Universidad Autónoma Metropolitana

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Reporte Final del Proyecto de Integración

Licenciatura en Ingeniería en Computación

Administración de listas de contactos de correo electrónico mediante servicios web

Edgar Garcia Velasco

205301322

Dra. Beatriz Adriana González Beltrán

Profesor Asociado

Departamento de Sistemas

Trimestre 2016 Otoño

14 de diciembre de 2016

Yo, **Dra. Beatriz Adriana González Beltrán**, *declaro que aprobé el contenido del presente Reporte de Proyecto de Integración* y *doy mi autorización* para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Dra. Beatriz Adriana González Beltrán

Asesor

Yo, **Edgar Garcia Velasco**, *doy mi autorización* a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Edgar Garcia Velasco

Alumno

Índice

Resumen

En el presente documento se describe de manera completa la forma en que se realizó el Sistema de Administración de listas de correo electrónico.

Este proyecto se debe a que existe la aplicación Majordomo para la gestión de listas de correo electrónico; sin embargo dicho software requiere de ciertos requisitos para poder instalarlo. Con el sistema desarrollado como resultado de esta propuesta se quitaron limitaciones que presentaba Majordomo, dado que se implementó con servicios web; además tiene la posibilidad de adaptar dichas funcionalidades a otros sistemas.

Los objetivos del desarrollo de estos servicios web, son los de poder crear diferentes tipos de listas, además de poder realizar las funcionalidades básicas que tienen otros programas que gestionan listas de correo electrónico. Además se implementó una aplicación cliente, la cual muestra con interfaces sencillas las funcioalidades necesarias para la gestión de listas de correo electrónico.

1. Introducción

Una lista de contactos de correo electrónico permite la distribución de mensajes a múltiples usuarios de forma simultánea. Si la lista se encuentra definida dentro de una aplicación cliente de correo electrónico (ej. Thunderbird) solo el usuario de la aplicación conoce los miembros de la lista. Existe otra manera de administrar las listas de contactos de correo electrónico donde además de permitir enviar un mensaje a una lista, éste llegará a la dirección de todas las personas inscritas en ella.

El objetivo de este proyecto es implementar la administración de listas de contactos de correo electrónico con servicios web. De esta manera las aplicaciones cliente podrán invocar estos servicios para obtener sus funcionalidades y así tener una organización de un conjunto de listas.

En las siguientes secciones se detalla como fue realizado el diseño de cada uno de los servicios web, así como la implementación de cada una de las funciones, además se muestran las interfaces de la aplicación cliente la cual ejemplifica de manera clara el potencial de las funciones realizadas por los servicios web.

2. Antecedentes

2.1 Referencias internas

1.- Gestión de calificaciones de cursos mediante servicios web

Este proyecto terminal tiene como objetivo el desarrollo de servicios web que permiten a un profesor llevar el control adecuado de las calificaciones de sus alumnos en los diferentes cursos que imparte **[1]**. La principal relación encontrada en este proyecto es la implementación de servicios web. Sin embargo, en este proyecto se implementaron servicios web para gestionar las calificaciones de cursos de un profesor.

2.-Extracción automatizada y representación de servicios web mediante ontologías

En este proyecto se diseñó e implementó un sistema que permite la extracción de la información más relevante de múltiples archivos de descripción de servicios web, información que está representada mediante un modelo ontológico **[2]**. Una de las relaciones principales encontradas es el uso de archivos de descripción de servicios web. Sin embargo, estos archivos se utilizan para obtener una representación de servicios y su propósito no es utilizar un servicio web.

3.- Solución de problemas de mediana dificultad utilizando composición de servicios web

En este proyecto se diseñó e implementó una aplicación para que una agencia de viajes pudiera comunicarse, mediante protocolos interoperables a través de Internet, con las diferentes aplicaciones de compañías de hoteles y vuelos así como con los bancos para realizar los pagos **[3]**. La relación existente con la propuesta planteada es el uso de servicios web. Sin embargo,

en este proyecto se diseñaron e implementaron módulos específicos para integrar los diferentes servicios web.

2.2 Referencias externas

1.- Majordomo

Majordomo es un programa encargado de gestionar las listas de contactos de correo electrónico; además, recibe mensajes para ser distribuidos entre los suscriptores de la lista y se utilizan comandos para realizar alguna acción sobre las listas [4]. La principal relación encontrada con mi proyecto es la administración de listas de distribución, pero este programa no utiliza ningún servicio web para su funcionamiento.

2.- LISTSERV

Es un gestor de listas de contactos de correo electrónico que permite gestionar y distribuir mensajes a los miembros de las listas [5]. La similitud con mi proyecto es la administración de listas de distribución pero su arquitectura no se basa en servicios web.

3.- Yahoo! Groups

Es un servicio gratuito que ofrece varias herramientas, tales como: discusiones, listas, archivo de mensajes, salas de pláticas, álbumes de fotos y la posibilidad de compartir archivos. Estas herramientas permiten publicar o desplegar información de cada uno de los miembros del grupo [6]. La semejanza primordial con mi proyecto es el manejo de listas. Sin embargo, Yahoo! Groups maneja una variedad de funciones que no están relacionadas con la administración de listas.

3. Justificación

Actualmente, la administración de listas de correo se puede realizar mediante la aplicación **Majordomo**. Este programa se encarga de gestionar las listas de correo, recibiendo mensajes para ser distribuidos entre los suscriptores de la lista y utiliza comandos para realizar alguna acción sobre las listas (hacer suscripciones, obtener información, etc.). Sin embargo, todas las acciones para administrar a una lista requieren ir dentro del cuerpo de un mensaje de correo electrónico, lo que no es amigable para el usuario.

Otra aplicación de gestión de listas es LISTSERV. Sin embargo, es comercial y no está construida como un servicio web, lo que dificulta su integración en un cliente de correo electrónico.

Este proyecto pretende implementar las funcionalidades que realiza la aplicación **Majordomo**, pero con las siguientes variantes:

- No se necesita asignar una cuenta de correo electrónico a la aplicación para la administración de las listas.

- El administrador no necesita tener un acceso directo al sistema donde está instalada la aplicación.
- La administración de las listas de correo electrónico, se implementarán con servicios web, ya que esta tecnología utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. De esta manera, las aplicaciones cliente podrán compartir las listas de manera independiente del lugar donde se encuentren guardadas.

La relevancia de este proyecto es crear servicios web para que puedan ser adaptados en aplicaciones cliente para facilitar la administración de información mediante las listas correo electrónico.

4. Objetivos

4.1 Objetivo general

- Diseñar e implementar servicios web que permitan la administración de listas de contactos de correo electrónico.

4.2 Objetivos específicos

- Diseñar, implementar y probar un servicio web que permita la creación, modificación y eliminación de diferentes tipos de listas.
- Diseñar, implementar y probar un servicio web que permita la gestión de listas.
- Diseñar, implementar y probar un servicio web que realice las funciones para gestionar contactos de correo electrónico.
- Diseñar, implementar y probar el servicio web que guarde los mensajes de las listas.

5 Marco Teórico

5.1 Listas de correo electrónico

Una lista de correo electrónico permite la distribución de mensajes entre contactos a través de un nombre de lista. Al enviar un mensaje a una lista, este llegará a la dirección de todas las personas inscritas en ella **[4]**.

Las operaciones que se pueden realizar en una lista son las siguientes:

- **Subscribir a una lista.** Esta función permite dar de alta a un usuario en una lista específica.
- **Anular suscripción a una lista.** Permite anular la suscripción de un usuario a una lista específica.
- **Listas que está suscrito un usuario.** Muestra las listas a las que está inscrito un usuario.

- **Enviar mensaje a una lista.** Envía un mensaje a una lista específica.
- **Usuarios registrados en una lista.** Muestra los usuarios que participan en una lista dada.
- **Descargar mensaje de una lista.** Permite descargar un mensaje relacionado con una lista determinada.
- **Mostrar mensajes de una lista.** Muestra los mensajes que contiene una determinada lista.

Además, una lista puede crearse con diferentes propiedades o políticas:

Política de suscripción. Determina quién puede suscribirse a la lista. De esta manera, las listas pueden ser abiertas o cerradas.

- **Listas abiertas.** Una inscripción o cancelación a una lista abierta puede ser aprobada automáticamente.
- **Listas cerradas.** Todos los requerimientos de inscripción o cancelación a una lista cerrada serán enviados al propietario de la lista para que los apruebe.

Política de distribución de mensajes. Determina como van a ser administrados y distribuidos los mensajes que se dirijan a una lista.

- **Listas moderadas.** Todos los mensajes que se envían a la lista son revisados por el propietario de la misma y es quien decide si se distribuyen o no los mensajes.
- **Listas no moderadas.** Todos los mensajes son mandados a las personas suscritas sin necesidad de ser aprobados.

Política de privacidad. Permite definir si un usuario puede obtener información de una lista.

- **Listas públicas.** Los suscriptores tienen acceso a la información de una lista.
- **Listas privadas.** En una lista privada los suscriptores no tienen acceso a la información de la lista.

Una lista puede tener una combinación de propiedades de cada una de las políticas; sin embargo, cada propiedad de una misma política es mutuamente excluyente. Por ejemplo, una lista puede ser cerrada, moderada y pública. Sin embargo, una lista no puede ser abierta y cerrada.

6. Desarrollo del Proyecto

En esta sección se aborda todo el proceso necesario para construir los servicios web, los cuales tienen la función de administrar listas de correo electrónico. La sección 6.1 Diseño del sistema contiene los artefactos que fueron utilizados para modelar el comportamiento de los servicios . La sección 6.2 Implementación describe las herramientas utilizadas para desarrollar el consumidor de los servicios web, así como los pasos que se siguieron para programar el cliente.

6.1 Diseño del sistema

En esta sección se presentan los artefactos de diseño que se utilizaron para modelar el comportamiento de los servicios web que administran las listas de correo electrónico. Dichos artefactos son:

- ✓ Diagramas de casos de uso general
- ✓ Diagrama de clases
- ✓ Diagrama de la base de datos
- ✓ Diagrama de la arquitectura del sistema

6.1.1 Diagramas de caso de uso general

Se determinó que para el correcto funcionamiento del sistema, existen tres actores:

1. **Administrador o Dueño:**

   - ✓ Es quien crea la lista.
   - ✓ Establece el moderador de la lista.
   - ✓ Define de qué forma se utilizara la lista.
   - ✓ Gestiona las suscripciones y de suscripciones
   - ✓ Controla todas las funcionalidades de la lista.
   - ✓ Puede crear, eliminar y modificar un moderador y usuario de la lista.

2. **Moderador:**

   - ✓ El moderador lo nombra el dueño de la lista.
   - ✓ El generalmente se encarga de controlar los mensajes enviados a su lista: después leerlos, elige si lo reenvía o no dependiendo de la característica de la lista.
   - ✓ Controla la suscripción y de suscripción de la lista.
   - ✓ Gestiona las funciones principales de cada lista.
   - ✓ Puede crear, eliminar y modificar el perfil de usuario de la lista.

3. **Usuario (o Suscriptor):**
    - ✓ Puede suscribirse y de suscribirse a una lista.
    - ✓ Tiene los permisos de usar todas las funciones de una lista.

Dado que existen muchos casos de uso para cada actor, se decidió dividir en diferentes diagramas los diferentes casos de uso de acuerdo al actor.

Observe que la Figura 1 muestra los casos de uso del actor Administrador relacionados con la gestión de los roles de los usuarios dentro del sistema de administración de listas.

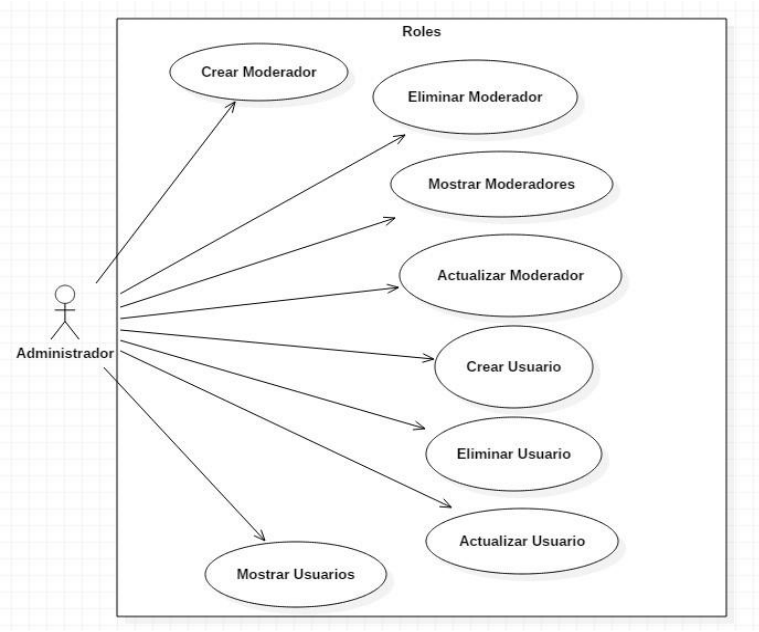

Figura 1: Muestra el diagrama de casos de uso del actor Administrador donde gestiona los roles de cada usuario.

