ESPECIFICACIÓN DE ARQUITECTURA DEL SISTEMA

para

Sistema de preprocesamiento y segmentación de células

Modelo 4 + 1

Preparado por:

José Alvarado Chaves, 201129079
Reggie Barker Guillén, 2014050578
Joel Barrantes Garro, 2013120962
Joel Schuster Valverde, 2014096796

Instituto Tecnológico de Costa Rica

3 de junio de 2017

Índice general

1 El Modelo 4 + 1

El modelo 4 + 1 es un modelo diseñado por Philippe Kruchten para describir la arquitectura de sistemas software, basados en el uso de múltiples vistas concurrentes. Las vistas suelen describir el sistema desde la perspectiva de los diferentes interesados, tales como usuarios finales, desarrolladores o directores de proyecto. Las cuatro vistas del modelo son: vista lógica, vista de despliegue, vista de proceso y vista física. Además, una selección de casos de uso o escenarios suele utilizarse para ilustrar la arquitectura sirviendo como una vista más.

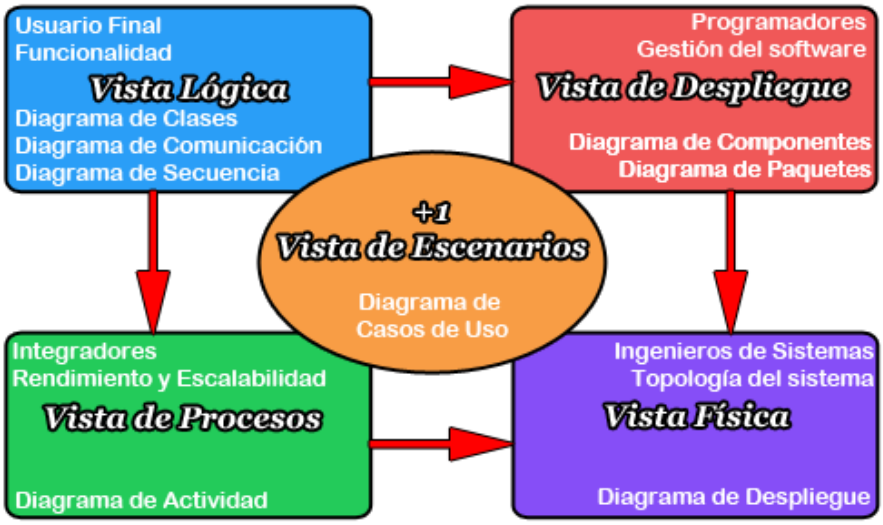

Figura 1.1: Ilustración del modelo 4 + 1 de Kruchten

2 Vista lógica

La vista lógica está enfocada en describir la estructura y funcionalidad del sistema. Los diagramas UML que se suelen utilizar para representar esta vista son los diagramas de clase, de estado, de comunicación y de secuencia.

