# CAPÍTULO 4

# ESTRUCTURA DEL SISTEMA

## 4.1 Diseño del Sistema.

En esta sección se presentará el diseño del sistema que será implementado para la generación de las gramáticas SGRS, mediante una aplicación Web. En la figura 4.1 se muestra un esquema general de la aplicación.

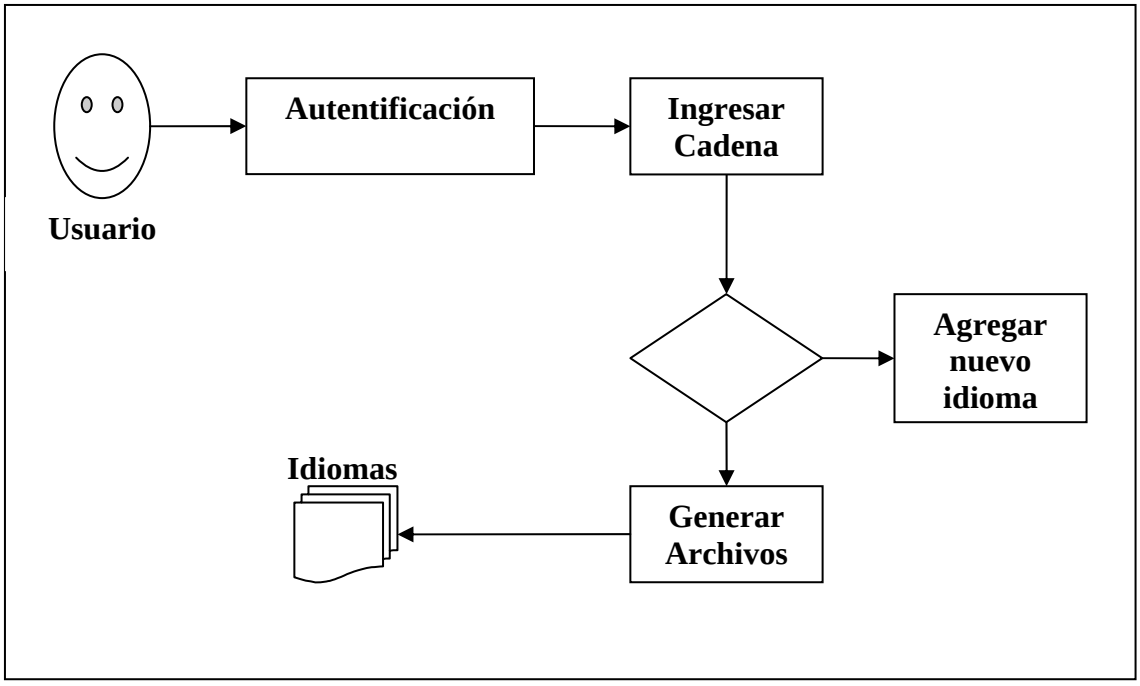


Figura 4.1 Esquema General de la aplicación.

La figura anterior representa de manera muy general el flujo del sistema, ya que, simplemente se quiere asentar el seguimiento de este.

En primer lugar se muestra una primera interfaz (Auntentificación) para ingresar password y id del usuario. Una vez que es validado el usuario, se ingresa a una segunda interfaz (Ingresa Cadena) en donde el usuario podrá ingresar su cadena junto con algunos parámetros y él o los idiomas en que quiere generar su gramática SGRS para la cadena; también en esta interfaz tendrá la opción de poder agregar otro idioma. En caso

de que no quiera agregar un nuevo idioma, se pasara a una tercera interfaz (Generar Archivos) en la que se mostraran los archivos generados por cada uno de los idiomas elegidos. Si se eligiera en la segunda interfaz la opción de agregar un nuevo idioma pasaría a una cuarta interfaz (Agregar nuevo idioma) en donde el usuario podrá visualizar el conjunto de idiomas existentes y podrá agregar uno nuevo.

### 4.1.1 Diagramas de Flujo

A continuación se muestra un diagrama de flujo de la aplicación.

