Capítulo IV. Diseño del sistema.

Este capítulo profundizará sobre la ingeniería de software necesaria para llevar a cabo la implementación del sistema. Por medio de UML podremos analizar de forma gráfica las clases que compondrán el sistema, la relación que existe entre ellas y la forma en que se comunican.

4.1 Arquitectura.

Como se mencionó en el capítulo anterior, la arquitectura que se seguirá para la implementación del sistema será *Model-View-Controller*, la cual permite clasificar todos y cada uno de los componentes del sistema en tres grupos según su función. A continuación se describe cada uno de los tres grupos así como los componentes que los conforman y una breve descripción de la función de cada uno de dichos componentes.

4.1.1 Modelo.

El modelo es el grupo que está compuesto por los datos del sistema, ya que es en este grupo en donde se encuentran las posibles operaciones o transformaciones que se deseen realizar a una o más imágenes.

Este grupo esta constituido por varias clases cuya función es la de recibir una o varias imágenes de entrada y aplicarle operadores para su transformación como son el caso de filtros, operadores aritméticos, métodos de binarisación o algoritmos de detección de

movimiento. Las clases que componen el grupo de modelo son: *TransIndiv, Bordes, Susan, Distancias, TransMult, AlgMov1* y *AlgMov2.*

Por un lado, las clases *TransIndiv, Bordes y Susan* constan de operadores unitarios, es decir, que necesitan de una sola imagen para poder ser aplicables. La clase *TransIndiv* contiene a los operadores de conversión de imágenes de color a escala de grises, invertir la intensidad de color en una imagen, el operador umbral para convertir una imagen en blanco y negro y un filtro. Por su parte la clase *Bordes* cuenta con tres métodos de extracción de bordes los cuales son: el operador de Sobel, el de Roberts y el de Prewitt. Por último, la clase *Susan* cuenta con el método SUSAN de detección de esquinas principales.

Por otro lado, las clases *TransMult, AlgMov1* y *AlgMov2* cuentan con operadores para más de una imagen. La clase *TransMult* cuenta con los operadores aritméticos más importantes en el tratamiento de imágenes los cuales son adición y sustracción de imágenes. *AlgMov1* y *AlgMov2*, son las clases que tienen los algoritmos de detección de movimiento entre imágenes.

4.1.2 Vista.

La vista es el grupo que se encarga de desplegar el resultado de las operaciones al usuario final, así como de las interfaces gráficas de usuario que serán las encargadas de proporcionarle de manera clara al usuario las opciones posibles a realizar.

Este grupo esta conformado por tres páginas Web creadas en HTML y por un servlet. Las páginas despliegan las interfaces de usuario. Estas páginas son *index.html* la cual despliega al usuario los cuatro posibles usos del sistema, los cuales se describen con mayor detalle en éste capítulo en la sección de casos de uso. *UnaImagen.html* la cual es una

forma para dar de alta una sola imagen al servidor para posteriormente poder trabajar con ella. Finalmente, *masImagenes.html* de la misma manera que *unaImagen.html* cuenta con una forma en la cual se pueden subir hasta cinco imágenes.

4.1.3 Control.

El control es el grupo intermediario entre la vista y el modelo, ya que es por medio de éste grupo que se informa al modelo la acción solicitada por un cliente y a la vista que el modelo ya tiene disponible el resultado para ser desplegado.

Este grupo consta de servlets, los cuales son clases escritas en Java que generan documentos HTML dinámicamente, los servlets pueden ser vistos como applets, sin embargo, no necesitan de una interfaz, corren sobre un servidor y al ser clases de Java cuentan con todas las ventajas que el lenguaje provee [Cevallos, 2002].

Los servlet que componen este grupo son: *CargaImagen* e *ImagenesDisponibles.* El primero es llamado cuando el usuario desea cargar ya sea una o varias imágenes al servidor, mientras que el segundo es llamado cuando el usuario desea realizar operaciones a imágenes que ya han sido cargadas en el servidor. Una vez que el cliente elige tanto la imagen a tratar como la operación a realizarle por medio de la vista, ambos servlets se encargan de enviar esta información al modelo.

Por otro lado el servlet *Comprobación* es el encargado de comunicar al modelo los métodos seleccionados por el cliente y generar dinámicamente una página Web la cual despliega el resultado obtenido por los operadores del modelo. Para mayor información sobre el uso del sistema revísese Apéndice A correspondiente a Manual de usuario.

