado.

  De respuesta sugerida→el entrevistador lee las preguntas al encuestado.

  De valoración→ el entrevistador lee una escala de intensidad creciente o decreciente de categorías de respuesta.

b) Según su función en el cuestionario:

- *Filtro* → se utilizan mucho en los cuestionarios para eliminar aquellas personas que no les afecten determinadas preguntas, es decir que marcan la realización o no de preguntas posteriores.
- *Batería* → todas las preguntas tratan sobre un mismo tema y que siempre deben ir juntas en el cuestionario en forma de batería, empezando por las + sencillas y luego las + complejas. Esto se denomina "embudo de preguntas".
- *De control* → se utilizan para comprobar la veracidad de las respuestas de los encuestados y normalmente lo que se hace en estos casos es colocar la misma pregunta pero redactada de forma distinta en lugares separados una de la otra.
- *Amortiguadoras* → se refieren a que cuando estamos preguntando temas escabrosos o pensamos que serán reticentes a contestar, hay que preguntar suavizando la pregunta y no preguntar de modo brusco y directo.

c) Según su contenido:

- *Identificación* → sitúan las condiciones en la estructura social. Ej. Edad, sexo, profesión.
- *Acción* → tratan sobre las accioens de los entrevistados. Ej. ¿Va al cine?¿fuma?.
- *Intención* → indagan sobre la intenciones de los encuestados. Ej. ¿Va a votar?
- *Opinión* → tratan sobre la opinión encuestados sobre determinados temas. Ej. ¿Qué piensa sobre...?
- *Información* → analizan el grado de conocimiento de los encuestados sobre determinados temas.
- *Motivos* → tratan de saber el porqué de determinadas opiniones o actos.

4. Reglas para la formulación de preguntas:

a) No deben ser excesivamente largo, porque en cuestionarios largos (+100 preguntas) disminuye el % de respuestas.
b) Tiene que ser sencillas y redactadas de tal forma que puedan comprenderse con facilidad (no utilizar términos técnicos).
c) No deben incorporar términos morales (juicios de valor).
d) Nunca sugerir la respuesta, incitando a contestar más en un sentido que en otra.
e) Todas deben referirse a 1 sóla idea.
f) Todas las que estén dentro de un mismo tema deben ir juntas en el cuestionario en forma de batería.
g) No juntar preguntas cuya contestación a 1 de ellas influya sobre la contestación de la otra, denominado efecto "halo".

Recomendaciones o Deformaciones al crear un cuestionario.

1. Deformación conservadora→ las personas tienen más tendencia a contestar "si" que a contestar "no". Una pregunta recibe + % de adhesiones cuando está formulada para contestar "si" que cuando está formulada para contestar "no".

2. Influjo predisponente de ciertas palabras→ hay ciertas palabras con una gran carga ideológica.

3. Evitar referencias a ciertas personalidades públicas.

5. Organización y preparación del cuestionario:

FASES

a) Formular hipótesis.
b) Establecer las variables intermedias (dimensiones que queramos analizar)
c) Operacionalizar las variables intermedias, dando lugar a las preguntas que serían los indicadores.

CONSTRUCCION

a) Introducción (quien nos encargó el estudio, el carácter anónimo de las respuestas, etc.)
b) Preguntas:
   - Preguntas de identificación (sexo, edad,...)
   - Preguntas sencillas para introducir las + complejas y terminar con sencillas.
   - Facilitar la transición de un tema a otro en el cuestionario y se debe escribir en éste.
   - Evitar muchas preguntas abiertas.
c) Elaborar o decidir sobre los aspectos formales.
d) Preparar determinados elementos decisorios (carta de presentación de los encuestadores)
e) Formar a los encuestadores y elaborar una guía de instrucciones para realizar el cuestionario.
f) Hacer un PRETEST (prueba del cuestionario antes de su lanzamiento definitivo) tiene por objeto ver si se entienden las preguntas, si hay problemas en la redacción,... y siempre tiene que hacerse. No interesan los resultados de este pretest. 150 personas son representativas de la prueba
g) Codificar el cuestionario

Algoritmos

Almacenar Aportes

Leer Aporte

    Conectar a la base de datos

    Crear Tabla de Aporte

    Si la Creación es correcta

        Mostrar “Aporte Realizado”

    Si no

        Mostrar “Hubo un Error, Aporte no Realizado”

        Mostrar pantalla de creación de aporte

Cerrar conexión a la base de datos

Leer Aportes

Conectar a la Base de datos

Seleccionar datos donde el campo grupo/tema sea igual al del usuario

Si la selección fue correcta

    Mostrar el resultado de la selección

    Seleccionar un aporte

    Seleccionar datos donde el campo id sea igual al que el usuario

    Selecciono

    Si la selección fue correcta

        Mostrar el resultado de la selección

    Si no

        Mostrar “Hubo un Error, La Selección no se Realizo”

        Mostra la pantalla de selección de aportes

Si no

    Mostrar “Hubo un Error, La Selección no se Realizo”

Mostrar la pantalla de selección de accion

Cerrar conexión a la base de datos

Borrar aporte

Leer Grupo/Tema

Conectar a la base de datos

Seleccionar los datos donde el campo Grupo/Tema sea igual al leído

Si la selección no fue correcta

Mostrar “La acción no se pudo realizar”

Mostrar la pantalla de introducción de Grupo/Tema

Si no

Mostrar los datos de la selección

Seleccionar el aporte a borrar

Borrar el aporte en la base de datos donde el id del aporte sea igual al aporte seleccionado

Si el borrado fue correcto

Mostrar “Aporte borrado”

Mostrar pantalla de introducción de Grupo/Tema

Si no

Mostrar “La acción no se pudo realizar”

Mostrar los datos de la selección anterior

Cerrar conexión a la base de datos

Loguear Usuario

Leer login

Leer password

Conectarse a la base de datos

Seleccionar datos donde los campos login y password sean iguales a los introducidos

Si la selección es diferente de blancos

Entrar al sistema

Si no

Mostrar “Usuario o Password erroneo”

Regresar a la pantalla de logueo

Cerrar conexión a la base de datos

Registrar usuario

Leer Informacion General

Conectar a la base de datos

Seleccionar datos donde el campo email sea igual al introducido

Si los datos son diferentes de blancos

Mostrar “Usuario ya existente”

Regresar a la pagina de registro

Si no

Crear tabla usuario

Si la creación fue correcta

Mostrar pantalla para introducion de datos particulaes

Si no

Mostrar “Error al introducir datos”

Mostrar la pantalla de introducción de datos generales

Leer datos particulares

Insertar datos en la tabla usuario

Si la inserción fue correcta

Mostrar “Usuario creado”

Mostrar pantalla de selección de acción

Si no

Mostrar “No se pudo realizar la acción”

Regresar a la pantalla de introducción de datos particulares

Cerrar conexión a la base de datos

Borrar Usuario

Leer Grupo/Tema

Conectar a la base de datos

Seleccionar los datos donde el campo Grupo/Tema sea igual al leído

Si la selección no fue correcta

Mostrar “La acción no se pudo realizar”

Mostrar la pantalla de introducción de Grupo/Tema

Si no

Mostrar los datos de la selección

Seleccionar el usuario a borrar

Borrar el usuario en la base de datos donde el id del aporte sea igual al aporte seleccionado

Si el borrado fue correcto

Mostrar “Usuario borrado”

Mostrar pantalla de introducción de Grupo/Tema

Si no

Mostrar “La acción no se pudo realizar”

Mostrar los datos de la selección anterior

Cerrar conexión a la base de datos

Cambio Usuario

Leer Grupo/Tema

Conectar a la base de datos

Seleccionar los datos donde el campo Grupo/Tema sea igual al leído

Si la selección no fue correcta

Mostrar “La acción no se pudo realizar”

Mostrar la pantalla de introducción de Grupo/Tema

Si no

Mostrar los datos de la selección

Hacer cambios en la información del usuario

Insertar los cambios en la tabla del usuario

Si el insertado fue correcto

Mostrar “Cambios Hechos”

Mostrar pantalla de Selección de Accion

Si no

Mostrar “La acción no se pudo realizar”

Mostrar los datos de la selección anterior

Cerrar conexión a la base de datos

Registrar Grupo/Tema

Leer nombre del Grupo/Tema

Conectar a la base de datos

Seleccionar datos donde el campo nombre sea igual al introducido

Si los datos son diferentes de blancos

Mostrar “Grupo/Tema ya existente”

Regresar a la página de registro

Si no

Crear tabla Grupo/Tema

Si la creación fue correcta

Mostrar “Grupo/Tema creado”

Si no

Mostrar “Error al introducir datos”

Mostrar la pantalla de introducción del nombre de Grupo/tema

Cerrar conexión a la base de datos

Borrar Grupo/Tema

Leer Grupo/Tema

Conectar a la base de datos

Seleccionar los datos donde el campo Grupo/Tema sea igual al leído

Si la selección no fue correcta

    Mostrar “La acción no se pudo realizar”

    Mostrar la pantalla de introducción de Grupo/Tema

Si no

    Mostrar los datos de la selección

    Borrar el Grupo/Tema en la base de datos

    Si el borrado fue correcto

        Mostrar “Grupo/Tema borrado”

        Mostrar pantalla de introducción de Grupo/Tema

    Si no

        Mostrar “La acción no se pudo realizar”

        Mostrar los datos de la selección anterior

Cerrar conexión a la base de datos

Cambio Grupo/Tema

Leer Grupo/Tema

Conectar a la base de datos

Seleccionar los datos donde el campo Grupo/Tema sea igual al leído

Si la selección no fue correcta

    Mostrar “La acción no se pudo realizar”

    Mostrar la pantalla de introducción de Grupo/Tema

Si no

    Mostrar los datos de la selección

    Hacer cambios en la información del Grupo/Tema

Insertar los cambios en la tabla del Grupo/Tema

Si el insertado fue correcto

Mostrar “Cambios Hechos”

Mostrar pantalla de Selección de Accion

Si no

Mostrar “La acción no se pudo realizar”

Mostrar los datos de la selección anterior

Cerrar conexión a la base de datos

Seleccionar otro Grupo

Conectar a la base de datos

Seleccionar un Grupo

Inserta en la base de datos idUsuario y idGrupo

Si la inserción fue correcta

Mostrar pantalla de “Acción Realizada”

Si no

Mostrar “La acción no se pudo realizar”

Mostrar la pantalla de selección de Grupo

Cerrar conexión a la base de datos

Asignación de Grupo

Sumar los puntos de las respuestas obtenidas del formulario de datos particulares

Obtener las palabras diferentes a las conjunciones, artículos, preposiciones

Comparar con las palabras claves y titulo del grupo que este en el rango de puntos

Con el grupo que tenga mayor grupo de coincidencias asignarlo

Si la acción fue correcta

Mostrar “Accion Realizada”

Si no

Mostrar “La acción no se pudo realzar”

Si el numero de coincidencias es igual para varios grupos

Mostrar Pantalla de Selección de Grupo

Seleccionar un Grupo

Si la selección fue correcta

Mostrar “Accion Realizada”

Si no

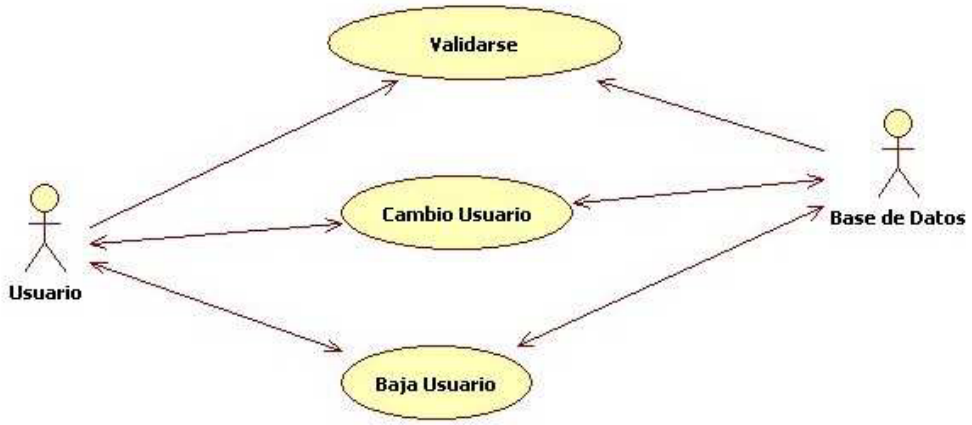
Mostrar “La acción no se pudo realizar”

Cerrar conexión con la base de datos.

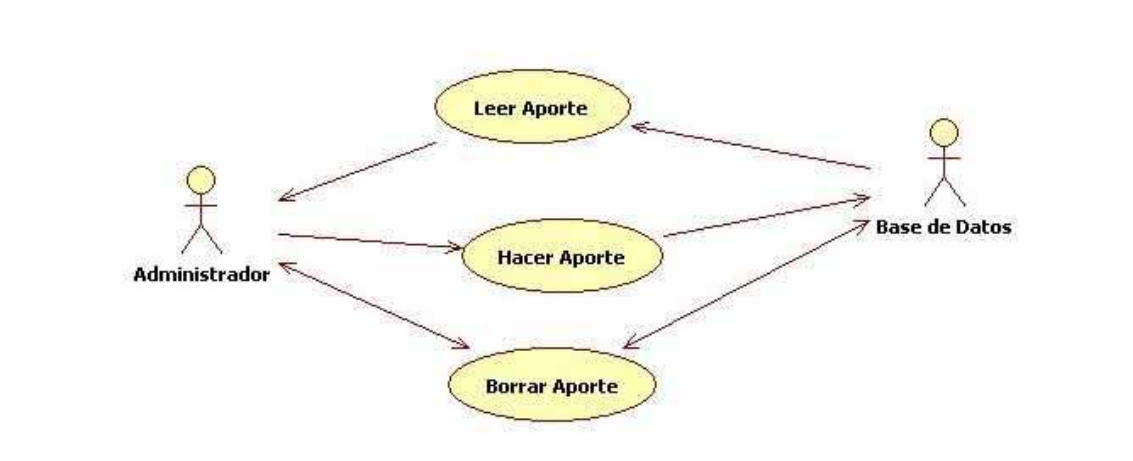
Diagramas de Caso de Uso

Registrar Usuario

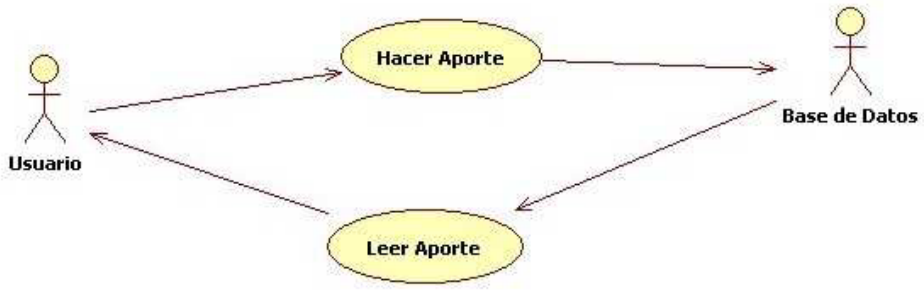
Cambio y Baja Usuario



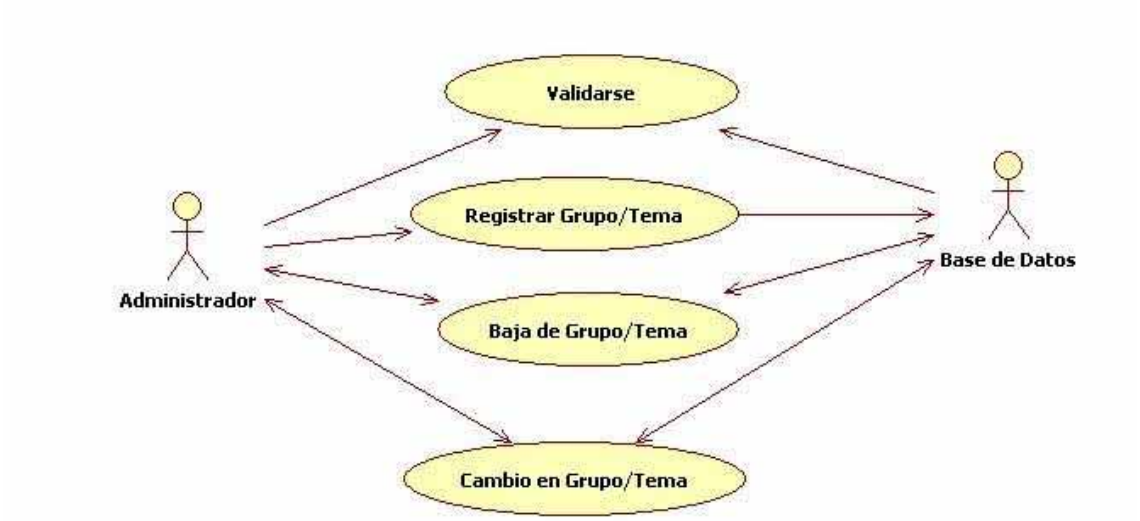
Alta Cambio Baja Aporte (Administrador)



Alta Baja Aporte (Usuario)



Alta Baja Cambio Grupo/Tema (Administrador)