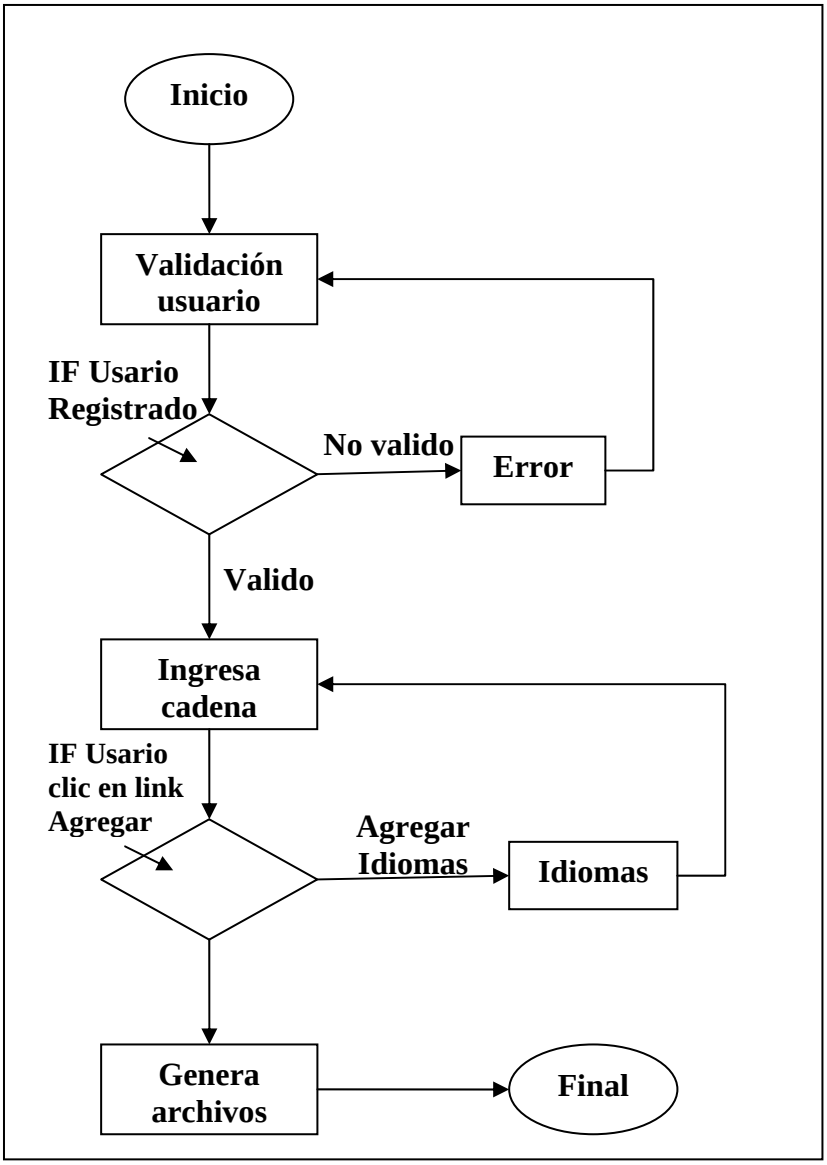


Figura 4.2 Diagrama de flujo del sistema.

A continuación se desglosarán cada uno de los módulos del diagrama de flujo:

***1.- Diagrama del módulo "Validación de usuario"***

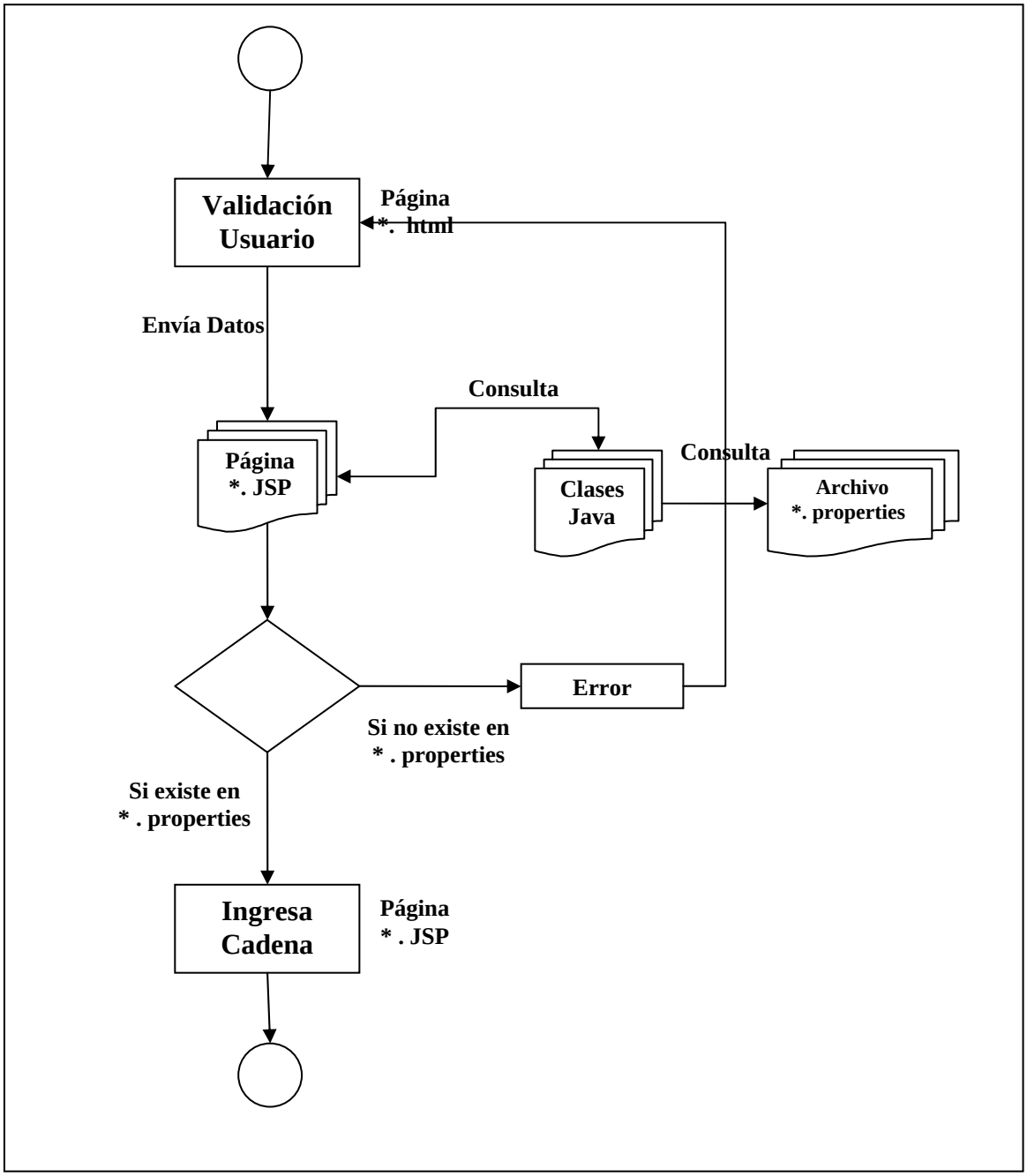


Figura 4.3 Esquema de módulo Validación Usuario.

En la figura 3.4 se representa como es que la interfaz "validación de usuario" será una página html, con la que se le pedirá al usuario ingresar su nombre de usuario y password para poder pasar a la siguiente interfaz (Ingresa datos). Logrando esto a través

de la interacción entre diferentes archivos como: clases java, JSP y archivos con extensión properties.

***2.- Diagrama del módulo "Ingresa Cadena"***

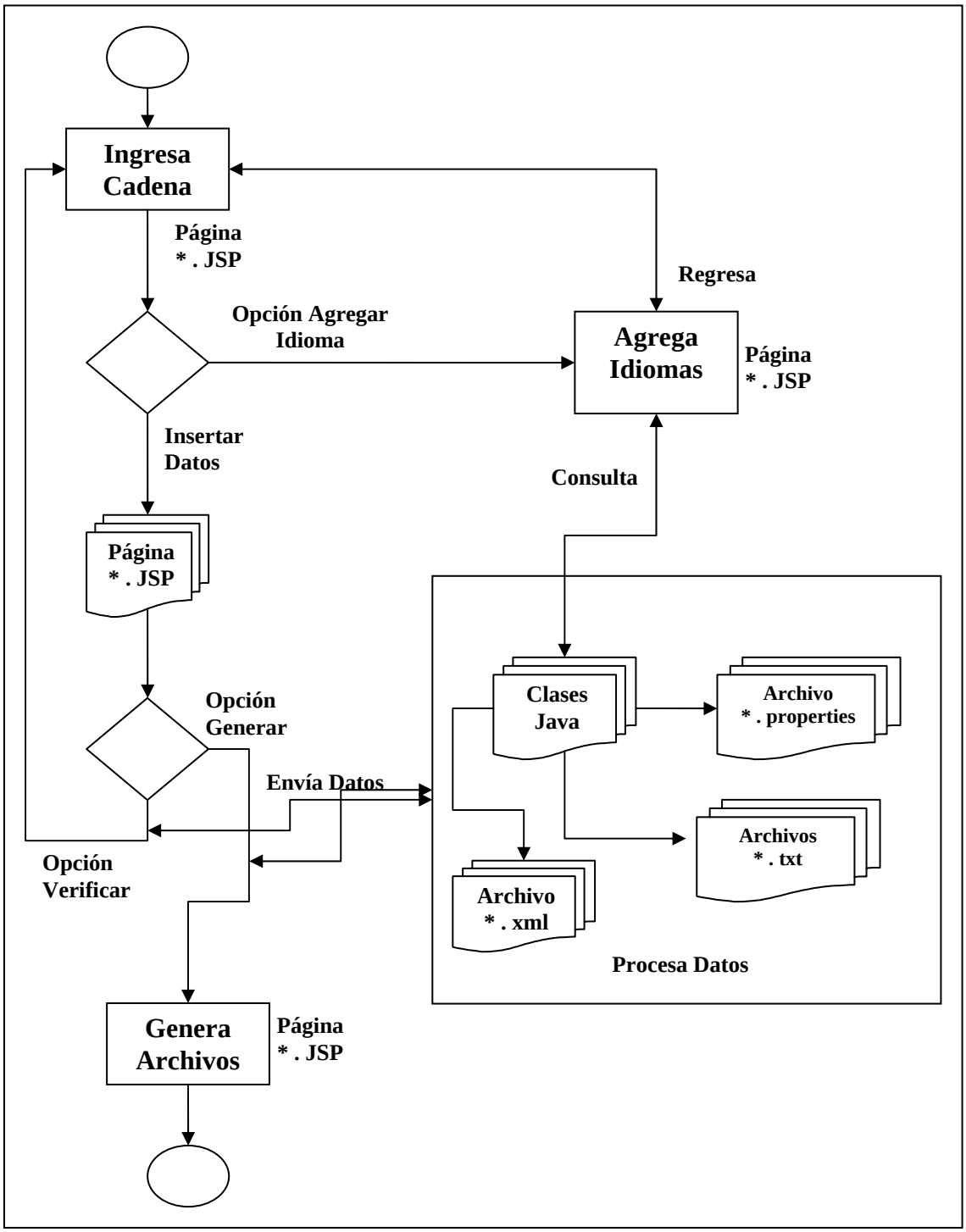


Figura 4.4 Esquema de módulo Ingresa Datos

En el anterior esquema se muestra de manera general como es que la interfaz “Ingresa Datos”, interactúa con los diferentes archivos que componen la aplicación, en este caso interactúa con clases en Java, paginas JSP, un archivo con extensión xml y otro con extensión properties. La forma en que esta interfaz se comunica con cada uno de estos archivos depende de la acción que se este realizando en ella.