4.2 Formato de imágenes soportadas.

Dentro del ámbito de imágenes digitales, existe una gran variedad de formatos para almacenar dichas imágenes en una computadora, entre las que destacan: bmp, jpg, gif, png, pnm, tiff, entre otras. La principal diferencia entre los distintos formatos, es la forma en que se codifica y almacena la imagen en una computadora.

Como se mencionó en capítulos anteriores, la aplicación será accesible vía Web y los formatos más utilizados para fines Web son JPG (Join Photographic Experts Group) y GIF (Graphics Interchange Format) los cuales utilizan métodos de compresión de datos para hacer más eficiente y veloz la transferencia de las imágenes.

La principal diferencia entre los formatos JPG y GIF es la cantidad de colores que pueden admitir, en el caso del formato GIF puede utilizar hasta 256 colores y es más utilizado en imágenes de menor tamaño, mientras que el formato JPG puede llegar a utilizar más de 16 millones de colores y es más utilizado para imágenes en las que se necesita almacenar el color verdadero de cada píxel.

Debido a lo anterior, los formatos que soporta el sistema son los mismos que soporta la plataforma Java 2. Para el correcto funcionamiento del sistema se probará con los formatos JPG y GIF y los resultados obtenidos por el sistema utilizaran el formato JPG.

4.3 Diseño del sistema.

En este punto se mostrará gráficamente tanto el diseño del sistema, es decir, como será construido, la relación que tienen cada una de las clases así como su funcionamiento. Para

esta tarea se utilizará el lenguaje de modelado unificado UML por sus siglas en inglés Unified Modeling Language.

UML es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprenden el desarrollo de un sistema. UML ofrece una forma de modelar gráficamente conceptos como son las funciones de proyectos de software [Salinas, et al., 2000].

4.3.1 Diagrama de casos de uso.

Por medio de éste diagrama podemos describir el o los procesos principales del sistema, así como las interacciones entre procesos (casos de uso) y sistemas externos o usuarios (actores). El siguiente diagrama muestra los casos de uso para el sistema [Salinas, et al., 2000] [Sommerville, 2002].

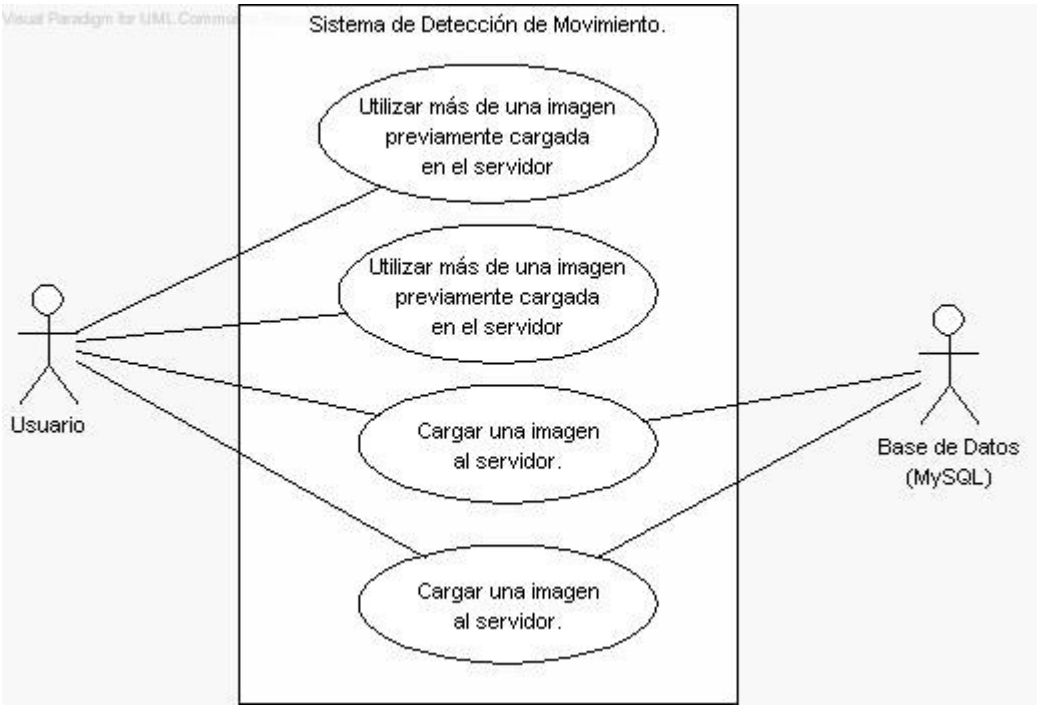

Figura 4.1 Diagrama de casos de uso.