2.1. Diagrama de clases

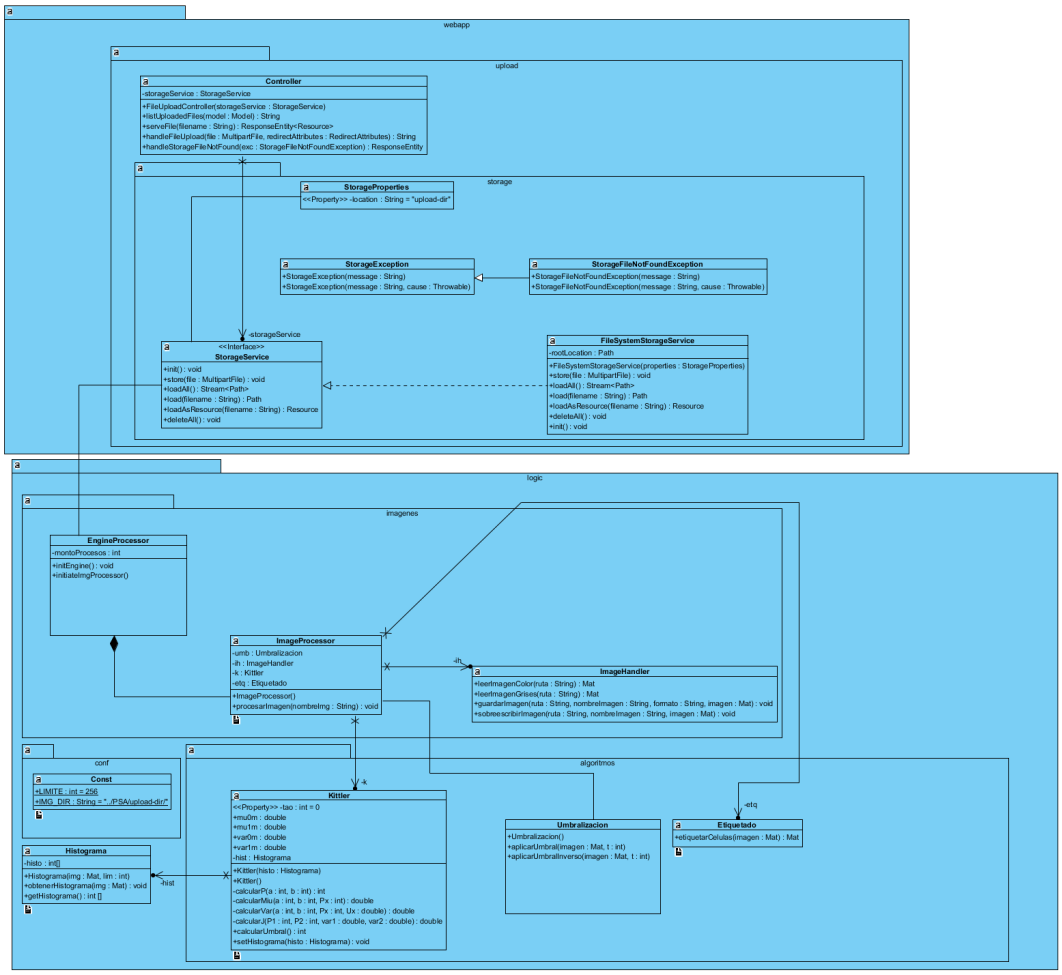

Figura 2.1: Diagrama de clases del sistema.

En el diagrama de clases se pueden observar dos grandes paquetes: *webapp* y *logic*. El paquete *webapp* posee los nodos de *Control* y *Storage*, los cuales utilizan el patron de MVC

para la comunicación entre los entes del sistema. Convenientemente, se dejó el paquete de *logic* por aparte debido a que el paquete de *webapp* se encarga de recibir, almacenar información y responder a las peticiones del cliente. Entonces, *logic* se enfocará en procesar imágenes que tienen un estado de incompleto tomando las imágenes directamente del *storage*. En otras palabras las imágenes vendrán en las peticiones, serán almacenadas, y luego van a ser tomadas por bloques en el nodo *logic*, para procesarlas y guardar los cambios (cada imagen tiene un estado en la base de datos, que determina si una imagen fue o no procesada).

La clase encargada del almacenamiento es *StorageService*, la cual se comunica y hace las peticiones con todos los encargados de almacenamiento o administración de datos. Proporciona funciones de lectura y escritura.

La clase *Controller* es la encargada de recibir peticiones por aprte del cliente y se comunica con el administrador de almacenamiento.

Nótese que *logic* es la capa de procesos del sistema sobre la cual podría aplicarse la paralelización, para que de esta manera el tiempo de procesamiento disminuya de manera considerable.

Para la implementación del sistema, se pretende seguir el patrón de diseño *command*, donde una petición se encapsula como un objeto y puede agregarse a una cola. De este modo, el sistema trata a cada uno de los objetos como una caja negra que retorna un resultado deseado. El patrón se ilustra en la figura 2.2.