***3.- Diagrama del módulo “Agregar Idiomas”.***

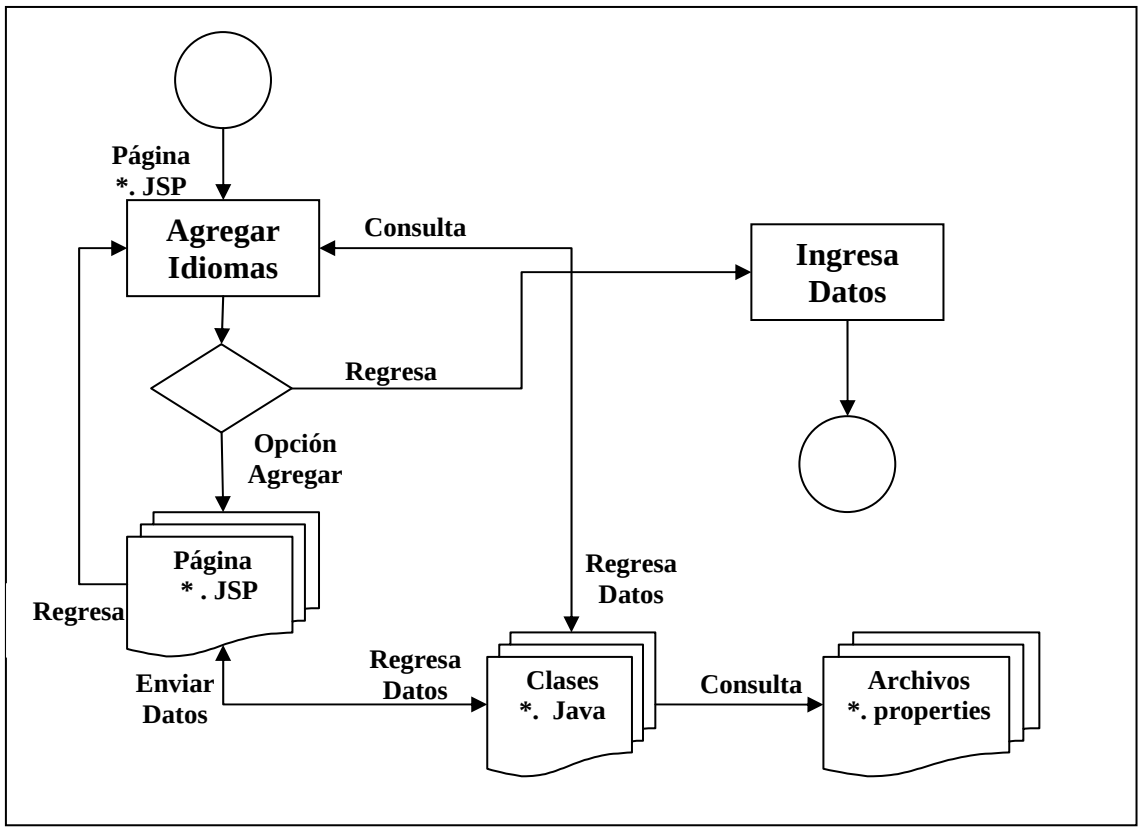


Figura 4.5 Esquema de módulo Agregar Idiomas.

El esquema anterior muestra la comunicación de la interfaz “agregar idiomas” con otros archivos como: JSP, clases de java y archivos con extensión properties.

***4.- Interfaz Generación de Archivo.***

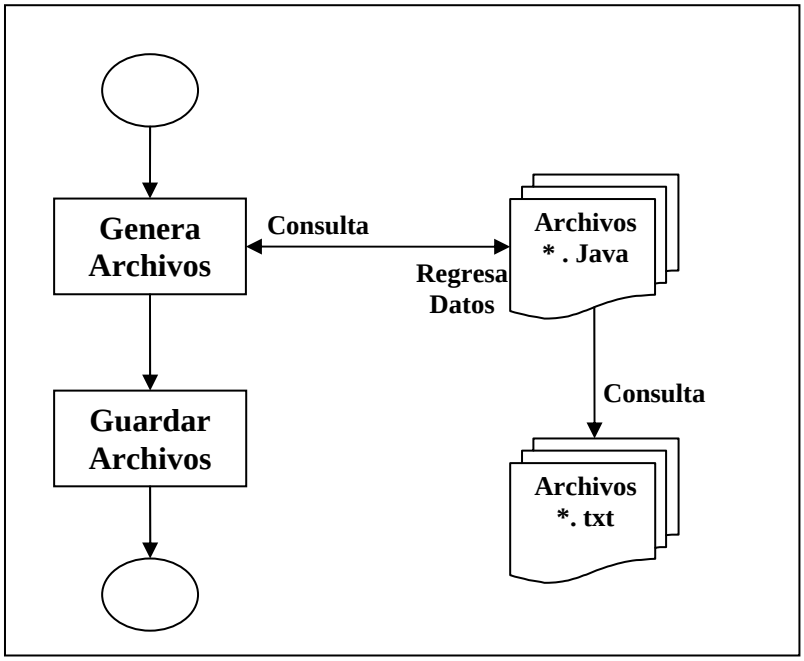


Figura 4.6 Esquema de módulo Genera Archivos.

