Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

ASISTENTE PARA EL ETIQUETADO DE OBJETOS PRESENTES EN IMÁGENES

Alumno: Francisco López Ortega

Tutor: Prof. D. FRANCISCO CHARTE OJEDA y
Dña. MARÍA DOLORES PÉREZ GODOY
Dpto: Departamento de Informática

Septiembre, 2020

Universidad de Jaén

Escuela Politécnica Superior de Jaén

Departamento de informática

Don Francisco Charte Ojeda y Doña María Dolores Pérez Godoy, tutores del Proyecto Fin de Grado titulado: **Asistente para el etiquetado de objetos presentes en imágenes**, que presenta Francisco López Ortega, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, septiembre de 2020

El estudiante

Firmado por LOPEZ ORTEGA FRANCISCO - 53914392T el día 02/09/2020

Francisco López Ortega

Los tutores

CHARTE OJEDA FRANCISCO - 25996290B

Firmado digitalmente por CHARTE OJEDA FRANCISCO - 25996290B
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-25996290B, givenName=FRANCISCO, sn=CHARTE OJEDA, cn=CHARTE OJEDA FRANCISCO - 25996290B
Fecha: 2020.09.02 17:36:02 +02'00'

Francisco Charte Ojeda

PEREZ GODOY MARIA DOLORES - 75093149N

Firmado digitalmente por PEREZ GODOY MARIA DOLORES - 75093149N
Fecha: 2020.09.02 18:40:50 +02'00'

María Dolores Pérez Godoy

Índice

Índice de figuras

1. Introducción y motivación

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos inspirados en los criterios y recomendaciones que establece la norma UNE 157801:2007 - "Criterios Generales para la elaboración de proyectos de Sistemas de Información".

A lo largo del documento se utilizarán términos y acrónimos cuya descripción aparecen en el apartado 9 (Definiciones y abreviaturas).

1.1. Introducción

Cumplir con las demandas en las tecnologías de la información es un objetivo vital para cualquier organización. Se requiere de una estrategia de extremo a extremo que garantice el manejo y transformación de estos datos en tiempo real.

Es necesaria una modernización rápida para poder manejar de forma efectiva estas grandes masas de datos. La solución podría estar en la inteligencia artificial. Para ello nos centramos en la característica más importante de la inteligencia artificial, su capacidad de aprender automáticamente. Utilizando técnicas de inteligencia artificial podemos hacer mucho más eficientes y efectivos determinados trabajos que se van a presentar en el día a día.

Entre los diferentes tipos de aprendizaje automático existentes, los dos más conocidos son el supervisado [2] y el no supervisado [7]. Los algoritmos de aprendizaje no supervisado se utilizan cuando únicamente se tienen datos definidos por un conjunto de variables de entrada, sin una salida asociada. El objetivo del aprendizaje no supervisado es modelar la estructura o distribución subyacente en los datos para aprender más sobre los mismos. Se denomina aprendizaje no superviso porque, a diferencia del aprendizaje supervisado, no hay un valor de salida asociado a cada conjunto de valores de entrada, por lo que no es posible generar modelos predictivos. Busca aumentar el conocimiento estructural de los datos disponibles (y posibles datos futuros que provengan del mismo fenómeno),

por ejemplo, conjuntos de datos agrupados según su similitud (*clustering*), reduciendo las dimensiones de los mismos manteniendo sus características fundamentales (reducción de dimensionalidad).

Con respecto al aprendizaje supervisado, gracias al enorme avance de la tecnología han surgido nuevos algoritmos supervisados, como las redes neuronales [16], las cuales necesitan de una gran cantidad de datos etiquetados para poder funcionar con la menor tasa de error posible. Debido al auge de esta tecnología, ha sido necesaria la creación de asistentes para la recolección de dichos datos.

El algoritmo necesita que le enseñemos la respuesta correcta. Respecto a esta respuesta correcta, el algoritmo aprende de sus errores. Al principio dan respuestas aleatorias pero a medida que van "aprendiendo" reducen la tasa de error.

El proceso del aprendizaje supervisado es el siguiente:

1. Recopilar datos históricos, incluyendo la respuesta correcta.
2. Construir un modelo predictivo con esos datos.
3. Evaluar el modelo, para entender qué rendimiento podemos esperar de él.
4. Usar este modelo con datos nuevos.

El aprendizaje supervisado o análisis predictivo cuenta con dos subcategorías diferentes [9]:

La clasificación es una subcategoría del aprendizaje supervisado. Su objetivo es predecir a que conjunto de categorías pertenece un nuevo elemento, basándonos en observaciones pasadas diferenciadas a través de variables categóricas. Los dos tipos de clasificación más conocidos son: clasificación binaria donde solo existen dos conjuntos y clasificación multi-clase como el clásico ejemplo del reconocimiento de cifras manuscritas (donde las

clases van de 0 a 9).

La regresión establece una relación entre una variable dependiente y un conjunto de variables independientes. Es un método para predecir una o más variables dependientes en función de los valores de las variables independientes ($X_i, i \in [1, m]$), siendo m el número de variables.

Se puede usar para los casos donde queremos predecir alguna cantidad continua, por ejemplo, predecir el tráfico de peticiones HTTP en una tienda minorista, predecir el tiempo de permanencia de un usuario en una página o el número de páginas visitas en un blog, etc.

El aprendizaje supervisado se aplica en entornos muy diversos. Por ejemplo, el traductor de Google usa aprendizaje supervisado. Clasificar correos, detectar objetos en una imagen o reconocer y sintetizar sonidos son otras de las muchas aplicaciones que tiene el aprendizaje supervisado

Por otro lado, también es importante hablar de las técnicas de visión artificial [11]. Es un área dentro de la Inteligencia Artificial, que tiene como objetivo procesar, analizar y comprender las imágenes, esto permite a un ordenador analizar una situación y que se pueda actuar de la manera más conveniente. Esto se consigue gracias a diferentes disciplinas como la geometría y estadística. La visión artificial tiene múltiples aplicaciones algunas son: el reconocimiento de objetos, la detección de sucesos, la reconstrucción de una escena y la restauración de imágenes.

Es importante también introducir el concepto de aprendizaje profundo [6]. Al igual que la visión artificial, se trata de un campo dentro de la Inteligencia Artificial. La visión artificial trata de "imitar" el enfoque de aprendizaje que usan los humanos. Los algoritmos de aprendizaje profundo se apilan en una jerarquía creciente de complejidad y abstracción. Cada algoritmo en la jerarquía aplica una transformación no lineal en su entrada y utiliza lo que aprende a través de un método iterativo para crear un modelo estadístico como salida. Entre otras aplicaciones, el aprendizaje profundo se puede combinar con

otras disciplinas como el procesamiento del lenguaje natural con el objetivo de tener un reconocimiento automático del habla.

Para más información acerca de las diferencias entre estas dos tecnologías se puede consultar [12] - "Deep learning vs. traditional computer vision".

1.2. Objetivos del trabajo

El desarrollo del proyecto, ASETOPI, se ha seguido teniendo en cuenta el cumplimiento de los siguientes objetivos.

- Obtención de una visión general del campo de la Minería de datos y, en particular, del aprendizaje supervisado a partir de datos etiquetados.
- Estudio de aplicaciones prácticas de las técnicas de visión artificial [11] tanto clásicas como basadas en los métodos modernos de aprendizaje profundo [6].
- Definición a priori de un conjunto de categorías/clases que se emplearán para proceder al etiquetado de lotes de imágenes.
- Diseño de un procedimiento que facilite la detección de objetos presentes en imágenes, estableciendo el área exacta que ocupan y la etiqueta que les corresponde.

1.3. Motivación

Como se indicaba en la introducción de este apartado, asistentes como ASETOPI surgen para ayudar en el proceso de etiquetado de imágenes. La principal motivación, que se trata a la vez de un objetivo, es facilitar todo lo posible una tarea tan tediosa como esta. El objetivo del asistente es facilitar el etiquetado y automatizarlo en la medida de lo posible, para evitar tener que hacerlo manualmente. El resultado será poder producir un conjunto de datos donde para cada imagen, se indicarán las áreas y sus correspondientes etiquetas, que especifican los objetos presentes en estas regiones. Esta es una información imprescindible para llevar a cabo el proceso de entrenamiento con algoritmos supervisados. Ya que es totalmente independiente del modelo: un mismo conjunto de imágenes ya

etiquetadas puede ser utilizando en el entrenamiento de múltiples modelos. Por este motivo se incentiva a etiquetar cuantas más imágenes mejor, lo cual resultará en una mayor variedad del conjunto de datos final.

1.4. Estructura de la memoria

La presente memoria se estructura de la siguiente manera:

- **Primer capítulo : *Introducción y motivación***. En este capítulo se describe el problema al que se le va a dar solución, así como el enfoque planteado para la solución propuesta de manera superficial.
- **Segundo capítulo : *Especificación del trabajo***. Este capítulo explica detalladamente la planificación y el desarrollo de la solución anteriormente planteada. Para ello se usan distintas técnicas de Ingeniaría del software.
- **Tercer y Cuarto capítulo : *Análisis de requisitos*** y ***Diseño***. El objetivo de estos capítulos es la descomposición pormenorizada de los requisitos identificados para el proyecto, así como la representación de los mismos mediante los distintos diagramas utilizados.
- **Quinto capítulo : *Desarrollo***. Se tratarán las diversas alternativas que se han presentado hasta este punto, y se escogerá de entre ellas la que mejor se adapte a este caso. Así como el proceso de desarrollo del proyecto, con el cual se desarrollará la aplicación final. También se detalla este proceso y las pruebas que se han llevado a cabo a lo largo del proyecto.
- **Sexto capítulo : *Conclusión y trabajos futuros***. Último capitulo de la memoria, en cuanto al proyecto se refiere. En él se llega a una conclusión tras el desarrollo del proyecto así como las posibles lineas de investigación que podrían surgir en un futuro no muy lejano.
- **Capítulos finales (del séptimo al noveno) : *Apéndices***, ***Manuales*** y ***Definiciones y abreviaturas***. Finalmente se adjuntan apéndices relevantes para la comprensión de la memoria, manuales de uso e instalación de la aplicación.

2. Especificación del trabajo

2.1. Descripción de la situación de partida

2.1.1. Descripción del entorno actual

Como ya se ha señalado, para la generación de modelos mediante aprendizaje automático es necesaria la generación de un conjunto de datos, entrenamiento y test. Lo cual deriva en la necesidad de crear un conjunto de datos para los algoritmos, que debe de indicar qué imágenes están involucradas, las regiones en las que se muestran los objetos y qué objeto contiene cada región, este proceso de etiquetado se ha estado realizando de manera manual, o utilizando material previamente ya generado.

2.1.2. Resumen de las deficiencias y carencias identificadas

Realizar la tarea descrita mediante un proceso manual resulta muy tedioso, a lo que se debe añadir los errores e inconsistencias de los datos provocados por el *error humano*. Todo esto se verá agravado si, además, se está trabajando con un gran volumen de datos. Por otro lado si se usa material previamente generado, estamos limitando la variedad del conjunto de datos que podríamos llegar a utilizar. Debido a estos problemas es por lo que surgen este tipo de herramientas que ayudan a realizar este procedimiento.

2.2. Restricciones

El TFG se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo.

Otra restricción es el no poder desarrollar la aplicación para Macintosh al no disponer de un dispositivo en el que realizar la pruebas y finalmente compilar los recursos.

2.3. Metodología de desarrollo software

La metodología es una parte crucial del proceso de desarrollo, por ello para poder escoger la mejor posible, será necesario revisar entre las múltiples disponibles. Para elegir la que mejor se adapte al escenario actual tendremos que escoger de entre una gran variedad, lo cual no será una tarea fácil.