El diagrama anterior nos muestra las cuatro opciones que tiene el usuario para trabajar con imágenes, ya sea que éstas estén cargadas en el servidor o que el usuario las tenga en su computadora y las quiera subir al servidor.

4.3.2 Arquitectura de grupos del sistema.

A continuación se muestra gráficamente la arquitectura de grupos del sistema y la interacción entre cada grupo. Para ver la función específica de cada grupo véase el capítulo III, sin embargo, la función general de cada grupo es:

- Modelo: Contiene los métodos de transformación de las imágenes.
- Vista: Contiene las interfaces gráficas de usuario y despliega resultados.
- Control: Lleva a cabo la comunicación entre el modelo y la vista y viceversa.

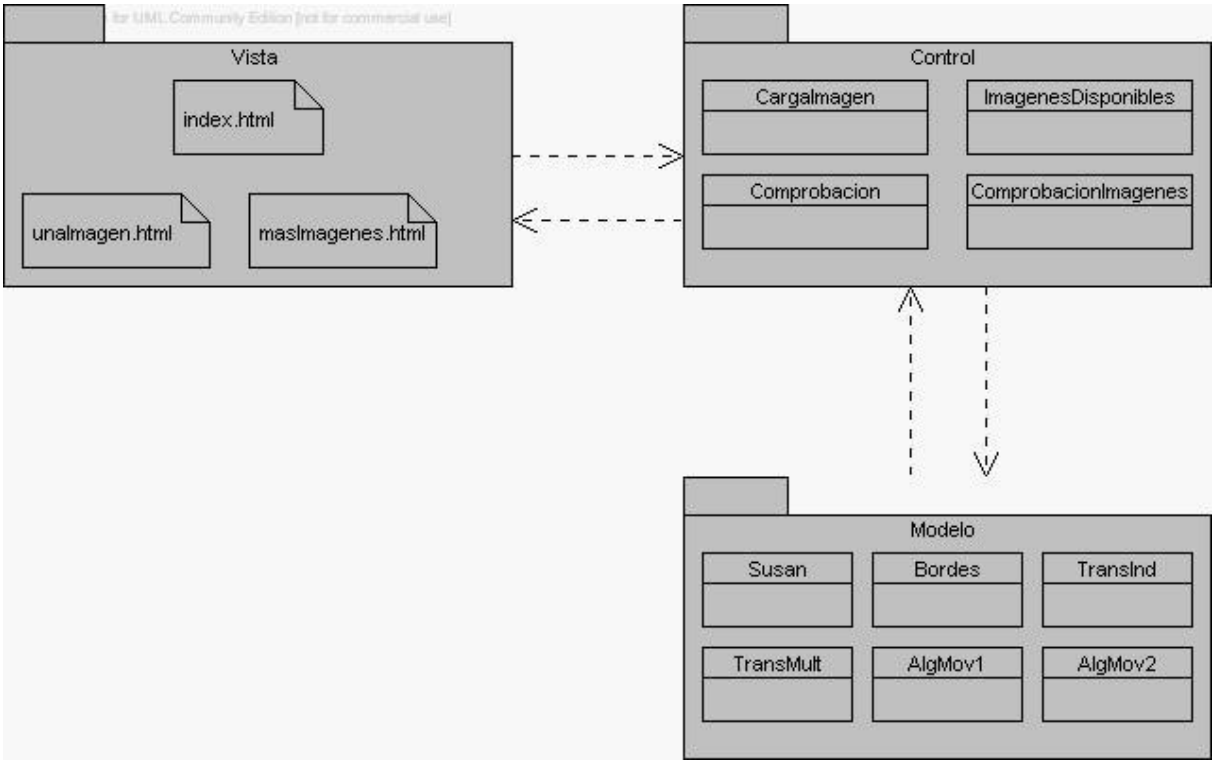

Figura 4.2 Diagrama de arquitectura de grupos.

El diagrama de la figura 4.2 nos muestra de forma gráfica los componentes que conforman cada grupo y la manera general en que se comunican entre ellos. Para ver de forma más específica la comunicación entre los diferentes componentes que forman cada grupo revísese el Apéndice B correspondiente a diagrama de flujo del sistema.