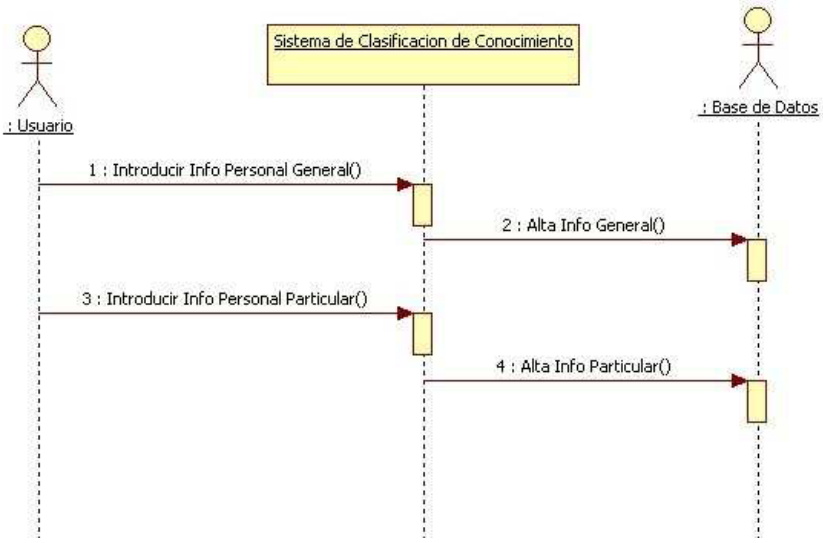


Baja Cambio Grupo/Tema (Usuario)

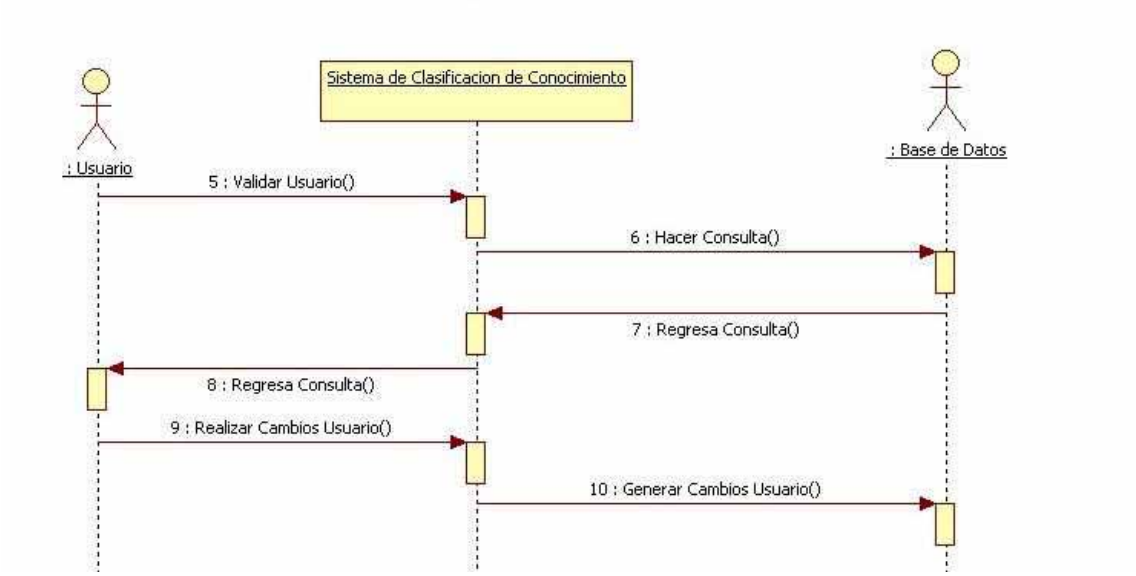
Baja Usuario (Administrador)

Diagramas de Secuencia

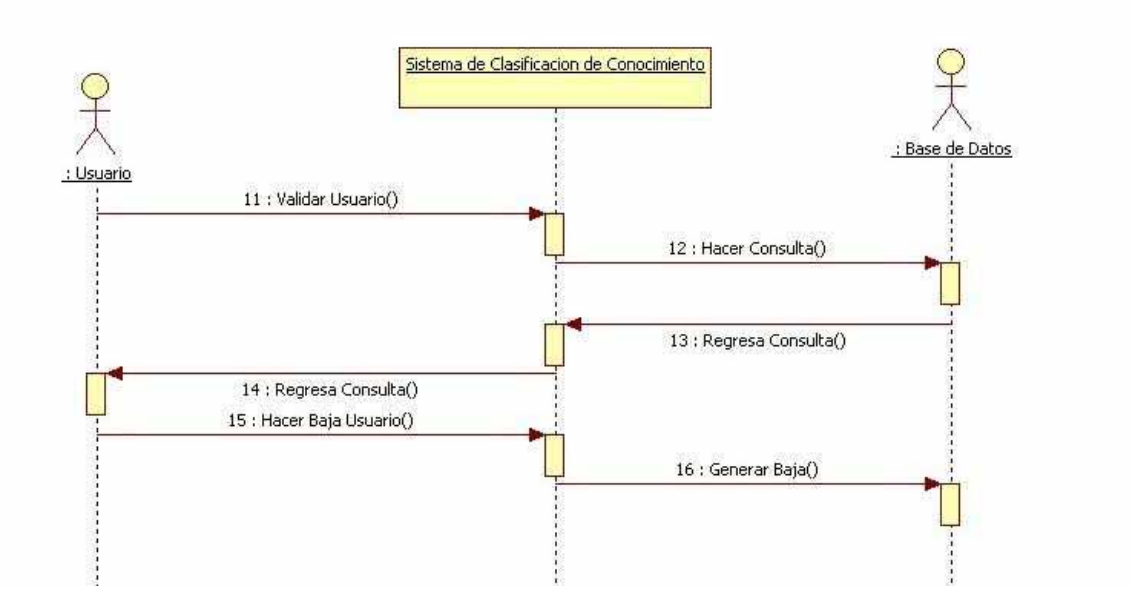
Registrar Usuario



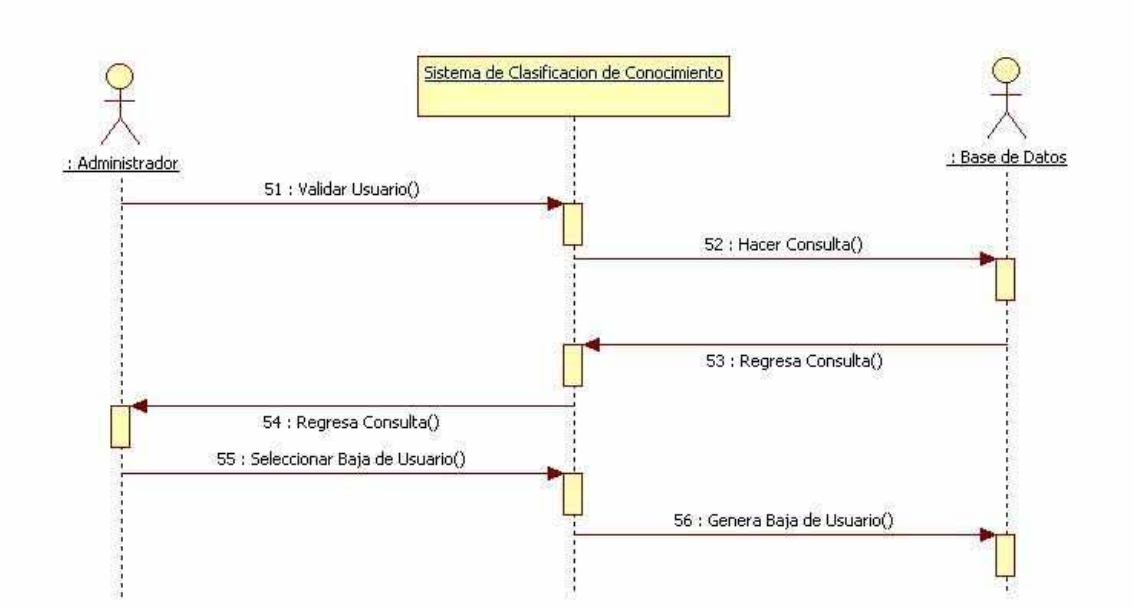
Cambio Usuario



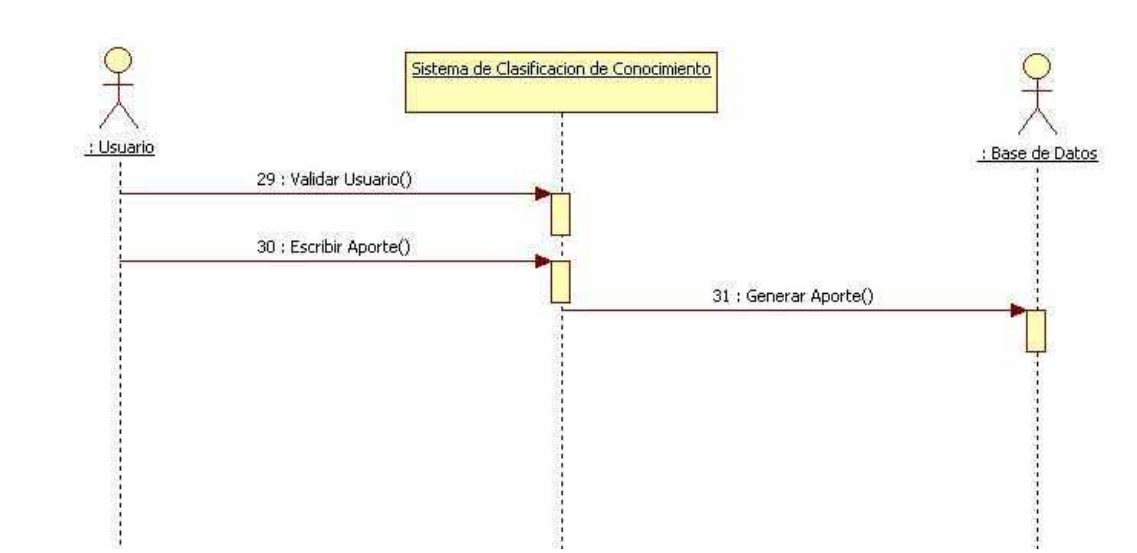
Baja Usuario (Usuario)



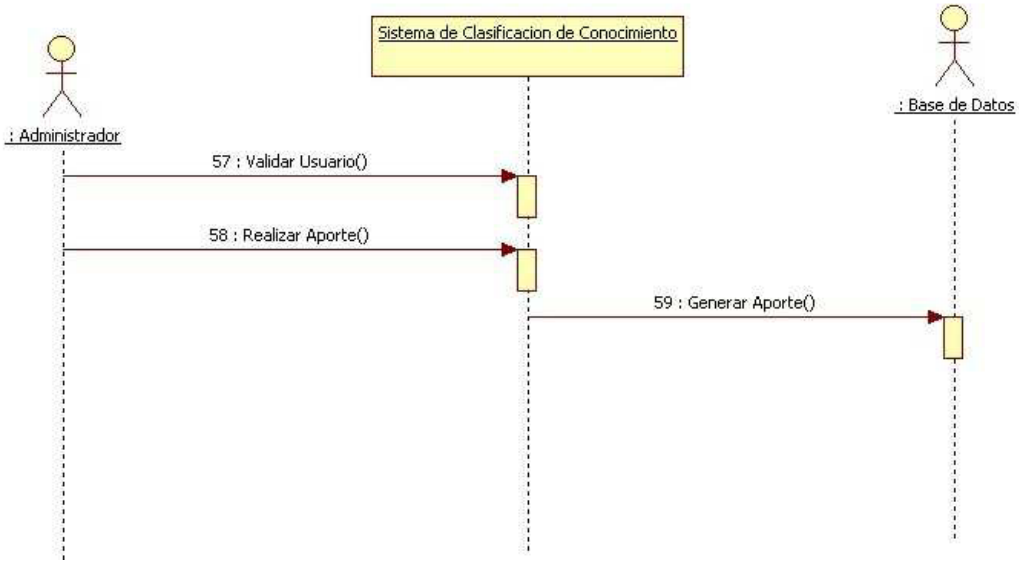
Baja Usuario (Administrador)



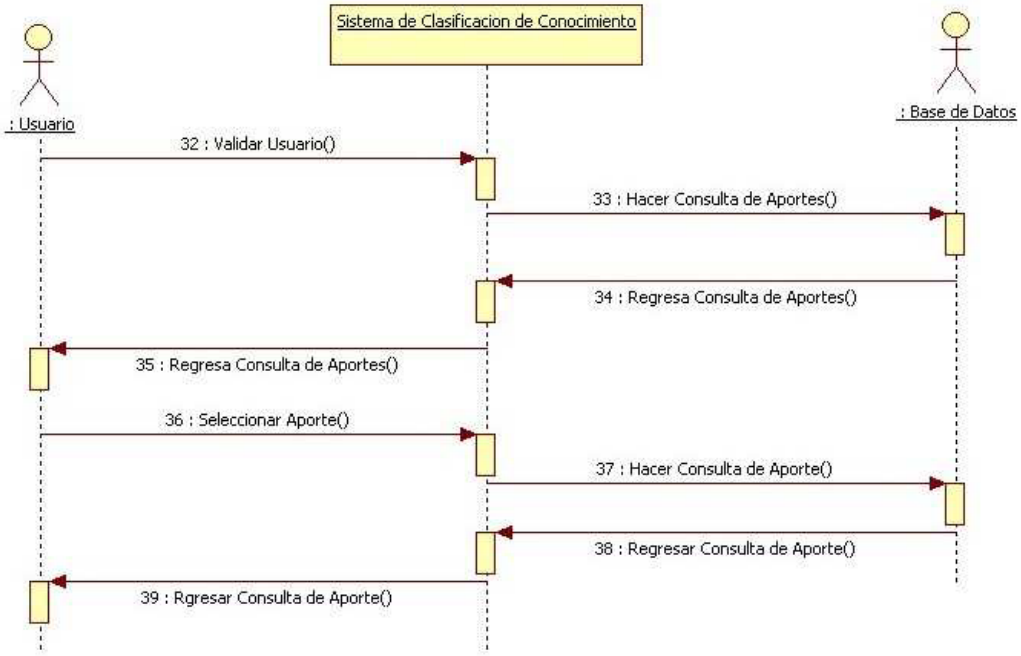
Hacer Aporte (Usuario)



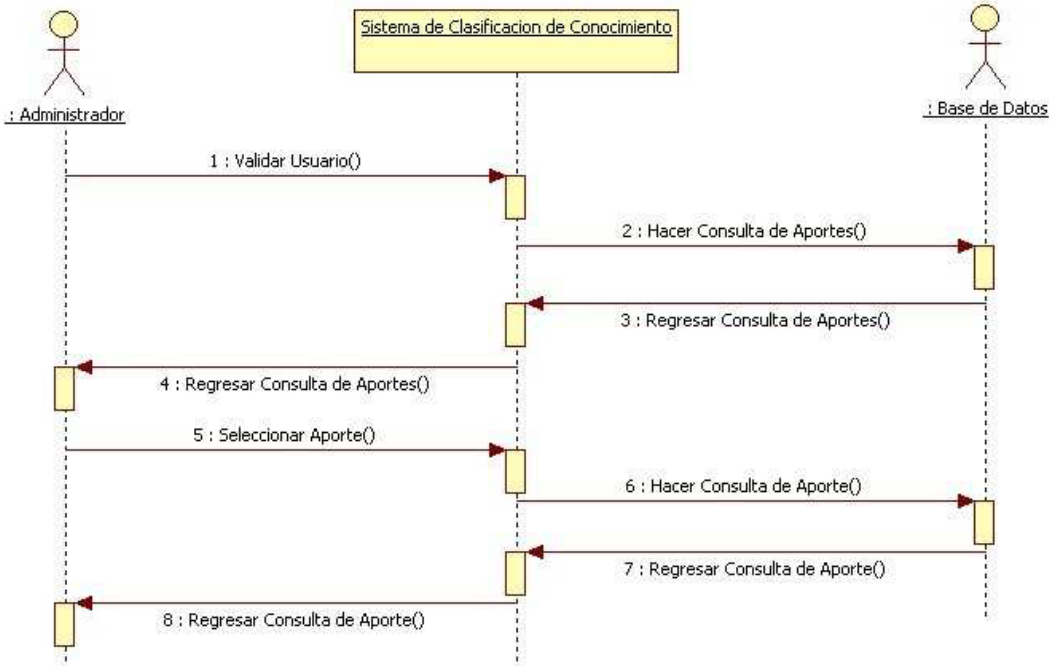
Hacer Aporte (Administrador)



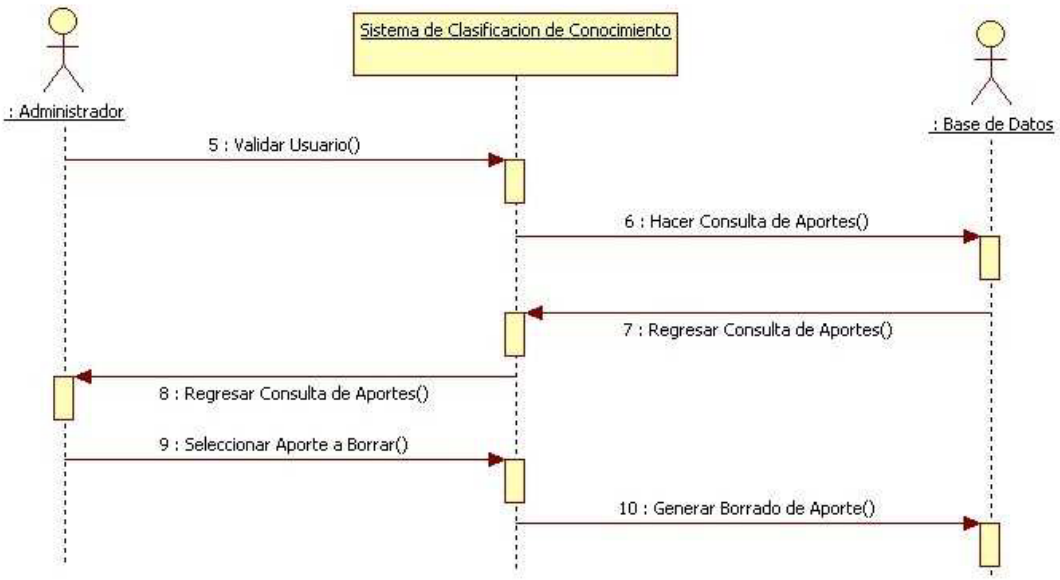
Leer Aporte (Usuario)