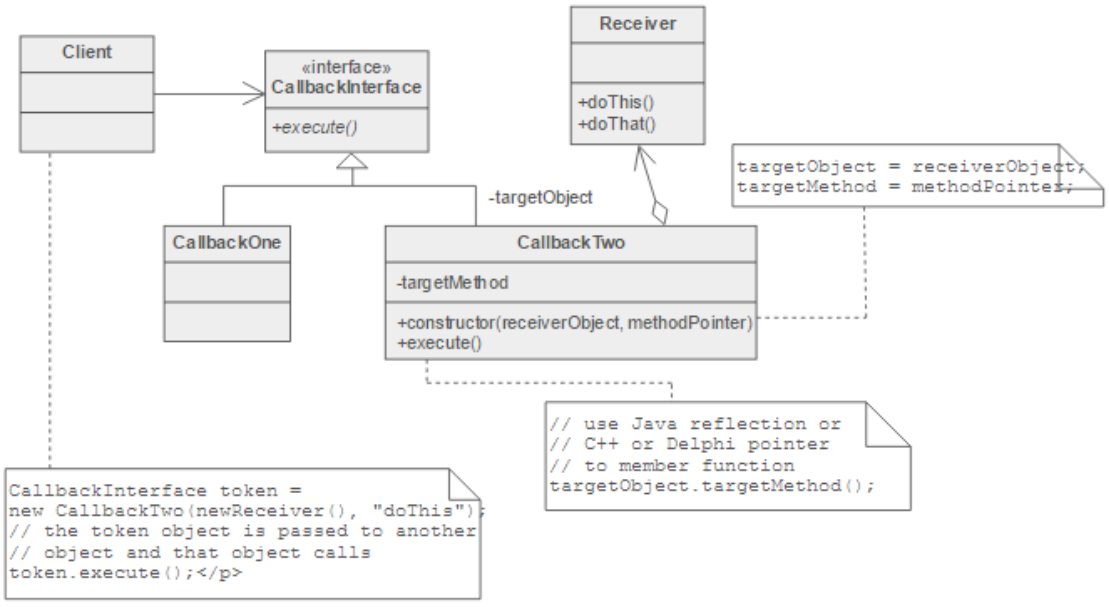

Figura 2.2: Patrón de comportamiento *command*.

2.2. Diagrama de estados

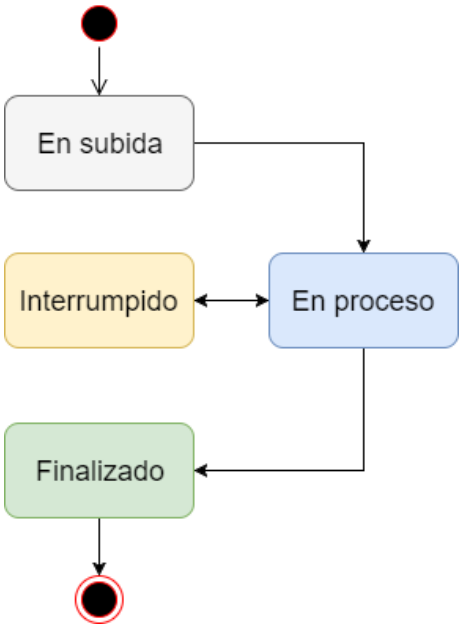

Figura 2.3: Diagrama de estados del procesamiento de un lote.

Este diagrama hace referencia de los estados de cada uno de los elementos subidos por el cliente.

- El estado **en subida** indica que un lote de imágenes está siendo cargado al sistema y que algunas de las imágenes pueden ya estar almacenadas, pero que no se ha completado el proceso.
- El estado **en proceso** indica que el lote de imágenes está siendo procesado por los algoritmos de preprocesamiento y segmentación, imagen por imagen.
- El estado **interrumpido** indica que un lote de imagen que estaba siendo procesado sufrió una interrupción ya sea de usuario o inesperada, es decir que se encuentra sin terminar.
- El estado **finalizado** indica que un lote de imágenes ha sido completamente procesado y puede ser descargado en su totalidad por el usuario.