La interfaz “Genera Archivos” tiene comunicación con clases de java y la opción de regresar a la interfaz anterior “Ingresa Datos”.

Con la presentación de estos esquemas se pretende mostrar la iteración entre cada una de las interfaces de la aplicación, así como su comunicación con las clases java, páginas JSP, archivos con extensión xml y properties, los cuales forman parte de la lógica en la aplicación.

### 4.1.2 Casos de Uso

El diagrama de Casos de Uso que se presenta a continuación muestra la interacción del usuario con cada uno de los procesos del sistema.

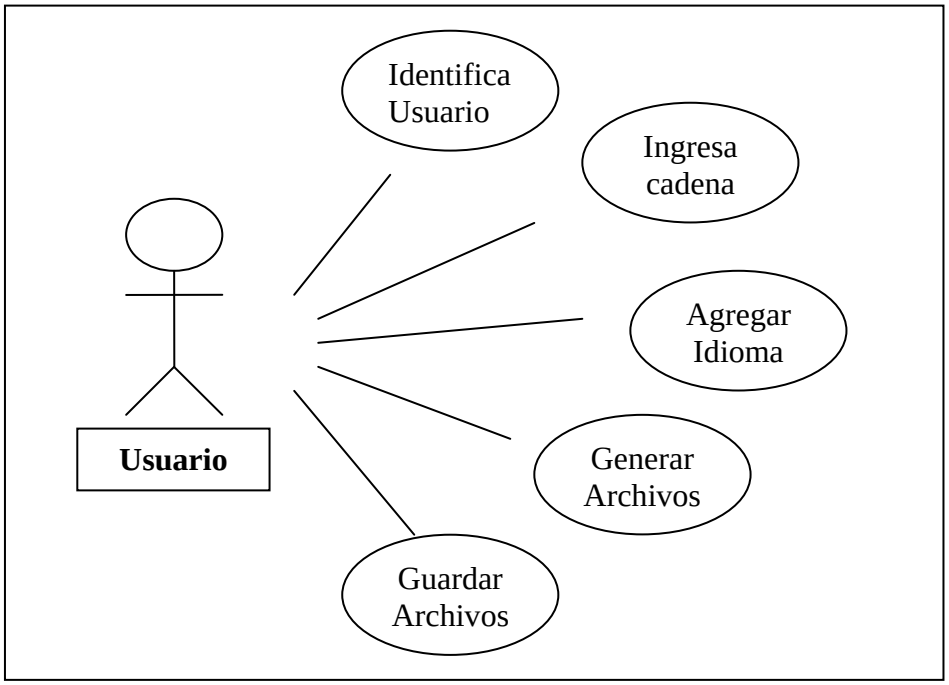


Figura 4.7 Diagrama de caso de uso de la aplicación.

A continuación se presenta un diagrama con la vinculación de cada una de la interfaces con los archivos, clases de java y paginas JSP con los que se apoyan para realizar sus tareas, para mostrar como es la organización de los archivos en el sistema.