A continuación discutiremos algunos de los modelos tradicionales más populares [14]:

Modelo	**Ventajas**	**Desventajas**
En cascada	Es un modelo lineal por lo que es más simple de implementar	No se puede volver atrás en caso de error, es rígido a modificaciones
	Cantidad de recursos mínimos	Si el cliente no tiene una idea clara producirá mucha confusión
	Después de una etapa importante se realizan pruebas	
Incremental	Se divide en múltiples incrementos	Una vez comenzado el incremento no se pueden modificar los requisitos
	El cliente puede ver el producto antes de estar acabado	Requiere de mucha planificación, tanto administrativa como técnica
		Requiere de metas claras para conocer el estado del proyecto
Desarrollo Rápido de Aplicaciones (DRA)	Mayor velocidad que el desarrollo en cascada al reutilizar componentes	Tiene inconvenientes para proyectos grandes
	Entrega temprana al cliente	Progreso más difícil de medir
	Ciclos de desarrollo más pequeños	Alto costo de herramientas integradas y equipo necesario

Tabla 1: Comparativa entre modelos tradicionales

En la tabla 1 se muestran múltiples modelos de desarrollo muy conocidos, de entre ellos el más conocido puede que sea el tradicional o cascada, pero también el menos flexible. Por este motivo el que mejor podría adaptarse de todos al proyecto sería uno incremental, al no necesitar una base como DRA.

En comparación, los métodos enumerados en la tabla 2 están enfocados a su uso por parte de equipos de trabajo, por lo que usarlos durante el desarrollo sería un poco

Metodología	Ventajas	Desventajas
Scrum	Resultados anticipados	Funciona mejor con equipos reducidos
	Flexibilidad y adaptación a los cambios en los requisitos	Requiere precisa definición de tareas y plazos
		Exige una alta formación
Kanban	Permite disminuir o eliminar stock entre procesos	No es una técnica específica del desarrollo software
	Facilita el control de la producción	
	Permite un sistema de producción flexible según la demanda	
TDD	Mayor simplicidad en el diseño	Dificil de implementar la capa de presentación
	Diseño enfocado en las necesidades	El hecho de pasar pruebas no significa que no existan errores
	Menos tiempo invertido en depuración de errores	Pronunciada curva de aprendizaje

Tabla 2: Comparativa entre metodologías ágiles

forzado. No por ello deben de pasarse por alto las cualidades que presentan, con el objetivo de buscar el mejor ciclo de vida para el proyecto se pueden llegar a fusionar múltiples procesos, para que de este modo se mitiguen las limitaciones individuales y se consiga un mejor desarrollo.

2.4. Organización y gestión

Para estimar el tiempo invertido durante el desarrollo de la aplicación se evaluará el tiempo de las tareas individuales y sus prioridades. El primer paso será conocer el tiempo que se espera tardar en cada tarea, para ello se usarán estrategias ágiles con las cuales será posible disminuir el error y el esfuerzo del cálculo. Existen múltiples alternativas, *planning poker*, tallas de camisetas, *Fibonacci*, etc [1]. Estos métodos al usar medidas relativas permiten comparar tareas para determinar si una es mayor que otra lo cual hace posible asignar un esfuerzo aproximado de manera efectiva y eficiente. Una muy conocida y fácil de aplicar es la de tallas de camisetas donde se asigna una talla (S,M,L,X,XL) a cada tarea dependiendo del tamaño relativo de esta y el esfuerzo para completarla.

De una manera similar se puede tomar una correcta decisión acerca de cómo estimar las prioridades, se puede realizar una estimación parecida a la hecha con las tallas de camisetas. Para ello utilizaremos el método `MOSCOW` como se explica en la página [15] y asignaremos a cada tarea una importancia. En la tabla 3 se usan las medidas de talla de camisetas y `MOSCOW`.

Tareas alto nivel	**Tareas**	**Estimación TALLAS**	**Estimación** `MOSCOW`
Gestión multiproyecto	Crear proyectos	L	M
	Borrar proyectos	L	S
	Abrir proyecto	M	M
	Cerrar proyecto	M	S
Gestionar proyecto	Guardar proyecto	M	M
	Renombrar proyecto	S	C
	Exportar proyecto	S	M
	Importar proyecto	M	C
Gestionar etiquetas	Modificar etiqueta	L	S
	Crear etiqueta	S	M
	Borrar etiqueta	M	C
Gestionar imagen	Cargar imagen	X	M
	Seleccionar imagen	M	M
	Quitar imagen	S	M
	Asignar etiqueta	S	M
Gestionar regiones	Sugerir regiones	XL	M
	Sustituir etiqueta	S	S
	Crear regiones	X	M
	Borrar regiones	S	M
	Seleccionar región	S	M
	Modificar coordenadas	M	M

Tabla 3: Tareas del proyecto

2.5. Estimación del tamaño y esfuerzo

Ya que el presente proyecto es un TFG, no existen tantas restricciones de tipo económico, como de tipo temporal. Por consiguiente, los cálculos de tamaño del proyecto están supeditados el tiempo disponible. En cuanto al esfuerzo, se dispone de tan un solo efectivo (la persona autora del trabajo).

Para conocer el tamaño usaremos estimaciones por Puntos de caso de uso (UCP), el cual se calcula como $UCP = UUCP \times TCF \times EF$, para más información consultar el artículo [3]. El primer paso es definir el peso de cada CU en función de la cantidad de

transacciones que se llevan a cabo.

Función	**Complejidad del CU**	**Peso**
Crear proyectos	Medio	10
Borrar proyectos	Medio	10
Abrir proyecto	Medio	10
Cerrar proyecto	Medio	10
Guardar proyecto	Medio	10
Renombrar proyecto	Medio	10
Exportar proyecto	Simple	5
Importar proyecto	Medio	10
Modificar etiqueta	Medio	10
Crear etiqueta	Medio	10
Borrar etiqueta	Medio	10
Cargar imagen	Simple	5
Seleccionar imagen	Medio	10
Quitar imagen	Medio	10
Asignar etiqueta	Medio	10
Sugerir regiones	Medio	10
Sustituir etiqueta	Medio	10
Crear regiones	Medio	10
Borrar regiones	Medio	10
Seleccionar región	Simple	5
Modificar coordenadas	Medio	10
	Total UUCW	185

Tabla 4: Estimación de Puntos de caso de uso

Una vez calculado el UUCW, ya podemos calcular el UAW [3], para el caso de esta aplicación, es un actor el cual utiliza únicamente interfaz de usuario, con lo cual UAW vale 3. Esto nos deja un UUCP de 188 puntos.

Ahora que tenemos el valor de los UUCP simplemente necesitamos saber los factores de ajuste TCF y EF. Para calcular el TCF asignaremos un valor del 0(bajo) al 5(alto) a cada elemento de la tabla 5.

Finalmente el valor del TCF será: $TCF = 0{,}6 + (0{,}01 \times \Sigma impactos)$. Por lo que en nuestro caso TCF = 0,885. Para calcular el EF se realizan los mismos pasos, en este caso evaluando las habilidades y experiencia del grupo de personas involucradas con el desarrollo del proyecto.

Al igual que para el caso del TCF, EF también tiene una fórmula: $EF = 1{,}4 + (-0{,}03 \times \Sigma impactos)$. Con lo que EF tiene un valor de EF = 0,68.

Descripción	**Peso**	**Valor asignado**	**Impacto** (peso×valor asignado)
Sistema distribuido	2	0	0
Tiempo de respuesta	1	5	5
Eficiencia del usuario final	1	5	5
Procesamiento interno complejo	1	2	2
El código debe ser reutilizable	1	4	4
Facilidad de instalación	0,5	4	2
Facilidad de uso	0,5	5	2,5
Portabilidad	2	0	0
Facilidad de cambio	1	4	4
Concurrencia	1	3	3
Incluye objetivos especiales de seguridad	1	0	0
Provee acceso directo a terceros	1	0	0
Se requiere de entrenamiento especial	1	1	1
		Total	28,5

Tabla 5: Factores técnicos

Factor	**Descripción**	**Peso**	**Valor asignado**	**Impacto**
E1	Familiaridad con el modelo de proyecto utilizado	1,5	3	4,5
E2	Experiencia en la aplicación	0,5	3	1,5
E3	Experiencia en programación orientación a objetos	1	5	5
E4	Capacidad del analista líder	0,5	4	2
E5	Motivación	1	5	5
E6	Estabilidad de los requisitos	2	5	10
E7	Personal a tiempo parcial	-1	4	-4
E8	Dificultad del lenguaje de programación	-1	0	0
			Total	24

Tabla 6: Factores ambientales

Finalmente obtenemos que $UCP = 188 \times 0{,}885 \times 0{,}68 = 113{,}138$. Por sí mismos los UCP no tienen mucho significado, por ejemplo un proyecto con 100 UCP podría tardar más que uno de 90, pero no podemos saber cuánto. Por este motivo es necesario conocer otros factores para calcular el tamaño del esfuerzo, tal y como se describe en [3].

2.6. Planificación temporal

Para obtener una aproximación del esfuerzo[1] (horas-hombre) del proyecto, nos basaremos en la estimación de UCP, calculada anteriormente. Inicialmente se invertía un punto de UCP durante 20 horas, pero con el tiempo se ha adaptado para ajustarse mejor, ahora se debe contar la cantidad de factores ambientales del E1 al E6 que tienen una

[1] `https://es.wikipedia.org/wiki/Puntos_de_caso_de_uso#Esfuerzo_horas-hombre_(E)`

puntuación menor a 3, y también la cantidad de factores del E7 y E8 que son mayores que 3. A partir de este resultado, el total de horas que tardará una persona en completar cada UCP, puede consultarse en la tabla 7.

Horas-persona	**Descripción**
20	El valor es ≤ 2
28	El valor es ≤ 4
36	El valor es ≥ 5

Tabla 7: Cantidad de horas-persona según EF

En mi caso un valor de 1, ya que solo E7 vale más que tres. Con esto se deduce que una persona debe de trabajar 20 horas, para completar un punto de UCP. Con lo cual el esfuerzo total es de $113{,}138 \times 20 = 2{,}262{,}76$ horas o 94,28 días de desarrollo.

2.7. Presupuesto

Para estimar el coste se han estudiado los diferentes ámbitos, tanto software como hardware así como los costes asociados con el desarrollo.

2.7.1. Estimación de recursos

- Equipamiento informático:
  - Procesador: Intel i7-7700HQ
  - RAM: 32 GB
  - Capacidad: 256 GB SSD y 1TB HDD
- Visual Paradigm:
  - Versión: 16.0 standard
  - Tipo: suscripción
- Salario:
  - 5 horas diarias
  - 131 días laborales

- Alquiler
- Servicios:
  - Luz
  - Agua
  - Material de oficina
  - Internet: fibra óptica 100MB simétricos

2.7.2. Estimación de costes

El proceso de desarrollo durará 7 meses de los cuales habrá 131 días laborales, es decir unos 4,367 meses. Para calcular los gastos de amortización en el proyecto se dividirá el importe unitario entre la amortización en meses. Estos resultados parciales se suman, y posteriormente se multiplican por la duración total del proyecto.

Concepto	**Importe unitario**	**Amortización**	**Total / mes**
Ordenador	1299 €	5 años	21,65 €
		Total	94,55 €

Tabla 8: Gastos amortizables

Concepto	**Importe unitario**	**Duración**	**Total**
Visual Paradigm Standard	19 € / mes	4,367 meses	82,97 €
Impuestos, salarios, seguro, etc	1.026,68[2] € / mes	4,367 meses	4.483,51 €
Oficina	450 € / mes	4,367 meses	1.965,15 €
		Total	6.531,63 €

Tabla 9: Gastos

Al sumar el resultado de ambas tablas, el precio final es de 6.626,18 €.

[2]Según el convenio colectivo estatal [5] el sueldo anual de un analista programador es de 24.640,37 € al año por lo que a media jornada serían 1.026,68 € / mes

2.8. Normas y referencias

A continuación, se presentan las normas, reglamentos y referencias de cualquier tipo que han sido aplicados tanto en la elaboración del trabajo como en el desarrollo del mismo.