4.3.3 Diagrama de clases.

El diagrama de clases es el diagrama principal dentro de UML, ya que es por medio estos diagramas que se pueden visualizar las clases, esto es, el formato que seguirá cada una de ellas para su implementación. Por medio de éstos diagramas podemos visualizar el contenido de cada clase, ya que los diagramas constan de tres partes como se ilustra en la figura 4.3 [Salinas, et al., 2000] [Sommerville, 2002].

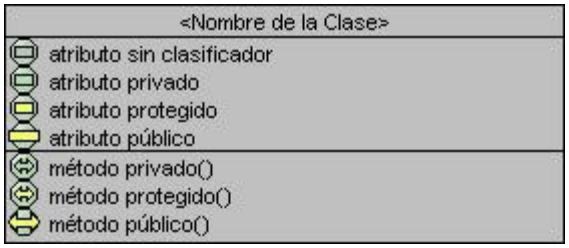

Figura 4.3 Muestra de diagrama de clase.

Dentro del grupo de Control se encuentran las siguientes clases:

- El servlet *CargaImagen* es el encargado de subir imágenes de la máquina de un cliente al servidor, para su futuro tratamiento. Además, obtiene de la base de datos las posibles operaciones de transformación. Una vez cargadas tanto las imágenes como las operaciones envía la petición al servlet *Comprobación.*

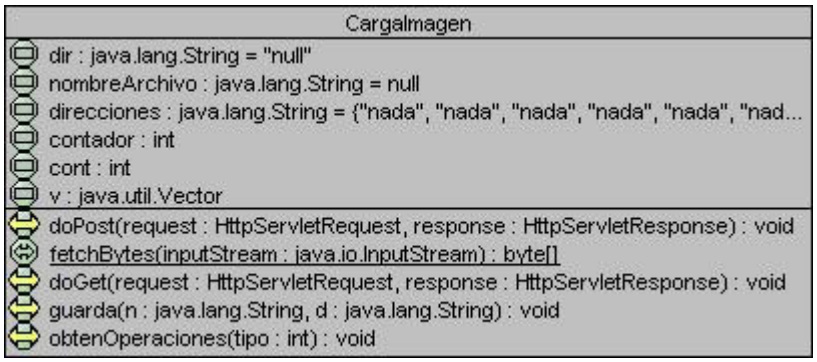

Figura 4.4 Diagrama de clase CargaImagen.

- El servlet *ImagenesDisponibles* presenta al cliente una lista de elección con las imágenes que se encuentran dentro de la base de datos del servidor. Finalmente envía la petición al servlet *ComprobacionImagenes.*

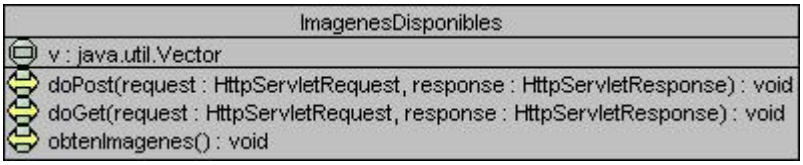

Figura 4.5 Diagrama de clase ImagenesDisponibles.

- El servlet *ComprobacionImagenes* recibe el nombre de las imágenes seleccionadas por el cliente y obtiene tanto la dirección de cada una como los operadores de transformación de imágenes disponibles para dichas imágenes.

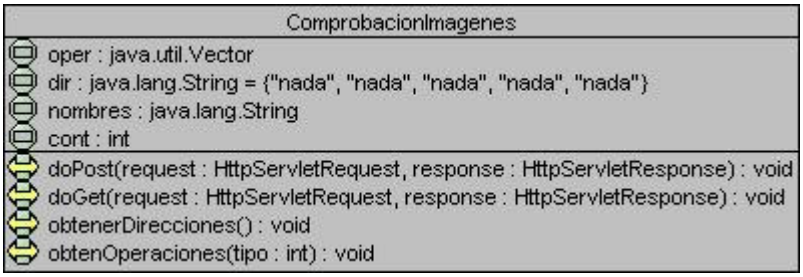

Figura 4.6 Diagrama de clase ComprobacionImagenes.

- El servlet *Comprobación* es el más importante dentro del grupo de control, debido a que es éste servlet el que recibe la petición del grupo de la vista y envía la petición correcta a la clase del modelo que resuelva el problema. Una vez que el modelo resolvió el problema, este servlet genera páginas dinámicas para desplegarle al usuario el resultado obtenido.