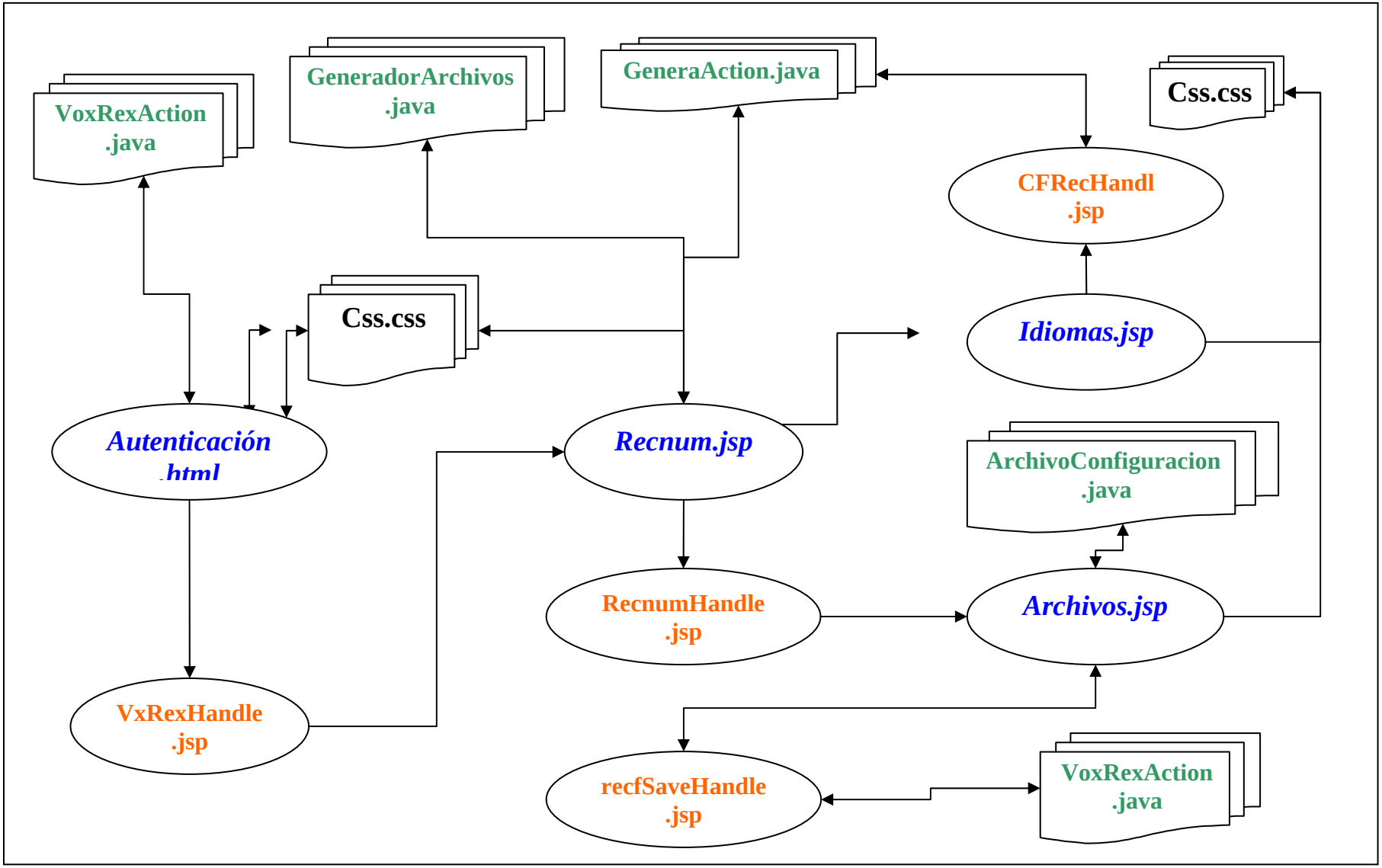


Figura 4.8 Diagrama de la estructura de archivos del sistema.

## 4.2 Procesos de la aplicación.

A continuación se describe la secuencia de pasos para llevar acabo cada uno de los procesos de la aplicación.

### 4.2.1 Proceso de Generación de las Gramáticas

El procedimiento para la generación de la o las gramáticas en la aplicación es:

- La interfaz recnum.jsp recolecta los datos como: cadena, rango, idioma y los envía recnumHandle.jsp.
- El JSP recnumHandle.jsp crea las listas de las combinaciones y los idiomas seleccionados, y las envía a archivos.jsp.
- El JSP archivos.jsp manda las listas a la clase ArchivoConfiguracion para generar cada una de las gramáticas correspondientes a cada idioma.
- La clase ArchivoConfiguración genera las gramáticas, de la siguiente forma:

  **1.-** Inserta encabezado de la gramática.

  **2.-** Inserta Reglas.

  ***1.-*** Crea matriz con las opciones de Combinación-Rango-Número opciones.

  ***2.-*** Crea árbol n-ario para generar todas las combinaciones de opciones posibles.

  ***3.-*** Se crea la matriz de todas las combinaciones de opciones.

  ***4.-*** Se obtiene la parte de texto correspondiente a cada celda de la matriz de combinaciones de opciones de los archivos idioma.txt seleccionados.

  **3.**- Se Inserta pie.

Para ver el diseño de las clases ver apéndice A.

### 4.2.2 Proceso de Agregar Idioma.

El proceso para poder incorporar un archivo idioma.txt nuevo a la aplicación, es el siguiente:

- La página recnum.jsp contiene el vínculo para la interfaz idiomas.jsp, donde se agrega un idioma nuevo.
- La interfaz idiomas.jsp muestra al usuario los idiomas existentes en la aplicación, además de encargarse de pedir al usuario la ruta del archivo idioma.txt.
- La ruta que el usuario introduce para agregar un idioma nuevo, es enviada a cFRecHandl.jsp.
- El JSP cFRecHandl, a través de la clase GeneraAction, recoge el archivo y lo sube a la carpeta contenedora de los archivos idioma.txt.
- La clase GeneraAction, mediante la clase Basexf, consulta el archivo rutaf.properties, para obtener la ruta de la carpeta idiomas.

Ver apéndice A.

### 4.2.3 Proceso de Autentificación

Para el proceso de verificación de usuario del sistema, se utilizó la clase Prorperties para almacenamiento de las contraseñas y los identificadores de usuario, que permite utilizar archivos de propiedades y una de sus principales ventajas es su gran manejo siempre y cuando se respete su sintaxis: clave=valor, además de asegurar un acceso eficiente, puesto que es una subclase de HashTable.