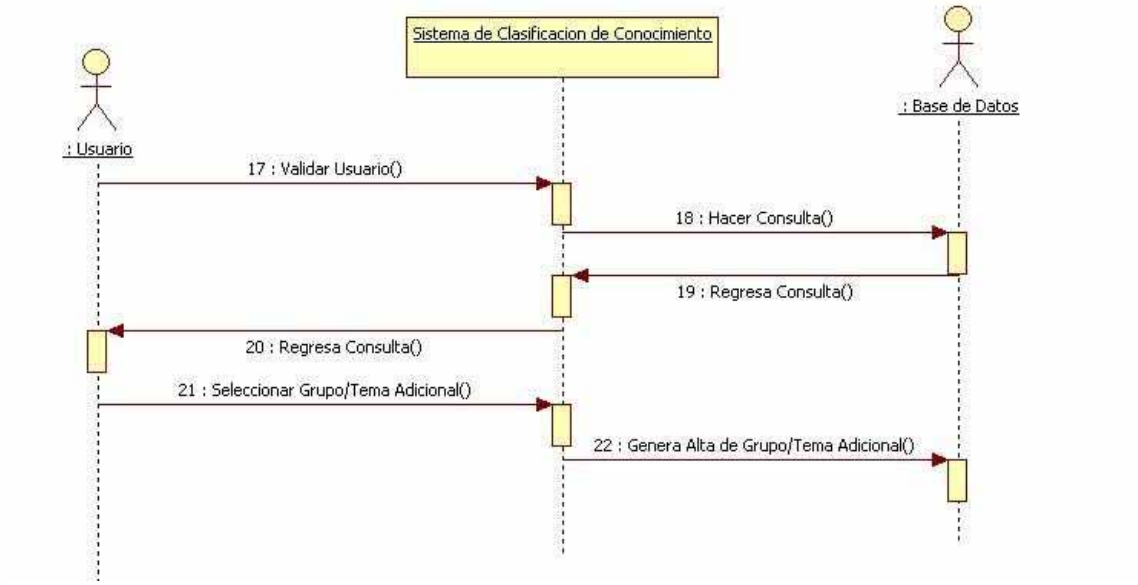
Leer Aporte (Administrador)



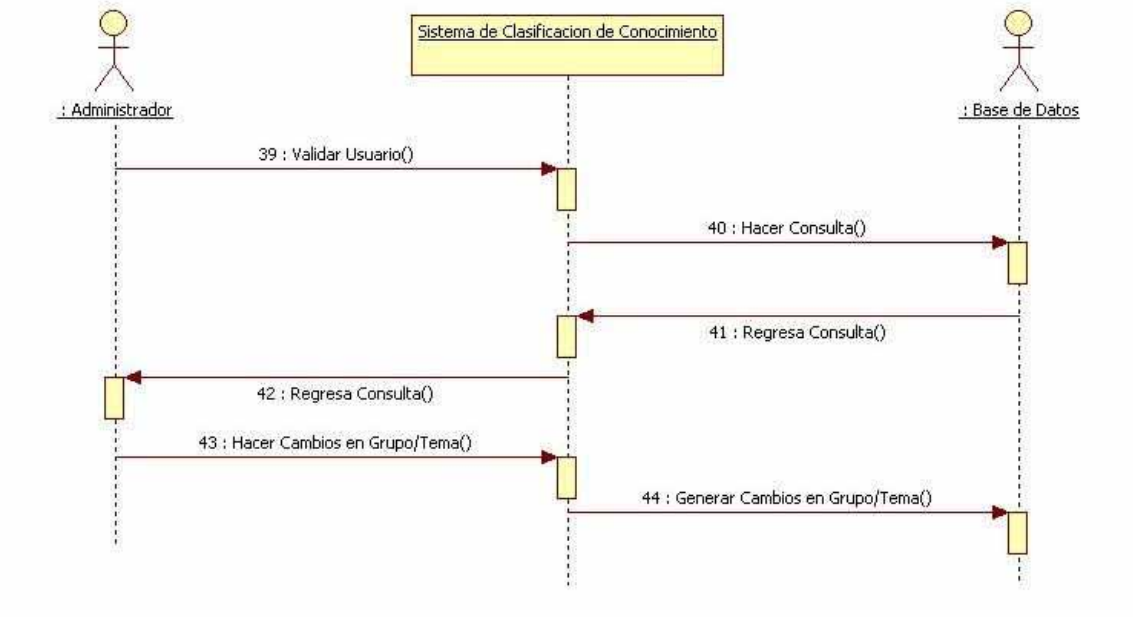
Borrar Aporte (Administrador)



Cambio de Grupo/Tema (Usuario)

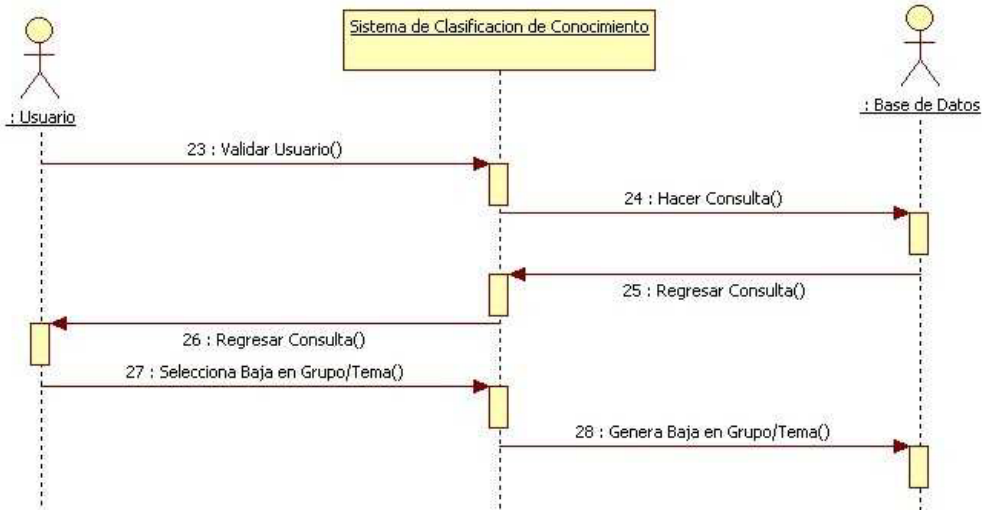


Cambio de Grupo/Tema (Administrador)

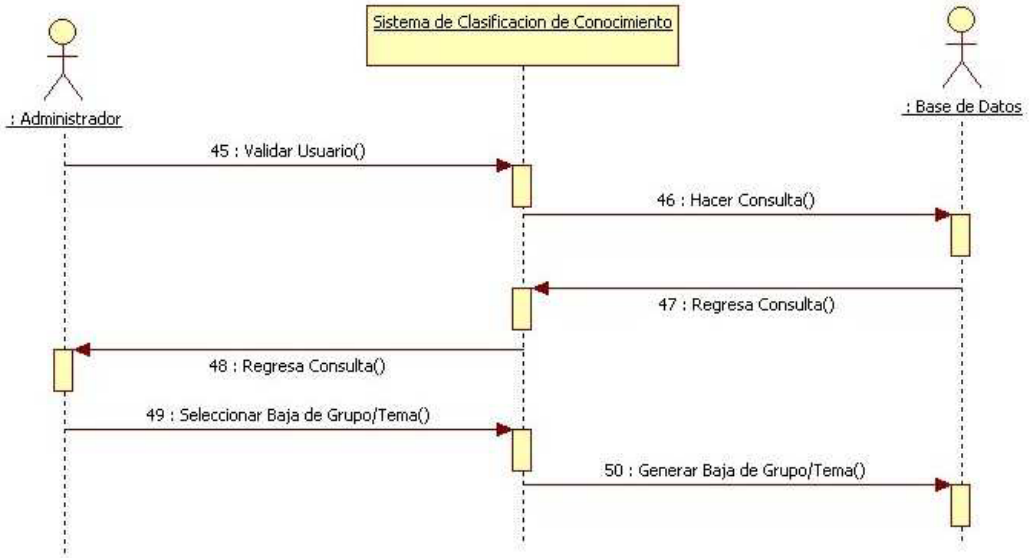


Baja Grupo/Tema (Usuario)

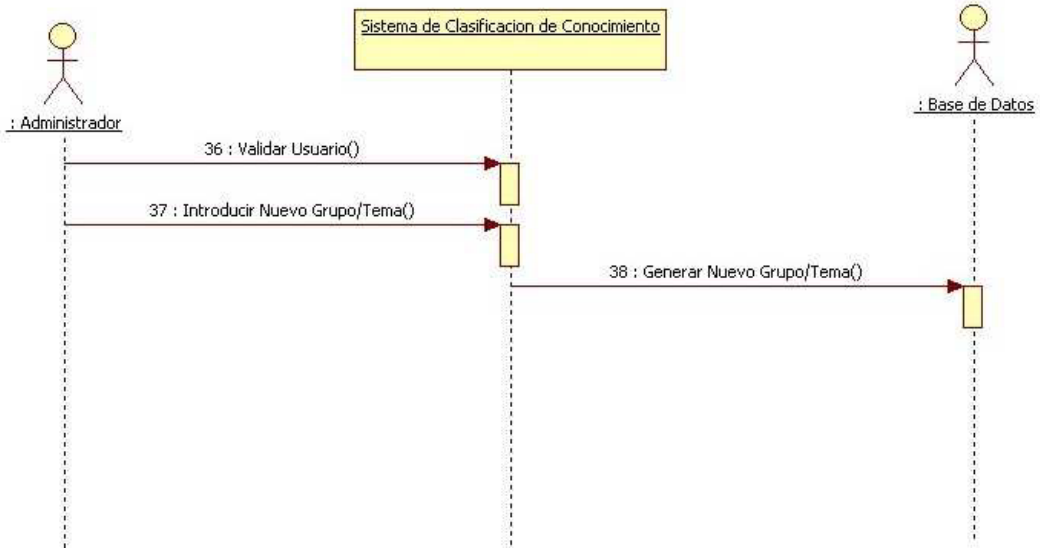
:



Baja Grupo/Tema (Administrador)