2.8.1. Disposiciones legales y normas aplicadas

Debido a que las imágenes que serán etiquetadas en la aplicación final son diferenciadas a través de la ruta de manera local, no se viola la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD), por lo tanto si los datos son exportados se deberá a acciones del usuario final.

Por otro lado las normas aplicadas de forma voluntaria son:

- UNE-ISO 21500:2013 - "Directrices para la dirección y gestión de proyectos".
- UNE 157801:2007 - "Criterios Generales para la elaboración de proyectos de Sistemas de Información".

Puede encontrarse más documentación acerca de estas normas en la página de AENOR[3].

2.8.2. Mecanismos de control de calidad

El asegurar la calidad en la redacción del trabajo implica la verificación de la completitud (falta de omisiones), la integridad de la documentación, así como una redacción clara, concisa y entendible por todos los participantes e interesados en dicho trabajo.

Al estar el trabajo desarrollado por una sola persona y verificado por un/a tutor/a, no es necesario llevar un documento de control de edición y revisión de la documentación. De esta forma, se han utilizado mecanismos básicos para la verificación de la integridad y completitud, que incluyen un control de versiones a nivel de documento y un control sencillo de la trazabilidad de especificaciones y requisitos.

[3] `https://www.aenor.com/normas-y-libros/buscador-de-normas`

3. Análisis de requisitos

Un buen diseño software permite el entendimiento de los requisitos, aplicar patrones de diseño y asegurar la calidad del producto final. Por este motivo, la parte del diseño es crucial para un buen desarrollo del producto y su posterior uso.

El sistema permitirá la gestión del trabajo a través del uso de proyectos, los cuales se podrán importar, exportar, borrar o modificar en cualquier momento. Para un proyecto dado, el sistema será capaz de asociarle imágenes, las cuales tendrán dos posibles estados etiquetadas o no. Para considerar una imagen como etiquetada esta debe de contener al menos una región que tenga asociada cualquiera de las etiquetas disponibles en el proyecto. Para agilizar la tarea de crear regiones, el sistema podrá identificar objetos en las imágenes para que el usuario pueda decidir qué etiqueta asignarles. El sistema también será capaz de crear/añadir, borrar o modificar cualquiera de los elementos mencionados (imágenes, regiones y etiquetas).

3.1. Captura de requisitos

Los requisitos se han plasmado a través de un diagrama de casos de uso del sistema (figura 1). Estos diagramas reflejan el comportamiento del sistema a través del punto de vista del cliente. Es por eso que los casos de uso fijan los requisitos funcionales del sistema (funciones que el sistema puede ejecutar).

En la fase inicial estos casos de uso deben ser desarrollados, para construir la base a partir de la cual, seremos capaces de guiar el resto del diseño, y con ello fijar el comportamiento de nuestro sistema, de una manera secuencial y estructurada.

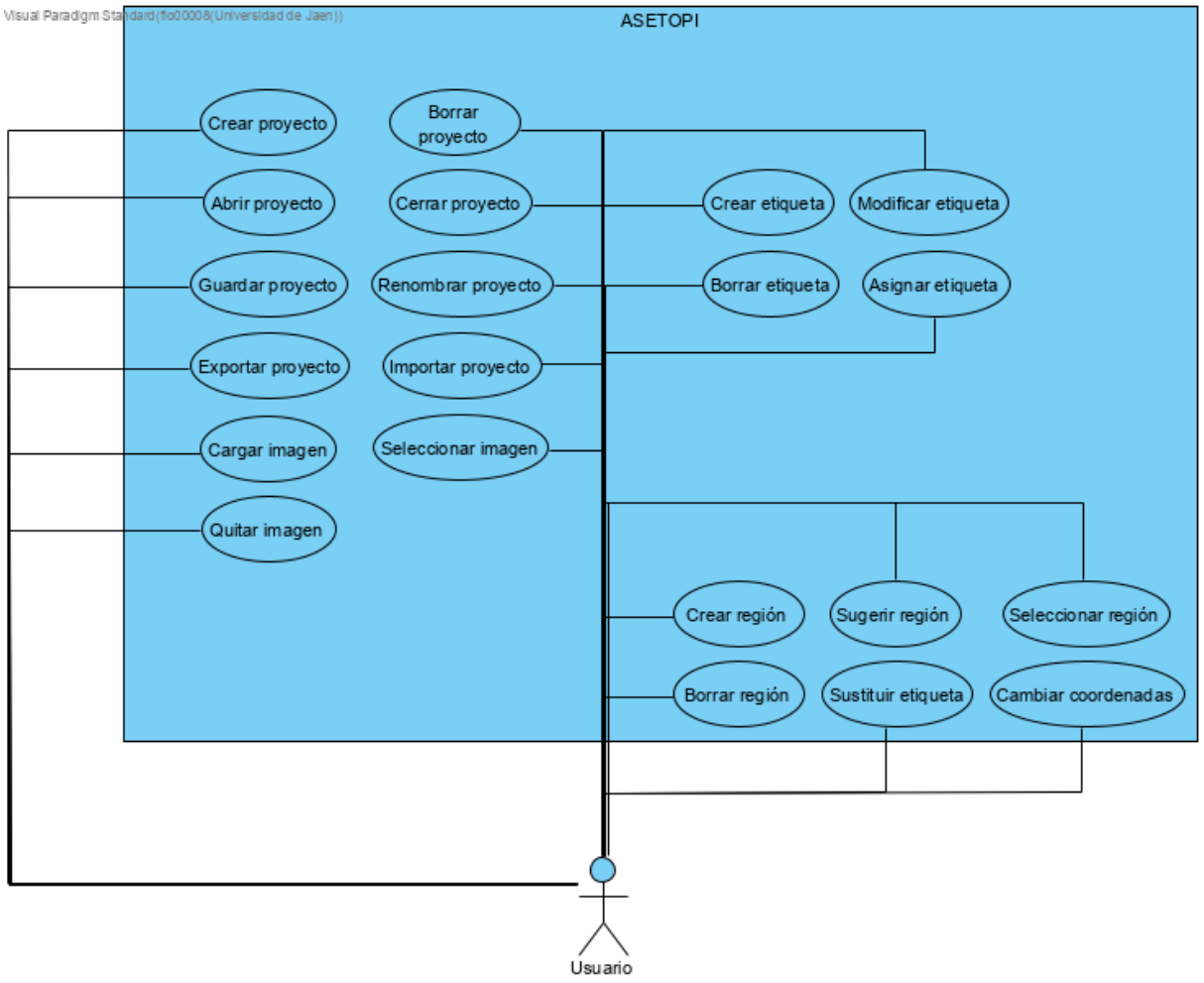

Figura 1: Diagrama de casos de uso del sistema

3.2. Requisitos funcionales

Código	Caso de uso
CU-01	Crear proyecto
CU-02	Abrir proyecto
CU-03	Cerrar proyecto
CU-04	Guardar proyecto
CU-05	Exportar proyecto
CU-06	Importar proyecto
CU-07	Renombrar proyecto
CU-08	Cargar imagen
CU-09	Seleccionar imagen
CU-10	Crear etiqueta
CU-11	Modificar etiqueta
CU-12	Borrar etiqueta
CU-13	Asignar etiqueta
CU-14	Crear región
CU-15	Sugerir región
CU-16	Seleccionar región
CU-17	Borrar región
CU-18	Cambiar coordenadas
CU-19	Borrar proyecto
CU-20	Quitar imagen

Tabla 10: Requisitos funcionales

En el primer Caso de Uso (tabla 11), se define la interacción acerca de cómo el usuario puede crear un proyecto, y los datos que son necesarios para su creación. A su vez el sistema informa continuamente al usuario del estado interno, o si se da el caso, qué datos introducidos son erróneos.

CU-01	**Crear proyecto**
Actores	Usuario, Sistema
Precondiciones	- Ninguna
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona la opción *Crear proyecto* 2. El usuario introduce el nombre del proyecto 3. El usuario selecciona aceptar 4. El sistema informa que el proyecto ha sido creado

	5. Fin del escenario con éxito
Escenarios alternativos	3a. Ya existe un proyecto con ese nombre 1. El sistema informa que ya existe un proyecto con ese nombre 2. Volver al paso 2 del escenario principal
Escenarios de excepción	3b. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin de escenario sin éxito

Tabla 11: CU-01 Crear proyecto

El CU-02 permite que una vez tengamos múltiples proyectos creados, se pueda continuar progresando en uno de ellos. Este CU es simple en su narrativa, pero complejo en su desarrollo (véase la figura 4), al necesitar de manera indirecta de otras funcionalidades.

CU-02	**Abrir proyecto**
Actores	Usuario, Sistema
Precondiciones	- Debe existir un proyecto - No hay proyectos abiertos - El sistema tiene permisos de lectura
Poscondiciones	- El sistema muestra al usuario el proyecto
Escenario principal	1. El usuario selecciona la opción *Abrir proyecto* 2. El sistema muestra la lista de proyectos creados 3. El usuario selecciona el proyecto 4. El usuario selecciona el botón aceptar 5. Fin del escenario con éxito
Escenarios alternativos	3a. El proyecto no es válido 1. El sistema notifica el error 2. Volver al paso 2 del escenario principal
Escenarios de excepción	4a. El usuario decide cancelar 1. El usuario pulsa el botón cancelar

	2. Fin de escenario sin éxito

Tabla 12: CU-02 Abrir proyecto

El siguiente CU es crucial para un sistema que permite trabajar con varios proyectos. Este CU es uno de los pocos que referencia a otros, pero gracias a estas referencias se simplifica mucho la descripción de los mismos.

CU-03	**Cerrar proyecto**
Actores	Usuario, Sistema
Precondiciones	- Hay un proyecto abierto
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona la opción *Cerrar proyecto* 2. El usuario selecciona aceptar 3. El sistema cierra el proyecto 4. Fin del escenario con éxito
Escenarios alternativos	1a. El proyecto no se ha guardado 1. El sistema advierte de que se va a guardar el proyecto 2. El usuario selecciona aceptar 3. Se realiza el caso de uso CU-04 Guardar proyecto 4. Volver al paso 2 del escenario principal 1a2a. El usuario decide no guardar 1. El usuario pulsa el botón cancelar 2. Volver al paso 2 del escenario principal
Escenarios de excepción	2a. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin de escenario sin éxito

Tabla 13: CU-03 Cerrar proyecto

Este es posiblemente el CU más importante de todos, de nada sirve que estén perfectamente definidos el resto de Casos de Uso. Sin un método para hacer persistentes los cambios, todo el esfuerzo es en vano. Del mismo modo si se cometen errores, se permite descartar los cambios.

CU-04	**Guardar proyecto**
Actores	Usuario, Sistema
Precondiciones	- Hay un proyecto abierto - Se han realizado cambios desde la última vez que se guardó - Se han realizado cambios desde la última vez que se abrió - El sistema tiene permisos de escritura
Poscondiciones	- Todos los cambios realizados se vuelven permanentes
Escenario principal	1. El usuario selecciona la opción *Guardar proyecto* 2. El sistema guarda el proyecto 3. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	2a. El sistema no pueden almacenar los datos 1. El sistema notifica del error 2. Fin de escenario sin éxito

Tabla 14: CU-04 Guardar proyecto

Al exportar un proyecto el sistema transforma el estado interno en datos, que el usuario puede utilizar en su favor. Un ejemplo es exportar en formato CSV donde para cada imagen se indican qué regiones aparecen, qué etiquetas tienen y sus posiciones.