Dentro del sistema el archivo consultado a través de la clase Properties se llama pxxx.properties y la forma de utilizar la sintaxis de la clase en este archivo fue: clave o identificador = valor o contraseña. Un ejemplo de la sintaxis utilizada es: “arturo = agt”, donde, “arturo” toma el lugar del la clave o identificador y “agt” el de valor o contraseña.

En general los pasos para la validación de usuario son:

- Se extraen el id y el password ingresados por el usuario.
- Se envían los datos capturados al JSP VxRexHandle.
- VxRexHandle utiliza el método verificaPx() de la clase voxRexAction.java
- voxRexAction accede al archivo pxxx.properties con el método getPxxxA() de la clase Basex.java.
- Finalmente al ser un usuario valido se envía a la siguiente página JSP recnum.jsp.

Ver apéndice A.

### 4.3 Interfaces del Sistema

Para la creación de estas interfaces se buscó fueran intuitivas, con colores claros para no lastimar la vista al momento de interactuar con ellas, además se buscó que cada una de las interfaces sólo contengan y pidan al usuario de forma clara la información necesaria para que la aplicación pueda realizar la operación que se le pide.

A continuación se presenta el diseño de cada una de las interfaces que conforman este sistema, así como la descripción de cada uno de los elementos de las interfaces.


Figura. 4.9 Diseño Interfaz Autenticación.

En esta interfaz se pide al usuario ingresar su clave y contraseña para poder ingresar al siguiente interfaz cuando el usuario oprime el botón “Ingresar” (véase figura 3.28).


Figura. 4.10 Diseño Interfaz Recnum.

Esta interfaz es la principal de la aplicación, donde como primer elemento se encuentra el espacio para que el usuario introduzca la cadena y oprima el botón de analizar que verifica si es correcta la cadena, para desplegar las combinaciones existentes en la cadena y su rango a elegir, además, se presentan los idiomas existentes en el sistema y un vínculo llamado “Agregar otro idioma”, el cual despliega una nueva ventana para agregar un idioma. (véase fig. 3.29). En el momento en que el usuario hace clic en el botón “Generar Gramáticas” se muestra el resultado en una nueva ventana.

(véase fig. 3.31) . Por ultimo se le presenta al usuario un vínculo llamado “Ayuda”, el cual presenta una ventada con la explicación de cómo utilizar la aplicación. (véase fig. 3.30).

.


Figura. 4.11 Diseño Interfaz Idiomas.

Los elementos de esta interfaz son la presentación de los idiomas existentes y dos botones, Examinar y Agregar, el primero despliega un explorador de archivos para que el usuario elija la ruta del archivo idioma que desea agregar, el segundo realiza la tarea de cargarlo a la carpeta contenedora de los archivos idioma.


Figura. 4.12 Diseño Página de Ayuda.

Esta página es la encargada de presentar al usuario la ayuda correspondiente a cada parte del sistema.


Figura. 4.13 Diseño Interfaz Archivos.

En esta interfaz se le presenta al usuario el resultado de su petición, a través de una serie de ventanas que contiene la gramática generada por idioma, además, de presentarle los botones “Guardar” y “Examinar” para que pueda salvar en su computadora la gramática que necesite.

## 4.4 Tecnología utilizada.

### 4.4.1 La tecnología Java Server Pages (JSP)

La tecnología Java Server Pages permite a los desarrolladores y a los diseñadores de Web desarrollar rápidamente y mantener fácilmente páginas dinámicas, ricas en información como lo son las que soportan a sistemas de negociación. La tecnología de los JSP separa la interfaz del usuario de la parte lógica del contenido, permitiendo a los diseñadores cambiar a su disposición las plantillas de la interfaz sin alterar el contenido dinámico subyacente [Deister, 2006].

En otras palabras, los JSP permiten la introducción de código para la generación dinámica de HTML dentro de las páginas Web, además se pueden localizar en cualquier parte del servidor y en cualquier servidor siempre y cuando exista una maquina virtual de java para él. También como ya se había mencionado, permiten crear el diseño de la página Web o interfaz separado de la parte lógica.

Dentro de la lógica que ayuda a los JSP a crear el contenido dinámico, existen varias herramientas como lo son: El uso de Java Beans, objetos de JDBC y objetos de RMI. Otra de las grandes ventajas que nos ofrecen los JSP es la compilación automática a través de un simple reload o refresh en el navegador, esto es, el JSP se carga según se vayan haciendo cambios en él sin tener la necesidad de recargar toda la lógica de la aplicación; los JSP cada vez que reciben una petición no comienzan a ejecutarse, sino que persisten de una petición a otra, de forma que no se pierde tiempo en invocarlos y esto da como resultado hacer una serie de cosas más eficiente como: conexiones a bases

de datos o bien el manejo de sesiones. En general su tarea es saber como procesar una solicitud para crear una respuesta.

Es por lo anterior que los JSP serán una tecnología que formara parte importante de este proyecto.