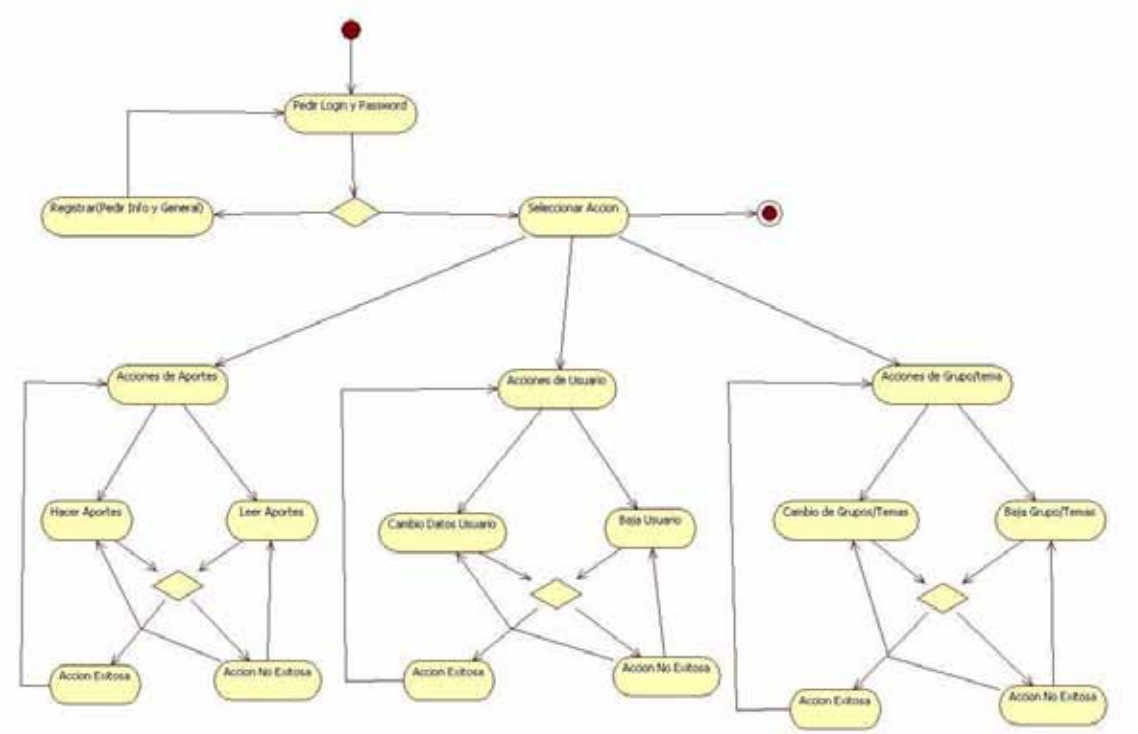


Registrar Grupo



Diagramas de Actividades

Usuario



Administrador

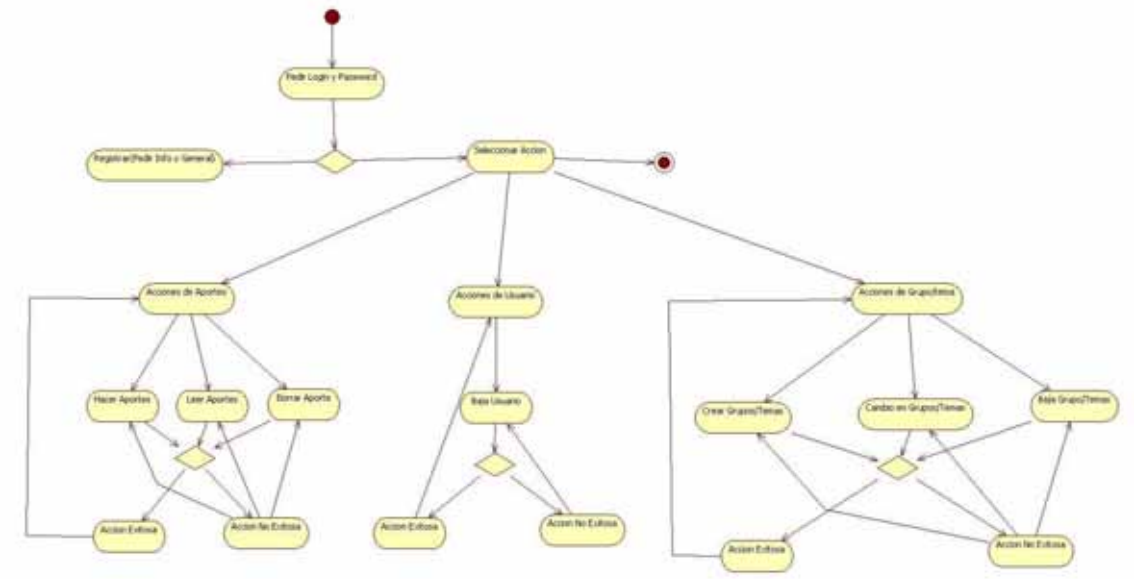


Diagrama de la Base de Datos

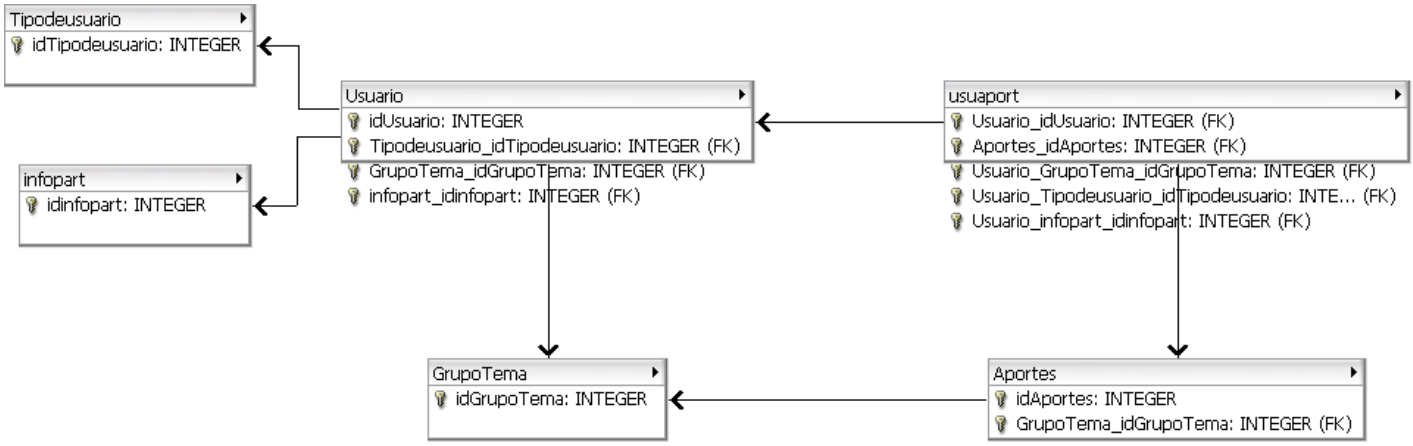


Diagrama de Clases

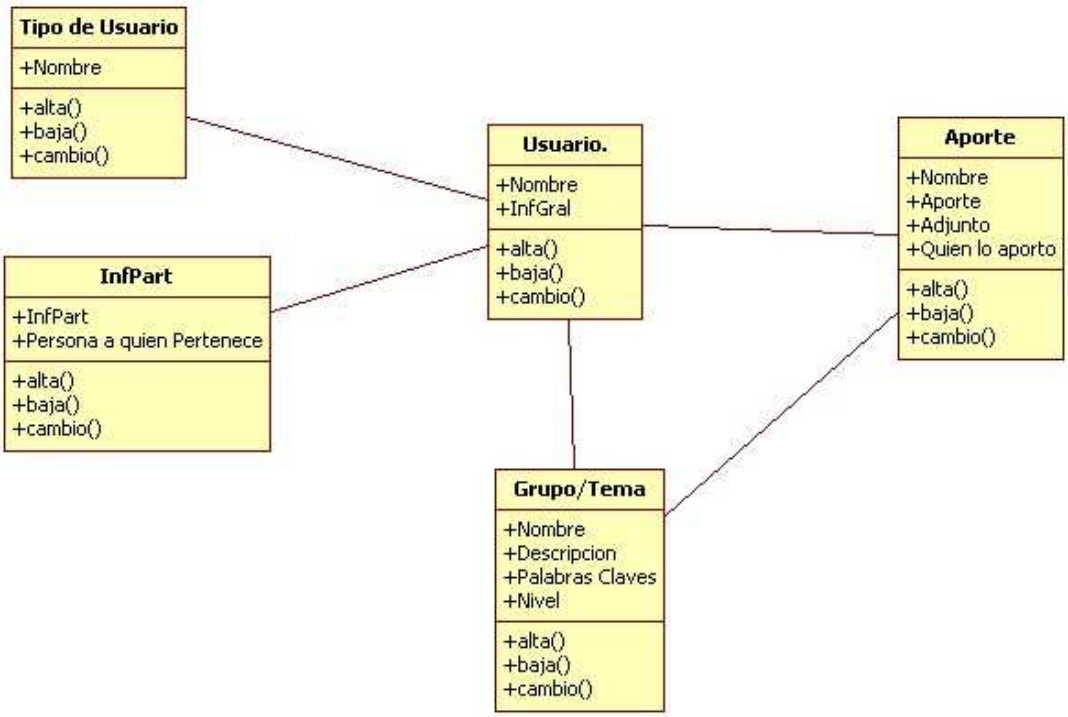
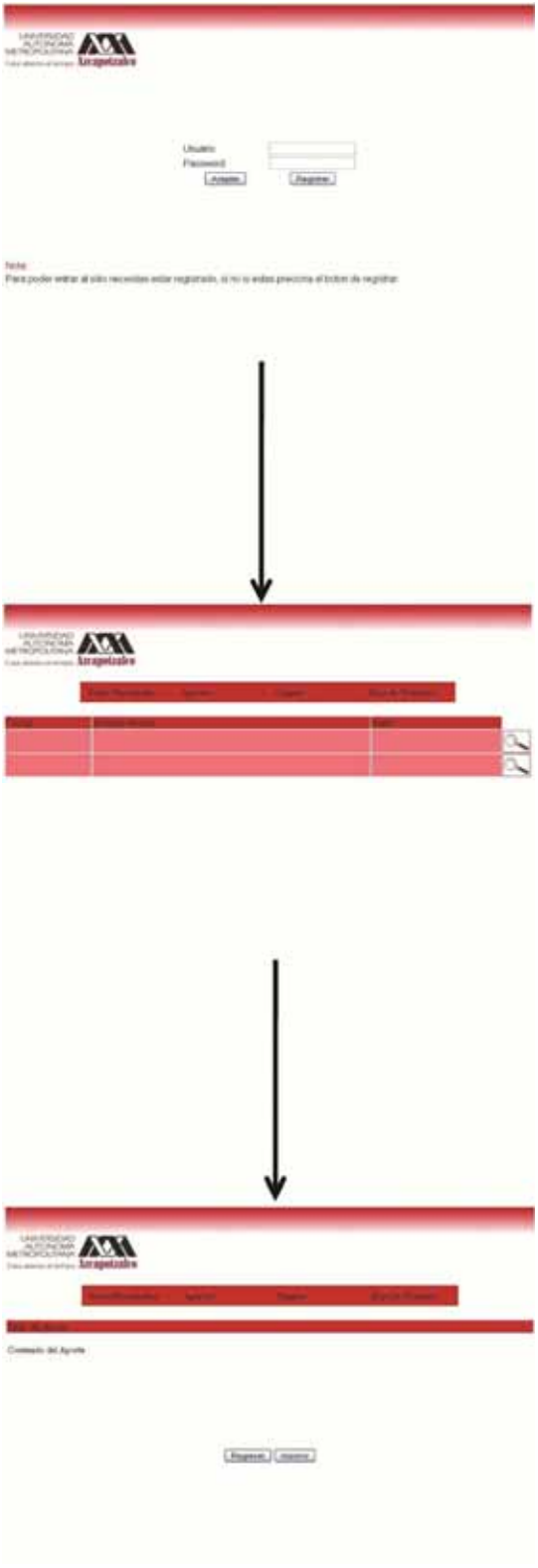
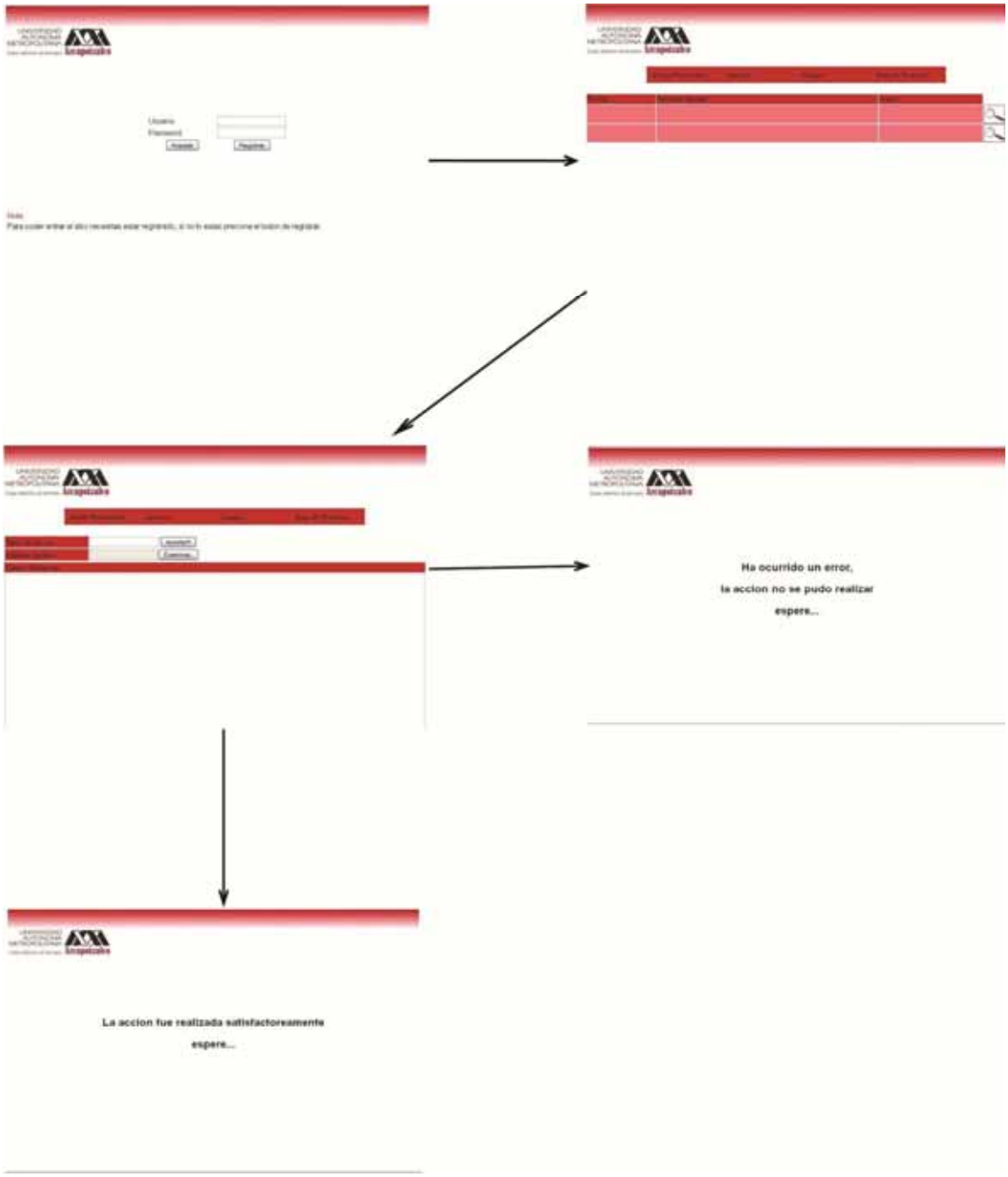