3 Vista de despliegue

La vista de despliegue ilustra el sistema de la perspectiva del programador y está enfocado en la administración de los artefactos de software. Esta vista también se conoce como vista de implementación. Utiliza el diagrama de componentes UML para describir los componentes de sistema.

3.1. Diagrama de componentes

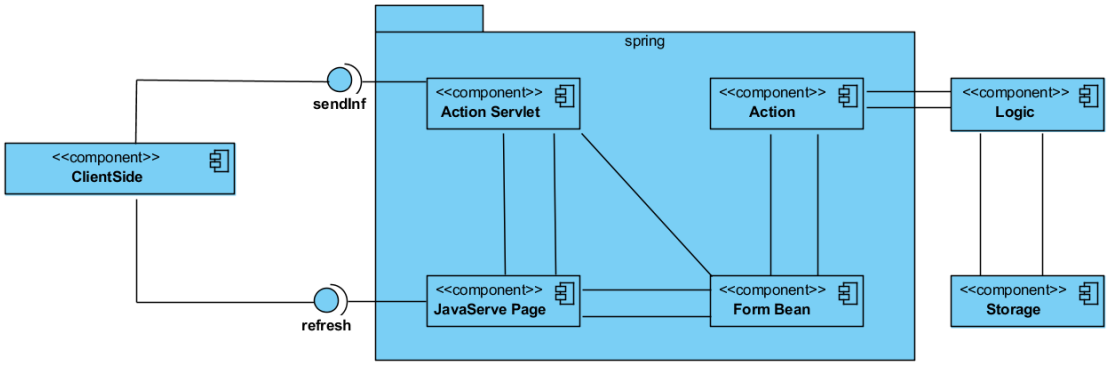

Figura 3.1: Diagrama de componentes

En el diagrama de componentes se denota la estructura a utilizar, el *framework* de Spring, y la intercomunicación entre sus elementos, mostrando a grandes rasgos el uso de patrones de arquitectura y diseño como MVC y *facade* para la estructura del sistema.

- *Client-side:* se actualiza constantemente con el JavaServe Page, mientras que puede enviar peticiones al Action Servlet, que es el encargado de recibir las peticiones.
- *Action Servlet y JavaServe Page:* encargados de las comunicaciones con el cliente.
- *Form bean y Action:* son los elementos intermediarios que determinan las acciones concretas para ejecutar en el área de *logic*. Puede encontrarse una especificación detallada en el diagrama de clases.
- *Área logic:* es la encargada de administración y procesos del sistema de datos.
- *Storage:* sistemas de bases de datos con las que se comunica el sistema.

4 Vista física

La vista física describe el sistema desde el punto de vista de un ingeniero de sistemas. Está relacionada con la topología de componentes de software en la capa física, así como las conexiones físicas entre estos componentes. En UML, se utiliza el diagrama de despliegue para representar esta vista.