Por otra parte, la Figura 2 muestra el diagrama del actor Administrador y cuyos casos de uso se encuentran relacionados con la gestión de listas.

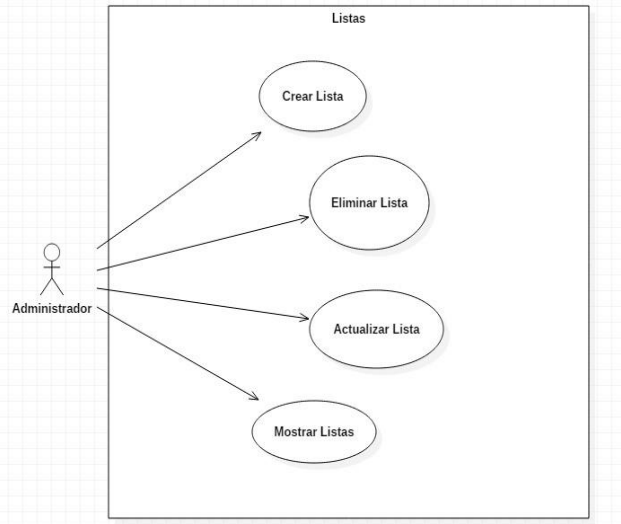

Figura 2: Muestra el diagrama de casos de uso del actor Administrador donde gestiona las listas.

La figura 3 muestra el diagrama del actor Administrador y cuyos casos de uso se encuentran relacionados a las funcionalidades de una lista.

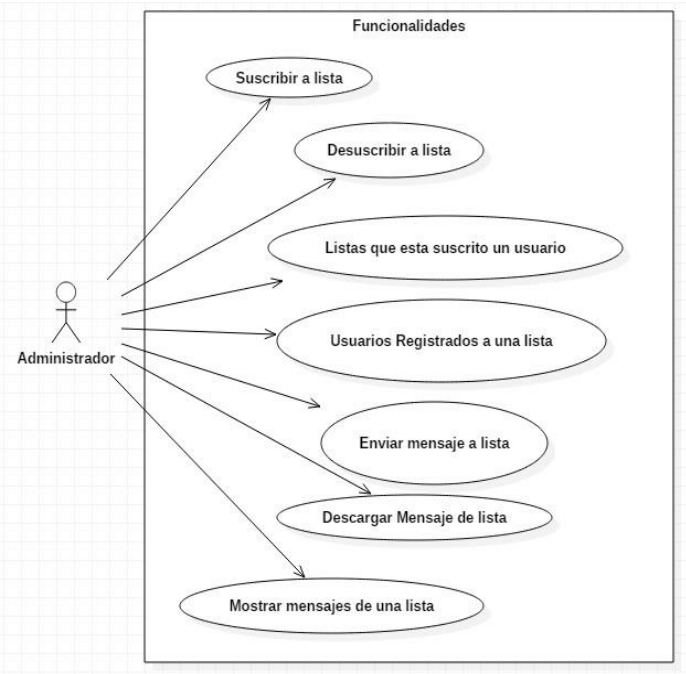

Figura 3: Muestra los principales casos de uso del actor Administrador, donde controla las funcionalidades de cada lista.

Observe que la Figura 4 muestra el diagrama del actor Moderador y cuyo casos de uso están relacionados con la gestión de los roles de los usuarios dentro del sistema de administración de listas.

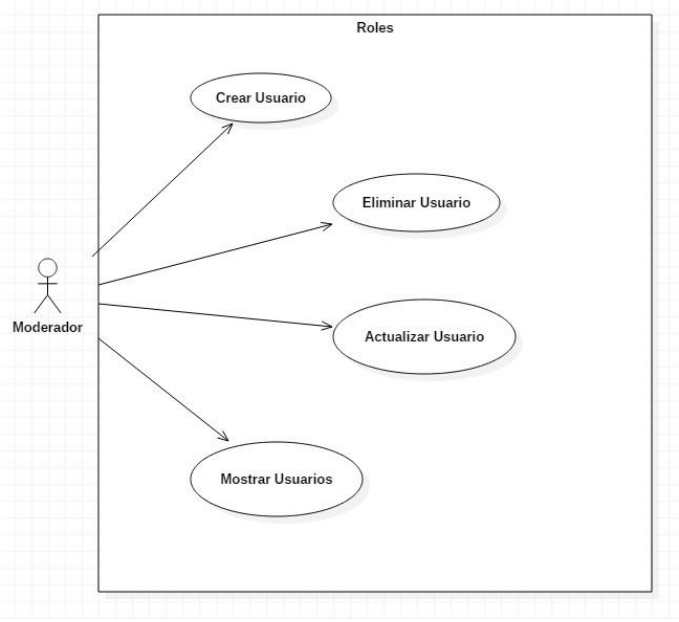

Figura 4: Muestra los principales casos de uso del actor Moderador, donde controla la creación, modificación y eliminación de un usuario.

Por otra parte, la Figura 5 muestra el diagrama del actor Moderador y cuyos casos de uso se encuentran relacionados con la gestión de listas.

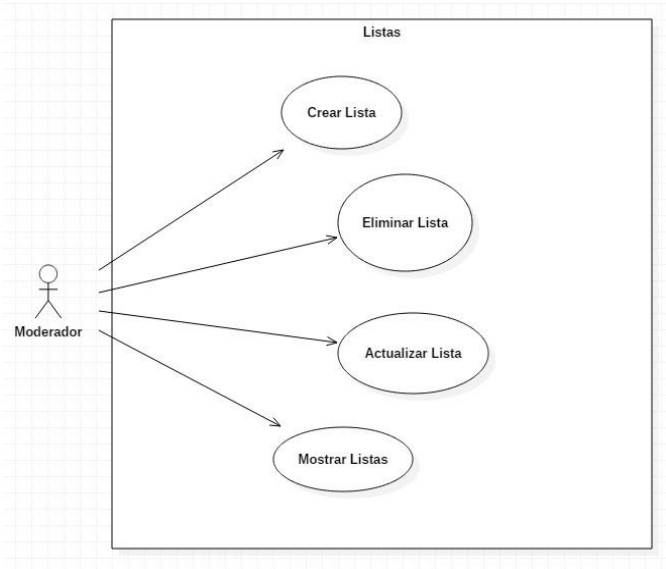

Figura 5: Muestra los principales casos de uso del actor Moderador, donde controla la creación, modificación y eliminación de las listas.

La figura 6 muestra el diagrama del actor Moderador y cuyos casos de uso se encuentran relacionados a las funcionalidades de una lista.

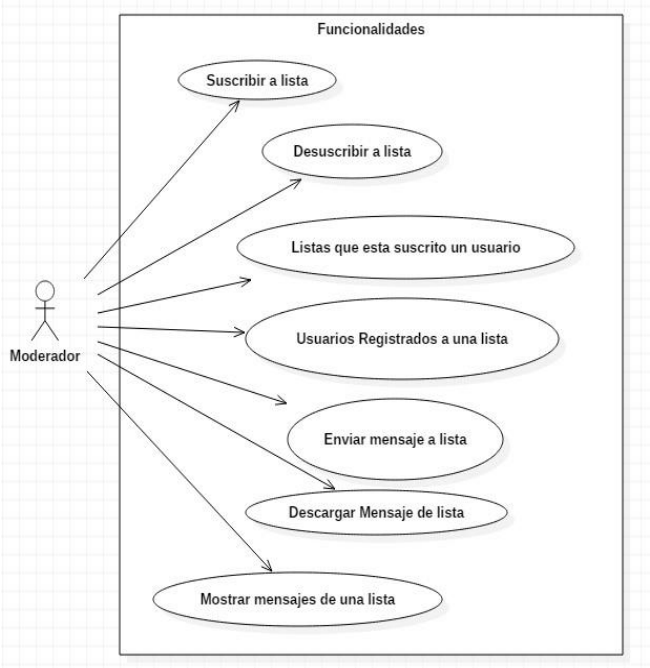

Figura 6: Muestra los principales casos de uso del actor Moderador, donde controla las funcionalidades de cada lista.

La figura 7 muestra el diagrama del actor Moderador y cuyos casos de uso se encuentran relacionados a los diferentes tipos de una lista.

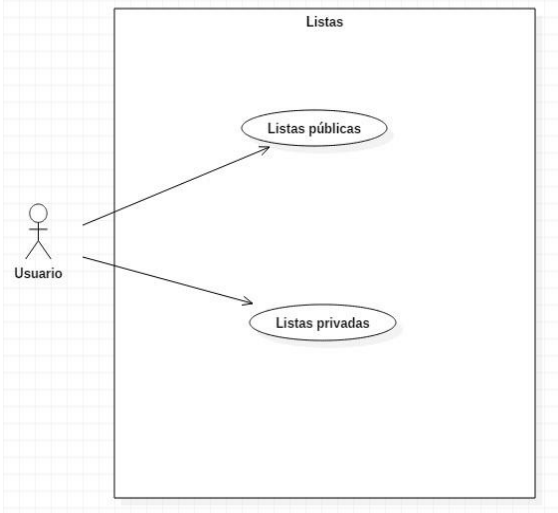

Figura 7: Muestra los principales casos de uso del actor Usuario, donde puede ver que listas están disponibles.

La figura 8 muestra el diagrama del actor Usuario o Suscriptor y cuyos casos de uso se encuentran relacionados a las funcionalidades de una lista.

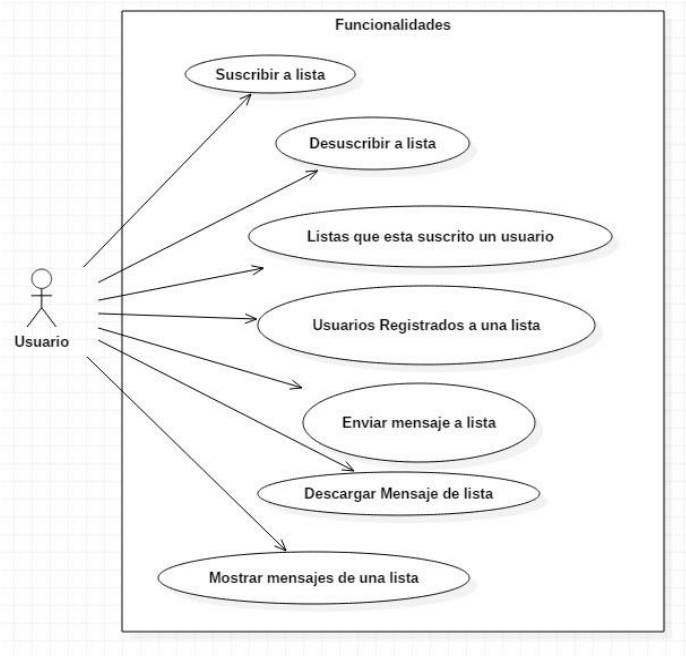

Figura 8: Muestra los principales casos de uso del actor Usuario, donde controla las funcionalidades de cada lista.

Documentación de los actores.

Actor	Administrador
Casos	• Crear Lista • Modificar Lista • Eliminar Lista • Crear Moderador • Modificar Moderador • Eliminar Moderador • Crear Usuario • Modificar Usuario • Eliminar Usuario • Suscribirse a una lista • Anular suscripción de una lista • Listas que esta suscrito un usuario • Enviar mensaje a una lista • Usuarios registrados en una lista • Mostrar mensajes de una lista
Tipo	Primario
Descripción	Las listas pueden contienen tres tipos de políticas (suscripción, distribución de mensajes y privacidad), las cuales determinan las diferentes propiedades de cada lista.

Tabla 1: Muestra los casos de uso del rol Administrador.

Actor	Moderador
Casos	• Crear Usuario • Modificar Usuario • Eliminar Usuario • Suscribirse a una lista • Anular suscripción de una lista • Listas que esta suscrito un usuario • Enviar mensaje a una lista • Usuarios registrados en una lista • Mostrar mensajes de una lista
Tipo	Secundario
Descripción	El moderador de la lista, podrá modificar los parámetros de cada usuario que este suscrito a una lista.

Tabla 2: Muestra los casos de uso del rol Moderador.

Actor	Usuario
Casos	• Listas Públicas • Listas Privadas • Suscribirse a una lista • Listas que está suscrito un usuario • Anular suscripción de una lista • Enviar mensaje a una lista • Mostrar mensajes de una lista
Tipo	Secundario
Descripción	El usuario tendrá acceso a la mayoría de las funcionalidades de una lista así como la función de crear su perfil.

Tabla 3: Muestra los casos de uso del rol Usuario.

Casos de uso para la gestión de listas

Las Tabla 4, 5 y 6 muestran las funcionalidades de los casos de uso "Suscribir a una lista", "Anular suscripción a una lista" y "Listas a las que está suscrito un usuario", respectivamente. Observe que los actores Administrador, Moderador y Usuario deben tener acceso a estos casos de uso.