CU-05	**Exportar proyecto**
Actores	Usuario, Sistema
Precondiciones	- Hay un proyecto abierto - El sistema tiene permisos de escritura
Poscondiciones	- El usuario puede importar usando ese fichero
Escenario principal	1. El usuario selecciona la opción *Exportar proyecto* 2. El usuario selecciona la ruta donde se desea exportar 3. El usuario selecciona el formato para exportar 4. El usuario pulsa el botón aceptar 5. El sistema exporta el proyecto 6. Fin del escenario con éxito
Escenarios alternativos	3a. El proyecto no está guardado 1. El sistema advierte de que se va a guardar el proyecto 2. El usuario selecciona aceptar 3. Se realiza el caso de uso CU-04 Guardar proyecto 4. Volver al paso 4 del escenario principal 3a2a. El usuario decide no guardar 1. El usuario pulsa el botón cancelar 2. Volver al paso 4 del escenario principal
Escenarios de excepción	3b. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin de escenario sin éxito

Tabla 15: CU-05 Exportar proyecto

Es en las situaciones en las que el trabajo se debe de continuar en otra máquina, es cuando resalta la importancia de este CU, de otro modo duplicar todo un proyecto sería

una tarea tediosa y larga.

CU-06	**Importar proyecto**
Actores	Usuario, Sistema
Precondiciones	- Existe un proyecto a importar - El sistema tiene permisos de lectura
Poscondiciones	- Hay un proyecto abierto
Escenario principal	1. El usuario selecciona la opción importar proyecto 2. El usuario selecciona el proyecto a importar 3. El usuario pulsa el botón aceptar 4. El sistema guarda el proyecto 4. El sistema abre el proyecto 5. Fin del escenario con éxito
Escenarios alternativos	1a. El usuario tiene un proyecto abierto 1. Se realiza el caso de uso CU-03 Cerrar proyecto 2. Volver al paso 1 del escenario principal
Escenarios de excepción	2a. El formato del fichero no es válido 1. El sistema informa de un error 2. Fin del escenario sin éxito 3a. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin del escenario sin éxito

Tabla 16: CU-06 Importar proyecto

En el siguiente CU se muestran los pasos necesarios a seguir, para que el usuario pueda cambiarle el nombre a un proyecto, en caso de cometer algún error.

CU-07	**Renombrar proyecto**
Actores	Usuario, Sistema

Precondiciones	- Hay un proyecto abierto
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona la opción *Renombrar proyecto* 2. El usuario introduce el nuevo nombre del proyecto 3. El usuario pulsa el botón aceptar 4. Fin del escenario con éxito
Escenarios alternativos	2a. El nombre no es válido o ya existe 1. El sistema informa de un error 2. Volver al paso 2 del escenario principal
Escenarios de excepción	3a. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin del escenario sin éxito

Tabla 17: CU-07 Renombrar proyecto

En este CU se declara una de las interacciones clave del sistema, y que da comienzo al uso del mismo. Se permite que para el proyecto en el que se está trabajando, se puedan registrar múltiples imágenes de manera simultanea, lo que facilita toda la labor de inserción de imágenes.

CU-08	**Cargar imagen**
Actores	Usuario, Sistema
Precondiciones	- Hay un proyecto abierto - Existen las imágenes a cargar - El sistema tiene permisos de lectura
Poscondiciones	- El sistema muestra una de las imágenes
Escenario principal	1. El usuario selecciona la opción *Cargar imágenes* 2. El usuario selecciona un conjunto de imágenes a cargar 3. El usuario pulsa el botón aceptar 4. El sistema carga las imágenes seleccionadas

	4.1. El sistema valida la siguiente imagen 4.2. El sistema añade esa imagen al proyecto 5. El sistema muestra la primera imagen exitosamente añadida 6. Fin del escenario con éxito
Escenarios alternativos	4.1a. Una imagen no es válida 1. El sistema informa al usuario del error 2. Volver al paso 4.1 del escenario principal 4.2a. Una imagen ya está incluida en el proyecto 1. El sistema informa al usuario del error 2. Volver al paso 4.1 del escenario principal
Escenarios de excepción	3a. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin del escenario sin éxito

Tabla 18: CU-08 Cargar imagen

Para empezar a trabajar con cualquier tipo de elemento en el sistema el usuario debe seleccionarlo. A continuación se detallan los pasos necesarios, para que se puedan añadir y modificar atributos gracias a la ayuda del resto de Casos de Uso.

CU-09	**Seleccionar imagen**
Actores	Usuario, Sistema
Precondiciones	- Al menos una imagen cargada
Poscondiciones	- Ninguna
Escenario principal	1. El usuario busca una imagen de entre las disponibles 2. El usuario selecciona la imagen 3. El sistema muestra la imagen 4. Fin del escenario con éxito

Escenarios alternativos	Ninguno
Escenarios de excepción	3a. Hay un error en la carga 1. El sistema muestra el error 2. Fin del escenario sin éxito

Tabla 19: CU-09 Seleccionar imagen

Antes de asignar etiquetas a las regiones, primero deben de crearse las mismas. Por este motivo, crear etiquetas debe de ser intuitivo y fácil. Con ese objetivo en mente se ha descrito el siguiente CU.

CU-10	**Crear etiqueta**
Actores	Usuario, Sistema
Precondiciones	- Hay un proyecto abierto
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona la opción *Crear etiqueta* 2. El usuario introduce el nombre de la etiqueta 3. El usuario pulsa el botón aceptar 4. El sistema añade la etiqueta al proyecto 5. Fin del escenario con éxito
Escenarios alternativos	3a. Ya existe una etiqueta con ese nombre 1. El sistema informa del error 2. Volver al paso 2 del escenario principal
Escenarios de excepción	3b. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin del escenario sin éxito

Tabla 20: CU-10 Crear etiqueta

Del mismo modo que se permite renombrar un proyecto, también cabe la posibilidad de renombrar una etiqueta. Esto plantea varias cuestiones como ¿Qué pasa si existen regiones que usan esa etiqueta? o ¿Existe ya otra etiqueta con ese nombre? No son cuestiones triviales y que comprometen la integridad de los datos, por este motivo, la simple tarea de renombrar una etiqueta, no es tan directa como pudiera parecer.

CU-11	**Modificar etiqueta**
Actores	Usuario, Sistema
Precondiciones	- Hay un proyecto abierto - Hay etiquetas creadas
Poscondiciones	- Se modifican las regiones que utilizan esa etiqueta
Escenario principal	1. El usuario selecciona una etiqueta 2. El usuario selecciona la opción *Modificar etiqueta* 3. El usuario introduce el nuevo nombre de la etiqueta 4. El usuario pulsa el botón aceptar 5. El sistema modifica la etiqueta en el proyecto 6. Fin del escenario con éxito
Escenarios alternativos	4a. Ya existe una etiqueta con ese nombre 1. El sistema informa del error 2. Volver al paso 3 del escenario principal
Escenarios de excepción	4b. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin del escenario sin éxito

Tabla 21: CU-11 Modificar etiqueta

Al igual que en el caso anterior para borrar una etiqueta deben de tomarse decisiones, que afectan al comportamiento del sistema de una manera u otra. Por ejemplo para este

CU si existe una región que utiliza la etiqueta, el escenario finaliza sin éxito, mientras que antes (tabla 21) se podían actualizar los valores sin problema.

CU-12	**Borrar etiqueta**
Actores	Usuario, Sistema
Precondiciones	- Hay un proyecto abierto - Hay etiquetas creadas
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona una etiqueta 2. El usuario selecciona la opción *Borrar etiqueta* 3. El usuario pulsa el botón aceptar 4. El sistema elimina la etiqueta del proyecto 5. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	3a. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin del escenario sin éxito 4a. La etiqueta está asignada a una región 1. El sistema informa del error 2. Fin del escenario sin éxito

Tabla 22: CU-12 Borrar etiqueta

Asignar etiquetas a las regiones es una tarea primordial para la aplicación, por ello necesita ser lo más simple posible de usar, para que se cometan el menor número de errores posibles.

CU-13	**Asignar etiqueta**
Actores	Usuario, Sistema
Precondiciones	- Hay un proyecto abierto

	- Existen etiquetas - Existen regiones
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona una etiqueta 2. El usuario selecciona una región 3. El sistema asigna la etiqueta a la región seleccionada 4. Fin del escenario con éxito
Escenarios alternativos	2a. La región ya tiene una etiqueta asignada 1. El sistema reemplaza la nueva etiqueta 2. Volver al paso 2 del escenario principal
Escenarios de excepción	Ninguno

Tabla 23: CU-13 Asignar etiqueta

El crear regiones es otro de los aspectos claves. Ya que es una tarea común debe de ser simple e intuitiva en este caso el usuario simplemente debe de seleccionar el área en el que se creará la región.

CU-14	**Crear región**
Actores	Usuario, Sistema
Precondiciones	- Hay una imagen seleccionada
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona la opción *Crear región* 2. El usuario marca una zona de la imagen 3. El sistema crea la región marcada para la zona seleccionada 4. Fin del escenario con éxito
Escenarios alternativos	Ninguno

Escenarios de excepción	Ninguno

Tabla 24: CU-14 Crear región

Una de las tareas más llamativas de la aplicación y que la diferencia de las demás, es la detección de objetos presentes. A través de varios algoritmos, es posible ayudar al usuario en la tarea del etiquetado, creando regiones donde se han detectado los objetos.

CU-15	**Sugerir región**
Actores	Usuario, Sistema
Precondiciones	- Hay una imagen seleccionada
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona la opción *Sugerir regiones* 2. El sistema busca objetos en la imagen 3. El sistema crea regiones para cada objeto detectado 4. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 25: CU-15 Sugerir región

A diferencia del CU para seleccionar una imagen (tabla 19) en este caso la selección de una región no solo da paso a las diferentes acciones que se pueden aplicar a una región, como asignar etiqueta o borrarla. Si no que también muestra sus propiedades de manera explicita.

CU-16	**Seleccionar región**
Actores	Usuario, Sistema

Precondiciones	- Hay una imagen cargada - Existen regiones en la imagen
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona una región de las disponibles 2. El sistema muestra las posibles acciones 4. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	Ninguno

Tabla 26: CU-16 Seleccionar región

Borrar regiones es una tarea sencilla pero importante. Borrar una región no surge solo de un error por parte del usuario al crearla, en ese caso simplemente podría modificarse. Donde es más útil es cuando el sistema ha sugerido regiones, pero el usuario no está conforme con estas, en este escenario quitar una región es indispensable.

CU-17	**Borrar región**
Actores	Usuario, Sistema
Precondiciones	- Hay una región seleccionada
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona la opción de *Borrar región* 2. El usuario pulsa el botón aceptar 3. El sistema borrar la región 4. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	3a. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin del escenario sin éxito

Tabla 27: CU-17 Borrar región

Del mismo modo que en el caso anterior, es necesario tener la posibilidad de modificar la posición y el área de una región creada previamente.

CU-18	**Cambiar coordenadas**
Actores	Usuario, Sistema
Precondiciones	- Hay una región seleccionada
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona la opción de *Modificar región* 2. El usuario mueve la región 3. El sistema modifica la región 4. Fin del escenario con éxito
Escenarios alternativos	2a. El usuario decide cambiar el tamaño 1. El usuario mueve los bordes de la región 2. Volver al paso 3 del escenario principal
Escenarios de excepción	Ninguno

Tabla 28: CU-18 Cambiar coordenadas

Otra de las operaciones cruciales de los objetos es el poder ser borrados, en el siguiente CU se describe cómo se pueden borrar los proyectos.

CU-19	**Borrar proyecto**
Actores	Usuario, Sistema
Precondiciones	- No hay ningún proyecto abierto - Hay al menos un proyecto creado
Poscondiciones	- Los cambios se guardan
Escenario principal	1. El usuario selecciona la opción *Borrar proyecto* de un proyecto 2. El usuario pulsa el botón aceptar 3. El sistema borra todos los datos del proyecto

	4. Fin del escenario con éxito
Escenarios alternativos	Ninguno
Escenarios de excepción	2a. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin del escenario sin éxito

Tabla 29: CU-19 Borrar proyecto

Este CU se llama quitar y no borrar, para enfatizar el hecho de que ninguna imagen será borrada simplemente el sistema dejará de referenciarla.