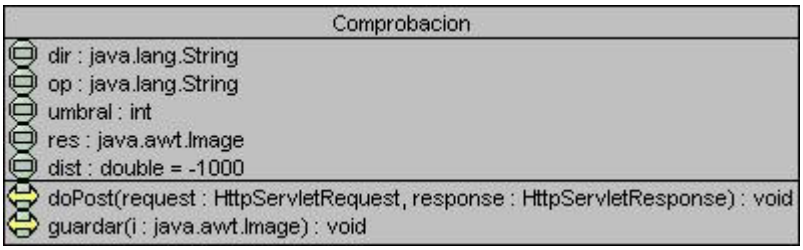

Figura 4.7 Diagrama de clase Comprobacion.

Dentro del grupo del modelo se encuentran las siguientes clases:

- La clase *TransIndiv* contiene operaciones realizables a una sola imagen, como son convertir una imagen de color a escala de grises, hacer binaria una imagen, invertir la intensidad de gris y convertir una matriz de intensidades a una imagen en formato JPG.

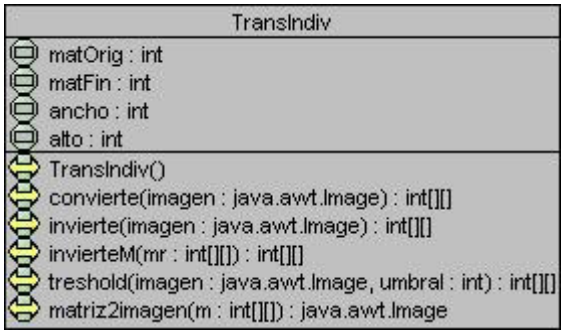

Figura 4.8 Diagrama de clase TransIndiv.

- La clase *TransMult* contiene operadores aritméticos realizables a más de una imagen.

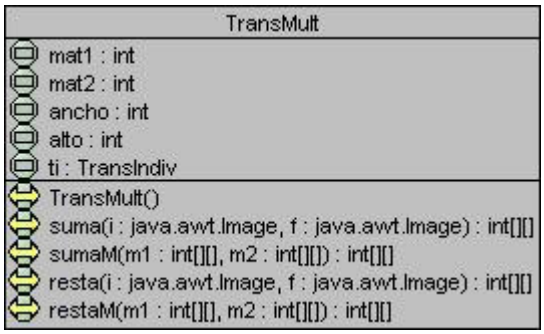

Figura 4.9 Diagrama de clase TransMult.

- La clase *Bordes* contiene las máscaras de Sobel, Prewitt y Roberts para la detección de bordes, además es utilizada por la clase *AlgMov2* para detectar desplazamiento de objetos.

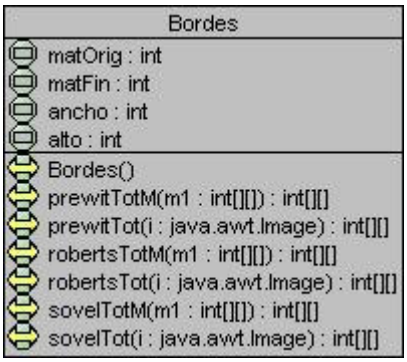

Figura 4.10 Diagrama de clase Bordes.

- La clase *Susan* contiene el método de detección de esquinas principales de una imagen y es utilizado por el algoritmo de detección de movimiento *AlgMov1*.

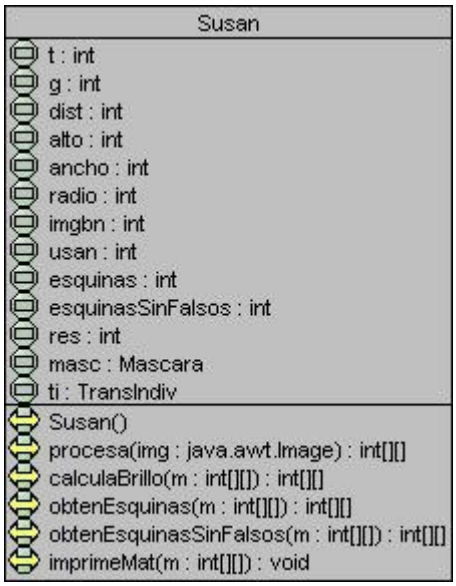

Figura 4.11 Diagrama de clase Susan.

- La clase *GeneraVector* recibe dos objetos de tipo java.awt.Point y una matriz. Su función es encontrar la distancia entre los dos puntos y generar el vector de desplazamiento entre los puntos.