Caso de uso	Suscribir a una lista
Actores	• Administrador • Moderador • Usuario
Tipo	Inclusión
Resumen	Esta función permitirá dar de alta a un usuario en una lista específica.
Parámetros de entrada	Nombre del usuario a suscribir y de la lista a la que se quiere suscribir.
Parámetros de retorno	Usuario registrado a la lista especificada.
Nombre	Suscribir_A_Lista
URL	http://localhost:8080/subscribe/

Tabla 4: Muestra las funcionalidades del caso de uso "Suscribir a una lista".

Caso de uso	Anular suscripción a una lista.
Actores	• Administrador • Moderador • Usuario
Tipo	Inclusión
Resumen	Permite anular la suscripción de un usuario a una lista específica.
Parámetros de entrada	Nombre de lista y usuario al que desea anular la suscripción.
Parámetros de retorno	Anulación de suscripción de la lista.
Nombre	Desuscribir_de_lista
URL	http://localhost:8080/Desuscribir_de_lista/

Tabla 5: Muestra las funcionalidades del caso de uso “Anular suscripción a una lista”.

Caso de uso	Listas que esta suscrito un usuario.
Actores	• Administrador • Moderador • Usuario
Tipo	Inclusión
Resumen	Muestra en qué listas está inscrito un usuario.
Parámetros de entrada	Nombre del usuario.
Parámetros de retorno	Nombre de las listas a la que está suscrito el usuario especificado.
Nombre	Listas_suscrito_usuario
URL	http://localhost:8080/Listas_suscrito_usuario/

Tabla 6: Muestra las funcionalidades del caso de uso “Listas que está suscrito un usuario”.

Las Tabla 7 al 10 muestran las funcionalidades de los casos de uso “Enviar mensaje a una lista”, “Usuarios registrados a una lista”, “Mostrar mensajes de una lista” y “Descargar mensajes de una lista”, respectivamente. Observe que los actores Administrador, Moderador y Usuario deben tener acceso a estos casos de uso.

Caso de uso	Enviar mensaje a una lista.
Actores	• Administrador • Moderador • Usuario
Tipo	Inclusión
Resumen	Envía un mensaje a una lista específica.
Parámetros de entrada	Nombre de la lista y mensaje a enviar.
Parámetros de retorno	Mensaje enviado a la lista.
Nombre	Enviar_mensaje_a_lista
URL	http://localhost:8080/Enviar_mensaje_a_lista/

Tabla 7: Muestra las funcionalidades del caso de uso “Enviar mensaje a una lista”.

Caso de uso	Usuarios registrados a una lista.
Actores	• Administrador • Moderador • Usuario (si la lista es moderada)
Tipo	Inclusión
Resumen	Muestra los usuarios que participan en una lista específica.
Parámetros de entrada	Nombre de la lista.
Parámetros de retorno	Nombre de usuarios registrados en la lista.
Nombre	Usuarios_registrados_en_lista
URL	http://localhost:8080/Usuarios_registrados_en_lista/

Tabla 8: Muestra las funcionalidades del caso de uso “Usuarios registrados a una lista”.

Caso de uso	Mostrar mensajes de una lista.
Actores	• Administrador • Moderador • Usuario
Tipo	Inclusión
Resumen	Muestra los mensajes que contiene una determinada lista.
Parámetros de entrada	Nombre de la lista.
Parámetros de retorno	Muestra los nombres de los mensajes contenidos en la lista.
Nombre	Mostrar_mensajes_de_lista
URL	http://localhost:8080/Mostrar_mensajes_de_lista/

Tabla 9: Muestra las funcionalidades del caso de uso “Mostrar mensajes de una lista”.

Caso de uso	Descargar mensajes de una lista.
Actores	• Administrador • Moderador • Usuario
Tipo	Inclusión
Resumen	Descarga el contenido de cualquier mensaje, si este tiene archivo adjunto.
Parámetros de entrada	Nombre de la lista, asunto del mensaje y ruta destino
Parámetros de retorno	Descarga archivo en ruta destino.
Nombre	Descargar_mensaje_de_lista
URL	http://localhost:8080/Descargar_mensaje_de_lista/

Tabla 10: Muestra las funcionalidades del caso de uso “Descargar mensaje de una lista”.

6.1.2 Diagrama de clases

Para obtener las entidades clave del sistema administrador de listas de correo electrónico, se analizaron las funciones que tiene la aplicación majordomo y LISTSERV. Para que se cumplieran las funciones más importantes y así poder administrar las listas.

De esta manera, quedaron solo cinco entidades clave:

- ✓ Listas
- ✓ Moderadores
- ✓ Usuarios
- ✓ Suscripción
- ✓ Mensajes de listas

Entidad Listas

La entidad **`Listas`** se refiere a las listas que van a estar disponibles en el sistema para que el usuario se pueda registrar a ellas. Los atributos que tiene son: idListas, Nombre_Lista, Descripción, Suscripción, Distribución, Privacidad y Moderador. Se determinó una relación con las entidades **`Moderadores`**, **`Suscripción y Mensajes de listas`**.

Entidad Moderadores

La entidad **`Moderadores`** se refiere al propietario de la lista, este va ser asignado por el Administrador el cual es el único que puede crear, eliminar y modificar una lista. Los atributos que tiene son: idModeradores, nombre_mod, email y Login. Se determinó una relación con la entidad **`Listas`**.

Entidad Suscripción

La entidad **`Suscripción`** corresponde a la relación de un usuario suscrito a una lista o listas determinadas. Los atributos que tiene son: idSuscripción, Nom_Lista y Nom_Usuario. Se determinó una relación con las entidades **`Listas y Usuarios`**.

Entidad Usuarios

La entidad **`Usuarios`** se refiere a los miembros de cada lista. Los atributos que tiene son: idUsuarios, Nombre_Usuario, email_usuario y password. Se determinó una relación con la entidad **`Suscripción`**.

Entidad `Mensajes de Listas`

La entidad **`Mensajes de Listas,`** corresponde a los mensajes que se envían a cada lista, donde se van almacenar los datos principales de cada mensaje, así como un campo dedicado a los archivos adjuntos, que se puedan o no enviar en el mismo mensaje. Los atributos que tiene son: idMensajes_List, Asunto, Mensaje, NombreArchivo, Contenido y listas_Nombre_Lista1. Se determinó una relación con la entidad **`Listas`**.

El diagrama de clases se muestra en la Figura 9. Observe que la entidad **`Moderadores`**, puede ser propietario de una o n listas. Una **`Lista`** puede tener a varios miembros suscritos a ella y a su vez puede tener varios **`Mensajes`** enviados por uno o distintos **`Usuarios`**. Por último, un **`Usuario`** puede tener una suscripción a distintas listas de manera simultánea.

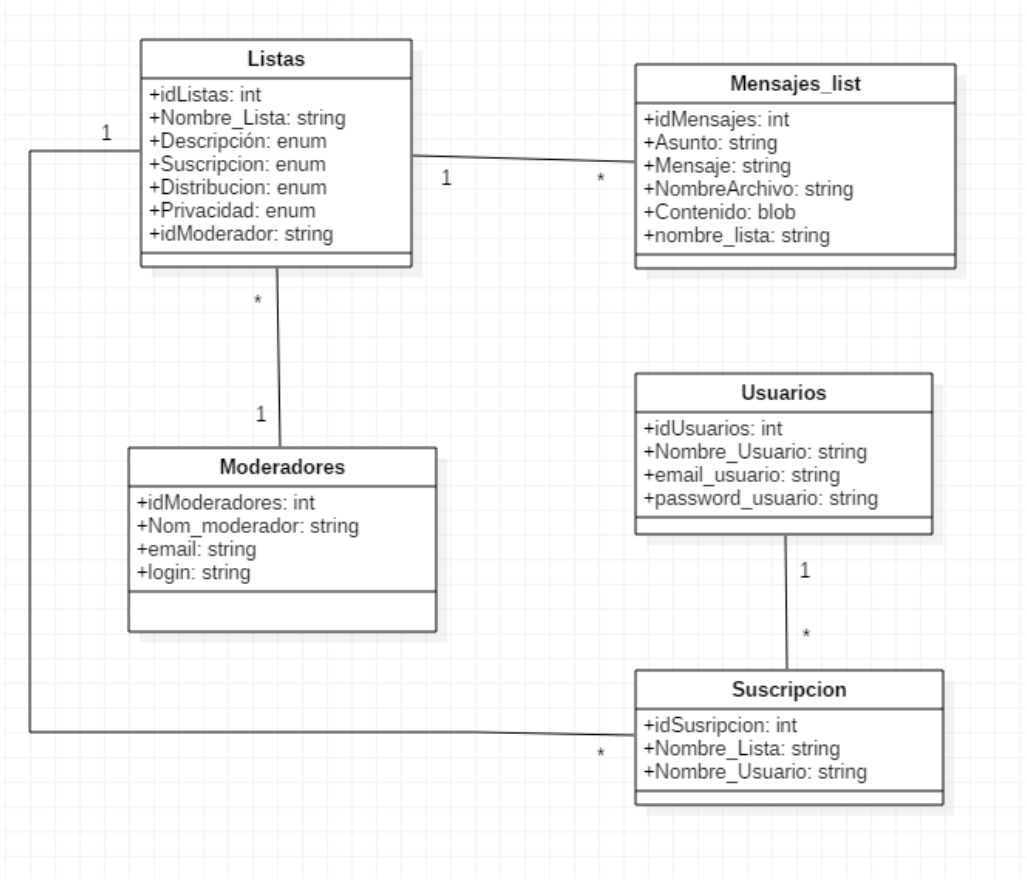

Figura 9. Diagrama de clases

6.1.3 Estructura de la base de datos

La base de datos del sistema de Administración de correo electrónico es de tipo relacional, tomando como referencia el diagrama de clases. En primer lugar, se transformó cada entidad del modelo de clases en una tabla; después, se definió una llave principal para cada tabla; por último se realizó la normalización de la base de datos. En la figura 10 se muestra el diagrama de la base de datos.

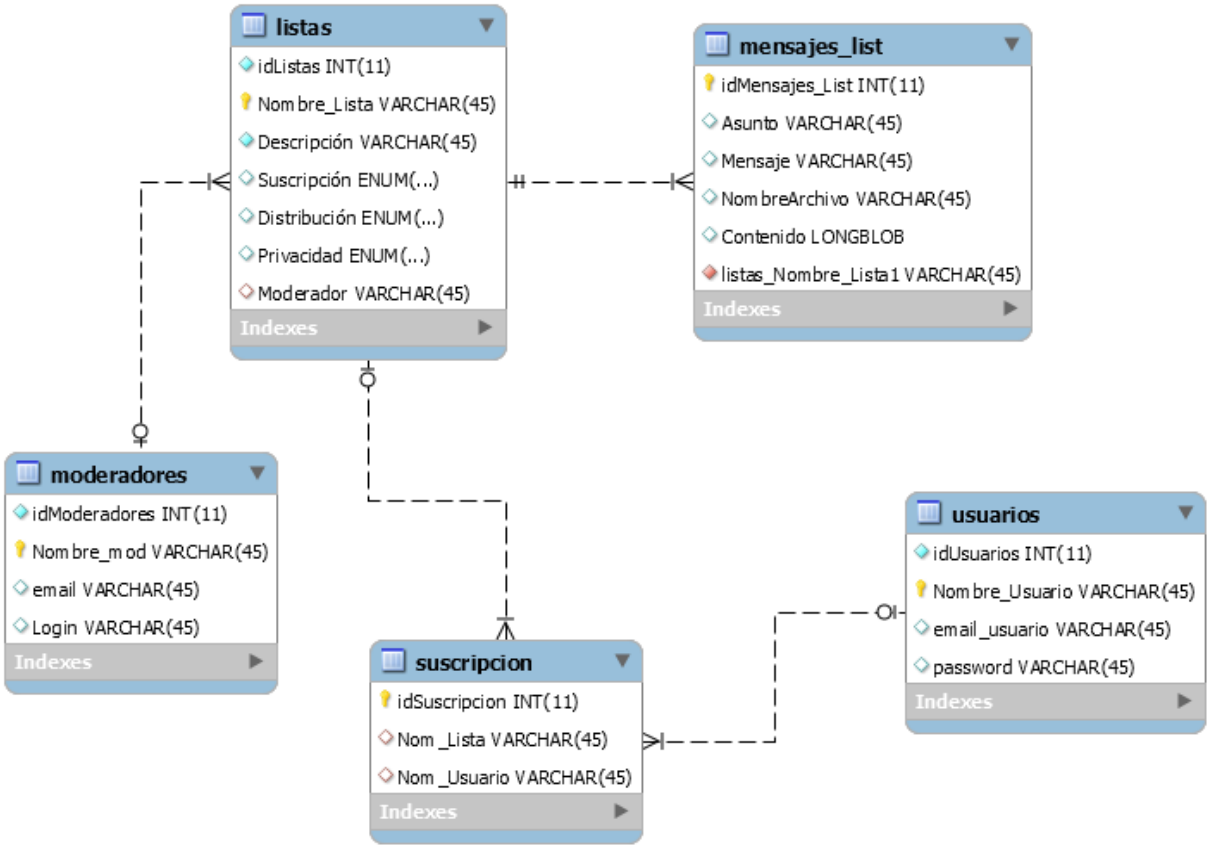

Figura 10. Estructura de la base de datos

Tabla `listas`

Almacena los datos correspondientes a los tipos de listas que se va tener en el sistema. Cuenta con una llave foránea para establecer la referencia a la tabla **`moderadores`**.