CU-20	**Quitar imagen**
Actores	Usuario, Sistema
Precondiciones	- Hay una imagen seleccionada
Poscondiciones	- Ninguna
Escenario principal	1. El usuario selecciona la opción *Quitar imagen* 2. El usuario pulsa el botón aceptar 3. El sistema quita esa imagen del proyecto 4. El sistema muestra la siguiente imagen 5. Fin del escenario con éxito
Escenarios alternativos	4a. No hay más imágenes en el proyecto 1. El sistema muestra que no hay imágenes en el proyecto 2. Volver al paso 5 del escenario principal
Escenarios de excepción	2a. El usuario decide cancelar la operación 1. El usuario pulsa el botón cancelar 2. Fin del escenario sin éxito

Tabla 30: CU-20 Quitar imagen

3.3. Requisitos no funcionales

Con frecuencia, los requisitos no funcionales son ignorados o subestimados en la fase de análisis, esto da lugar a que más adelante en la fase de implementación intentar remediar este error implica más esfuerzo. En ellos se describen características del funcionamiento del sistema y no qué información se debe guardar o las funciones que realiza. A continuación se listan los requisitos no funcionales:

1. El nuevo sistema debe desarrollarse aplicando patrones y recomendaciones de programación que incrementen la seguridad de datos.
2. El sistema deberá poder gestionar proyectos, de gran volumen.
3. La aplicación debe tener interfaces intuitivas, de fácil uso.
4. El sistema deberá tener una interfaz *responsive* [4].
5. El sistema debe contar con manuales de usuario estructurados adecuadamente.
6. El sistema debe proporcionar mensajes de error que sean informativos y orientados a usuario final.
7. El sistema debe ser tolerante ante los fallos y las operaciones a realizar deben ser transaccionales.

[4] `https://developer.mozilla.org/es/docs/Learn/CSS/CSS_layout/Dise%C3%B1o_receptivo`

3.4. Especificación de requisitos

A partir de los casos de uso ya puede modelarse un el diagrama del modelo de dominio (véase imagen 2) para poder visualizar el entorno del problema y así poder entenderlo mejor y con ello aplicar una buena solución.

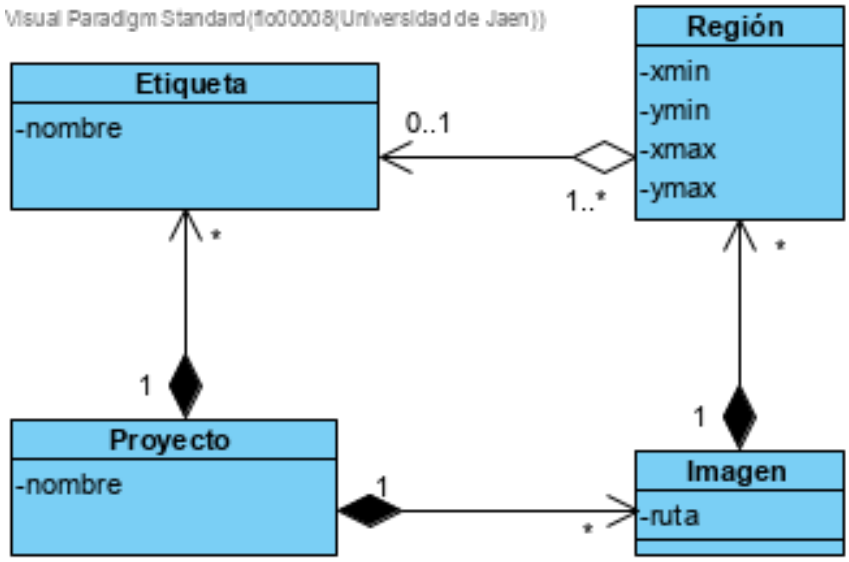

Figura 2: Modelo de dominio

4. Diseño

4.1. Diagramas de secuencia

A partir de esta primera aproximación ya se podría empezar a modelar una solución al problema de etiquetado de imágenes, esto nos daría paso a los diagramas de secuencia, utilizando como apoyo los casos de uso y el modelo de dominio. Una gran cantidad de diagramas de secuencia resultantes podría generar aspectos negativos como una mayor complejidad y mayor esfuerzo en la planificación. Sin embargo aunque se invierta un mayor esfuerzo en la etapa de diseño, será tiempo ahorrado en problemas durante el desarrollo.

En el primer Diagrama de secuencia 3, se diferencia la VistaUsuario del resto de la aplicación, y se define como una interfaz con la que el usuario interactúa. Esto se hace con el objetivo de crear una abstracción que establezca qué operaciones se deben de realizar, pero dando libertad.

El resto del diagrama simplemente muestra la serie de operaciones por las que pasa el usuario, desde validación hasta el proceso final en el que se crea el proyecto y se le muestra al actor.

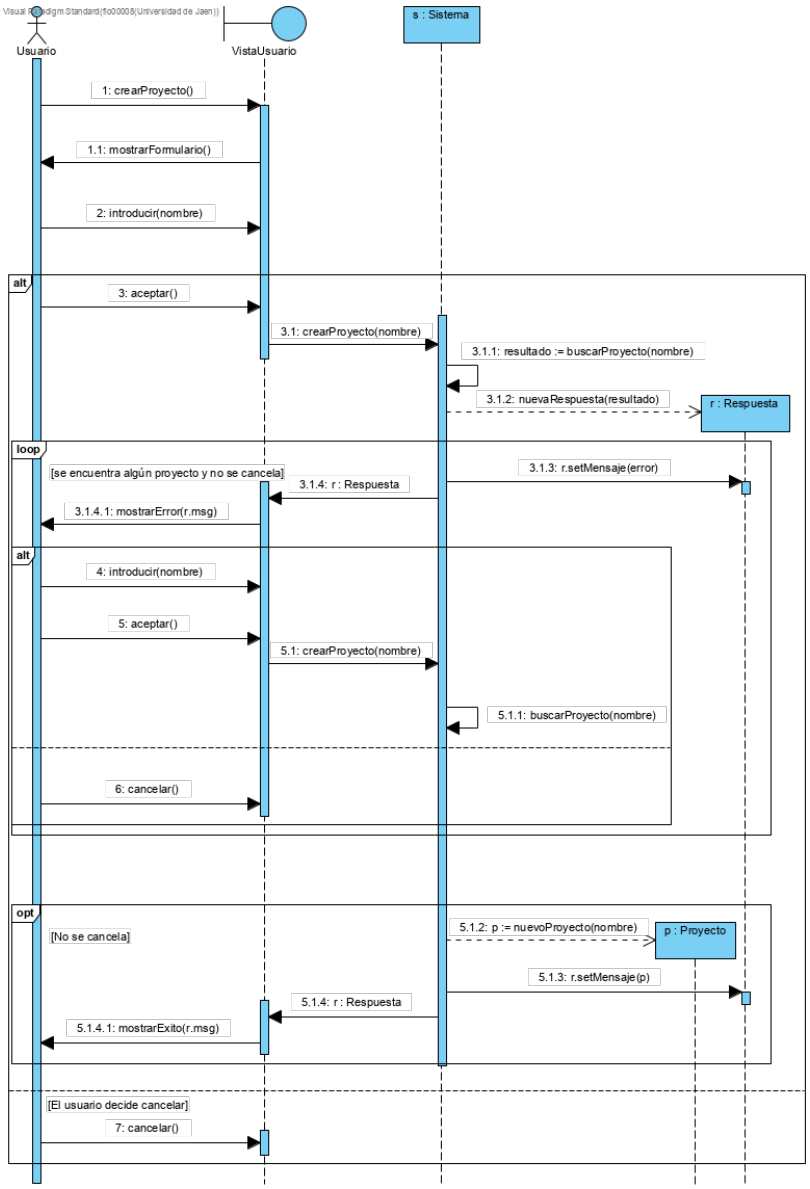

Figura 3: Diagrama de secuencia del caso de uso CU-1

Este DS (4) muestra por primera vez la creación de un componente independiente, el GestorArchivos que ayudará a separar toda la interacción con los ficheros. Manteniendo la clase principal libre de esta lógica, esto aporta cohesión y disminuye el acoplamiento al especializar una clase para un objetivo concreto.

Para este diagrama, también cabe destacar que es importante la gestión de los errores ya que protege al sistema en caso de que el proyecto este corrompido.

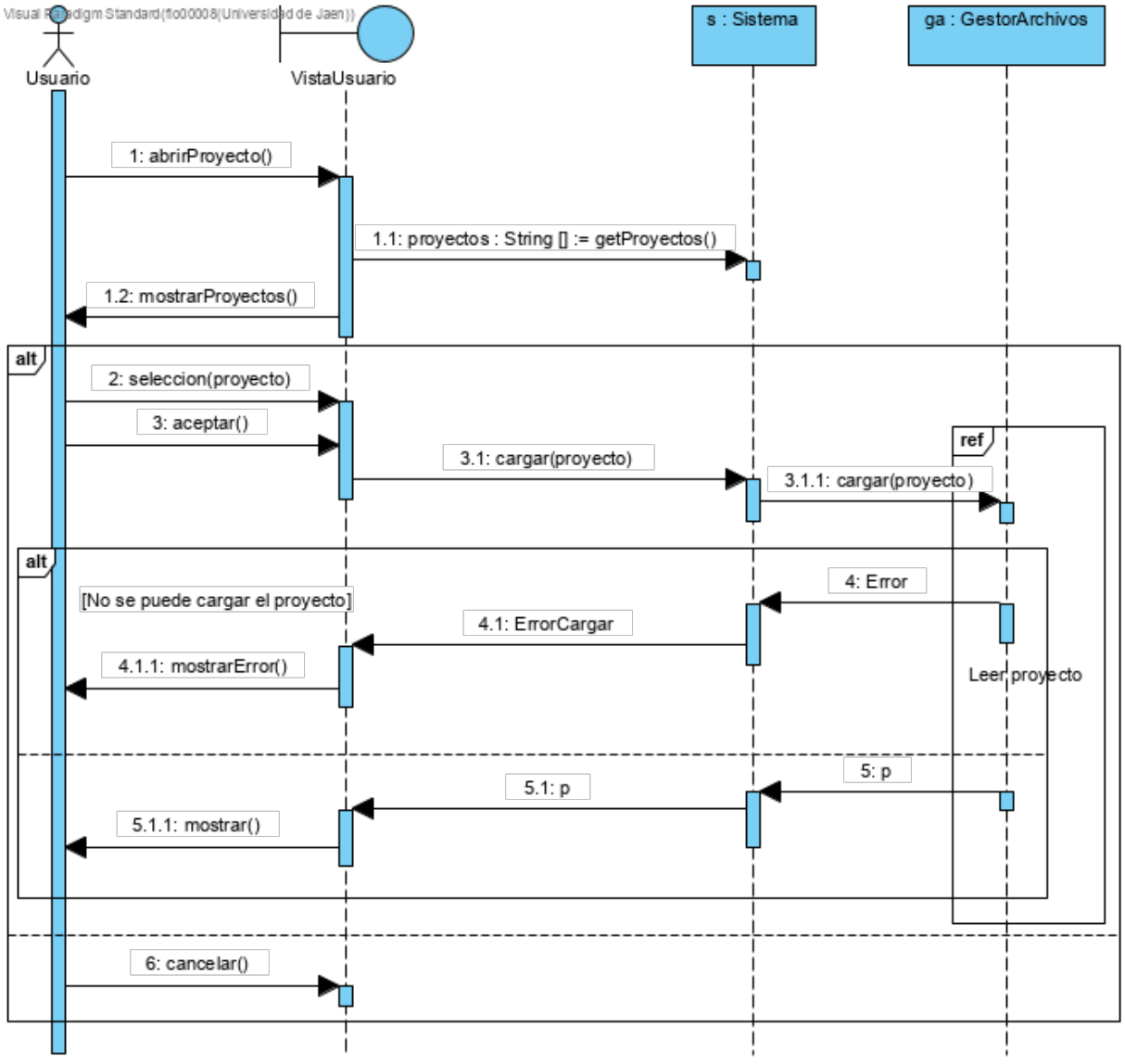

Figura 4: Diagrama de secuencia del caso de uso CU-2

En el caso de cerrar un proyecto ya sea para cerrar la aplicación o para volver al menú con todos los proyectos, es posible que el actor olvide guardar o decida no hacerlo y cancelar una transacción por ese motivo es importante confirmar la acción antes de perder los datos por culpa de un error.