### 4.4.2 Java Beans

En los JSP existen diferentes acciones, una de ellas son los Java Beans. Los Java Beans suelen emplearse como parte de un nivel intermedio que tiene como misión dividir la lógica de presentación de la lógica de acceso a los datos, esto es, los Java Beans serán los encargados de llevar la lógica que accede directamente a los datos y no las páginas JSP.

En otras palabras los Java Beans permiten encapsular diversos valores en una clase Java para que su contenido sea manipulable fácilmente en los JSP. [JavaBeans,2007]

### 4.4.3 Javascript

Javascript es un lenguaje de programación flexible y escaso de reglas por lo que se le considera un lenguaje script o bien un lenguaje de archivos y comandos basado en objetos, que se utiliza para ampliar la capacidad funcional de las paginas HTML. Este lenguaje o script no tiene ninguna dependencia funcional bajo ninguna plataforma y sólo está vinculado al navegador que lo interpreta, es por esta razón que se pueden encontrar diferencias entre los distintos navegadores. [Murray y Pappas, 1998]

### 4.4.4 Tomcat.

Tomcat es un servidor Web que brinda el soporte de servlets y JSP el cual fue hecho en java y por lo tanto funciona bajo cualquier sistema operativo que disponga de una máquina virtual Java. Este servidor Web incluye el compilador Jasper encargado de llevar acabo la compilación de los JSP.

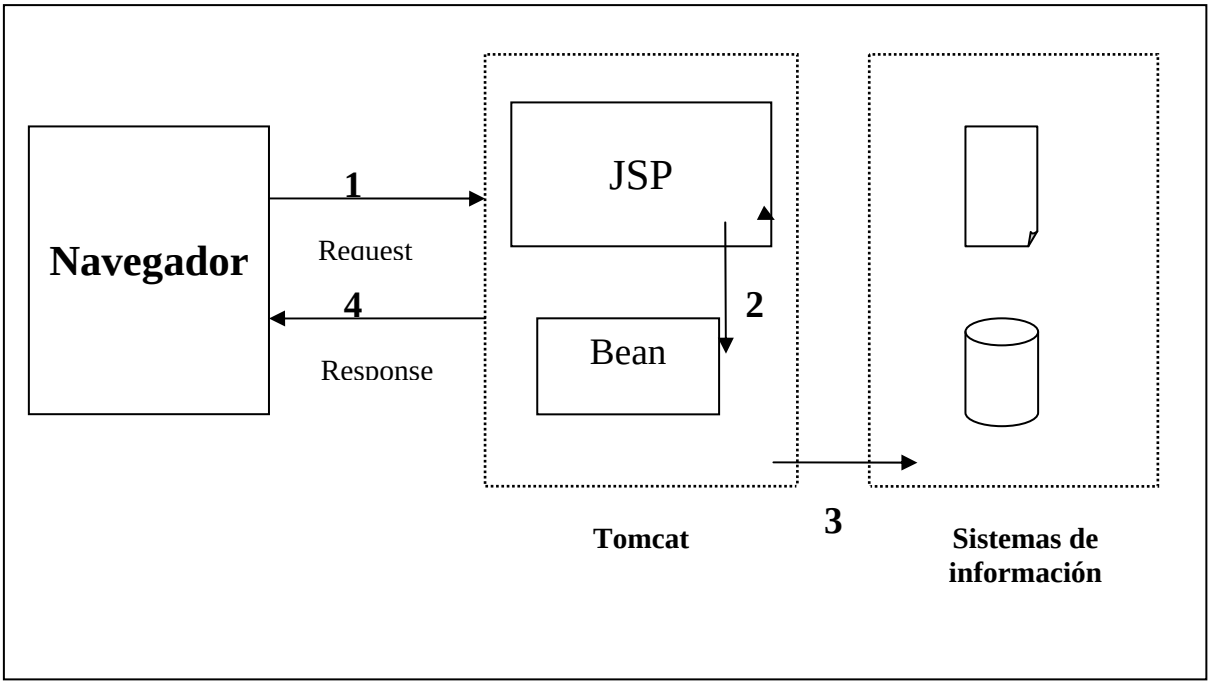


Figura 4.14 Modelo de arquitectura JSP.

## 4.5 Software Utilizado en el Proyecto.

El software y Hardware con el que se cuenta para la realización del proyecto de tesis es:

- Computadora portátil con un procesador AMD Duron a 1.20GHz.
- JDK 1.5 para crear aplicaciones J2SE 5. JDK.

Java Development Kit, es decir, JDK 1.5 es el conjunto de herramientas, utilidades, documentación y ejemplos para desarrollar aplicaciones Java.

- Servidor de web Jakarta Tomcat 5.5.

  Servidor de aplicaciones Web con soporte de servlets y JSPs. Funciona como un contenedor de servlets e implementa las especificaciones de los servlets y JSPs

- El ambiente de desarrollo Eclipse.

  IDE utilizado para el desarrollo del código fuente de la aplicación Web.

- Microsoft Visio.

  Para le desarrollo de los diagramas.

- Dreamwever

  Para facilitar el desarrollo de la interfaz gráfica web.

- Open Speech Recognizer SDK

  Software que se usará para la realización de las pruebas de las gramáticas generadas por la aplicación web.