4.1. Diagrama de despliegue

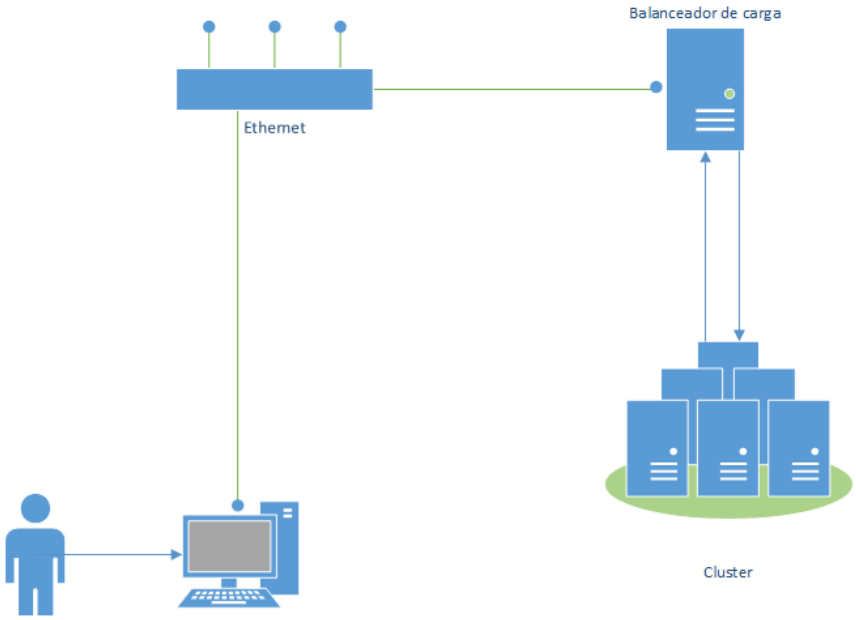

Figura 4.1: Diagrama de despliegue del sistema.

En el diagrama de despliegue muestra un bosquejo de la arquitectura de comunicación del sistema en general. Se le da un enfoque en este diagrama al Ethernet, que representa el enlace de red entre la computadora que usará el usuario y un balanceador de carga que distribuye los trabajos en un clúster. Se propone el uso de una red local, puesto que es más conveniente para el cliente disponer de un complejo computacional potente para correr sus análisis sobre grandes volúmenes de información.

5 Vista de procesos

La vista de procesos trata los aspectos dinámicos del sistema, explica los procesos de sistema y cómo se comunican. se enfoca en el comportamiento del sistema en tiempo de ejecución. La vista considera aspectos de concurrencia, distribución, rendimiento, escalabilidad, entre otros. En UML se utiliza, por ejemplo, el diagrama de actividad para representar esta vista.

5.1. Diagrama de actividad

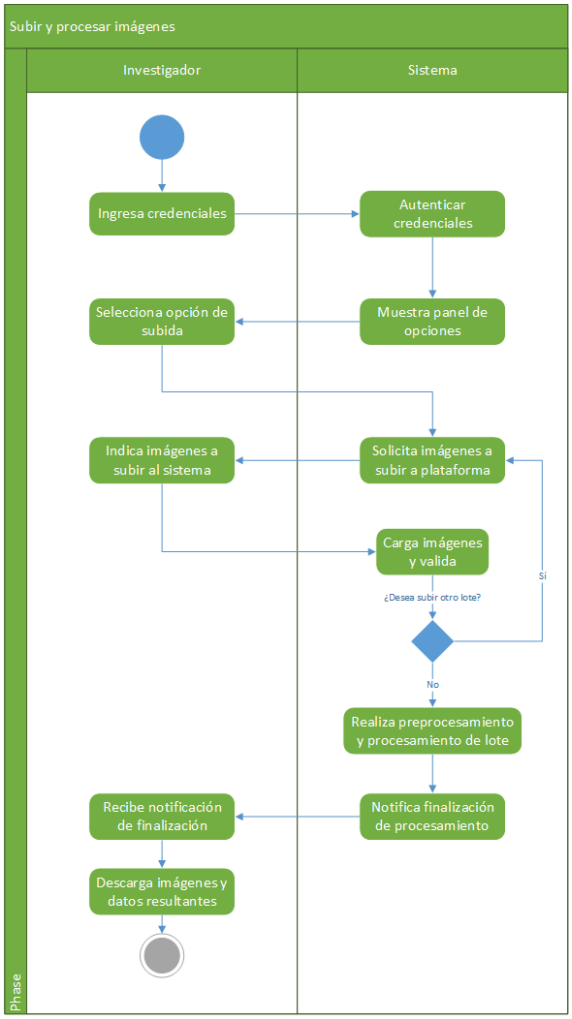

Figura 5.1: Interacción entre el usuario y el sistema.

El diagrama de actividad muestra el proceso básico de actividad entre el sistema y el usuario, para exponer los pasos de interacción entre ambos, la funcionalidad del sistema y cómo se comporta cada ente en dicha interacción. Se muestra además el flujo de mensajes entre ambos y las decisiones que puede tomar el usuario ïnvestigador". Este diagrama muestra el flujo básico para que el usuario pueda iniciar el procesamiento de un lote o varios. Se ha mostrado un flujo sencillo que permita la usabilidad como prioridad para el sistema.