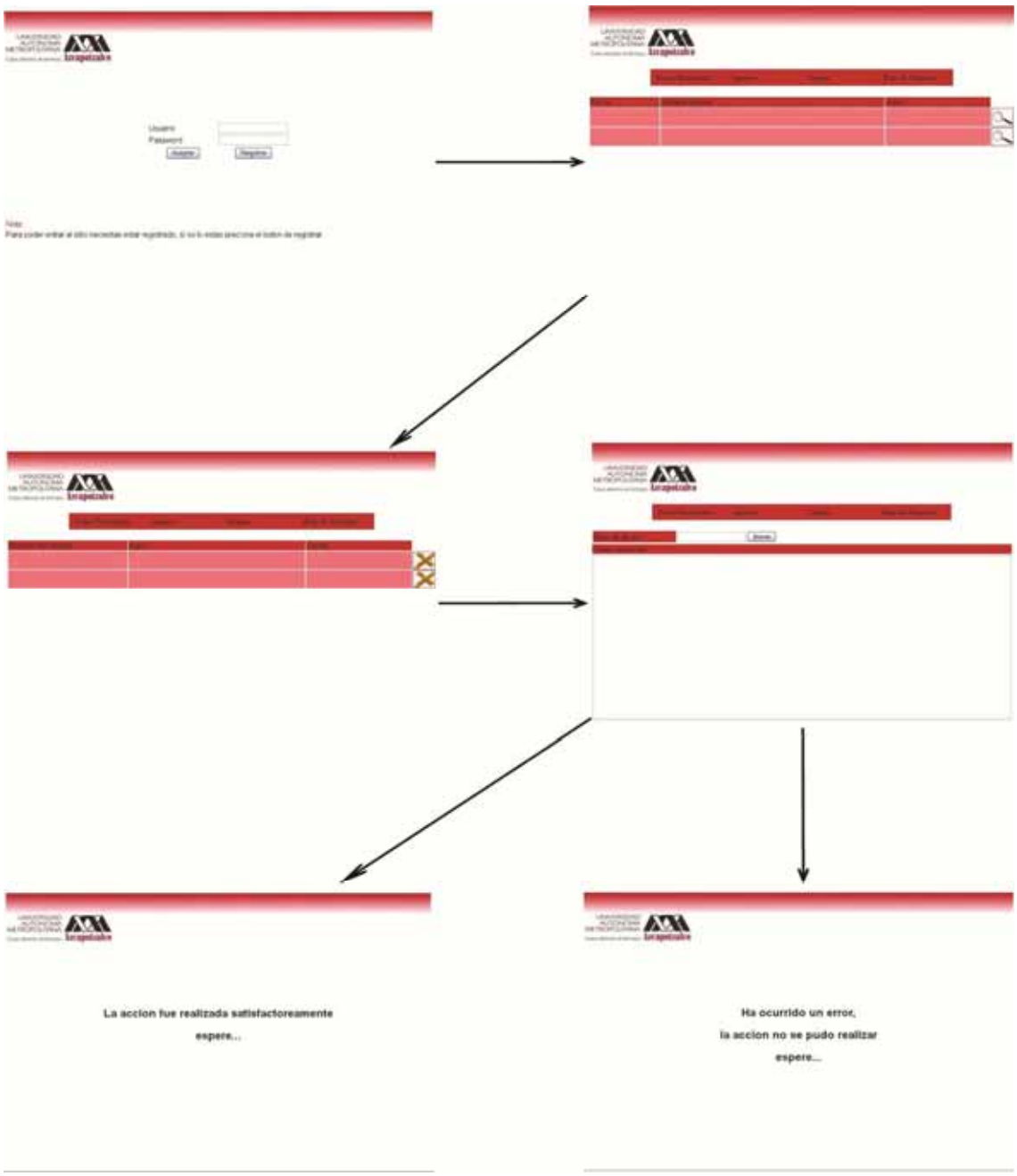
Diagrama de Navegación entre Pantallas



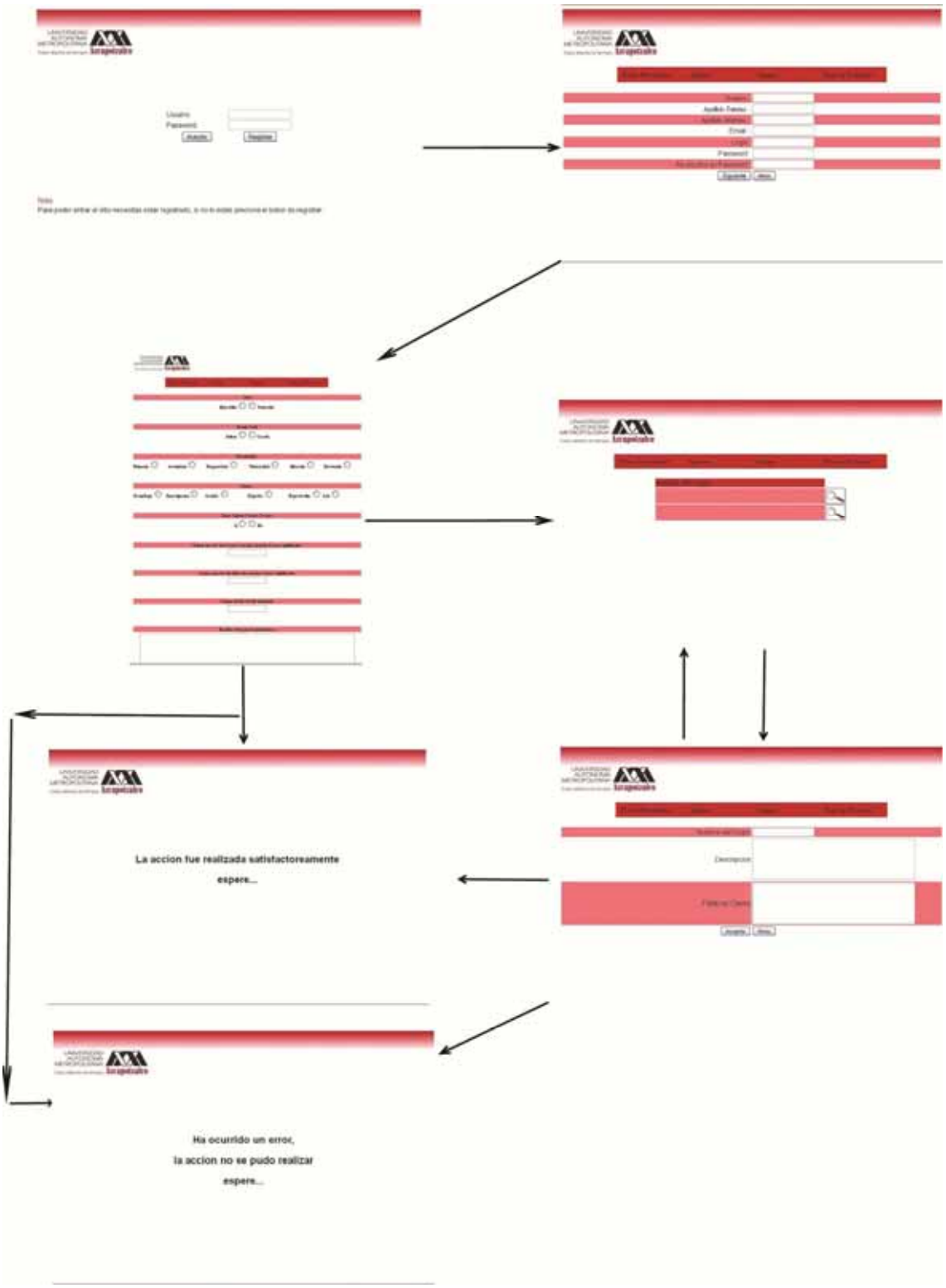
Leer Aporte



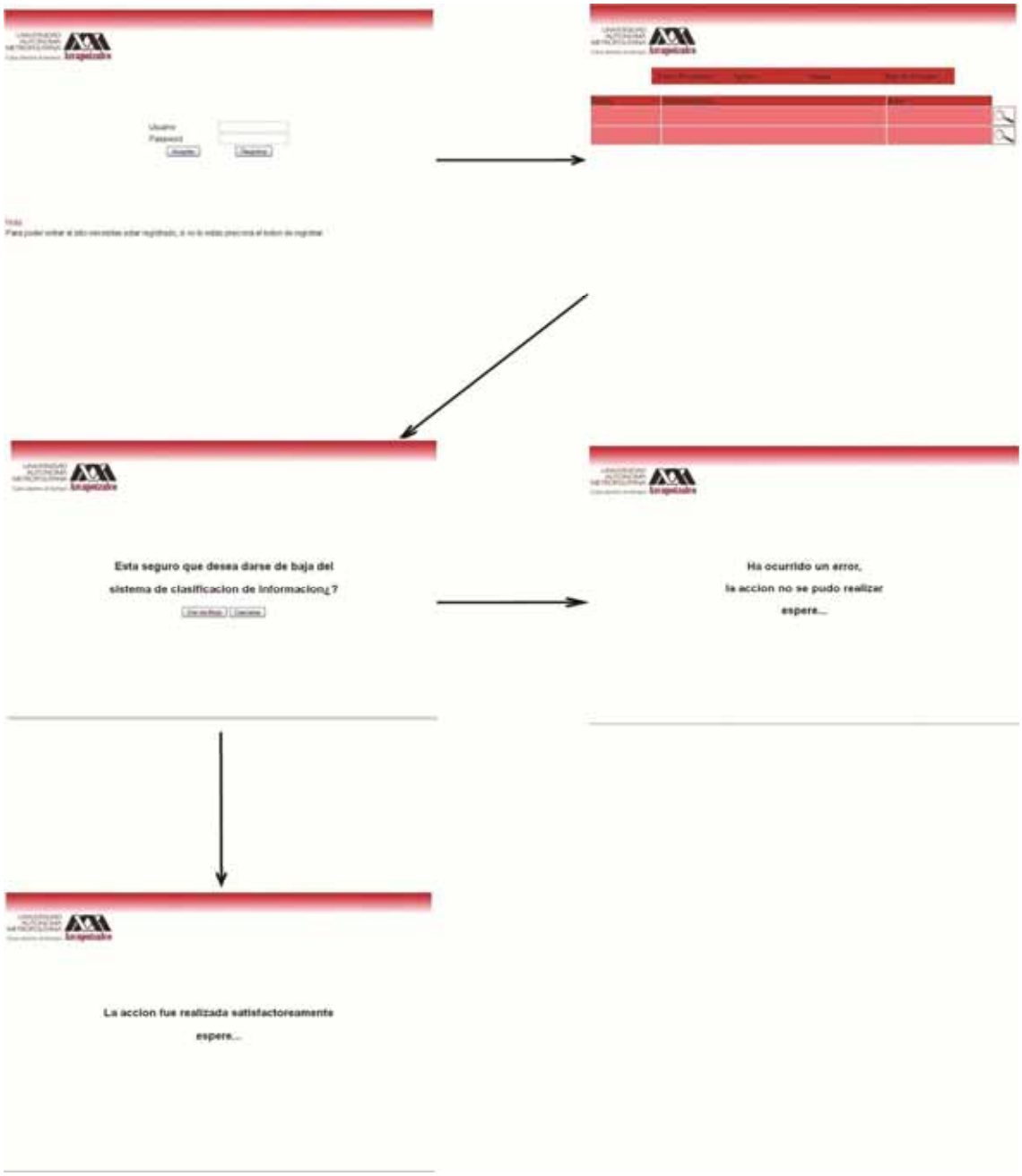
Escribir Aporte



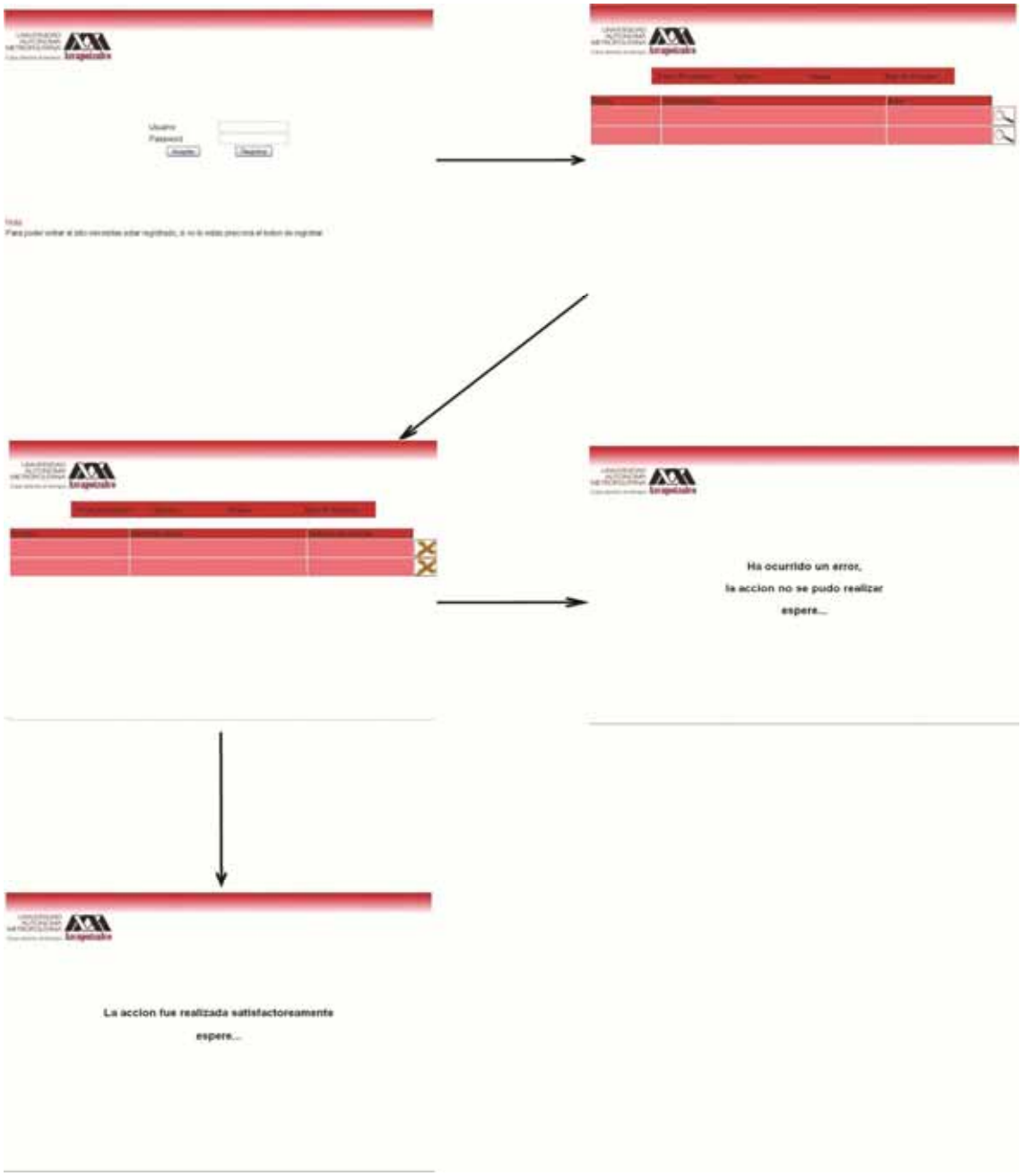
Borrar Aporte



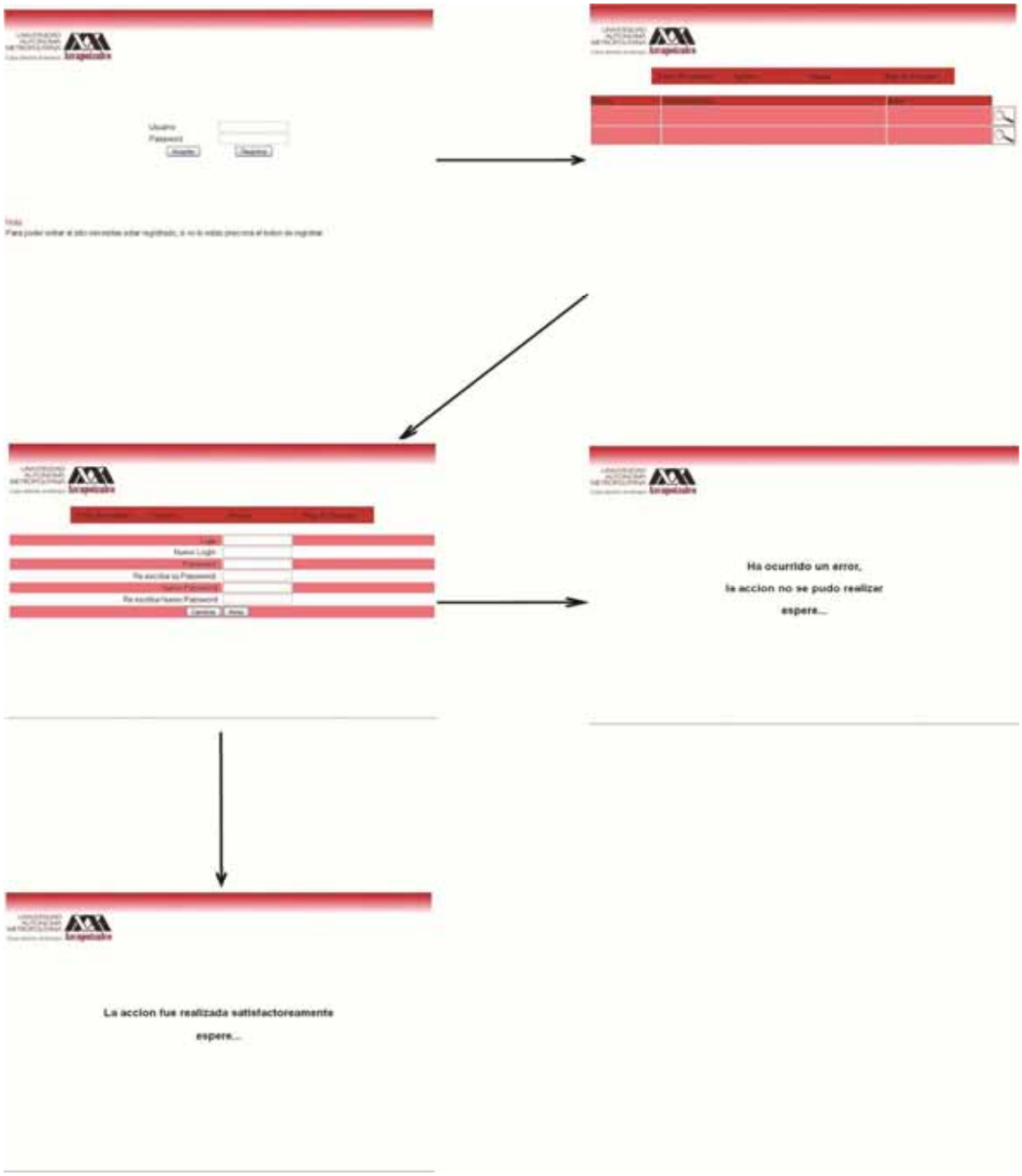
Alta Usuario



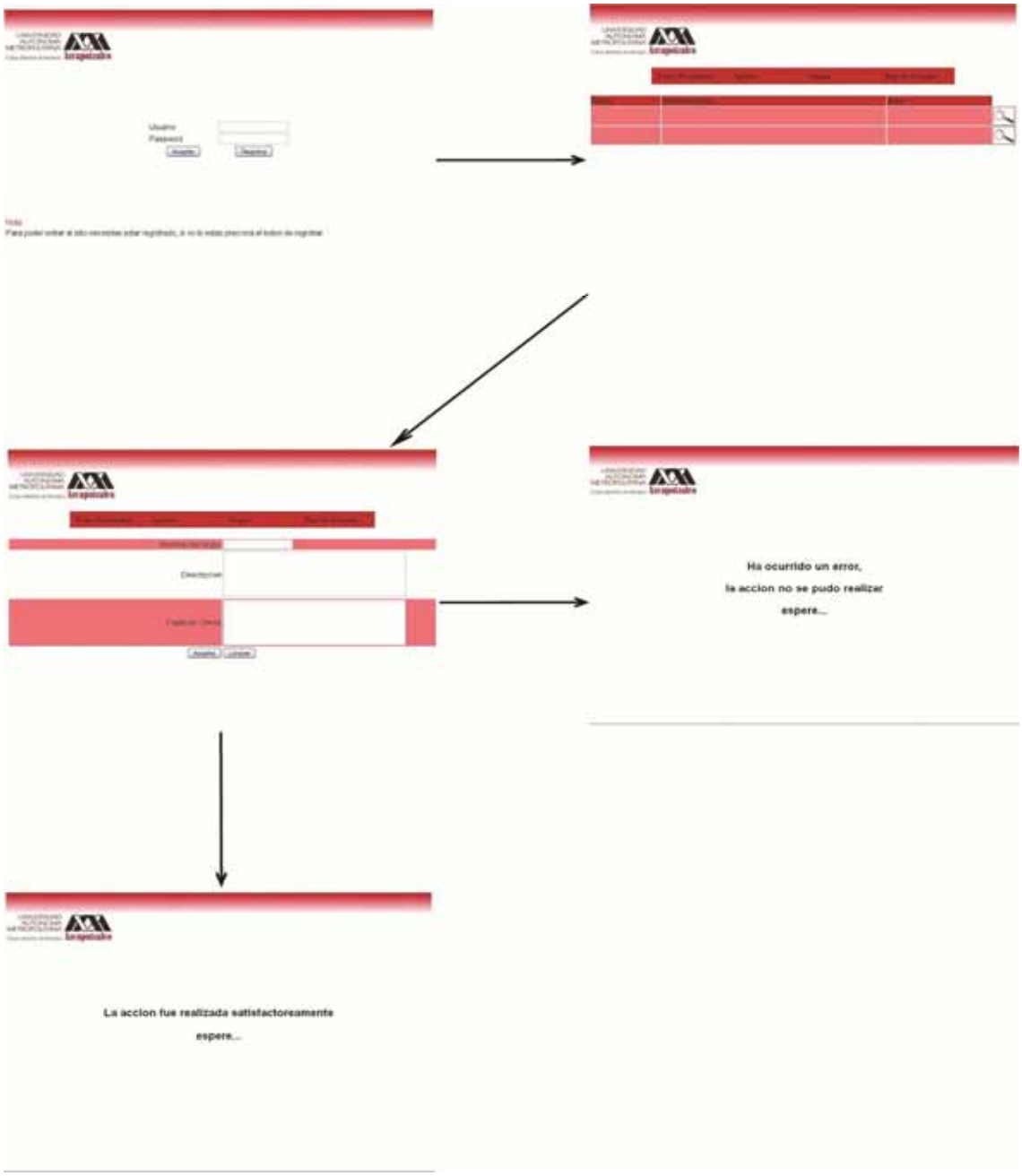
Borrar Usuario (Usuario)



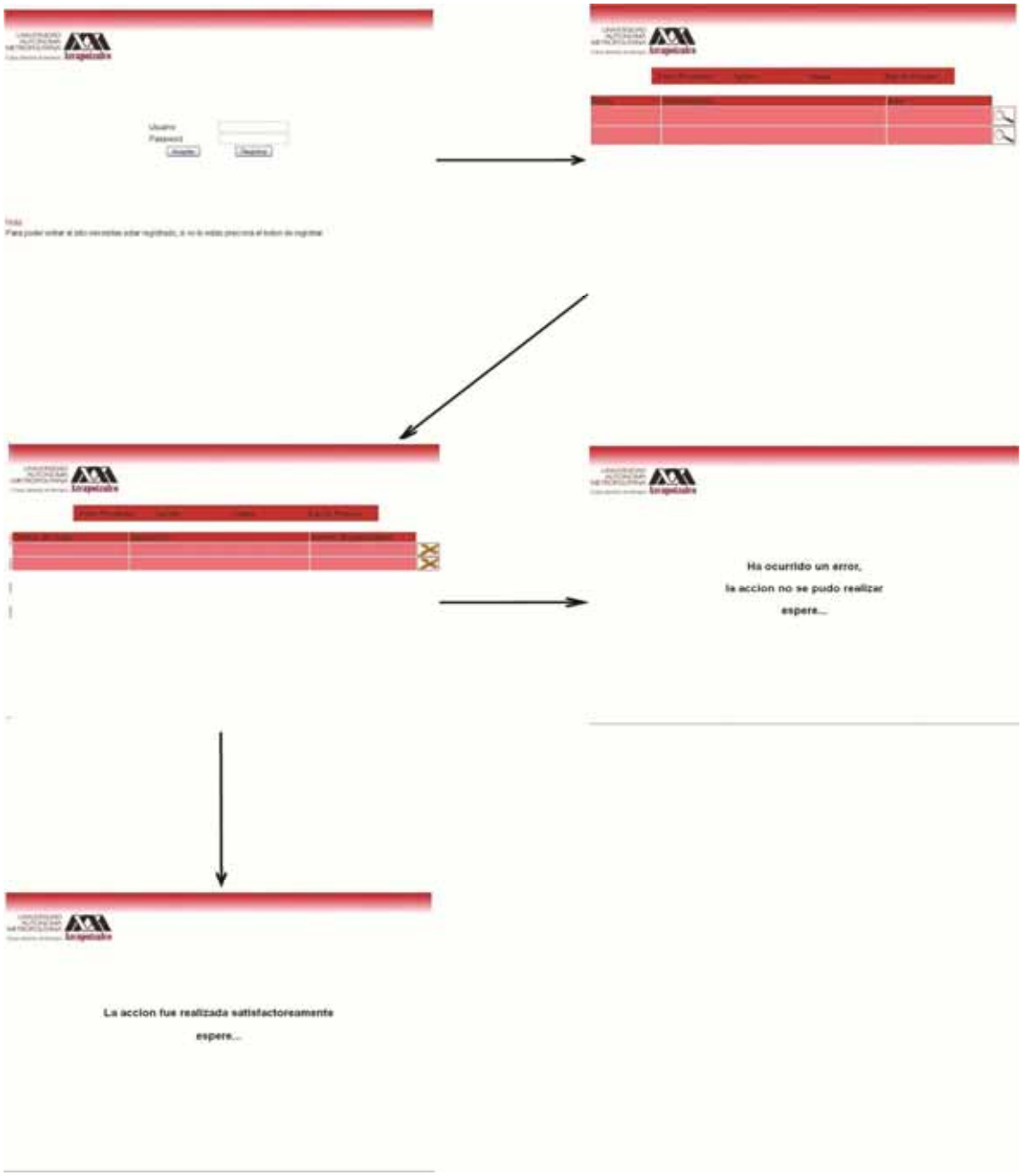
BajaUsuario (Administrador)