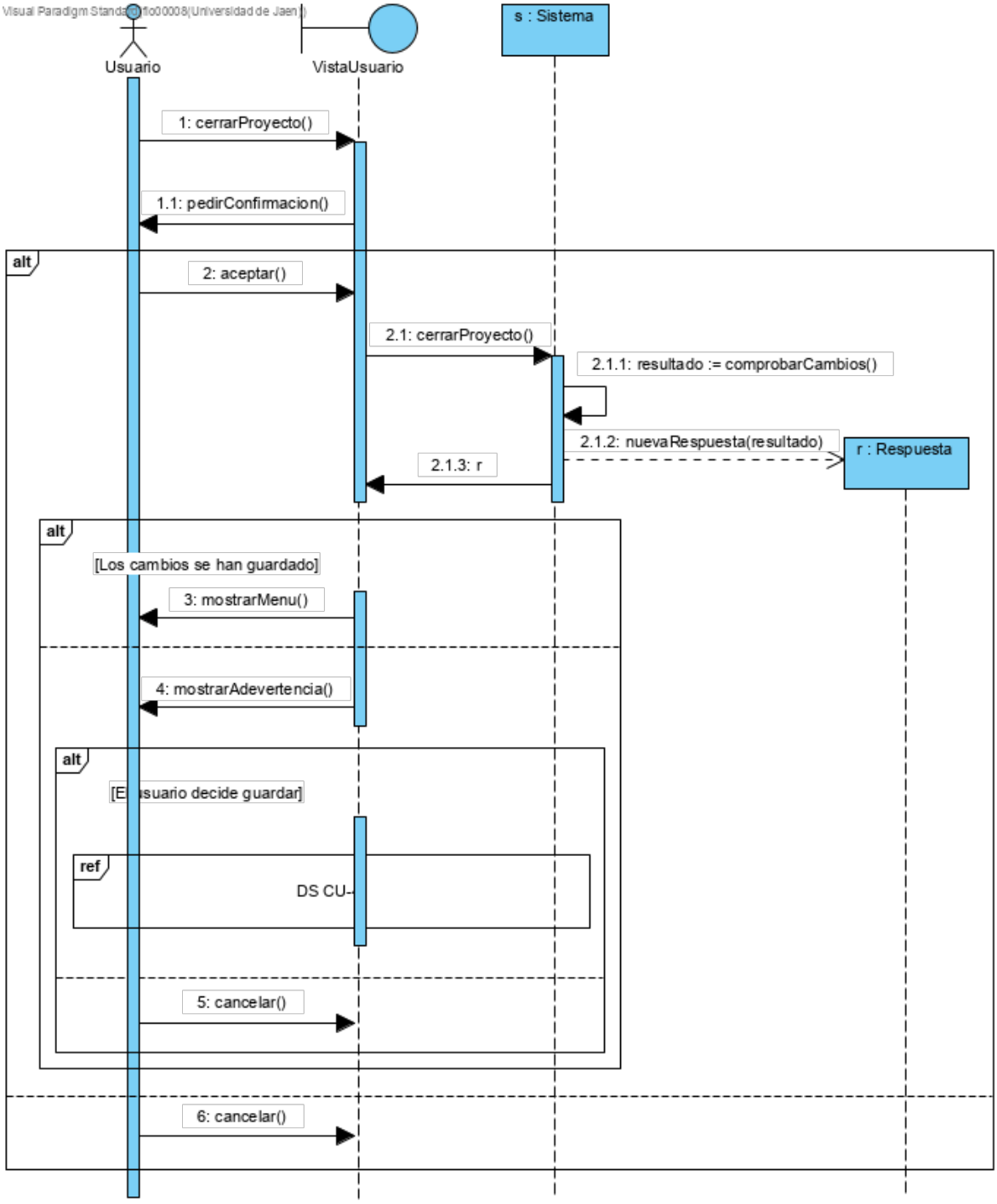

Figura 5: Diagrama de secuencia del caso de uso CU-3

Con el objetivo de que guardar sea una operación rápida y habitual el sistema guardará el proyecto que tiene en memoria a disco, sin pedir confirmación al Usuario.

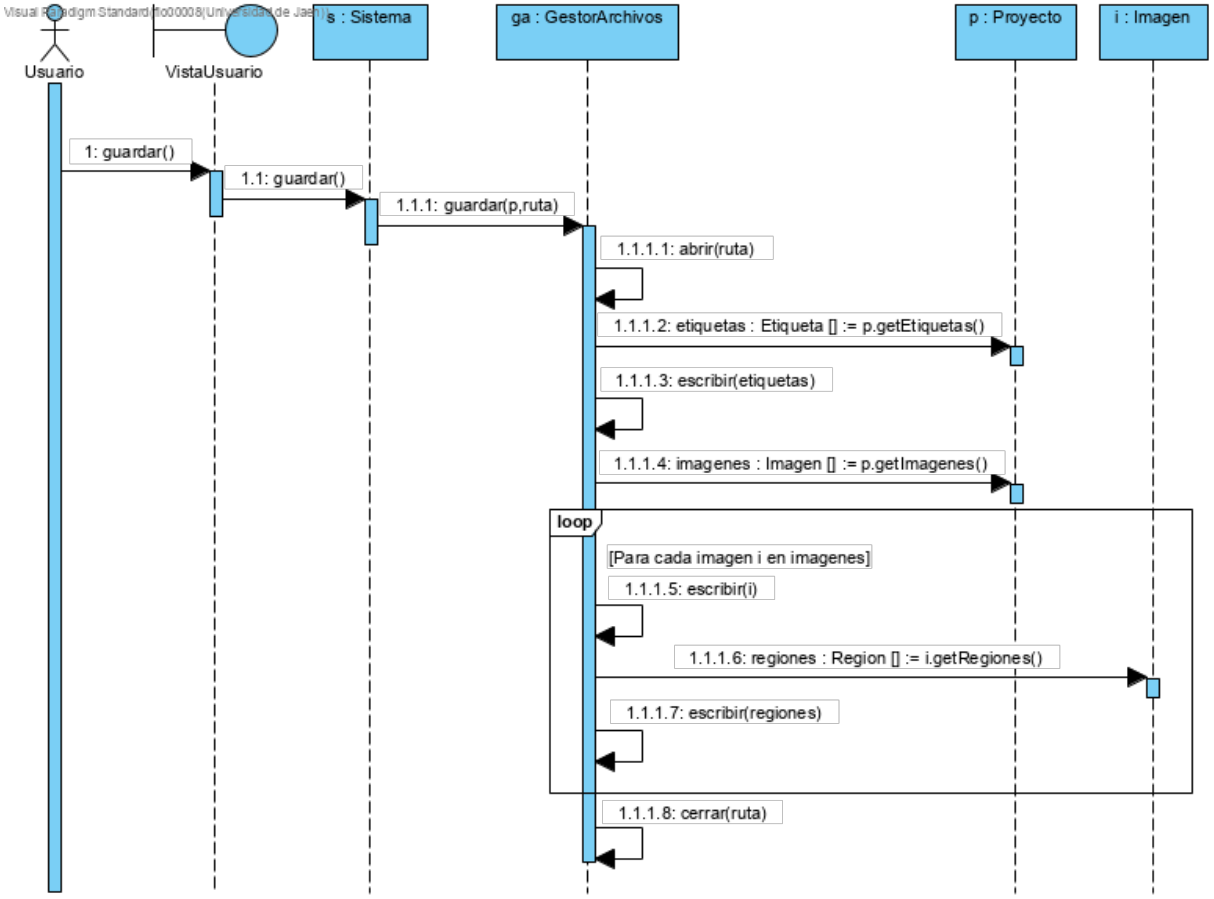

Figura 6: Diagrama de secuencia del caso de uso CU-4

Para exportar un proyecto, se realizan unos pasos parecidos a los de guardar, pero en este caso el actor define una ruta para generar los ficheros con los datos procesados.

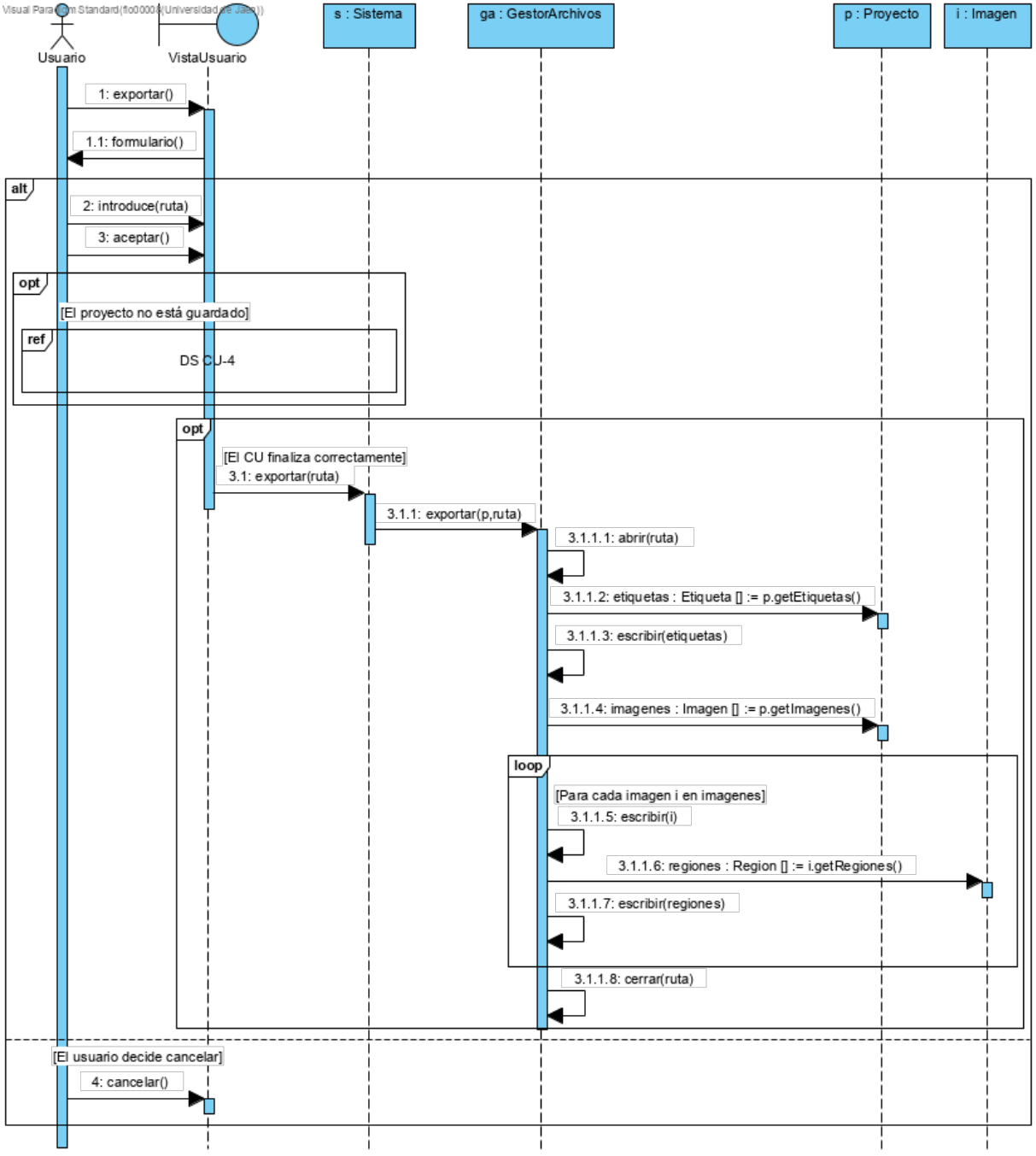

Figura 7: Diagrama de secuencia del caso de uso CU-5

Al igual que anteriormente importar se parece a otro CU, en este caso se reutiliza la parte de cargar un proyecto. Esto acortará el desarrollo al reusar partes.