6 Vista de escenarios

La descripción de la arquitectura se ilustra utilizando un conjunto de casos de uso, o escenarios lo que genera una quinta vista. Los escenarios describen secuencias de interacciones entre objetos, y entre procesos. Se utilizan para identificar y validar el diseño de arquitectura. También sirven como punto de partida para pruebas de un prototipo de arquitectura. Esta vista es también conocida como vista de casos de uso.

6.1. Diagrama de casos de uso

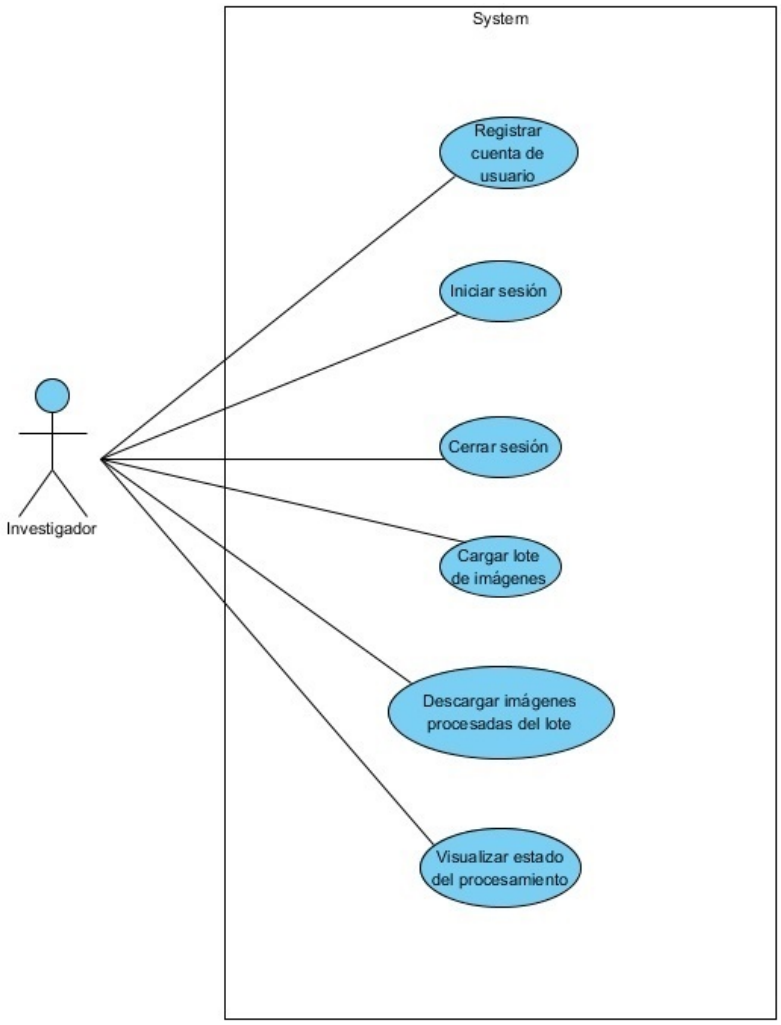

Figura 6.1: Diagrama de casos de uso

7 Vista adicional: datos

Esta vista adicional no forma parte del modelo 4 + 1 por definición, pero se ha incluido por su importancia para el sistema. Un modelo de datos permite describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí. En este apartado, se muestra la arquitectura de entidad-relación y el esquema de datos del sistema. Se maneja un modelo tanto relacional como no relacional.