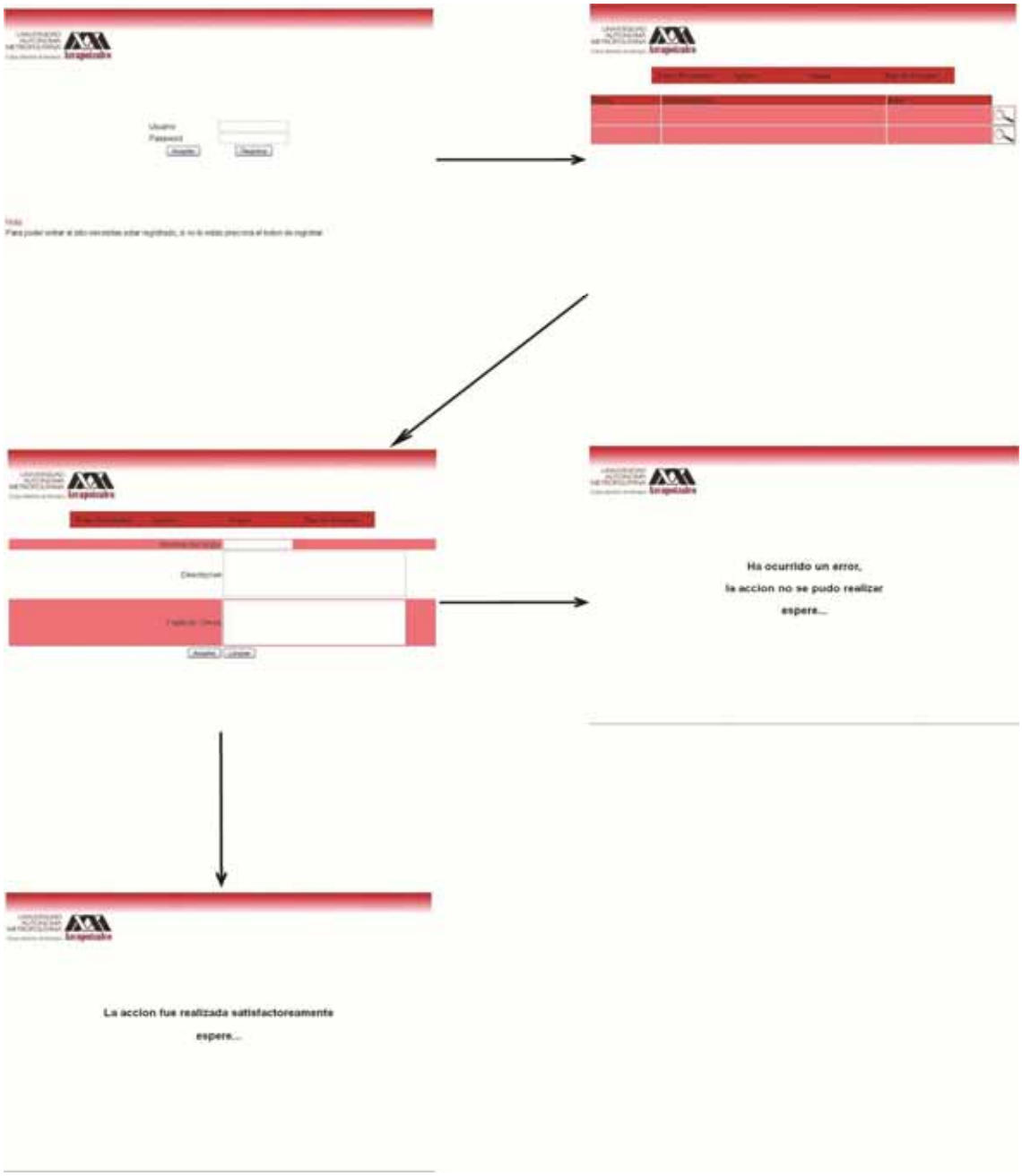
Cambio Usuario


Ha ocurrido un error,
la accion no se pudo realizar
espere...
La accion fue realizada satisfactoreamente
espere...

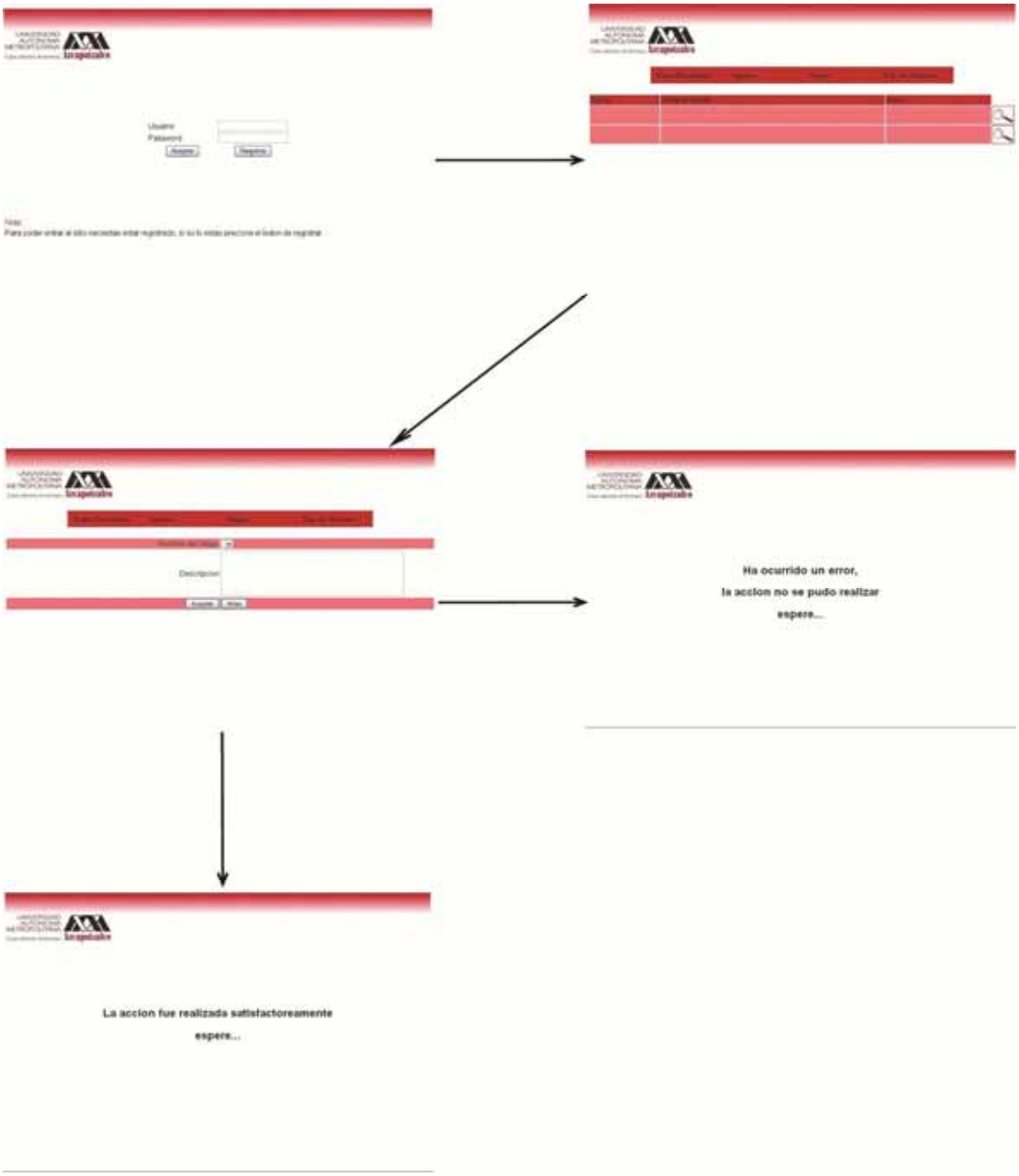
Crear Grupo



Borrar Grupo



Cambio en Grupo



Seleccionar Otro Grupo

Diccionario de Datos

Registro Aporte Form

aporte (String): En este campo se almacena el contenido del aporte.

archivo (String): En este campo se almacena el archivo adjunto del aporte.

fecha (date): En este campo se almacena la fecha en la que se realiza el aporte.

idusu (int): En este campo se almacena el id del aporte.

nombre (String): En este campo se almacena el nombre del usuario que realizo el aporte.

titulo (String): En este campo se almacena el nombre del aporte.

Registro Cuestionario Form

carrera (int): En este campo se almacena el id de la carrera del usuario.

doctorado (int): En este campo se almacena el id del doctorado del usuario.

ec (int): En este campo se almacena el id del estado civil del usuario.

escolaridad (int): En este campo se almacena el id de la escolaridad del usuario.

gusto (int): En este campo se almacena el id del gusto del usuario.

iduser (int): En este campo se almacena el id del usuario.

leer (String): En este campo se almacena lo que al usuario le gustaría leer.

maestria (int): En este campo se almacena el id de la maestría del usuario.

sexo (int): En este campo se almacena el id del sexo del usuario.

tec (int): En este campo se almacena el id de si tiene carrera técnica el usuario.

tec2 (int): En este campo se almacena el id de la carrera técnica del usuario.

Registro Grupo Form

lic (int): En este campo se almacena el id de la carrera del grupo.

docto (int): En este campo se almacena el id del doctorado del grupo.

ec (int): En este campo se almacena el id del estado civil del grupo.

escolaridad (int): En este campo se almacena el id de la escolaridad del grupo.

gusto (int): En este campo se almacena el id del gusto del grupo.

idusu (int): En este campo se almacena el id del grupo.

descripcion (String): En este campo se almacena la descripción del grupo.

maestria (int): En este campo se almacena el id de la maestría del grupo.

sexo (int): En este campo se almacena el id del sexo del grupo.

ct (int): En este campo se almacena el id de si tiene carrera técnica el grupo.

ct2 (int): En este campo se almacena el id de la carrera técnica del grupo..

tags (String): En este campo se almacena las palabras claves del grupo.

nombre (String): En este campo se almacena el nombre del grupo.

Registro Usuario Form

apellidoMaterno (String): En este campo se almacena el apellido materno del usuario.

apellidoPaterno (String): En este campo se almacena el apellido paterno del usuario.

aportes (int): En este campo se almacena el numero de aportes del usuario.

email (String): En este campo se almacena el e-mail del usuario.

fecha (date): En este campo se almacena la fecha de alta del usuario.

idusu (int): En este campo se almacena el id del usuario.

login (String): En este campo se almacena el login del usuario.

login2 (String): En este campo se almacena el login del usuario.

nombre (String): En este campo se almacena el nombre del usuario.

password (String): En este campo se almacena el password del usuario.

password2 (String): En este campo se almacena el password del usuario.

Validacion Usuario Form

idgrupo (int): En este campo se almacena el id del grupo al que pertenece el usuario.

iduser (int): En este campo se almacena el id del usuario.

password (String): En este campo se almacena el password del usuario.

tipousu (int): En este campo se almacena el tipo de usuario del usuario.

usuario (String): En este campo se almacena el nombre del usuario.

Casos de Uso Desglosado

Caso de Uso: Registrar Usuario

Descripción: El usuario entra en la página de login del sistema, selecciona el link de registrarse, se le muestra la pantalla donde debe de llenar un formulario con sus datos, Nombre, Apellidos, Nombre de Usuario y Password, se presiona el botón de registra, el sistema verificara que no exista ese Nombre de Usuario, si ya existe no se registrara al usuario, si no existe, se muestra la siguiente pantalla donde llena otro formulario en el cual se hacen pregunta sobre el tipo de sexo del usuario, estado civil, nivel de educación, gustos, carreras técnicas, carreras, maestrías y doctorados obtenidos y por ultimo una descripción sobre lo que le gustaría leer, y se presiona el botón de registrar, el sistema ejecutar una consulta donde con todos los datos obtenidos los utilizara como parámetros para encontrar un grupo, si encuentra uno o más grupos dividirá la descripción dada en palabras y las comparara con las palabras clave de cada grupo, en el grupo donde haya más coincidencias en donde lo inscribirá, si en la consulta no encontró ningún grupo, lo inscribirá pero sin grupo.

Caso de Uso: Cambio de Usuario

Descripción: El usuario entra en la página de login del sistema, introduce su usuario y password, se le muestra la pantalla de inicio, en el menú de la parte superior se elige la opción de cambio de usuario, en la pantalla se le mostrara un formulario donde se podrá cambiar el usuario y contraseña, se introducen los nuevos datos, el sistema verificara que estos datos no existan, si existen, la acción no se podrá realizar y si no existen el cambio se realizara.

Caso de Uso: Baja de Usuario

Descripción:

Si el usuario tiene derechos de administrador:

El usuario entra en la página de login del sistema, introduce su usuario y password, se le muestra la pantalla de inicio, en el menú de la parte superior se elige la opción baja de usuario, el sistema le mostrara en pantalla una lista con todos los usuarios existentes en el sistema, el administrador elegirá al usuario y presiona sobre el icono de la “X”, con ello el sistema cambiara el estatus del usuario a baja.

Si el usuario no tiene derechos de administrador:

El usuario entra en la página de login del sistema, introduce su usuario y password, se le muestra la pantalla de inicio, en el menú de la parte superior se elige la opción baja de usuario y con ello el sistema cambio el estatus del usuario a baja.

Caso de Uso: Alta de aporte:

Descripción: El usuario entra en la página de login del sistema, introduce su usuario y password, se le muestra la pantalla de inicio, en el menú de la parte superior se elige la opción hacer aporte, el sistema le mostrara una pantalla en la cual en un formulario introducirá el nombre del aporte, su contenido y si se quiere anexar un archivo, se presiona le botón de aportar y el sistema verificara que los campos de nombre de aporte y contenido contengan algo, si no es así, no se realizara la operación, en caso contrario, el sistema registrara el aporte.

Caso de Uso: Baja de Aporte

Descripción: Si el usuario tiene derechos de administrador:

El usuario entra en la página de login del sistema, introduce su usuario y password, se le muestra la pantalla de inicio, en el menú de la parte superior se elige la opción baja de aporte, se le mostrar un listado con todos los grupos existentes, el usuario seleccionara un grupo y con ello se le mostrara todos los aportes de ese grupo, ahí elegirá el aporte que quiere dar de baja.

Caso de Uso: Leer Aporte

Descripción: El usuario entra en la página de login del sistema, introduce su usuario y password, se le muestra la pantalla de inicio, en la pantalla de inicio se le muestra todos los aportes existentes del grupo al cual está inscrito, el usuario selecciona el aporte q desee leer.

Caso de Uso: Registrar Grupo

Descripción:

Si el usuario tiene derechos de administrador:

El usuario entra en la página de login del sistema, introduce su usuario y password, se le muestra la pantalla de inicio, en el menú de la parte superior se elige la opción registrar grupo, se le mostrara una pantalla con un formulario en el introducirá el nombre del grupo, la descripción del grupo, las palabras claves del grupo y los parámetros sobre a quién va dirigido ese grupo, estos parámetros son, sexo, estado civil, nivel de estudios, intereses, carreras técnicas, carreras, maestrías y doctorados de los usuario, se presiona el botón registrar y el sistema verificara que los campos de nombre, descripción y palabras claves no estén vacías, si lo están el sistema no realizara la acción, en caso contrario el sistema quedara registrado.

Caso de Uso: Cambio en Grupo

Descripción:

Si el usuario tiene derechos de administrador:

El usuario entra en la página de login del sistema, introduce su usuario y password, se le muestra la pantalla de inicio, en el menú de la parte superior se elige la opción cambio en grupo, se le mostrara una pantalla con la lista de grupos existentes, el usuario selecciona el grupo que desee modificar, después se le mostrar una pantalla con el mismo formulario que cuando registra un grupo, solo que ahora con los datos actuales del grupo, los que podrá modificar y presionando el botón validar, el sistema verificara que el campo de nombre, descripción y palabras claves no estén vacios, si lo están la acción no se realizara, en caso contrario la acción se realizara

Caso de Uso: Cambio de Grupo

Descripción: El usuario entra en la página de login del sistema, introduce su usuario y password, se le muestra la pantalla de inicio, en el menú de la parte superior se elige la opción cambio de grupo, se le mostrara una lista con todos los grupos existentes, el usuario selecciona un grupo y con ello se le mostrara el nombre y descripción del grupo, si al usuario presiona el botón validar, se modificara su grupo al que está inscrito, el que tiene será substituido por este que acaba de elegir.

Caso de Uso: Baja de Grupo

Descripción:

Si el usuario tiene derechos de administrador:

El usuario entra en la página de login del sistema, introduce su usuario y password, se le muestra la pantalla de inicio, en el menú de la parte superior se elige la opción baja de grupo, se le mostrara una lista con todos los grupos existentes, el usuario selecciona un grupo y con ello el estatus del grupo cambiara a baja.

Manual de Instalación

Software necesario para la ejecución del Sistema de Clasificación de Información en Equipos de Trabajos:

Apache TomCat

MySQL

Eclipse

Primero Instalamos el programa Apache TomCat.

Después se instala el MySQL, una vez instalado se abre una ventana de comando del MySQL en esta ventana lo que se va a hacer es pegar el script del esquema físico de la base de datos y las tablas para crearlas ya que son necesarias para almacenar la información del Sistema de Clasificación de Información en Equipos de Trabajos.

Por último se ejecuta el Eclipse, una vez dentro del programa, se importa y se ejecuta el Sistema de Clasificación de Información en Equipos de Trabajos, se elige a TomCat como contenedor del sistema y listo.