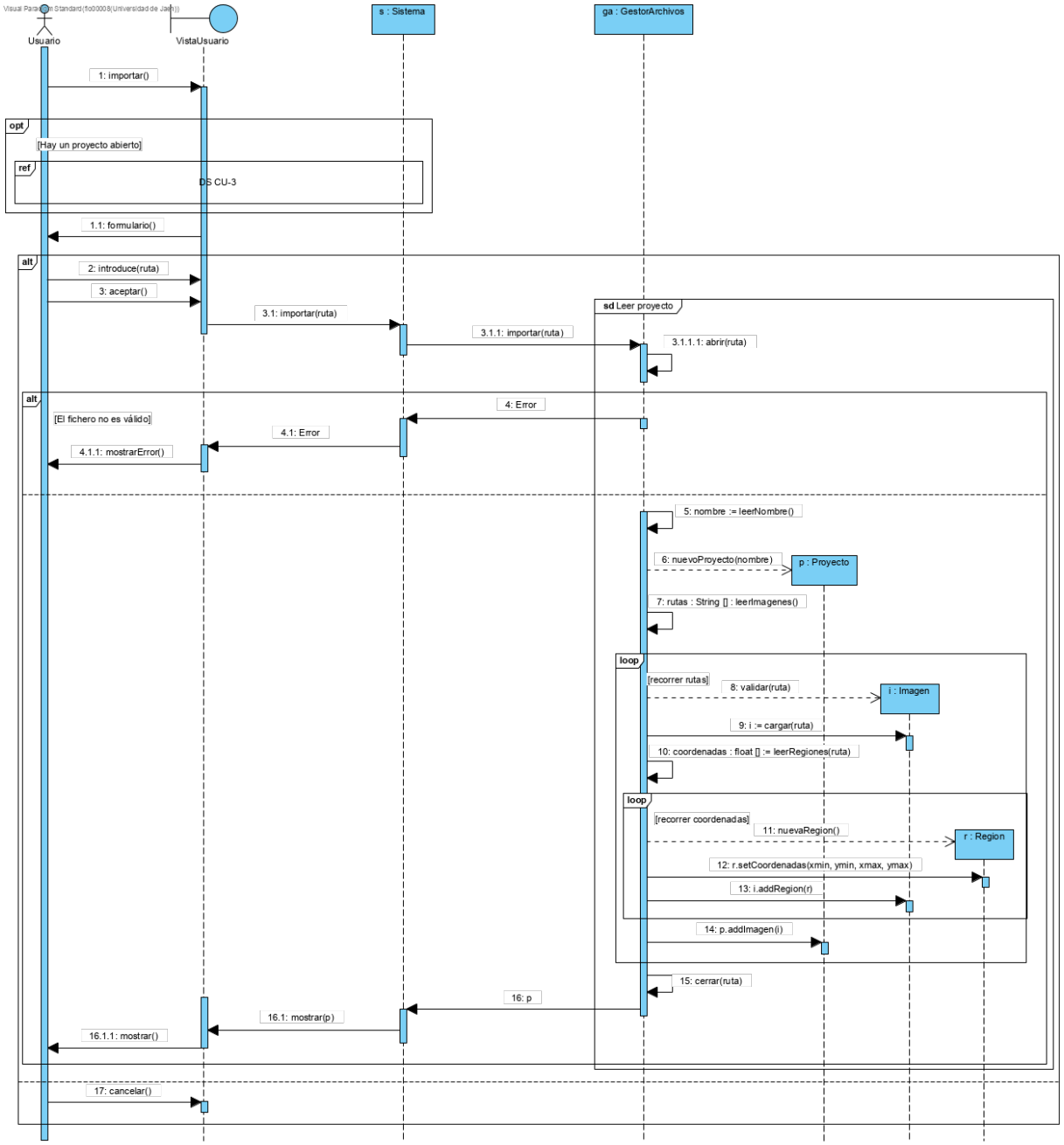

Figura 8: Diagrama de secuencia del caso de uso CU-6

Renombrar, como en otros CU, deben de validarse los datos introducidos por el usuario, en este caso no se permitirá renombrar un proyecto si ya existe otro proyecto con ese nombre. Al renombrar el atributo del proyecto, permitirá que renombrar tenga efecto si finalmente se guarda.

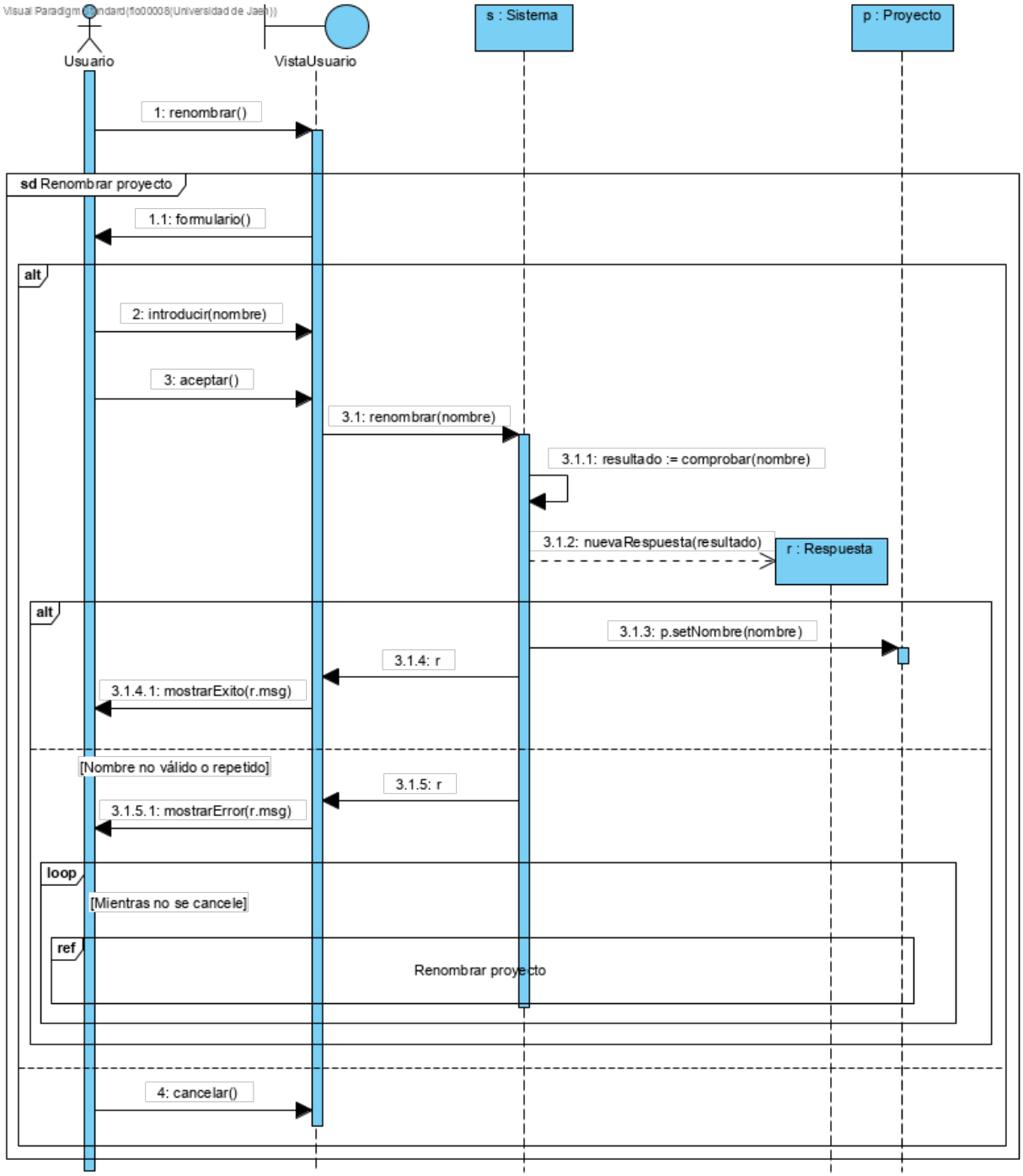

Figura 9: Diagrama de secuencia del caso de uso CU-7

El objetivo del proyecto es contener gran cantidad de imágenes por ese motivo el DS debe plantearse para que se añadan de manera simultanea. Al tratar con múltiples objetos es posible que una imagen ya estuviera insertada, por lo que en ese caso se descarta y se notifica al Usuario. Por el contrario las añadidas correctamente se le muestran en pantalla.

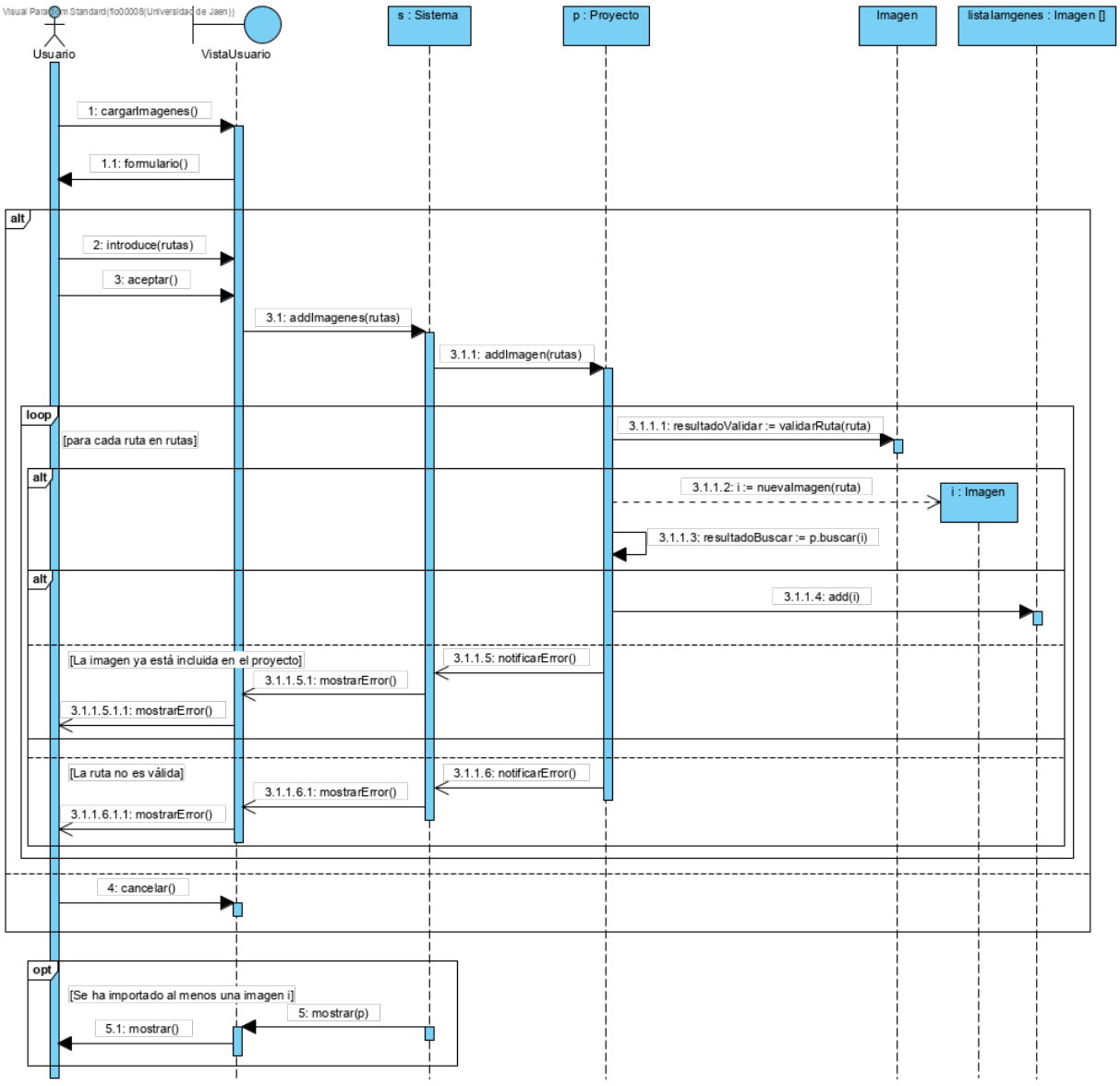

Figura 10: Diagrama de secuencia del caso de uso CU-8

La mecánica de seleccionar objetos simplificará operaciones al no tener que especificar constantemente a qué se está haciendo referencia. Al seleccionar una imagen en este caso debe de buscarse de entre todas las demás del proyecto.