7.1. Diagrama de Entidad-Relación

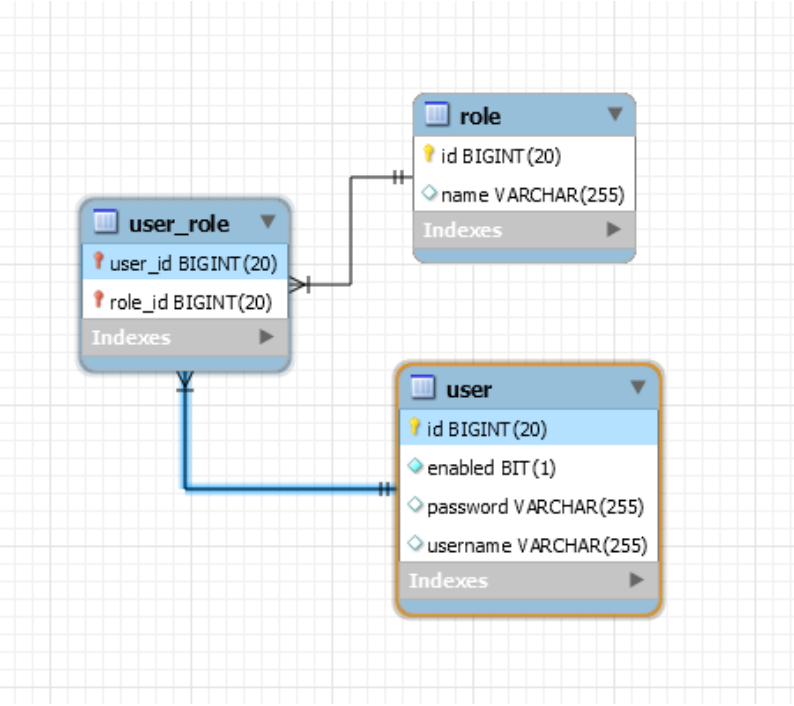

Figura 7.1: Diagrama de Entidad-Relación

El diagrama de entidad relación mostrado muestra la estructura de la base de datos relacional que va a tener el sistema. Para el manejo de sesiones se tiene la tabla login, que contiene un Username y el password, los cuales permiten el acceso a cada uno de los clientes registrados. El lote es la representación conceptual de un conjunto de imágenes, los cuales se hacen en un pedido del usuario. Los mismos tienen una fecha de inicio y además un contador que permite verificar cuánto es el monto de imagenes relacionados a cada uno de ellos.

Es importante añadir que la tabla ïmagenes.[en] la base de datos relacional contiene por cada tupla un identificador, un numero de células (dato que se actualiza cuando la imagen

fue procesada) y el tiempo transcurrido por imagen.

Nota: no se almacenarán en ningún momento del proceso las imágenes en la base de datos relacional, puesto que se tiene las base de datos NoSQL dedicadas al almacenamiento de las imágenes obtenidas.

7.2. Esquema de datos (NoSQL)

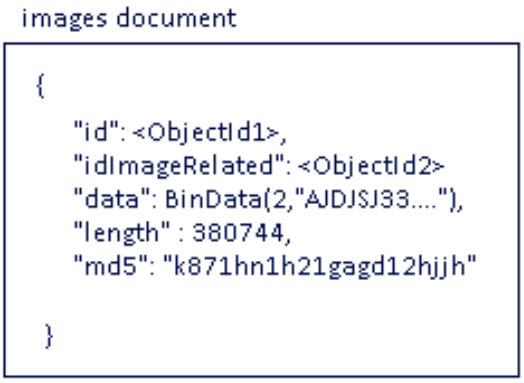

Figura 7.2: Esquema de datos (mongodb)

El esquema de datos es una representación de la base de datos no relacional, mongodb. La cual se va a utilizar para el almacenamiento de las imagenes en el sistema. En el diagrama se muestra las propiedades que contiene este:

- *Id:* es un identificador que tiene cada elemento dentro de la base de datos.
- *IdImageRelated:* es un identificador que esta asociado a un elemento en la base de datos relacional.
- *Data:* Almacena en dato binario la imagen obtenida del usuario.
- *Lenght:* determina el largo del archivo recibido, como medida de seguridad.
- *Md5:* es el cifrado de la imagen en formato Md5.