Manual de Usuario:

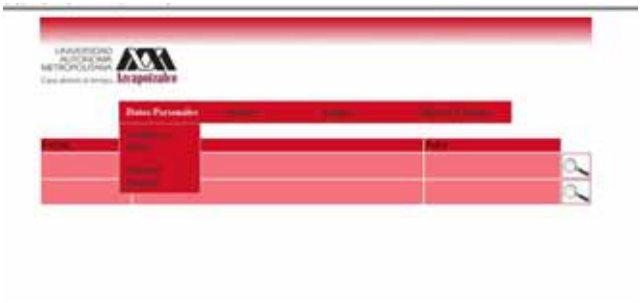
El usuario entra en la página de login del sistema, selecciona el link de registrarse.

Se le muestra la pantalla donde debe de llenar un formulario con sus datos, Nombre, Apellidos, Nombre de Usuario y Password, se presiona el botón de registra.

El sistema verificara que no exista ese Nombre de Usuario, si ya existe no se registrara al usuario, si no existe, se muestra la siguiente pantalla donde llena otro formulario en el cual se hacen pregunta sobre el tipo de sexo del usuario, estado civil, nivel de educación, gustos, carreras técnicas, carreras, maestrías y doctorados obtenidos y por ultimo una descripción sobre lo que le gustaría leer, y se presiona el botón de registra.

Cambio de datos de Usuario

En la pantalla de inicio, en el menú de la parte superior se elige la opción de cambio de usuario.



En la pantalla se le mostrara un formulario donde se podrá cambiar el usuario y contraseña, se introducen los nuevos datos.

Baja de Usuario

En la pantalla de inicio, en el menú de la parte superior se elige la opción baja de usuario y con ello será dado de baja del sistema.

Alta de aporte:

En la pantalla de inicio, en el menú de la parte superior se elige la opción hacer aporte, el sistema le mostrara una pantalla en la cual en un formulario introducirá el nombre del aporte, su contenido y si se quiere anexar un archivo, se presiona le botón de aportar.

Leer Aporte

En la pantalla de inicio se le muestra todos los aportes existentes del grupo al cual está inscrito, el usuario selecciona el aporte q desee leer.

Cambio de Grupo

En la pantalla de inicio, en el menú de la parte superior se elige la opción baja de grupo.

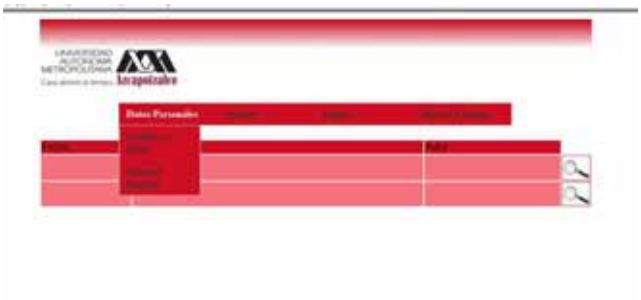
Se le mostrara una lista con todos los grupos existentes.

El usuario selecciona un grupo y con ello se le mostrara el nombre y descripción del grupo, si al usuario presiona el botón validar, se modificara su grupo al que está inscrito, el que tiene será substituido por este que acaba de elegir.

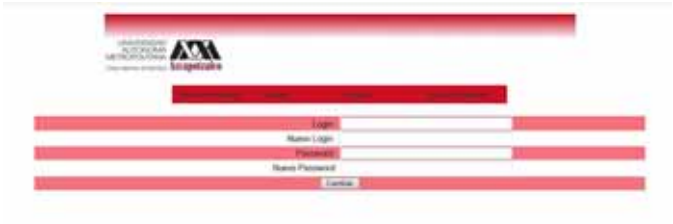
Manual de Administrador:

Cambio de datos de Usuario

En la pantalla de inicio, en el menú de la parte superior se elige la opción de cambio de usuario.



En la pantalla se le mostrara un formulario donde se podrá cambiar el usuario y contraseña, se introducen los nuevos datos.



Baja de Usuario

En el menú de la parte superior se elige la opción baja de usuario

El sistema le mostrara en pantalla una lista con todos los usuarios existentes en el sistema

El administrador elegirá al usuario y presiona sobre el icono de la “X”, con ello el sistema cambiara el estatus del usuario a baja.

Alta de aporte:

En la pantalla de inicio, en el menú de la parte superior se elige la opción hacer aporte, el sistema le mostrara una pantalla en la cual en un formulario introducirá el nombre del aporte, su contenido y si se quiere anexar un archivo, se presiona le botón de aportar.

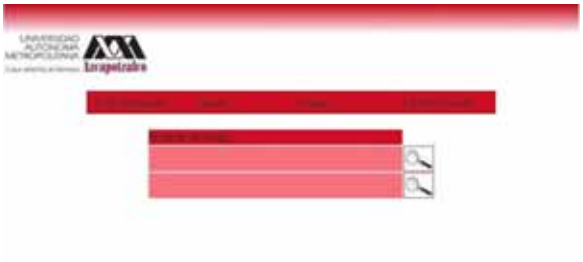
Leer Aporte

En la pantalla de inicio se le muestra todos los aportes existentes del grupo al cual está inscrito, el usuario selecciona el aporte q desee leer.

Baja Aporte

En la pantalla de inicio, en el menú de la parte superior se elige la opción baja de aporte

Se le mostrar un listado con todos los grupos existentes.



El usuario seleccionara un grupo y con ello se le mostrara todos los aportes de ese grupo, ahí elegirá el aporte que quiere dar de baja.

Registrar Grupo

En la pantalla de inicio, en el menú de la parte superior se elige la opción registrar grupo.

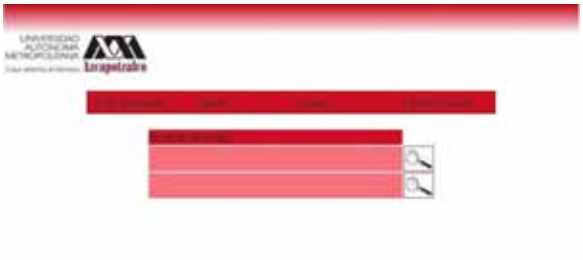
Se le mostrara una pantalla con un formulario en el introducirá el nombre del grupo, la descripción del grupo, las palabras claves del grupo y los parámetros sobre a quién va dirigido ese grupo, estos parámetros son, sexo, estado civil, nivel de estudios,

intereses, carreras técnicas, carreras, maestrías y doctorados de los usuario, se presiona el botón registrar.

Cambio en Grupo

En la pantalla de inicio, en el menú de la parte superior se elige la opción cambio en grupo.

Se le mostrara una pantalla con la lista de grupos existentes.



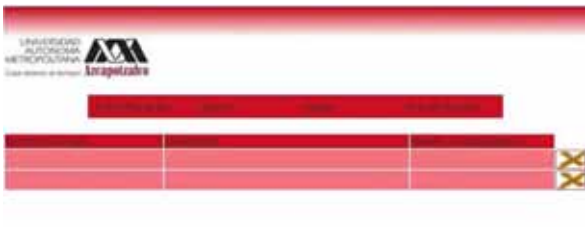
El usuario selecciona el grupo que desee modificar.

Después se le mostrar una pantalla con el mismo formulario que cuando registra un grupo, solo que ahora con los datos actuales del grupo, los que podrá modificar y presionando el botón validar, el sistema verificara que el campo de nombre, descripción y palabras claves no estén vacios, si lo están la acción no se realizara, en caso contrario la acción se realizara

Baja de Grupo

En la pantalla de inicio, en el menú de la parte superior se elige la opción baja de grupo.

Se le mostrara una lista con todos los grupos existentes.

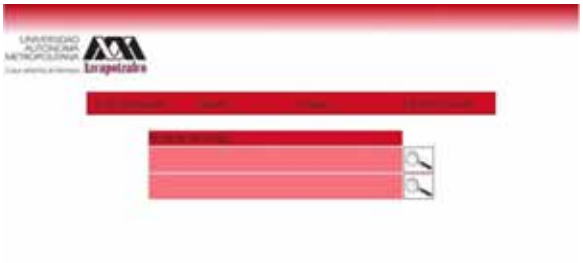


El usuario selecciona un grupo y con ello el estatus del grupo cambiara a baja.

Cambio de Grupo

En la pantalla de inicio, en el menú de la parte superior se elige la opción baja de grupo.

Se le mostrara una lista con todos los grupos existentes.



El usuario selecciona un grupo y con ello se le mostrara el nombre y descripción del grupo, si al usuario presiona el botón validar, se modificara su grupo al que está inscrito, el que tiene será substituido por este que acaba de elegir.