Columnas:

- idListas: tipo int, autoincrement.
- Nombre_Lista: llave primaria de la tabla,tipo:varchar.
- Descripción: tipo varchar.
- Suscripción: tipo enum(abierta o cerrada).
- Distribución: tipo enum(moderada o no moderada).
- Privacidad: tipo enum(publica o privada).
- Moderador: tipo varchar

Tabla `moderadores`

Almacena los datos de cada moderador, el cual será el propietario de una o más listas.

Columnas:
- idModeradores: tipo int
- Nombre_Mod: Llave primaria de la tabla, tipo varchar
- email: tipo varchar
- Login: tipo varchar

Tabla `suscripción`

Almacena las suscripciones que realice cualquiera de los usuarios a los diferentes tipos de listas. Tiene 2 llaves foráneas para establecer relación a las tablas **`listas`** y **`usuarios`**.

Columnas:
- idSuscripcion: tipo int
- Nom_Lista: llave foránea, tipo varchar
- Nom_Usuario: llave foránea, tipo varchar

Tabla `usuarios`

Almacena los datos de cada usuario, el cual tendrá los permisos asignados por el sistema del uso de las funciones de listas.

Columnas:
- idUsuarios: tipo int
- Nombre_Usuario: Llave primaria de la tabla, tipo: varchar
- email_usuario: tipo varchar
- password: tipo varchar

Tabla `mensajes_list`

Almacena los datos de los mensajes que se enviaran a cada lista. Tiene una llave foránea que establece la relación con la tabla **`lista`**.

Columnas:
- idMensajes_List: Llave primaria de la tabla, tipo int
- Asunto: tipo varchar
- Mensaje: tipo varchar
- NombreArchivo: tipo varchar
- Contenido: tipo blob
- Listas_Nombre_Lista1: Llave foránea, tipo int

6.1.4 Arquitectura del sistema

En la propuesta se estableció solo el diseño e implementación de los servicios web que realizarían las funcionalidades de administración de listas (ver Figura 11). Se determinó crear una aplicación cliente para poder visualizar las funcionalidades de estos servicios.

Capa datos

Esta capa se encarga de almacenar los datos de las entidades de nuestro sistema en la Base de datos; así como de acceder a ellos a través de consultas.

Capa de aplicación

En esta capa residen los recursos y servicios Web que los clientes van a utilizar para realizar sus actividades. Dependiendo de la petición del usuario, a través de la aplicación cliente, los servicios Web acceden a los recursos y envía al cliente los resultados para responder así a la petición. Esta capa también se comunica con la capa de datos, para almacenar o recuperar datos.

Capa de presentación

También se le puede llamar Cliente o Consumidor de los servicios web y es la capa con la cual interactúa el usuario de la aplicación web. Esta capa captura los datos proporcionados por el usuario y luego los envía a la capa de aplicación. En un proceso inverso, esta capa recibe los datos proporcionados por la capa de aplicación y los envía hacia la capa de presentación.

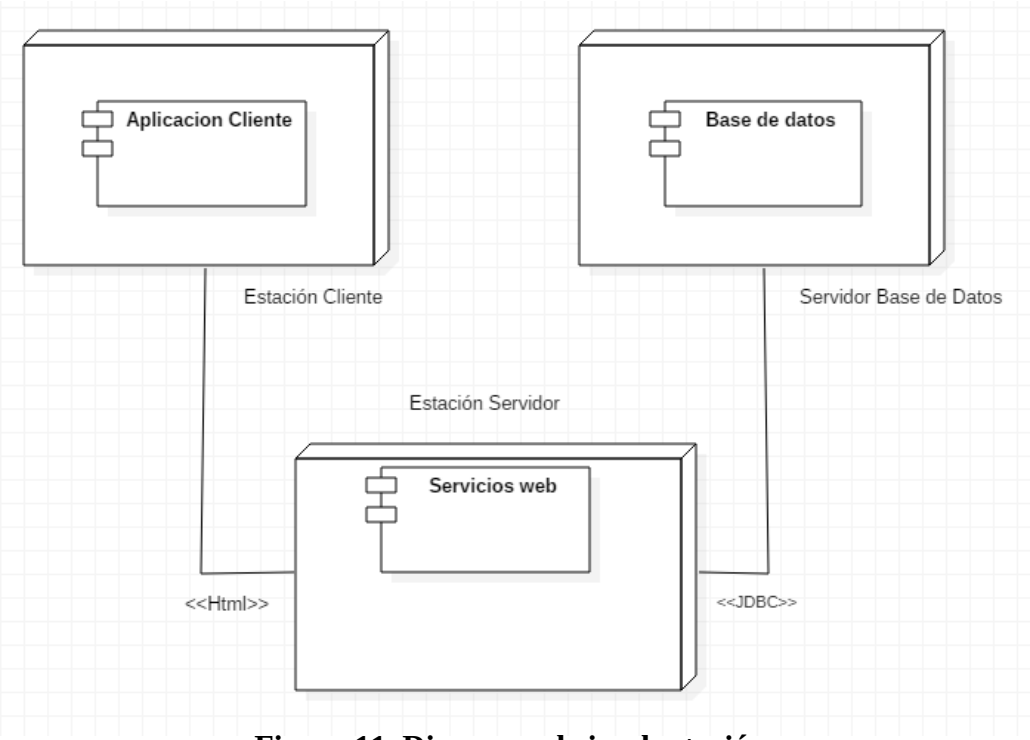

Figura 11. Diagrama de implantación.

6.2 Implementación

6.2.1 Tecnología empleada

Hardware

Para el desarrollo del proyecto se utilizó una computadora personal con los siguientes recursos:

- ✓ Memoria RAM de 3 Gb.
- ✓ Disco duro de 500 Gb.
- ✓ Procesador AMD Athlon(tm) 64 X2 Dual Core Procesador 5000+ 2.60 Ghz. ✓ Sistema Operativo Windows 10 .

El proyecto se desarrolló usando las siguientes tecnologías:

Software

Ambiente de desarrollo

El ambiente de desarrollo (IDE) utilizado fue NetBeans. Existen dos razones importantes para elegir este IDE; en primer lugar, porque es un producto de libre distribución, por lo cual no hay restricciones en su uso. En segundo lugar, porque contiene una gran cantidad de componentes de software, con los cuales, se pueden construir aplicaciones Web. La versión utilizada para desarrollar el proyecto fue la 8.0.

La implementación se dividió en dos partes: los servicios web y la aplicación cliente. A continuación se describe el software necesario en cada parte.

Servicios web

Los servicios Web fueron implementados utilizando el API JAX-WS.

Servidor de aplicaciones

El servidor de aplicaciones utilizado fue GlassFish porque implementa la plataforma Java EE6, y soporta las últimas versiones de tecnologías como: JSP, JSF, Servlets, EJBs, Java API para servicios Web (JAX-WS), entre otras tecnologías. La versión de GlassFish utilizada para

desarrollar el proyecto fue la versión 4.1.1, debido a que para el proyecto se utilizaron algunas características de Java EE6[1].

Manejador de base de datos

Se utilizó MySQL como sistema de gestión de bases de datos, utiliza el modelo relacional, es multihilo y multiusuario. MySQL es ofrecido bajo la licencia GNU GLP para cualquier uso compatible con esta licencia. La versión utilizada para el desarrollo del proyecto fue la 5.1.

MySQL Workbench

MySQL Workbench es una herramienta visual de diseño de bases de datos que integra: diseño, creación y gestión de bases de datos MySQL. Para usar esta herramienta, es necesario tener instalado previamente, el manejador de base de datos MySQL. La versión utilizada para este proyecto fue la 6.3.

Lenguaje de programación

El código tanto de la aplicación cliente como de los servicios Web fue implementado con el lenguaje de programación orientado a objetos Java.

CLIENTE

HTML 5

HTML5 es un lenguaje de etiquetas usado para estructurar y presentar el contenido para la web.

CSS

Es un lenguaje para definir el estilo o la apariencia de las páginas web, escritas con HTML o de los documentos XML. CSS se creó para separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas. La versión utilizada para este proyecto es la 3.

Lenguaje de programación

El código de la aplicación cliente como de los servicios Web, fue implementado con el lenguaje de programación orientado a objetos Java.

[1] Java Plataform Enterprise Edition. Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

6.3 Funcionalidad del sistema

Hasta el momento en este documento solo se han abordado temas de diseño; se mostraron, entre otras cosas, los casos de uso del Sistema de Administración de correo electrónico , el diagrama de clases y la descripción de las entidades, el modelo de la base de datos con la descripción de las tablas y el diagrama de la arquitectura. A continuación se mostrará una de las funcionalidades del sistema, indicando cómo interactúan tanto la aplicación cliente como el servidor, el cual proporciona los recursos necesarios para la administración de listas. Para explicar la funcionalidad del sistema se mostraran las pantallas necesarias para el etendimiento del uso de cada una de las funciones.

Comenzamos con la pantalla principal (Figura 1) del consumidor de los servicios web, es una interfaz sencilla pero con los elementos necesarios para que se puedan accesar cualquiera de los 3 roles que se tienen en el sistema.

El usuario Admon, ya esta asignado predeterminadamente, ya que será el que tendrá el control total de las funcionalidades del sistema, y en el caso del Moderador, el administrador creara el perfil para poder asignarlo como propietario de una lista.

Figura 1: Muestra la pantalla principal del sistema.

El rol de usuario (o suscriptor), si no esta registrado en el sistema, debe dar click en el botón de color rojo, donde ingresara a otra pantalla (Figura 2) donde se podrá registrar con los permisos admitidos para su rol.

Figura 2: Muestra la pantalla de registro de usuario (o suscriptor).

En el caso que ingrese el usuario "Administrador", se mostrara la siguiente interfaz (Figura 3), donde se visualizaran todas las funciones que tiene acceso. El administrador tendrá prácticamente el control de todo el sistema, podrá crear, editar y eliminar perfiles de Moderadores y usuarios (o suscriptores), asi como la libertad de usar todas las funcionalidades de una lista.

Figura 3: Muestra la pantalla del usuario administrador.

Por su parte el usuario Moderador, cuando ingrese saldrá la siguiente pantalla (Figura 4) con sus funciones habilitadas para su perfil.

Figura 4: Muestra la pantalla del usuario Moderador.

Y por ultimo tendremos la pantalla del usuario (o suscriptor) (Figura 5).

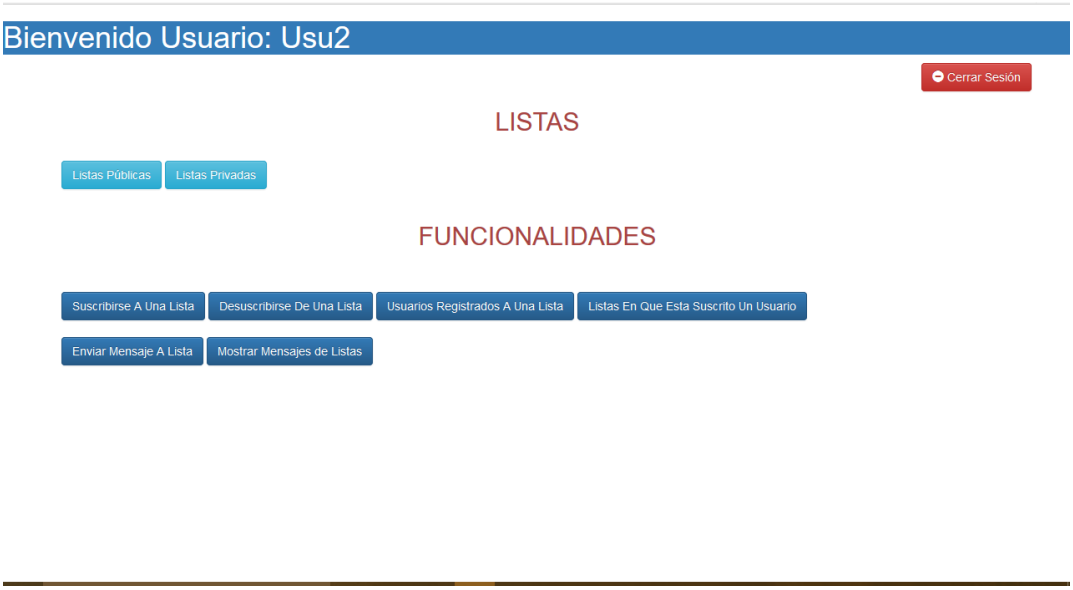

Figura 5: Muestra la pantalla del usuario (o Suscriptor).