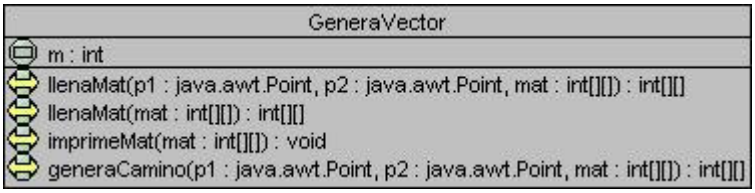

Figura 4.12 Diagrama de clase GeneraVector.

- La clase *AlgMov1* es la encargada de detectar el desplazamiento de un objeto en una secuencia de imágenes. Este algoritmo utiliza el método SUSAN de detección de esquinas principales para la detección de dicho desplazamiento.

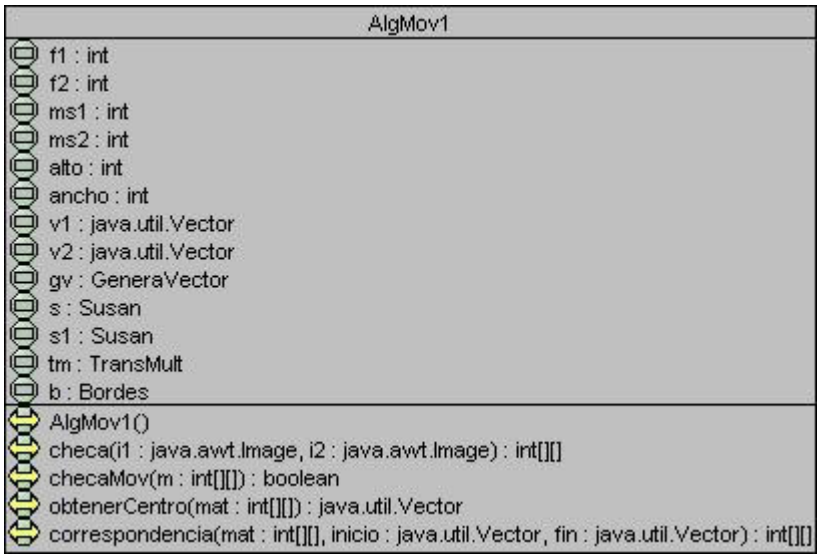

Figura 4.13 Diagrama de clase AlgMov1.

- La clase *AlgMov2* también se encarga de determinar el desplazamiento de un objeto dentro de una secuencia de imágenes. A diferencia de *AlgMov1*, éste método utiliza la detección de bordes del objeto.

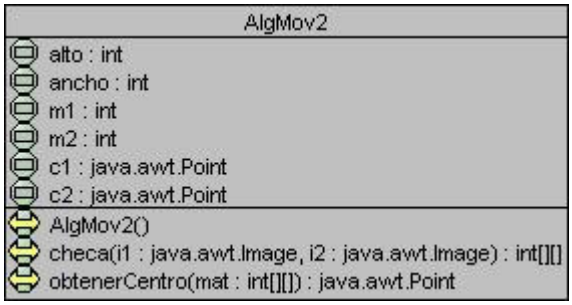

Figura 4.14 Diagrama de clase AlgMov2.

- La clase *Distancias* es utilizada por la clase *AlgMov1*. Su función es almacenar cada esquina del objeto y determinar su distancia al centro geométrico del objeto.

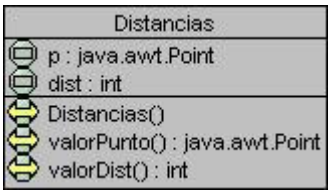

Figura 4.15 Diagrama de clase Distancias.

4.3.4 Diagrama de secuencia.

Los diagramas de secuencia muestran la forma en como un cliente (actor) u objeto (clases) se comunican entre sí en petición a un evento. Lo anteriormente mencionado implica recorrer toda la secuencia de llamadas necesarias para resolver la petición del cliente. Estos diagramas pueden ser obtenidos de dos formas distintas, por medio de los casos de uso o por medio del diagrama de clases. Debido a la complejidad del sistema, este diagrama se descompondrá en dos partes esenciales [Salinas, et al., 2000].