Código del Sistema

```
package ts.struts.javabeans;
import java.sql.Date;
import java.util.ArrayList;

import org.apache.struts.action.ActionForm;

public class ListaAporte extends ActionForm{

	private String titulo;
	private String archivo;
	private String aporte;
	private int idusu;
	private Date fecha;
	private String nombre;
	private ArrayList lista;

	public String getNombre(){
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
```

```
}

public int getIdUsu(){
        return idusu;
}

public void setIdUsu(int idusu) {
        this.idusu = idusu;
}

public Date getFecha() {
        return fecha;
}

public void setFecha(Date fecha) {
        this.fecha =  fecha;
}

public ArrayList getLista(){
   return lista;
     }

public void setLista(ArrayList lista){
   this.lista=lista;
```

```
    }

public String getArchivo() {
        return archivo;
}

public String getAporte() {
        return aporte;
}

public String gettitulo() {
        return titulo;
}

public void setArchivo(String archivo) {
        this.archivo = archivo;
}

public void setAportes(String aporte) {
        this.aporte = aporte;
}

public void settitulo(String titulo) {
        this.titulo = titulo;
```

```
        }
}

package ts.struts.javabeans;
import java.sql.Date;
import java.util.ArrayList;

import org.apache.struts.action.ActionForm;

public class ListaGrupo extends ActionForm{

        private String nombre;
        private int idusu;
        private ArrayList lista;

        public int getIdUsu(){
                return idusu;
        }

        public void setIdUsu(int idusu) {
                this.idusu = idusu;
        }

        public ArrayList getLista(){
```

```
        return lista;
          }

    public void setLista(ArrayList lista){
      this.lista=lista;
          }

    public String getNombre() {
            return nombre;
    }
            public void setNombre(String nombre) {
            this.nombre = nombre;
    }
}

package ts.struts.javabeans;
import java.sql.Date;
import java.util.ArrayList;

import org.apache.struts.action.ActionForm;

public class ListaUsuarios extends ActionForm{

    private String nombre;
```

```
private String apellidoPaterno;
private String apellidoMaterno;
private String email;
private String login;
private String password;
private String password2;
private Date fecha;
private int aportes;
private int idusu;
private ArrayList lista;

public int getIdUsu(){
        return idusu;
}

public void setIdUsu(int idusu) {
        this.idusu = idusu;
}

public ArrayList getLista(){
   return lista;
     }

public void setLista(ArrayList lista){
```

```
   this.lista=lista;
    }

public Date getFecha() {
        return fecha;
}

public int getAportes() {
        return aportes;
}

public String getNombre() {
        return nombre;
}

public String getApellidoPaterno() {
        return apellidoPaterno;
}

public String getApellidoMaterno() {
        return apellidoMaterno;
}

public String getEmail() {
```

```
        return email;
}

public String getLogin() {
        return login;
}

public String getPassword() {
        return password;
}

public String getPassword2() {
        return password2;
}

public void setFecha(Date fecha) {
        this.fecha = fecha;
}

public void setAportes(int aportes) {
        this.aportes = aportes;
}

public void setNombre(String nombre) {
```

```
        this.nombre = nombre;
}

public void setApellidoPaterno(String apellidoPaterno) {
        this.apellidoPaterno = apellidoPaterno;
}

public void setApellidoMaterno(String apellidoMaterno) {
        this.apellidoMaterno = apellidoMaterno;
}

public void setEmail(String email) {
        this.email = email;
}

public void setLogin(String login) {
        this.login = login;
}

public void setPassword(String password) {
        this.password = password;
}

public void setPassword2(String password2) {
```

```
            this.password2 = password2;
        }
}

package ts.struts.javabeans;
import java.util.Date;

import org.apache.struts.action.ActionForm;

public class RegistroAporteForm extends ActionForm{

        private String titulo;
        private String archivo;
        private String aporte;
        private int idusu;
        private Date fecha;
        private String nombre;

        public int getIdusu(){
            return idusu;
        }

        public String getNombre(){
            return nombre;
```

```
}

public void setNombre(String nombre) {
        this.nombre = nombre;
}

public void setIdUsu(int idusu) {
        this.idusu = idusu;
}

public Date getFecha(){
        return fecha;
}

public void setFecha(Date fecha) {
        this.fecha = fecha;
}

public String gettitulo() {
        return titulo;
}

public String getArchivo() {
        return archivo;
```

```
        }

        public String getAporte() {
                return aporte;
        }

        public void settitulo(String titulo) {
                this.titulo = titulo;
        }

        public void setArchivo(String archivo) {
                this.archivo = archivo;
        }

        public void setAporte(String aporte) {
                this.aporte = aporte;
        }
}

package ts.struts.servlets;
import javax.servlet.http.*;
import org.apache.struts.action.*;
import ts.struts.DTO.PersonaDTO;
import ts.struts.javabeans.*;
```

```
import ts.struts.modelo.*;


        public class ValidarAction extends Action{


                public ActionForward execute(ActionMapping
mapping,ActionForm form,HttpServletRequest
request,HttpServletResponse response)

                throws Exception{


                        ValidarUsuario v = new ValidarUsuario();

                        ValidacionUsuarioForm vf =
(ValidacionUsuarioForm)form;

                        int r;

                        r=v.validar(vf);

                        if(r==2)

                                return mapping.findForward("inicio");

                        else

                                return
mapping.findForward("errorValidacion");


                }

}


package ts.struts.servlets;

import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.*;

import ts.struts.javabeans.RegistroUsuarioForm;

import ts.struts.javabeans.ValidacionUsuarioForm;

import ts.struts.modelo.RegistrarUsuario;


        public class RegistrarUsuarioAction extends Action{


                public ActionForward execute(ActionMapping
mapping,ActionForm form,HttpServletRequest
request,HttpServletResponse response)

                throws Exception{


                        ValidacionUsuarioForm vf = new
ValidacionUsuarioForm();

                        RegistroUsuarioForm rf =
(RegistroUsuarioForm)form;

                        RegistrarUsuario registro = new
RegistrarUsuario();




                        if(rf.getLogin().length()!=0 &&
rf.getPassword().length()!=0 && rf.getNombre().length()!=0)

                        {


                                vf=registro.registraUsuario(rf);
```

```
                        if (vf!=null){

                                System.out.println("este mi nuevo
id"+vf.getIduser());



                                request.setAttribute("datos",vf);


                                return
mapping.findForward("cuestionario");
                        }
                        else{
                                return mapping.findForward("error");
                        }
                }else{
                        return mapping.findForward("error");
                }
        }
}

package ts.struts.operacionesBasesDeDatos;

import java.sql.Connection;
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.Iterator;
import java.util.List;
import java.util.StringTokenizer;

import ts.struts.javabeans.*;
import ts.struts.DTO.*;
import ts.struts.clasesPlanas.Persona;

public class OperacionesPersona {

	private ManejoDeBasesdeDatos baseDeDatos;

	public OperacionesPersona(){
		baseDeDatos=new ManejoDeBasesdeDatos();
		}

	public ValidacionUsuarioForm
insertarElemento(RegistroUsuarioForm rf) {
		Statement s;
```

```
            ValidacionUsuarioForm vf = new
ValidacionUsuarioForm();



            try {
                baseDeDatos=new ManejoDeBasesdeDatos();
                Connection con=baseDeDatos.crearConexion();
                s=con.createStatement();


                ResultSet rs = s.executeQuery("SELECT * FROM
usuario where login='"+rf.getLogin()+"'");


                if(!rs.first()){


                    String parametro = construirParametros(rf);
                    s.executeUpdate("INSERT INTO usuario
(apepa,apema,nombre,email,login,pass, fecha_alta, estatus,
id_tipousuario)"+ parametro);


                    ResultSet rs1 = s.executeQuery("SELECT *
FROM usuario where login='"+rf.getLogin()+"' and
pass='"+rf.getPassword()+"'");
                    rs1.beforeFirst();
                    while (rs1.next()) {
```

```
                        vf.setPassword(rs1.getString("pass"));

                        vf.setIdGrupo(rs1.getInt("id_grupo"));
                        vf.setIduser(rs1.getInt("idusu"));

        vf.setTipouser(rs1.getInt("id_tipousuario"));

                    }
                }

                else{
                    vf=null;
                }
                s.close();
                baseDeDatos.cerrarConexion();
                }
            catch (Exception e1) {
                e1.printStackTrace();
                }
            return vf;
            }
        //-------------------------------------------------------------------------
-------------------------------------------------------------------------------
        public int insertarCuestionario(RegistroCuestionarioForm rc) {
            Statement s;
```

```
int band=0;
try {
baseDeDatos=new ManejoDeBasesdeDatos();
Connection con=baseDeDatos.crearConexion();
s=con.createStatement();
String parametro =
construirParametrosCuestionario(rc);
s.executeUpdate("INSERT INTO cuestionario
(doctorado,ec,escolaridad,gusto,leer,maestria,sexo,tec,tec2,id_user)
"+ parametro);

int bandera=4;

ResultSet rs1;
String qry;
String parametro2;

do{

parametro2=construirParametroBusqueda(rc,bandera);
qry="SELECT * FROM grupo where
ec='"+rc.getEc()+"' and escolaridad='"+rc.getEscolaridad()+"' and
gusto='"+rc.getGusto()+"'"+parametro2;
System.out.println(qry);
rs1= s.executeQuery(qry);
if(!rs1.first()){
```

```
                    bandera--;
                }else{
                    bandera=-3;
                }
                if(bandera==0){
                    rc.setSexo(3);
                }
            }
            while (bandera>=-1);

        if(bandera==-3){
            System.out.println("si encontre grupo");


            band = 1;
            int coin;
            int palabras=0;
            int ayuda=0;
            int id_grupo=0;
            String tags;
            String leer;
            rs1.beforeFirst();
            while( rs1.next() ){
```

```
                    StringTokenizer tokens = new
StringTokenizer(rs1.getString("tags"));
                    while(tokens.hasMoreTokens()){
                        tags=tokens.nextToken();
                        StringTokenizer tokens1 = new
StringTokenizer(rc.getLeer());
                        while(tokens1.hasMoreTokens()){
                            leer=tokens1.nextToken();

    System.out.println("tags"+tags);

    System.out.println("leer"+leer);

    coin=tags.compareToIgnoreCase(leer);
                                if(coin==0){
                                    palabras++;
                                }
                        }
                }
                    System.out.println(palabras);
                    if(palabras>ayuda){
                        id_grupo=rs1.getInt("idgrupo");
                        palabras=ayuda;
                    }

                    palabras=0;
```

```
                        }
                        s.executeUpdate("update usuario set
id_grupo='"+id_grupo+"' where idusu='"+rc.getIduser()+"'");
                    }
                    else {
                        s.executeUpdate("update usuario set
id_grupo=0 where idusu='"+rc.getIduser()+"'");
                        band=2;


                    }




                    s.close();
                    baseDeDatos.cerrarConexion();
                    }
                catch (Exception e1) {
                    e1.printStackTrace();
                    }
                return band;
                }
//-------------------------------------------------------------------------------------
-------------------------------------------------------------
        public void insertarGrupo(RegistroGrupoForm rg) {
```

```
		Statement s;
		try {
			baseDeDatos=new ManejoDeBasesdeDatos();
			Connection con=baseDeDatos.crearConexion();
			s=con.createStatement();
			String parametro = construirParametrosGrupo(rg);
			String qry="INSERT INTO grupo
(nombre,descripcion,tags, ct, ct1, estatus, docto, ec, escolaridad,
gusto, lic, maestria, sexo)"+ parametro;
			s.executeUpdate(qry);
			s.close();
			baseDeDatos.cerrarConexion();
			}
		catch (Exception e1) {
			e1.printStackTrace();
			}
		}


	public void insertarAporte(RegistroAporteForm ra, int
id_grupo, int id_usuario) {
		Statement s;
		int aporte=0;
		try {
			baseDeDatos=new ManejoDeBasesdeDatos();
			Connection con=baseDeDatos.crearConexion();
```

```
                    s=con.createStatement();

                    String parametro = construirParametrosAporte(ra,
id_grupo, id_usuario);

                    s.executeUpdate("INSERT INTO aporte
(id_usuario,id_grupo,titulo,adjunto,texto,estatus,fecha_aporte)"+
parametro);

                    ResultSet rs1 = s.executeQuery("SELECT * FROM
usuario where idusu='"+id_usuario+"'");

                    rs1.beforeFirst();

                    while (rs1.next()) {

                        aporte=rs1.getInt("aportes");

                    }

                        aporte++;

                        s.executeUpdate("update usuario set
aportes='"+aporte+"' where idusu='"+id_usuario+"'");


                    s.close();

                    baseDeDatos.cerrarConexion();

                    }
                catch (Exception e1) {

                    e1.printStackTrace();

                    }

                }


        public ArrayList lista() throws Exception {
```

```
RegistroUsuarioForm contacto = null;
ArrayList contactos = new ArrayList();
Statement s = null;
try {
    baseDeDatos=new ManejoDeBasesdeDatos();
    Connection con = baseDeDatos.crearConexion();
    s=con.createStatement();
    ResultSet rs = s.executeQuery("SELECT * FROM
usuario where estatus=1 ORDER BY fecha_alta");

    while( rs.next() ){

        contacto = new RegistroUsuarioForm();

        contacto.setIdUsu(rs.getInt("idusu"));

        contacto.setNombre(rs.getString("nombre"));

        contacto.setFecha(rs.getDate("fecha_alta"));

        contacto.setAportes(rs.getInt("aportes"));

        contactos.add(contacto);
    }
```

```
            }catch(Exception sse){
                        throw sse;
                  }
            finally{
            s.close();
            baseDeDatos.cerrarConexion();
      }


      return contactos;
}

public void borrausuario(String idusua) throws Exception {

      Statement s = null;

            try {
                  baseDeDatos=new ManejoDeBasesdeDatos();
                  Connection con=baseDeDatos.crearConexion();
                  s=con.createStatement();
                  s.executeUpdate("update usuario set estatus=0
where idusu='"+idusua+"'");
                  s.close();
                  baseDeDatos.cerrarConexion();
```

```
                }
            catch (Exception e1) {
                e1.printStackTrace();
                }
}

public void borrausuario2(int idusua) throws Exception {

    Statement s = null;

            try {
                baseDeDatos=new ManejoDeBasesdeDatos();
                Connection con=baseDeDatos.crearConexion();
                s=con.createStatement();
                s.executeUpdate("update usuario set estatus=0
where idusu="'+idusua+"'");
                s.close();
                baseDeDatos.cerrarConexion();
                }
            catch (Exception e1) {
                e1.printStackTrace();
                }
}
```

```
public void actualizardatosusuario(RegistroUsuarioForm rf, int
idusuario) throws Exception {


        Statement s = null;


                try {
                        baseDeDatos=new ManejoDeBasesdeDatos();
                        Connection con=baseDeDatos.crearConexion();
                        s=con.createStatement();
                        s.executeUpdate("update usuario set
login='"+rf.getLogin2()+"', pass='"+rf.getPassword2()+"'  where
idusu='"+idusuario+"'");
                        s.close();
                        baseDeDatos.cerrarConexion();
                        }
                catch (Exception e1) {
                        e1.printStackTrace();
                        }
}


public ArrayList listagrupos() throws Exception {


                RegistroGrupoForm grupo = null;
                ArrayList grupos = new ArrayList();
                Statement s = null;
```

```
try {
        baseDeDatos=new ManejoDeBasesdeDatos();
        Connection con = baseDeDatos.crearConexion();
        s=con.createStatement();
        ResultSet rs = s.executeQuery("SELECT * FROM
grupo where estatus=1 ORDER BY idgrupo");

        while( rs.next() ){

            grupo = new RegistroGrupoForm();

            grupo.setIdusu(rs.getInt("idgrupo"));

            grupo.setNombre(rs.getString("nombre"));



            grupos.add(grupo);
        }
    }catch(Exception sse){
            throw sse;
        }
    finally{
    s.close();
    baseDeDatos.cerrarConexion();
```

```
        }


        return grupos;
}

public void borragrupo(String idgrupo) throws Exception {

        Statement s = null;

                try {
                        baseDeDatos=new ManejoDeBasesdeDatos();
                        Connection con=baseDeDatos.crearConexion();
                        s=con.createStatement();
                        s.executeUpdate("update grupo set estatus=0
where idgrupo='"+idgrupo+"'");
                        s.close();
                        baseDeDatos.cerrarConexion();
                        }
                catch (Exception e1) {
                        e1.printStackTrace();
                        }
}
```

```
public ArrayList listaaportes(int idgrupo) throws Exception {

        RegistroAporteForm aporte = null;
        ArrayList aportes = new ArrayList();
        Statement s = null;
        Statement s1 = null;

        try {
                baseDeDatos=new ManejoDeBasesdeDatos();
                Connection con = baseDeDatos.crearConexion();
                s=con.createStatement();
                s1=con.createStatement();

                String query = ("SELECT * FROM aporte where
estatus=1 and id_grupo = '"+idgrupo+"' ORDER BY fecha_aporte
desc");

                ResultSet rs = s.executeQuery(query);

                while( rs.next() ){

                        aporte = new RegistroAporteForm();

                        aporte.setIdUsu(rs.getInt("id_aporte"));
                        aporte.settitulo(rs.getString("titulo"));
```

```
                    aporte.setFecha(rs.getDate("fecha_aporte"));


                    String query1 = ("SELECT * FROM usuario where
idusu = '"+rs.getInt("id_usuario")+"'");
                    ResultSet rs1 = s1.executeQuery(query1);
                    while( rs1.next() ){
                    aporte.setNombre(rs1.getString("nombre"));
                    }
                    System.out.println(aporte.getNombre());
                    aportes.add(aporte);
            }
        }catch(Exception sse){
                    throw sse;
            }
        finally{
        s.close();
        baseDeDatos.cerrarConexion();
}



return aportes;
}

public ArrayList listaaportes2(String idgrupo) throws Exception {
```

```
        RegistroAporteForm aporte = null;
        ArrayList aportes = new ArrayList();
        Statement s = null;

        try {
                baseDeDatos=new ManejoDeBasesdeDatos();
                Connection con = baseDeDatos.crearConexion();
                s=con.createStatement();

                String query = ("SELECT * FROM aporte where
estatus=1 and id_grupo = '"+idgrupo+"' ORDER BY fecha_aporte
desc");

                ResultSet rs = s.executeQuery(query);

                while( rs.next() ){

                        aporte = new RegistroAporteForm();

                        aporte.setIdUsu(rs.getInt("id_aporte"));
                        aporte.setNombre(rs.getString("titulo"));
                        aporte.setFecha(rs.getDate("fecha_aporte"));
                        aportes.add(aporte);
                }
```

```
        }catch(Exception sse){
                throw sse;
            }
        finally{
        s.close();
        baseDeDatos.cerrarConexion();
}



return aportes;
}

public RegistroAporteForm muestraaporte(String idgrupo) throws
Exception {

        Statement s = null;
        RegistroAporteForm aporte = new RegistroAporteForm();

            try {
                baseDeDatos=new ManejoDeBasesdeDatos();
                Connection con=baseDeDatos.crearConexion();
                s=con.createStatement();

                String query = ("SELECT * FROM aporte where
id_aporte = '"+idgrupo+"'");
```

```
                ResultSet rs = s.executeQuery(query);

                while( rs.next() ){

                        aporte.setNombre(rs.getString("titulo"));
                        aporte.setAporte(rs.getString("texto"));

                }
        }catch(Exception sse){
                        throw sse;
                }
        finally{
        s.close();
        baseDeDatos.cerrarConexion();
    }


    return aporte;
    }

public void borraraporte(String idaporte) throws Exception {

Statement s = null;
```

```
        try {
                baseDeDatos=new ManejoDeBasesdeDatos();
                Connection con=baseDeDatos.crearConexion();
                s=con.createStatement();
                s.executeUpdate("update aporte set estatus=0 where
id_aporte='"+idaporte+"'");
                s.close();
                baseDeDatos.cerrarConexion();
                }
        catch (Exception e1) {
                e1.printStackTrace();
                }
}

public RegistroGrupoForm muestragrupo(String idgrupo) throws
Exception {

        Statement s = null;
        RegistroGrupoForm grupo = new RegistroGrupoForm();

                try {
                        baseDeDatos=new ManejoDeBasesdeDatos();
                        Connection con=baseDeDatos.crearConexion();
                        s=con.createStatement();
```

```
                String query = ("SELECT * FROM grupo where
idgrupo = '"+idgrupo+"'");

                ResultSet rs = s.executeQuery(query);

                while( rs.next() ){
                        //Datos para Mostrar Grupo
                        grupo.setIdusu(rs.getInt("idgrupo"));
                        grupo.setNombre(rs.getString("nombre"));

        grupo.setDescripcion(rs.getString("descripcion"));
                        grupo.setTags(rs.getString("tags"));
                        grupo.setCt(rs.getInt("ct"));
                        grupo.setCt1(rs.getInt("ct1"));
                        grupo.setDocto(rs.getInt("docto"));
                        grupo.setEc(rs.getInt("ec"));

        grupo.setEscolaridad(rs.getInt("escolaridad"));
                        grupo.setGusto(rs.getInt("gusto"));
                        grupo.setLic(rs.getInt("lic"));
                        grupo.setMaestria(rs.getInt("maestria"));
                        grupo.setSexo(rs.getInt("sexo"));
```

```
                }
            }catch(Exception sse){
                    throw sse;
                }
            finally{
            s.close();
            baseDeDatos.cerrarConexion();
        }



        return grupo;
        }

public void actualizardatosgrupo(RegistroGrupoForm rf) throws
Exception {

        Statement s = null;


            try {
                baseDeDatos=new ManejoDeBasesdeDatos();
                Connection con=baseDeDatos.crearConexion();
                s=con.createStatement();
                String query="update grupo set
nombre='"+rf.getNombre()+"', descripcion='"+rf.getDescripcion()+"',
tags='"+rf.getTags()+"'  where idgrupo='"+rf.getIdusu()+"'";
```

```
                System.out.println(query);
                s.executeUpdate(query);
                s.close();
                baseDeDatos.cerrarConexion();
                }
            catch (Exception e1) {
                e1.printStackTrace();
                }

}

public void actualizargrupo(RegistroGrupoForm rf, int idusu) throws
Exception {

    Statement s = null;

        try {
                baseDeDatos=new ManejoDeBasesdeDatos();
                Connection con=baseDeDatos.crearConexion();
                s=con.createStatement();
                String query="update usuario set
id_grupo='"+rf.getIdusu()+"' where idusu='"+idusu+"'";
                System.out.println(query);
                s.executeUpdate(query);
                s.close();
```

```
                baseDeDatos.cerrarConexion();
                }
            catch (Exception e1) {
                e1.printStackTrace();
                }

}
    public ValidacionUsuarioForm
validausu(ValidacionUsuarioForm vf) {
            Statement s;
            Connection con=baseDeDatos.crearConexion();

            System.out.println(vf.getUsuario());

                try {

                    s=con.createStatement();

                    String qry="SELECT pass, id_grupo,
id_tipousuario, idusu FROM usuario WHERE login='" +
vf.getUsuario() +"'";

                    ResultSet rs = s.executeQuery(qry);
```

```
                    if(!rs.first()){

                        vf = null;

                    }

                    else{
                        rs.beforeFirst();
                            while (rs.next()) {

vf.setPassword(rs.getString("pass"));

vf.setIdGrupo(rs.getInt("id_grupo"));

vf.setIduser(rs.getInt("idusu"));

vf.setTipouser(rs.getInt("id_tipousuario"));
                                    }
                    }
                    s.close();
                baseDeDatos.cerrarConexion();
            return vf;
            }
            catch (Exception e) {
```

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

    public String construirParametrosAporte(RegistroAporteForm
p, int id_grupo, int id_usuario){
            Calendar c = Calendar.getInstance();
            int dia = c.get(Calendar.DATE);
            int mes = c.get(Calendar.MONTH);
            mes++;
            int annio = c.get(Calendar.YEAR);
            String fecha=annio+"-"+mes+"-"+dia;
            int estatus=1;
            String parametro="";
            parametro = parametro.concat(" VALUES
("+"'"+id_usuario+"',"+"'"+id_grupo+"',"+"'"+p.getNombre()+"',"+"'"+p.
getArchivo()+"',"+"'"+p.getAporte()+"',"+"'"+estatus+"',"+"'"+fecha+"')"
);
            return parametro;
            }

    public String construirParametrosGrupo(RegistroGrupoForm
p){
            int estado=1;
            String parametro="";
```

```
            parametro = parametro.concat(" VALUES
("+"'"+p.getNombre()+"',"+"'"+p.getDescripcion()+"',"+"'"+p.getTags()
+"', '"+p.getCt()+"', '"+p.getCt1()+"', '"+estado+"', '"+p.getDocto()+"',
'"+p.getEc()+"', '"+p.getEscolaridad()+"', '"+p.getGusto()+"',
'"+p.getLic()+"', '"+p.getMaestria()+"', '"+p.getSexo()+"')");

            return parametro;

            }


        public String
construirParametrosCuestionario(RegistroCuestionarioForm p){

            String parametro="";

            parametro = parametro.concat(" VALUES
("+"'"+p.getDoctorado()+"',"+"'"+p.getEc()+"',"+"'"+p.getEscolaridad()
+"',"+"'"+p.getGusto()+"',"+"'"+p.getLeer()+"',"+"'"+p.getMaestria()+"',
"+"'"+p.getSexo()+"',"+"'"+p.getTec()+"',"+"'"+p.getTec2()+"',
'"+p.getIduser()+"')");

            return parametro;

            }


        public String construirParametros(RegistroUsuarioForm p){

            Calendar c = Calendar.getInstance();

            int dia = c.get(Calendar.DATE);

            int mes = c.get(Calendar.MONTH);

            mes++;

            int annio = c.get(Calendar.YEAR);

            String fecha=annio+"-"+mes+"-"+dia;

            int estatus=1;
```

```
            int tipo_usu=2;
            String parametro="";
            parametro = parametro.concat(" VALUES
("+"'"+p.getApellidoPaterno()+"',"+"'"+p.getApellidoMaterno()+"',"+"'"
+p.getNombre()+"',"+"'"+p.getEmail()+"',"+"'"+p.getLogin()+"',"+"'"+p.
getPassword()+"', '"+fecha+"', '"+estatus+"', '"+tipo_usu+"')");
            return parametro;
            }
//--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

            public String
construirParametroBusqueda(RegistroCuestionarioForm p, int i){
                  String parametro=" ";

                  int j;
                  int [] param;
                  param= new int [6];

                  String [] param1;
                  param1= new String [6];

                  param[1]=p.getSexo();
                  param[2]=p.getTec2();
                  param[3]=p.getCarrera();
                  param[4]=p.getMaestria();
```

```
                param[5]=p.getDoctorado();

                param1[1]="and sexo='";
                param1[2]="and ct1='";
                param1[3]="and lic='";
                param1[4]="and maestria='";
                param1[5]="and docto='";

                if(i==0){

        parametro=parametro+param1[1]+param[1]+"'";
                }else{

                for (j=i; j>=1; j--){

                        if(param[j]!=0)

        parametro=parametro+param1[j]+param[j]+"'";

                }
            }
            return parametro;
}

}
```