Funciones principales

El usuario Administrador tiene una sección llamada “Roles”, que son las pantallas donde podrá crear, eliminar, editar y mostrar tanto un Moderador como un usuario (o suscriptor). A continuación se mostrara una de las pantallas (Figura 6) de los casos de uso correspondientes.

El usuario Moderador solo podrá crear, eliminar, editar y mostrar los usuarios (o suscripctores).

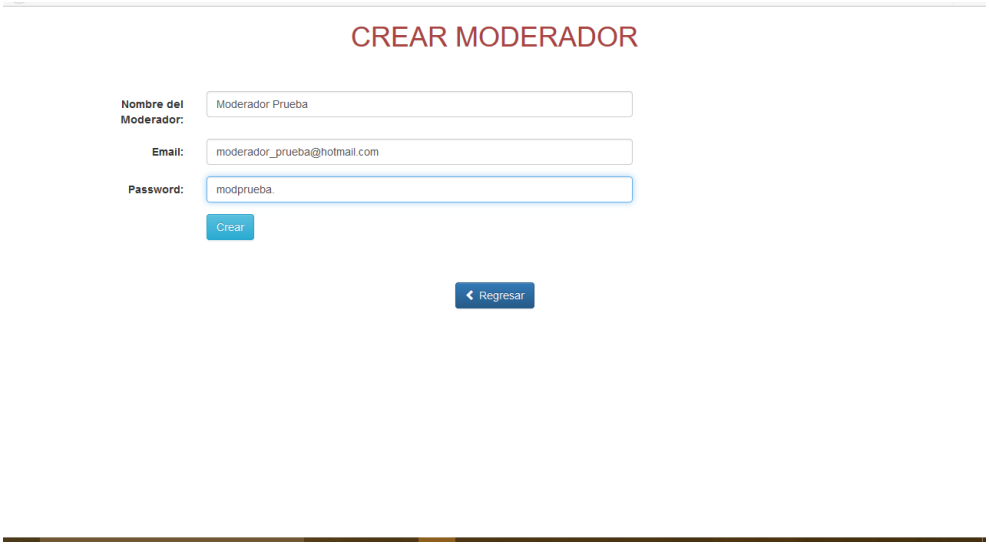

Figura 6: Muestra la pantalla del caso de uso crear moderador.

Los casos de uso de actualizar y eliminar son muy similares a la anterior pantalla, la única que cambia es la de mostrar moderadores o usuarios en su caso (Figura 7).

Figura 7: Muestra la pantalla del caso de uso mostrar usuarios.

Listas

El rol de administrador podrá crear, editar, eliminar y mostrar las listas existentes en el sistema, la siguiente pantalla (Figura 8) muestra los campos que se tienen que insertar para poder crear una lista. Los casos de uso restantes tienen pantallas muy similares.

Figura 8: Muestra la pantalla del caso de uso crear lista para el usuario Administrador.

El usuario Moderador solo tendrá la opción de ver las listas disponibles, como se muestra en la siguiente pantalla (Figura 9).

Nombre de la lista	Descripción	Suscripción	Distribución	Privacidad	Moderador
Aplicaciones Web	Javascrip	abierta	no moderada	publica	Cinco
Edgar	Tema Relevante	abierta	moderada	publica	Uno
FUEGO	Incendios	abierta	no moderada	privada	Uno
Lista 10	Tema 10	cerrada	no moderada	privada	Tres
Lista1	Tema 1	abierta	moderada	publica	Uno
Lista2	Tema 2	abierta	no moderada	privada	Dos
Lista3	Tema 3	cerrada	moderada	publica	Tres
Lista4	Tema 4	abierta	no moderada	publica	Cuatro
Lista5	Tema 5	cerrada	no moderada	privada	Cinco
Lista6	Tema 6	abierta	no moderada	publica	Uno
Lista7	Tema 7	cerrada	moderada	privada	Dos
Lista8	Tema 8	abierta	no moderada	publica	Cinco
Lista9	Tema 9	abierta	no moderada	publica	Uno

Figura 9: Muestra la pantalla del caso de uso Listas disponibles para el usuario Moderador.

En el caso del usuario (o suscriptor), tiene dos casos de uso asignados para poder suscribirse a una lista, la pantalla de listas públicas donde se muestran las listas donde se puede mostrar toda su información y la de pantalla de listas privadas (Figura 10) donde solo se mostrara el nombre y la descripción.

Figura 10: Muestra la pantalla del caso de uso Listas privadas para el usuario (O suscriptor).

En el caso de las listas públicas y privadas al darle click al nombre automáticamente nos enviara a la pantalla de suscripción.

Funcionalidades de listas

El usuario Administrador y el Moderador tienen los mismos privilegios en cuanto a las funcionalidades de listas.

En el caso de uso **Suscribir usuario a una lista** se muestra la siguiente pantalla (Figura 11) donde se deben llenar los campos pertinentes.

Figura 11: Muestra la pantalla del caso de uso Suscribir a una lista.

Para el caso del usuario (O suscriptor), la pantalla es igual con la única diferencia que en el campo de Usuario a suscribir, tendrá su nombre y solo tendrá que llenar el campo de la lista a la que se quiere suscribir y en caso que esta lista sea cerrada, privada o ambas, saldrá un mensaje de error además de recibir un correo por parte del Moderador de la lista el cual, tendrá que autorizar o no a suscripción a la lista.

Para el caso de **Desuscribir usuario de lista**, la pantalla y los campos son los mismos, solo que hara el proceso inverso.

El caso de uso Usuarios registrados a lista, es el mismo para cada uno de los roles en el sistema, la pantalla será la siguiente (Figura 12).

Figura 12: Muestra la pantalla del caso de uso Usuarios registrados a una lista.

El caso de uso similar al anterior, llamado **Listas en que esta suscrito un usuario,** la pantalla es idéntica con la única diferencia que en vez de poner el nombre de la lista, será el nombre del usuario y asi desplegara las listas que esta registrado el usuario elegido.

Para la función de **Enviar mensaje a lista**, se utiliza la siguiente pantalla (Figura 13), donde se tendrán que llenar todos los campos indicados. Cabe aclarar que en el campo “Ruta del archivo”, se tiene que introducir la ruta absoluta de a ubicación donde se encuentra el documento o archivo que se quiere adjuntar con el mensaje.

Figura 13: Muestra la pantalla del caso de uso Enviar mensaje a una lista.

La siguiente pantalla, tiene dos casos de uso incluidos los cuales son “Mostrar mensajes de lista” y “Descargar contenido de mensaje”. El primer caso de uso se muestra de la siguiente manera (Figura 14). Y el segundo también se muestra en la parte inferior de la pantalla donde se debe de introducir los datos solicitados. Una vez eligiendo que archivo se desea descargar, proporcionando la ruta donde se va almacenar.

Figura 14: Muestra la pantalla del caso de uso Mostrar mensajes de una lista.

7 Resultados

En la realización del proyecto se obtuvieron los siguientes productos:

Actividad	Producto
Diseñar el servicio web que permita la creación, modificación y eliminación de diferentes tipos de listas.	Documentación de diseño
Implementar el servicio web que permita la gestión de listas.	Servicio web implementado
Probar el servicio web que permita la gestión de listas.	Servicio web funcionando
Diseñar el servicio web que realice las funciones para gestionar una lista de contactos de correo electrónico.	Documentación de diseño
Implementar el servicio web que realice las funciones para gestionar una lista de contactos de correo electrónico.	Servicio web implementado
Probar el servicio web que realice las funciones para gestionar una lista de contactos de correo electrónico.	Servicio web funcionando
Diseñar el servicio web que guarde los mensajes de las listas.	Documentación de diseño
Implementar el servicio web que guarde los mensajes de las listas.	Servicio web implementado
Probar el servicio web que guarde los mensajes de las listas.	Servicio web funcionando
Escritura de Reporte Final.	Documentación final completa

8 Análisis y discusión de resultados

Durante la elaboración de este proyecto se desarrollaron los servicios web que fueran capaces de administrar listas de correo electrónico, donde efectivamente se cumplieron los objetivos.

Los servicios web son capaces de crear diferentes tipos de listas, además realizan las funciones específicas como lo son:

- Suscribir un usuario.
- Desuscribir un usuario.
- Enviar mensaje a una lista.
- Mostrar mensajes de una lista.
- Descargar contenido de mensaje de una lista.
- Ver en qué listas esta suscrito un usuario.
- Ver que usuarios están registrados a una lista.
- Mostrar listas existentes.

Para poder visualizar de manera correcta la funcionalidad de los servicios web, se optó por programar un cliente el cual tiene las interfaces necesarias donde se muestran los diferentes roles existentes en el sistema, ya que cada uno habilita las funciones que le corresponden.

Algunos límites que tiene el sistema y que se podría mejorar en algún proyecto de continuidad, es la posibilidad de enviar varios archivos adjuntos por mensaje. En este caso solo fue diseñado para adjuntar un solo archivo por mensaje.

Se realizaron diferentes pruebas para comprobar su funcionalidad, mismas que se describen a continuación:

Comprobación de funciones por rol: En esta prueba se verificaron las funcionalidades que se ofrecen para el Administrador, moderador y usuario.

Funcionalidades: Se probaron cada una de las diferentes opciones de funciones de una lista, donde cada una fue ejecutada con éxito.

Por ejemplo, en la funcionalidad de “Suscribir a una lista”, cuando es ejecutada por un usuario, si la lista seleccionada es de tipo privada, cerrada o ambas, se le enviará un correo electrónico al moderador de la lista, para su autorización. Esto mismo aplica para la función “Desuscribir a una lista”.

En el caso de uso “enviar mensaje”, si la lista a la que se le envía el mensaje es moderada, se le enviará un correo electrónico al moderador para poder autorizar el envío del mensaje a la lista.

9 Conclusiones

El proyecto que se presenta en este documento tiene como objetivo la implementación de un sistema capaz de administrar la gestión de listas de correo electrónico.

Aún cuando el objetivo era solo implementar los servicios web, para probar los servicios se implementó una aplicación web. Dicha aplicación tiene una interfaz de inicio de sesión. Si el usuario aún no está registrado, permite su inscripción. Si el usuario ya está registrado, mostrará las funciones y consultas que pueden realizar con el rol que posea.

El sistema cuenta con tres roles: Administrador, Moderador y Usuario, dependiendo de cada modo de usuario, este tendrá acceso a diferentes funcionalidades en el sistema.

Durante la programación del sistema he adquirido diferentes conocimientos que me han ayudado a crear las interfaces con mayor funcionalidad y diseño. Entre ellos aprendí a crear aplicaciones web, con distintas tecnologías como el uso correcto de HTML 5, CSS 3 y Javascript. También aprendí a utilizar servlets en Java.

Bibliografía

[1] Avendaño Méndez Sergio Enrique, **"Gestión de calificaciones de cursos mediante servicios web"**, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma
Metropolitana Azcapotzalco, México, 2012.

[2] Pascual Martínez Jorge, **"Extracción automatizada y representación de servicios web mediante ontologías"**, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2012.

[3] Malagón Mercado, Daniel Armando, **"Solución de problemas de mediana dificultad utilizando composición de servicios web"**, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2013.

[4] Documentación listas distribución correo (Majordomo). http://www.rediris.es/mail/gt/foroses/tr/documaj/listas2.html, 2013.

[5] LISTSERV Email List Management Software, http://www.lsoft.se/products/listserv.asp , 2013.

[6] Yahoo! Grupos - crear grupos y comunidades virtuales gratis,. http://mx.groups.yahoo.com/ , 2013.

Apéndice A – Documentación de Servicios Web.

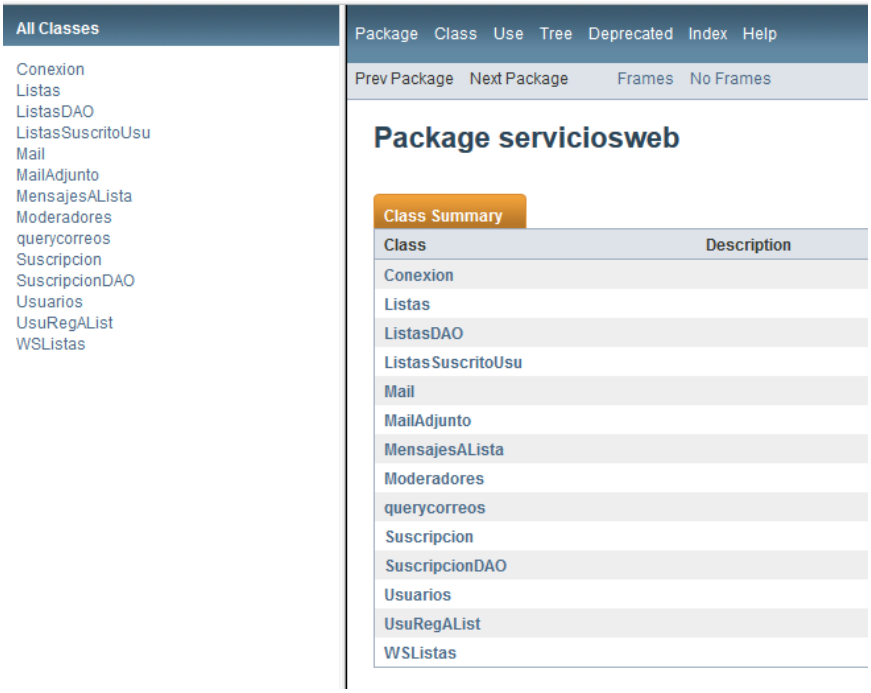

Figura 1: Muestra todas las clases usadas en los servicios web.