La primera, es obtenida de los casos de uso, ya que el usuario puede escoger entre cuatro posibles usos del sistema. Según el uso seleccionado es la llamada a una clase distinta, pero al final todas las llamadas llegan a la clase *Comprobación* y es esta clase quién hace la solicitud al modelo. El diagrama se muestra a continuación:

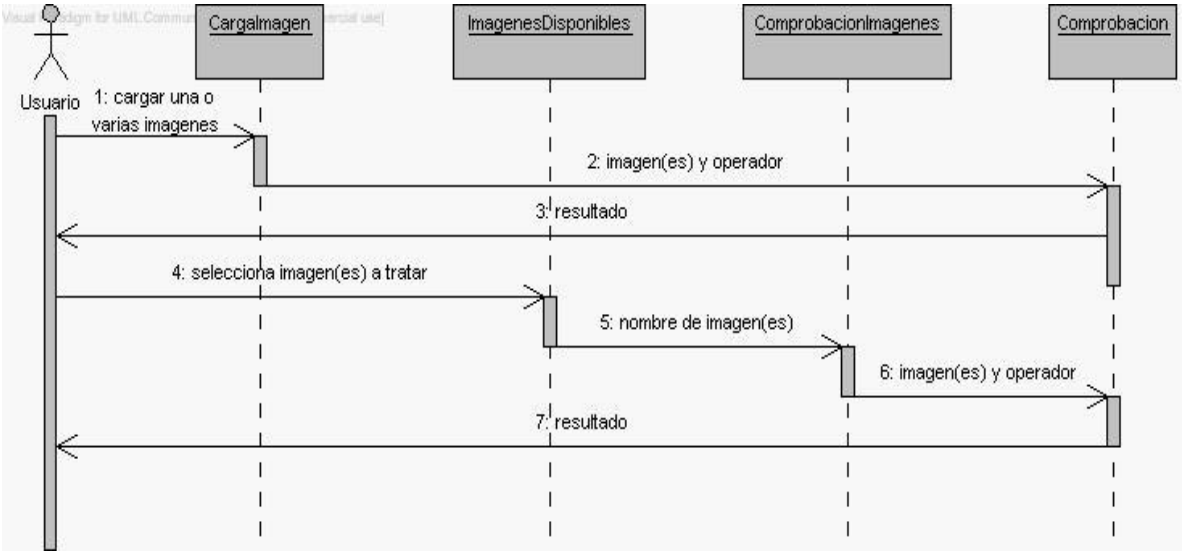

Figura 4.16 Diagrama de secuencia general.

Una vez que la o las imágenes y el operador que se les desea aplicar son elegidos por el usuario, la petición llega a *Comprobación* y ésta última envía los datos a la clase correspondiente en el modelo.

La segunda parte corresponde al problema de estudio de ésta tesis, los algoritmos de movimiento. Ambos algoritmos son llamados por el servlet *Comprobación* y sus diagramas de secuencia se presentan a continuación:

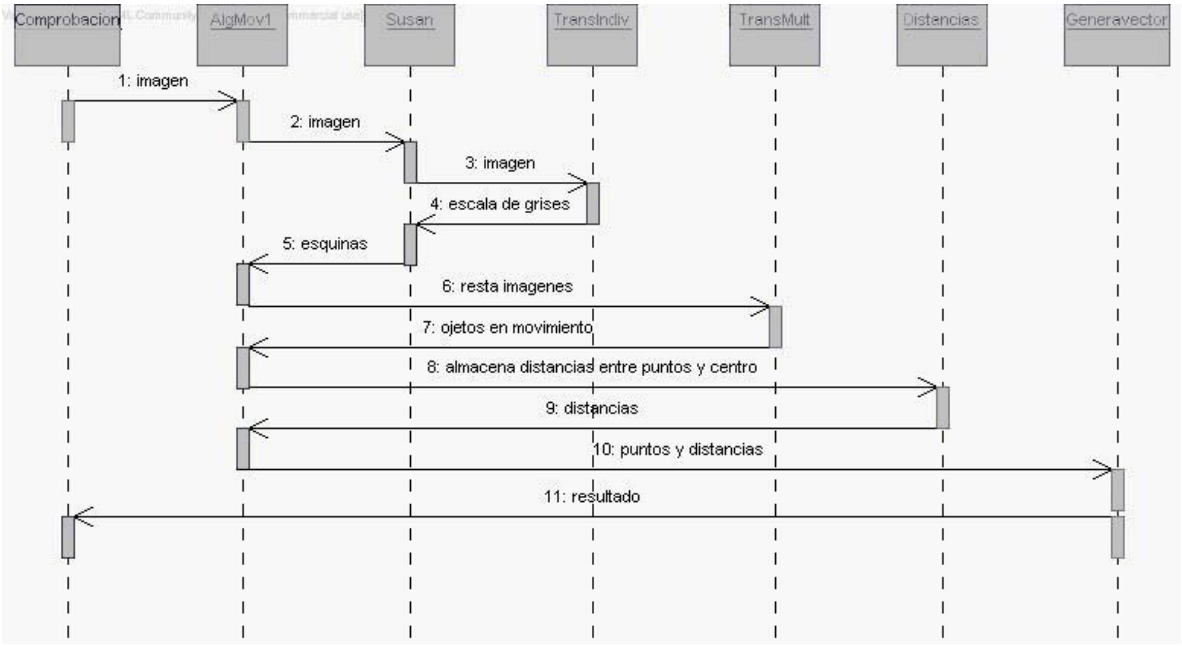

Figura 4.17 Diagrama de secuencia AlgMov1.

El diagrama de secuencia para el algoritmo de movimiento 2 se presenta a continuación:

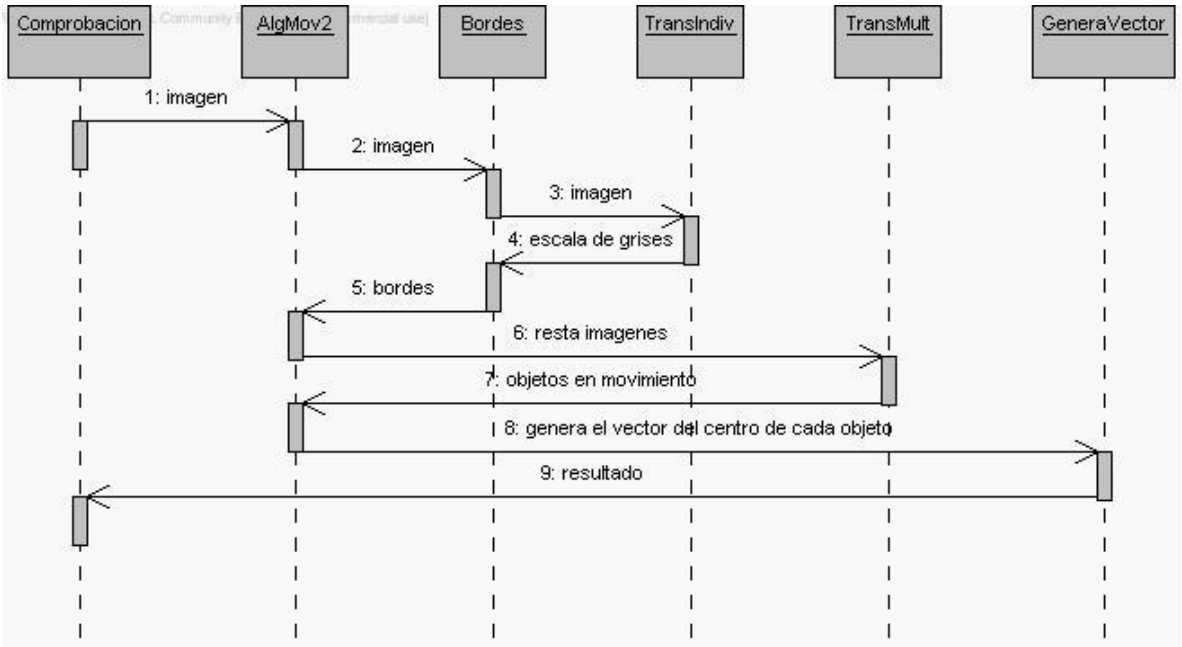

Figura 4.18 Diagrama de secuencia AlgMov2.

Los diagramas de secuencia de cada algoritmo de movimiento muestran la comunicación entre clases necesaria para poder solucionar el problema. Para una descripción más a fondo del funcionamiento de cada clase, revísese el apéndice B.

4.4 Conclusiones.

Este capítulo se dividió en tres partes, la primera corresponde a la arquitectura que seguirá el sistema y la separación de las clases en los tres grupos que componen la arquitectura *Model-View-Controller.* La segunda parte corresponde a los formatos de imágenes que puede procesar el sistema, así como el formato de salida que codificará el programa. Por último, el diseño del sistema.

Por medio de los diagramas UML podemos ver la forma en que estarán constituidas las clases, por medio de los diagramas de clases. Con los diagramas de secuencia podemos ver claramente la forma en que se interrelacionan las clases para responder a una petición del usuario. Los casos de uso nos muestran las posibles opciones que tiene el usuario al acceder al sistema. Finalmente, la arquitectura de grupos nos muestra de forma más general la comunicación entre cada uno de los grupos que compondrán al sistema.