Class Conexión

- java.lang.Object
  - serviciosweb.Conexion

```
public class Conexion
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
`Conexion()`

Method Summary

Methods	
Modifier and Type	**Method and Description**
`java.sql.Connection`	**`getConnection`**`()`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

`Conexion`
`public Conexion()`

Method Detail

`getConnection`
`public java.sql.Connection getConnection()`

Class Listas

- java.lang.Object
  - serviciosweb.Listas

- Direct Known Subclasses:
  SuscripcionDAO

```
public class Listas
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
`Listas``()`
`Listas``(int id, java.lang.String nombre, java.lang.String descripcion, java.lang.String suscripcion, java.lang.String distribucion, java.lang.String privacidad, java.lang.String moderador)`

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**getDescripcion**()
java.lang.String	**getDistribucion**()
int	**getId**()
java.lang.String	**getModerador**()
java.lang.String	**getNombre**()
java.lang.String	**getPrivacidad**()
java.lang.String	**getSuscripcion**()
Void	**setDescripcion**(java.lang.String descripcion)
Void	**setDistribucion**(java.lang.String distribucion)
Void	**setId**(int id)
Void	**setModerador**(java.lang.String moderador)
Void	**setNombre**(java.lang.String nombre)
Void	**setPrivacidad**(java.lang.String privacidad)
Void	**setSuscripcion**(java.lang.String suscripcion)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Listas
public Listas()

Listas
- public Listas(int id,
- java.lang.String nombre,
- java.lang.String descripcion,
- java.lang.String suscripcion,
- java.lang.String distribucion,
- java.lang.String privacidad,
  java.lang.String moderador)

Method Detail

- **getId**

```
public int getId()
```

- **setId**

```
public void setId(int id)
```

- **getNombre**

```
public java.lang.String getNombre()
```

- **setNombre**

```
public void setNombre(java.lang.String nombre)
```

- **getDescripcion**

```
public java.lang.String getDescripcion()
```

- **setDescripcion**

```
public void setDescripcion(java.lang.String descripcion)
```

- **getSuscripcion**

```
public java.lang.String getSuscripcion()
```

- **setSuscripcion**

```
public void setSuscripcion(java.lang.String suscripcion)
```

- **getDistribucion**

```
public java.lang.String getDistribucion()
```

- **setDistribucion**

```
public void setDistribucion(java.lang.String distribucion)
```

- **getPrivacidad**

```
public java.lang.String getPrivacidad()
```

- **setPrivacidad**

```
public void setPrivacidad(java.lang.String privacidad)
```

- **getModerador**

```
public java.lang.String getModerador()
```

- **setModerador**

```
public void setModerador(java.lang.String moderador)
```

Class ListasDAO

- java.lang.Object
  -
    - serviciosweb.ListasDAO

```
public class ListasDAO
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
ListasDAO()

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**Actualizarlista**(java.lang.String Nombre, java.lang.String Descripcion, java.lang.String Suscripción, java.lang.String Distribución, java.lang.String Privacidad, java.lang.String Moderador)
java.lang.String	**Crearlista**(java.lang.String Nombre, java.lang.String Descripcion, java.lang.String Suscripción, java.lang.String Distribución, java.lang.String Privacidad, java.lang.String Moderador)
java.lang.String	**Eliminarlista**(java.lang.String Nombre)
Listas[]	**mostrarlistas**()
Listas	**verificaLista**(int id, java.lang.String nombre)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **ListasDAO**

```
public ListasDAO()
```

Method Detail

- **verificaLista**
  - public Listas verificaLista(int id,
    java.lang.String nombre)
- **mostrarlistas**

  public Listas[] mostrarlistas()

- **Crearlista**
  - public java.lang.String Crearlista(java.lang.String Nombre, java.lang.String Descripcion,java.lang.String Suscripción,java.lang.String Distribución,java.lang.String Privacidad,java.lang.String Moderador)

- **Eliminarlista**

  public java.lang.String Eliminarlista(java.lang.String Nombre)

- **Actualizarlista**
  - public java.lang.String Actualizarlista(java.lang.String Nombre,java.lang.String Descripcion,java.lang.String Suscripción,java.lang.String Distribución,java.lang.String Privacidad,java.lang.String Moderador)

Class ListasSuscritoUsu

- java.lang.Object
  -
    - serviciosweb.ListasSuscritoUsu

```
public class ListasSuscritoUsu
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
ListasSuscritoUsu()
ListasSuscritoUsu(java.lang.String nombre_Usuario, java.lang.String nombre_Lista)

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**getNombre_Lista**()
java.lang.String	**getNombre_Usuario**()
Void	**setNombre_Lista**(java.lang.String nombre_Lista)
Void	**setNombre_Usuario**(java.lang.String nombre_Usuario)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **ListasSuscritoUsu**

public ListasSuscritoUsu()

- **ListasSuscritoUsu**

  - public ListasSuscritoUsu(java.lang.String nombre_Usuario, java.lang.String nombre_Lista)

Method Detail

- **getNombre_Usuario**

public java.lang.String getNombre_Usuario()

- **setNombre_Usuario**

public void setNombre_Usuario(java.lang.String nombre_Usuario)

- **getNombre_Lista**

public java.lang.String getNombre_Lista()

- **setNombre_Lista**

public void setNombre_Lista(java.lang.String nombre_Lista)

Class Mail

- java.lang.Object
  -
    - serviciosweb.Mail

```
public class Mail
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
Mail()

Method Summary

Methods	
Modifier and Type	**Method and Description**
boolean	**enviarEmail**(java.lang.String para, java.lang.String asunto, java.lang.String mensaje)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **Mail**

```
public Mail()
```

Method Detail

- **enviarEmail**
  - public boolean enviarEmail(java.lang.String para,java.lang.String asunto,java.lang.String mensaje)

Class MailAdjunto

- java.lang.Object
  -
    - serviciosweb.MailAdjunto

```
public class MailAdjunto
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
MailAdjunto()

Method Summary

Methods	
Modifier and Type	**Method and Description**
boolean	**enviarEmailA**(java.lang.String para, java.lang.String asunto, java.lang.String mensaje, java.lang.String ruta)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **MailAdjunto**

```
public MailAdjunto()
```

Method Detail

- **enviarEmailA**
  - public boolean enviarEmailA(java.lang.String para,java.lang.String asunto,java.lang.String mensaje,java.lang.String ruta)throws java.io.IOException

    Throws:java.io.IOException

Class MensajesALista

- java.lang.Object
  -
    - serviciosweb.MensajesALista

```
public class MensajesALista
extends java.lang.Object
```

Constructor Summary

Constructors	
Constructor and Description	
MensajesALista()	
MensajesALista(int idMensaje_list, java.lang.String Asunto, java.lang.String NombreArchivo, java.lang.String Contenido, java.lang.String NombreLista)	

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**getAsunto**()
java.lang.String	**getContenido**()
Int	**getIdMensaje_list**()
java.lang.String	**getNombreArchivo**()
java.lang.String	**getNombreLista**()
Void	**setAsunto**(java.lang.String Asunto)
Void	**setContenido**(java.lang.String Contenido)
Void	**setIdMensaje_list**(int idMensaje_list)
Void	**setNombreArchivo**(java.lang.String NombreArchivo)
Void	**setNombreLista**(java.lang.String NombreLista)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **MensajesALista**

public MensajesALista()

- **MensajesALista**

- public MensajesALista(int idMensaje_list,java.lang.String Asunto,java.lang.String NombreArchivo,java.lang.String Contenido,java.lang.String NombreLista)

Method Detail

- **getIdMensaje_list**

public int getIdMensaje_list()

- **setIdMensaje_list**

public void setIdMensaje_list(int idMensaje_list)

- **getAsunto**

public java.lang.String getAsunto()

- **setAsunto**

public void setAsunto(java.lang.String Asunto)

- **getNombreArchivo**

public java.lang.String getNombreArchivo()

- **setNombreArchivo**

public void setNombreArchivo(java.lang.String NombreArchivo)

- **getContenido**

public java.lang.String getContenido()

- **setContenido**

public void setContenido(java.lang.String Contenido)

- **getNombreLista**

public java.lang.String getNombreLista()

- **setNombreLista**

public void setNombreLista(java.lang.String NombreLista)

Class Moderadores

- java.lang.Object
  -
    - serviciosweb.Moderadores

```
public class Moderadores
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
Moderadores()
Moderadores(int idModeradores, java.lang.String email, java.lang.String Nombre_Moderador, java.lang.String Login)

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**getEmail**()
Int	**getIdModeradores**()
java.lang.String	**getLogin**()
java.lang.String	**getNombre_Moderador**()
void	**setEmail**(java.lang.String email)
void	**setIdModeradores**(int idModeradores)
void	**setLogin**(java.lang.String Login)
void	**setNombre_Moderador**(java.lang.String Nombre_Moderador)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **Moderadores**

```
public Moderadores()
```

- **Moderadores**

```
public Moderadores(int idModeradores,java.lang.String email
,java.lang.String Nombre_Moderador,java.lang.String Login)
```

Method Detail

- **getIdModeradores**

```
public int getIdModeradores()
```

- **setIdModeradores**

```
public void setIdModeradores(int idModeradores)
```

- **getEmail**

```
public java.lang.String getEmail()
```

- **setEmail**

```
public void setEmail(java.lang.String email)
```

- **getNombre_Moderador**

```
public java.lang.String getNombre_Moderador()
+
```

- **setNombre_Moderador**

```
public void setNombre_Moderador(java.lang.String Nombre_Mod
erador)
```

- **getLogin**

```
public java.lang.String getLogin()
```

- **setLogin**

```
public void setLogin(java.lang.String Login)
```

Class querycorreos

- java.lang.Object
  -
  - serviciosweb.querycorreos

```
public class querycorreos
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
querycorreos()
querycorreos(java.lang.String email_usuarios)

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**getEmail_usuarios**()
void	**setEmail_usuarios**(java.lang.String email_usuarios)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **querycorreos**

public querycorreos()

- **querycorreos**

public querycorreos(java.lang.String email_usuarios)

Method Detail

- **getEmail_usuarios**

public java.lang.String getEmail_usuarios()

- **setEmail_usuarios**

public void setEmail_usuarios(java.lang.String email_usuarios)

Class Suscripcion

- java.lang.Object
  -
  - serviciosweb.Suscripcion

public class Suscripcion
extends java.lang.Object

Constructor Summary

Constructors
Constructor and Description
Suscripcion()
Suscripcion(int idSuscripcion,java.lang.String Nom_Lista, java.lang.String Nom_Usuario)

Method Summary

Methods	
Modifier and Type	**Method and Description**
int	**getIdSuscripcion**()
java.lang.String	**getNom_Lista**()
java.lang.String	**getNom_Usuario**()
void	**setIdSuscripcion**(int idSuscripcion)
void	**setNom_Lista**(java.lang.String Nom_Lista)
void	**setNom_Usuario**(java.lang.String Nom_Usuario)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **Suscripcion**

public Suscripcion()

- **Suscripcion**

  - public Suscripcion(int idSuscripcion,java.lang.String Status,java.lang.String Nom_Lista,java.lang.String Nom_Usuario)

Method Detail

- **getNom_Usuario**

public java.lang.String getNom_Usuario()

- **setNom_Usuario**

public void setNom_Usuario(java.lang.String Nom_Usuario)

- **getIdSuscripcion**

public int getIdSuscripcion()

- **setIdSuscripcion**

```
public void setIdSuscripcion(int idSuscripcion)
```

- **getNom_Lista**

```
public java.lang.String getNom_Lista()
```

- **setNom_Lista**

```
public void setNom_Lista(java.lang.String Nom_Lista)
```

Class SuscripcionDAO

- java.lang.Object
  - Listas
    -
      - serviciosweb.SuscripcionDAO

```
public class SuscripcionDAO
extends Listas
```

Constructor Summary

Constructors
Constructor and Description
SuscripcionDAO()

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**ActualizarModerador**(java.lang.String Nombre_Moderador, java.lang.String email_moderador, java.lang.String Login)
java.lang.String	**ActualizarUsuario**(java.lang.String Nombre_Usuario, java.lang.String email_usuario, java.lang.String password)
java.lang.String	**CrearModerador**(java.lang.String Nombre_Moderador, java.lang.String email_moderador, java.lang.String login)
java.lang.String	**CrearUsuario**(java.lang.String Nombre_Usuario, java.lang.String email_usuario, java.lang.String password)

java.lang.String	**DescargarMsj**(java.lang.String Lista, java.lang.String Asunto, java.lang.String Ruta)
java.lang.String	**Desuscribir_JSP**(java.lang.String Nom_Lista, java.lang.String Nom_Usuario)
java.lang.String	**Desuscribir**(java.lang.String Nom_Lista, java.lang.String Nom_Usuario)
java.lang.String	**EliminarModerador**(java.lang.String Nombre_Moderador)
java.lang.String	**EliminarUsuario**(java.lang.String Nombre_Usuario)
java.lang.String	**EnviarMensaje**(java.lang.String Asunto, java.lang.String Mensaje, java.lang.String NombreArchivo, java.lang.String Contenido, java.lang.String NombreLista, java.lang.String N_usuario)
java.lang.String	**EnviarMensajeJSP**(java.lang.String Asunto, java.lang.String Mensaje, java.lang.String NombreArchivo, java.lang.String Contenido, java.lang.String NombreLista, java.lang.String N_usuario)
ListasSuscritoUsu[]	**Listas Sus Usuario**(java.lang.String Nom_Usuario)
MensajesALista[]	**Mensajes de una lista**(java.lang.String listas_Nombre_Lista1)
java.lang.String	**prueba**(java.lang.String Nom_Lis, java.lang.String Contenido)
java.lang.String	**Suscribir_JSP**(java.lang.String Nom_Lista, java.lang.String Nom_Usuario)
java.lang.String	**Suscribir**(java.lang.String Nom_Lista, java.lang.String Nom_Usuario)
UsuRegAList[]	**usuariosregistrados**(java.lang.String Nom_Lista)
Moderadores	**ValidarModerador**(java.lang.String Nombre_Moderador, java.lang.String password_mod)
Usuarios	**ValidarUsuario**(java.lang.String Nombre_Usuario, java.lang.String password)

Methods inherited from class serviciosweb.Listas

getDescripcion, getDistribucion, getId, getModerador, getNombre, getPrivacidad, getSuscripcion, setDescripcion, setDistribucion, setId, setModerador, setNombre, setPrivacidad, setSuscripcion

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **SuscripcionDAO**

public SuscripcionDAO()

Method Detail

- **Suscribir**

- public java.lang.String Suscribir(java.lang.String Nom_Lista,java.lang.String Nom_Usuario)

- **Suscribir_JSP**

- public java.lang.String Suscribir_JSP(java.lang.String Nom_Lista,java.lang.String Nom_Usuario)

- **usuariosregistrados**

public UsuRegAList[] usuariosregistrados(java.lang.String Nom_Lista)

- **Listas_Sus_Usuario**

public ListasSuscritoUsu[] Listas_Sus_Usuario(java.lang.String Nom_Usuario)

- **EnviarMensaje**

- public java.lang.String EnviarMensaje(java.lang.String Asunto,java.lang.String Mensaje,java.lang.String NombreArchivo,java.lang.String Contenido,java.lang.String NombreLista,java.lang.String N_usuario)

- **Mensajes_de_una_lista**

- public MensajesALista[] Mensajes_de_una_lista(java.lang.String listas_Nombre_Lista1)throwsjava.io.FileNotFoundException,java.io.IOException

  Throws:java.io.FileNotFoundException java.io.IOException

- **ValidarUsuario**

- public Usuarios ValidarUsuario(java.lang.String Nombre_Usuario,java.lang.String password)

- **CrearUsuario**

- public java.lang.String CrearUsuario(java.lang.String Nombre_Usuario,java.lang.String email_usuario,java.lang.String password)

- **EliminarUsuario**

public java.lang.String EliminarUsuario(java.lang.String Nombre_Usuario)

- **ActualizarUsuario**

- public java.lang.String ActualizarUsuario(java.lang.String Nombre_Usuario,java.lang.String email_usuario,java.lang.String password)

- **ValidarModerador**

- public Moderadores ValidarModerador(java.lang.String Nombre_Moderador,java.lang.String password_mod)

- **CrearModerador**

- public java.lang.String CrearModerador(java.lang.String Nombre_Moderador,java.lang.String email_moderador,java.lang.String login)

- **EliminarModerador**

public java.lang.String EliminarModerador(java.lang.String Nombre_Moderador)

- **ActualizarModerador**

- public java.lang.String ActualizarModerador(java.lang.String Nombre_Moderador,java.lang.String email_moderador,java.lang.String Login)

- **Desuscribir**

- public java.lang.String Desuscribir(java.lang.String Nom_Lista,java.lang.String Nom_Usuario)

- **prueba**

- public java.lang.String prueba(java.lang.String Nom_Lis,java.lang.String Contenido)throws java.io.IOException

Throws:java.io.IOException

- **Desuscribir_JSP**

- public java.lang.String Desuscribir_JSP(java.lang.String Nom_Lista,java.lang.String Nom_Usuario)

- **EnviarMensajeJSP**

- public java.lang.String EnviarMensajeJSP(java.lang.String Asunto,java.lang.String Mensaje,java.lang.String NombreArchivo,java.lang.String Contenido,java.lang.String NombreLista,java.lang.String N_usuario)

- **DescargarMsj**

- public java.lang.String DescargarMsj(java.lang.String Lista,java.lang.String Asunto,java.lang.String Ruta)

Class Usuarios

- java.lang.Object
  -
    - serviciosweb.Usuarios

```
public class Usuarios
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
Usuarios()
Usuarios(int idUsuarios, java.lang.String Nombre_Usuario, java.lang.String email_usuario, java.lang.String password)

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**getEmail_usuario**()
Int	**getIdUsuarios**()
java.lang.String	**getNombre_Usuario**()
java.lang.String	**getPassword**()
Void	**setEmail_usuario**(java.lang.String email_usuario)
Void	**setIdUsuarios**(int idUsuarios)
Void	**setNombre_Usuario**(java.lang.String Nombre_Usuario)
Void	**setPassword**(java.lang.String password)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **Usuarios**

```
public Usuarios()
```

- **Usuarios**

```
public Usuarios(int idUsuarios,java.lang.String Nombre_Usua
rio,java.lang.String email_usuario,java.lang.String passwor
d)
```

Method Detail

- **getPassword**

```
public java.lang.String getPassword()
```

- **setPassword**

```
public void setPassword(java.lang.String password)
```

- **getIdUsuarios**

```
public int getIdUsuarios()
```

- **setIdUsuarios**

```
public void setIdUsuarios(int idUsuarios)
```

- **getNombre_Usuario**

```
public java.lang.String getNombre_Usuario()
```

- **setNombre_Usuario**

```
public void setNombre_Usuario(java.lang.String Nombre_Usuar
io)
```

- **getEmail_usuario**

```
public java.lang.String getEmail_usuario()
```

- **setEmail_usuario**

```
public void setEmail_usuario(java.lang.String email_usuario
)
```

Class UsuRegAList

- java.lang.Object
  -
  - serviciosweb.UsuRegAList

```
public class UsuRegAList
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
UsuRegAList()
UsuRegAList(java.lang.String nombre_lista, java.lang.String Nom_Usuario)

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**getNom_Usuario**()
java.lang.String	**getNombre_lista**()
Void	**setNom_Usuario**(java.lang.String Nom_Usuario)
Void	**setNombre_lista**(java.lang.String nombre_lista)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **UsuRegAList**

public UsuRegAList()

- **UsuRegAList**

▪ public UsuRegAList(java.lang.String nombre_lista,java.lang.String Nom_Usuario)

Method Detail

- **getNombre_lista**

public java.lang.String getNombre_lista()

- **setNombre_lista**

public void setNombre_lista(java.lang.String nombre_lista)

- **getNom_Usuario**

public java.lang.String getNom_Usuario()

- **setNom_Usuario**

public void setNom_Usuario(java.lang.String Nom_Usuario)

Class WSListas

- java.lang.Object
  -
    - serviciosweb.WSListas

```
public class WSListas
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
WSListas()

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**Actualizar Moderador**(java.lang.String Nombre_Moderador, java.lang.String email_moderador, java.lang.String login_moderador)
java.lang.String	**Actualizar Usuario**(java.lang.String Nombre_Usuario, java.lang.String email_usuario, java.lang.String password)
java.lang.String	**Actualizar**(java.lang.String Nombre, java.lang.String Descripcion, java.lang.String Suscripcion, java.lang.String Distribucion, java.lang.String Privacidad, java.lang.String Moderador)
java.lang.String	**Crear Moderador**(java.lang.String Nombre_Moderador, java.lang.String email_moderador, java.lang.String login_moderador)
java.lang.String	**Crear Usuario**(java.lang.String Nombre_Usuario, java.lang.String email_usuario, java.lang.String password)
java.lang.String	**Crear**(java.lang.String Nombre, java.lang.String Descripcion, java.lang.String Suscripcion, java.lang.String Distribucion, java.lang.String Privacidad, java.lang.String Moderador) Web service operation

java.lang.String	**DescargarMensaje**(java.lang.String Lista, java.lang.String Asunto, java.lang.String Ruta)
java.lang.String	**Desuscribir_JSP**(java.lang.String Nom_Lista, java.lang.String Nom_Usuario)
java.lang.String	**Desuscribir**(java.lang.String Nom_Lista, java.lang.String Nom_Usuario)
java.lang.String	**Eliminar_Moderador**(java.lang.String Nombre_Moderador)
java.lang.String	**Eliminar_Usuario**(java.lang.String Nombre_Usuario)
java.lang.String	**Eliminar**(java.lang.String Nombre) Web service operation
ListasSuscritoUsu[]	**ListasSuscritoUsuario**(java.lang.String Nom_Usuario)
java.lang.String	**Mensajelista**(java.lang.String Asunto, java.lang.String Mensaje, java.lang.String NombreArchivo, java.lang.String Contenido, java.lang.String NombreLista, java.lang.String NombreUsuario)
java.lang.String	**MensajelistaJSP**(java.lang.String Asunto, java.lang.String Mensaje, java.lang.String NombreArchivo, java.lang.String Contenido, java.lang.String NombreLista, java.lang.String NombreUsuario)
MensajesALista[]	**Mostrar_Mensajes_Lista**(java.lang.String listas_Nombre_Lista1)
Listas[]	**Mostrar**() Web service operation
java.lang.String	**Suscribir_Mod**(java.lang.String Nom_Lista, java.lang.String Nom_Usuario)
java.lang.String	**Suscribir**(java.lang.String Nom_Lista, java.lang.String Nom_Usuario) FUNCIONALIDADES DE LISTAS
UsuRegAList[]	**Usuarios_Registrados_En_Lista**(java.lang.String Nom_Lista)
Moderadores	**Validar_Moderador**(java.lang.String Nombre_Moderador, java.lang.String Login)
Usuarios	**Validar_Usuario**(java.lang.String Nombre_Usuario, java.lang.String password)
Listas	**Validar**(int idListas, java.lang.String Nombre) Web service operation

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **WSListas**

public WSListas()

Method Detail

- **Validar**

- public Listas Validar(int idListas,java.lang.String Nombre)

Web service operation

- **Mostrar**

public Listas[] Mostrar()

Web service operation

- **Crear**

- public java.lang.String Crear(java.lang.String Nombre,java.lang.String Descripcion,java.lang.String Suscripcion,java.lang.String Distribucion,java.lang.String Privacidad,java.lang.String Moderador)

Web service operation

- **Actualizar**

- public java.lang.String Actualizar(java.lang.String Nombre,java.lang.String Descripcion,java.lang.String Suscripcion,java.lang.String Distribucion,java.lang.String Privacidad,java.lang.String Moderador)

- **Eliminar**

public java.lang.String Eliminar(java.lang.String Nombre)

Web service operation

- **Suscribir**

- public java.lang.String Suscribir(java.lang.String Nom_Lista,java.lang.String Nom_Usuario)

FUNCIONALIDADES DE LISTAS

- **Suscribir_Mod**

- public java.lang.String Suscribir_Mod(java.lang.String Nom_Lista,java.lang.String Nom_Usuario)

- **Desuscribir**

- public java.lang.String Desuscribir(java.lang.String Nom_Lista,java.lang.String Nom_Usuario)

  - **Desuscribir_JSP**
- public java.lang.String Desuscribir_JSP(java.lang.String Nom_Lista,java.lang.String Nom_Usuario)

  - **Mensajelista**
- public java.lang.String Mensajelista(java.lang.String Asunto,java.lang.String Mensaje,java.lang.String NombreArchivo,java.lang.String Contenido,java.lang.String NombreLista,java.lang.String NombreUsuario)

  - **DescargarMensaje**
- public java.lang.String DescargarMensaje(java.lang.String Lista,java.lang.String Asunto,java.lang.String Ruta)

  - **MensajelistaJSP**
- public java.lang.String MensajelistaJSP(java.lang.String Asunto,java.lang.String Mensaje,java.lang.String NombreArchivo,java.lang.String Contenido,java.lang.String NombreLista,java.lang.String NombreUsuario)

  - **Mostrar_Mensajes_Lista**
- public MensajesALista[] Mostrar_Mensajes_Lista(java.lang.String listas_Nombre_Lista1)throws
  java.io.FileNotFoundException,
  java.io.IOException
  Throws:java.io.FileNotFoundException java.io.IOException

  - **Usuarios_Registrados_En_Lista**

  public UsuRegAList[] Usuarios_Registrados_En_Lista(java.lang.String Nom_Lista)

  - **ListasSuscritoUsuario**

  public ListasSuscritoUsu[] ListasSuscritoUsuario(java.lang.String Nom_Usuario)

  - **Validar_Usuario**
- public Usuarios Validar_Usuario(java.lang.String Nombre_Usuario,java.lang.String password)

  - **Crear_Usuario**
- public java.lang.String Crear_Usuario(java.lang.String Nombre_Usuario,java.lang.String email_usuario,java.lang.String password)

  - **Eliminar_Usuario**

  public java.lang.String Eliminar_Usuario(java.lang.String Nombre_Usuario)

  - **Actualizar_Usuario**
- public java.lang.String Actualizar_Usuario(java.lang.String Nombre_Usuario,java.lang.String email_usuario,java.lang.String password)

- **Validar_Moderador**

  - public Moderadores Validar_Moderador(java.lang.String Nombre_Moderador,java.lang.String Login)

- **Crear_Moderador**

  - public java.lang.String Crear_Moderador(java.lang.String Nombre_Moderador,java.lang.String email_moderador,java.lang.String login_moderador)

- **Eliminar_Moderador**

  public java.lang.String Eliminar_Moderador(java.lang.String Nombre_Moderador)

- **Actualizar_Moderador**

  - public java.lang.String Actualizar_Moderador(java.lang.String Nombre_Moderador,java.lang.String email_moderador, java.lang.String login_moderador)

Apéndice B – Documentación de Cliente.

Servlets

Class EliminarUsuario

- java.lang.Object
  - javax.servlet.GenericServlet
    -
      - javax.servlet.http.HttpServlet
        - servlets.EliminarUsuario

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="EliminarUsuario",
            urlPatterns="/EliminarUsuario")
public class EliminarUsuario
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
EliminarUsuario()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.

protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet

destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EliminarUsuario

```
public EliminarUsuario()
```

Method Detail

- processRequest
- protected void processRequest(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Processes requests for both HTTP GET and POST methods.

Parameters:

request - servlet request

response - servlet response

Throws:

javax.servlet.ServletException - if a servlet-specific error occurs

java.io.IOException - if an I/O error occurs

- doGet
- protected void doGet(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Handles the HTTP GET method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- `doPost`
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

getServletInfo

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SActualizarLista

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SActualizarLista

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SActualizarLista",
            urlPatterns="/SActualizarLista")
public class SActualizarLista
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SActualizarLista()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet

destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SActualizarLista
public SActualizarLista()

Method Detail

- processRequest
- protected void processRequest(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Processes requests for both HTTP `GET` and `POST` methods.

Parameters:
`request` - servlet request
`response` - servlet response
Throws:
`javax.servlet.ServletException` - if a servlet-specific error occurs
`java.io.IOException` - if an I/O error occurs

- *doGet*
- protected void doGet(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Handles the HTTP `GET` method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- `doPost`
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

`getServletInfo`

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SActualizarModerador

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SActualizarModerador

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SActualizarModerador",
            urlPatterns="/SActualizarModerador")
public class SActualizarModerador
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SActualizarModerador()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet

destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **SActualizarModerador**

public SActualizarModerador()

Method Detail

- processRequest
- protected void processRequest(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Processes requests for both HTTP GET and POST methods.

Parameters:

request - servlet request

response - servlet response

Throws:

javax.servlet.ServletException - if a servlet-specific error occurs

java.io.IOException - if an I/O error occurs

- **doGet**
- protected void doGet(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Handles the HTTP `GET` method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- `doPost`
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **`getServletInfo`**

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SActualizarUsuario

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SActualizarUsuario

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SActualizarUsuario",
            urlPatterns="/SActualizarUsuario")
public class SActualizarUsuario
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SActualizarUsuario()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

`doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service`

Methods inherited from class javax.servlet.GenericServlet

`destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

SActualizarUsuario
`public SActualizarUsuario()`

Method Detail

- `processRequest`
- `protected void processRequest(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Processes requests for both HTTP `GET` and `POST` methods.

Parameters:
`request` - servlet request
`response` - servlet response
Throws:
`javax.servlet.ServletException` - if a servlet-specific error occurs
`java.io.IOException` - if an I/O error occurs

- *doGet*
- `protected void doGet(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException,`

```
                                    java.io.IOException
```

Handles the HTTP `GET` method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- `doPost`
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

`getServletInfo`

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SCrearLista

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SCrearLista

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SCrearLista",
            urlPatterns="/SCrearLista")
public class SCrearLista
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SCrearLista()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet

destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SCrearLista

```
public SCrearLista()
```

Method Detail

- processRequest
- protected void processRequest(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Processes requests for both HTTP `GET` and `POST` methods.

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- doGet
- protected void doGet(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Handles the HTTP `GET` method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- `doPost`
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

`getServletInfo`

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SCrearModerador

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SCrearModerador

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SCrearModerador",
            urlPatterns="/SCrearModerador")
public class SCrearModerador
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SCrearModerador()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet

destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- SCrearModerador

public SCrearModerador()

Method Detail

- processRequest
- protected void processRequest(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Processes requests for both HTTP GET and POST methods.

Parameters:

request - servlet request

response - servlet response

Throws:

javax.servlet.ServletException - if a servlet-specific error occurs

java.io.IOException - if an I/O error occurs

- **doGet**
- protected void doGet(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Handles the HTTP `GET` method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **doPost**
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **getServletInfo**

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SCrearUsuario

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SCrearUsuario

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SCrearUsuario",
            urlPatterns="/SCrearUsuario")
public class SCrearUsuario
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SCrearUsuario()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response)

	Processes requests for both HTTP `GET` and `POST` methods.

Methods inherited from class javax.servlet.http.HttpServlet

`doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service`

Methods inherited from class javax.servlet.GenericServlet

`destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

- **SCrearUsuario**

```
public SCrearUsuario()
```

Method Detail

- processRequest
- protected void processRequest(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Processes requests for both HTTP `GET` and `POST` methods.

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **doGet**
- protected void doGet(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException,

```
                                    java.io.IOException
```

Handles the HTTP `GET` method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **`doPost`**
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **`getServletInfo`**

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SDescargarMsj

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SDescargarMsj

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SDescargarMsj",
            urlPatterns="/SDescargarMsj")
public class SDescargarMsj
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SDescargarMsj()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet

destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **SDescargarMsj**

public SDescargarMsj()

Method Detail

- processRequest
- protected void processRequest(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Processes requests for both HTTP GET and POST methods.

Parameters:

request - servlet request

response - servlet response

Throws:

javax.servlet.ServletException - if a servlet-specific error occurs

java.io.IOException - if an I/O error occurs

- **doGet**
- protected void doGet(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Handles the HTTP GET method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **doPost**
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **getServletInfo**

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SDesuscribirse

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SDesuscribirse

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SDesuscribirse",
            urlPatterns="/SDesuscribirse")
public class SDesuscribirse
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SDesuscribirse()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet

destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **SDesuscribirse**

public SDesuscribirse()

Method Detail

- **processRequest**
- protected void processRequest(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Processes requests for both HTTP GET and POST methods.

Parameters:

request - servlet request

response - servlet response

Throws:

javax.servlet.ServletException - if a servlet-specific error occurs

java.io.IOException - if an I/O error occurs

- **doGet**
- protected void doGet(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException,

```
                                    java.io.IOException
```

Handles the HTTP `GET` method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **`doPost`**
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **`getServletInfo`**

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SEliminarLista

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SEliminarLista

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SEliminarLista",
            urlPatterns="/SEliminarLista")
public class SEliminarLista
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SEliminarLista()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet

destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **SEliminarLista**

public SEliminarLista()

Method Detail

- **processRequest**
- protected void processRequest(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Processes requests for both HTTP GET and POST methods.

Parameters:

request - servlet request

response - servlet response

Throws:

javax.servlet.ServletException - if a servlet-specific error occurs

java.io.IOException - if an I/O error occurs

- **doGet**
- protected void doGet(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Handles the HTTP `GET` method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **doPost**
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **getServletInfo**

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SEliminarModerador

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SEliminarModerador

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SEliminarModerador",
            urlPatterns="/SEliminarModerador")
public class SEliminarModerador
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SEliminarModerador()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

`doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service`

Methods inherited from class javax.servlet.GenericServlet

`destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

- **SEliminarModerador**

```
public SEliminarModerador()
```

Method Detail

- **processRequest**
- `protected void processRequest(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Processes requests for both HTTP `GET` and `POST` methods.

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **doGet**
- `protected void doGet(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `GET` method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **doPost**
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **getServletInfo**

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SMensajeALista

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SMensajeALista

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SMensajeALista",
            urlPatterns="/SMensajeALista")
public class SMensajeALista
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SMensajeALista()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet

destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- **SMensajeALista**

public SMensajeALista()

Method Detail

- **processRequest**
- protected void processRequest(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Processes requests for both HTTP GET and POST methods.

Parameters:

request - servlet request

response - servlet response

Throws:

javax.servlet.ServletException - if a servlet-specific error occurs

java.io.IOException - if an I/O error occurs

- **doGet**
- protected void doGet(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException, java.io.IOException

Handles the HTTP `GET` method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **doPost**
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **getServletInfo**

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description

Class SSuscribirse

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SSuscribirse

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SSuscribirse",
            urlPatterns="/SSuscribirse")
public class SSuscribirse
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SSuscribirse()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response)

	Processes requests for both HTTP `GET` and `POST` methods.

Methods inherited from class javax.servlet.http.HttpServlet

`doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service`

Methods inherited from class javax.servlet.GenericServlet

`destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

SSuscribirse
`public SSuscribirse()`

Method Detail

- **processRequest**
- `protected void processRequest(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Processes requests for both HTTP `GET` and `POST` methods.

Parameters:
`request` - servlet request
`response` - servlet response
Throws:
`javax.servlet.ServletException` - if a servlet-specific error occurs
`java.io.IOException` - if an I/O error occurs

- **doGet**
- `protected void doGet(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`

- throws javax.servlet.ServletException,
  java.io.IOException

Handles the HTTP GET method.

Overrides:

doGet in class javax.servlet.http.HttpServlet

Parameters:

request - servlet request

response - servlet response

Throws:

javax.servlet.ServletException - if a servlet-specific error occurs

java.io.IOException - if an I/O error occurs

- **doPost**
- protected void doPost(javax.servlet.http.HttpServletRequest request,
- javax.servlet.http.HttpServletResponse response)
- throws javax.servlet.ServletException,
  java.io.IOException

Handles the HTTP POST method.

Overrides:

doPost in class javax.servlet.http.HttpServlet

Parameters:

request - servlet request

response - servlet response

Throws:

javax.servlet.ServletException - if a servlet-specific error occurs

java.io.IOException - if an I/O error occurs

- **getServletInfo**

public java.lang.String getServletInfo()

Returns a short description of the servlet.

Specified by:

getServletInfo in interface javax.servlet.Servlet

Overrides:

getServletInfo in class javax.servlet.GenericServlet

Returns:

a String containing servlet description

Class SValidacionRol

- java.lang.Object
  - javax.servlet.GenericServlet
    - javax.servlet.http.HttpServlet
      - servlets.SValidacionRol

- All Implemented Interfaces:
  java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

```
@WebServlet(name="SValidacionRol",
            urlPatterns="/SValidacionRol")
public class SValidacionRol
extends javax.servlet.http.HttpServlet
```

See Also:
Serialized Form

Constructor Summary

Constructors
Constructor and Description
SValidacionRol()

Method Summary

Methods	
Modifier and Type	**Method and Description**
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP GET method.
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Handles the HTTP POST method.
java.lang.String	**getServletInfo**() Returns a short description of the servlet.
protected void	**processRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Processes requests for both HTTP GET and POST methods.

Methods inherited from class javax.servlet.http.HttpServlet

`doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service`

Methods inherited from class javax.servlet.GenericServlet

`destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletName, init, init, log, log`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

- **SValidacionRol**

```
public SValidacionRol()
```

Method Detail

- **processRequest**
- `protected void processRequest(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Processes requests for both HTTP `GET` and `POST` methods.

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **doGet**
- `protected void doGet(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `GET` method.

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **doPost**
- `protected void doPost(javax.servlet.http.HttpServletRequest request,`
- `javax.servlet.http.HttpServletResponse response)`
- `throws javax.servlet.ServletException, java.io.IOException`

Handles the HTTP `POST` method.

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Parameters:

`request` - servlet request

`response` - servlet response

Throws:

`javax.servlet.ServletException` - if a servlet-specific error occurs

`java.io.IOException` - if an I/O error occurs

- **getServletInfo**

`public java.lang.String getServletInfo()`

Returns a short description of the servlet.

Specified by:

`getServletInfo` in interface `javax.servlet.Servlet`

Overrides:

`getServletInfo` in class `javax.servlet.GenericServlet`

Returns:

a String containing servlet description