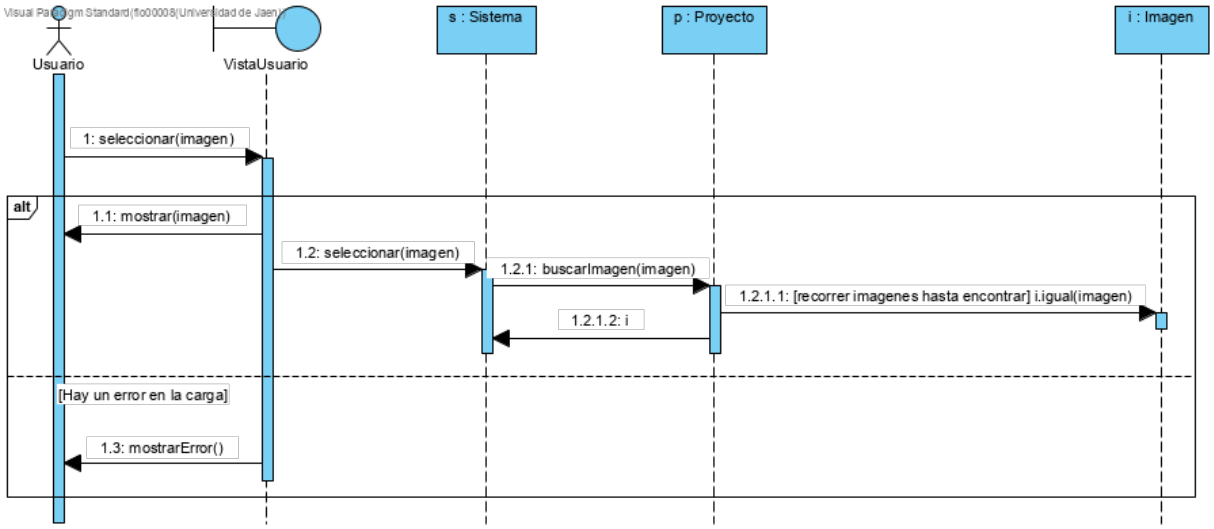

Figura 11: Diagrama de secuencia del caso de uso CU-9

Para la creación de objetos se han utilizado patrones como el patrón creador[5] que asigna la responsabilidad de crear la etiqueta a objetos de tipo proyecto. Con respecto al resto del DS, se trata de lo mismo donde se valida la entrada del Usuario para que no esté repetida y se generen inconsistencias.

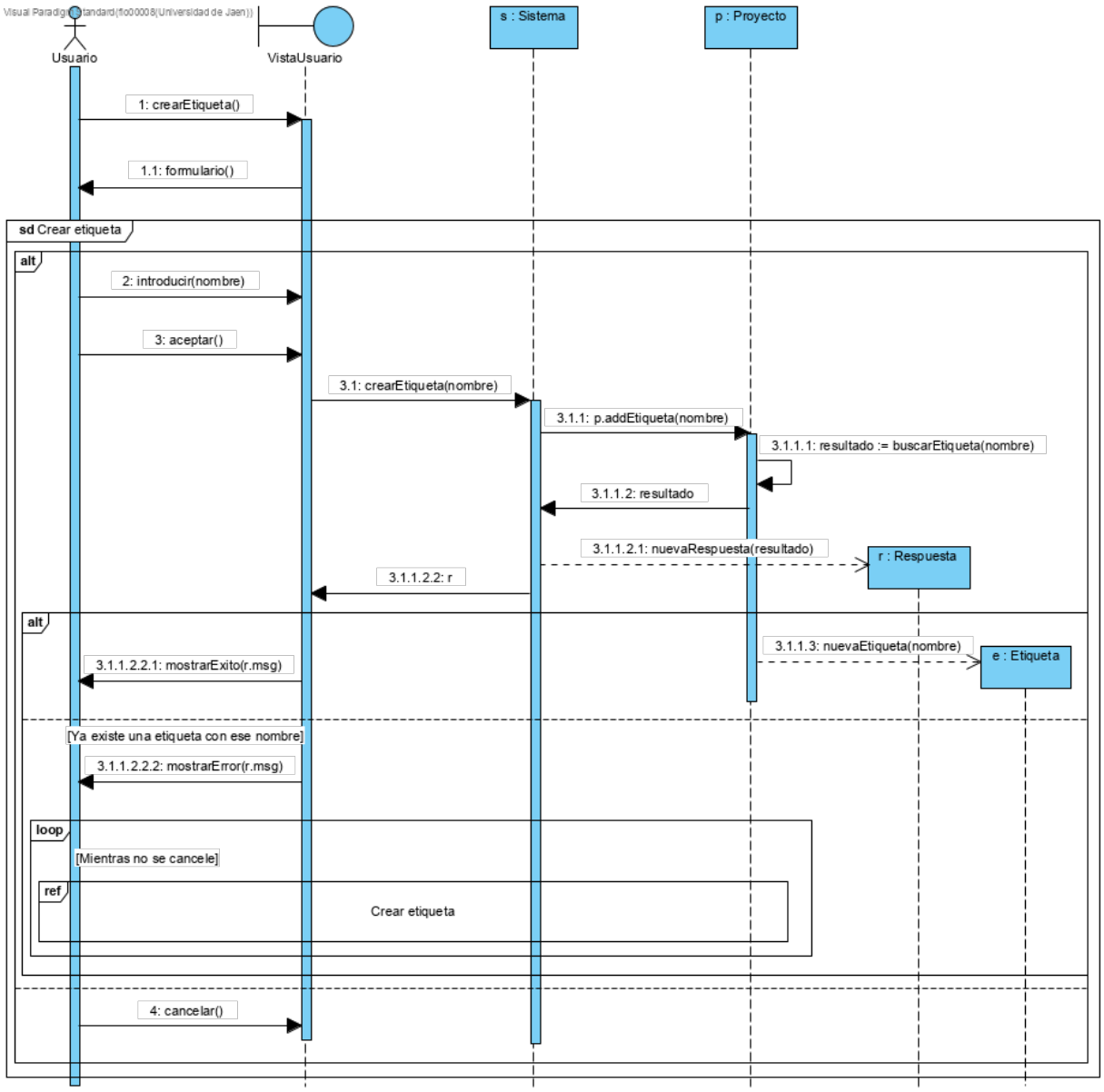

Figura 12: Diagrama de secuencia del caso de uso CU-10

[5] https://www.sourcecodeexamples.net/2018/06/creator-grasp-pattern.html

Para modificar atributos de un objeto el primer paso es seleccionarlo, esto permitirá simplificar operaciones. Una vez seleccionada la etiqueta se modifica el nombre, como las regiones utilizan una referencia a la etiqueta bastaría con modificar el nombre de la original. También comprobando una vez más que no exista otra etiqueta con ese nombre.

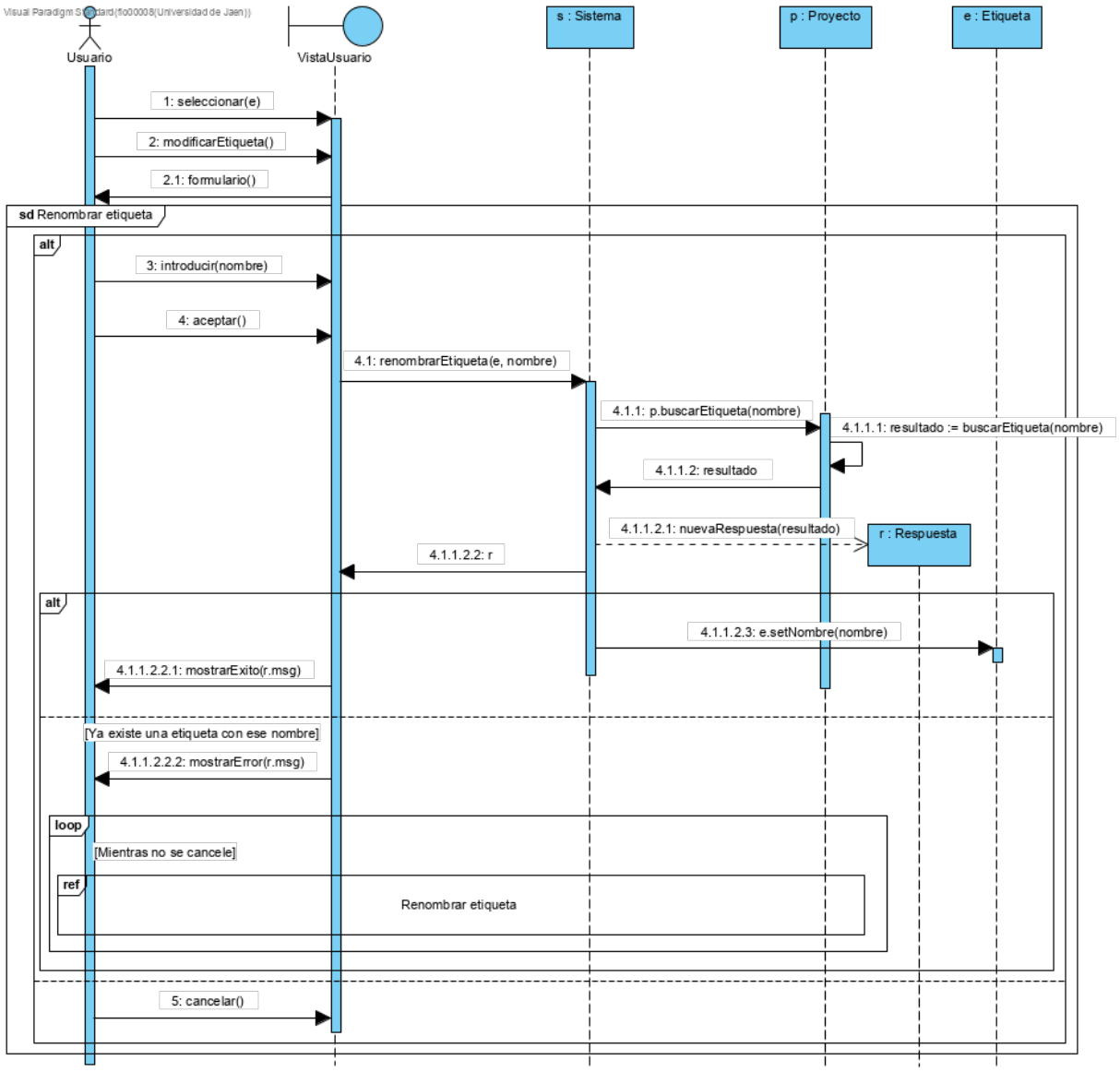

Figura 13: Diagrama de secuencia del caso de uso CU-11

Dado que el encargado de crear la etiqueta es el proyecto, también será el responsable de borrarla. El DS muestra las diferentes llamadas e interacciones para ver si se podría borrar la etiqueta, ya que si existe una región que la utilice no se podrá borrar. En cualquiera de los dos casos se le notifica al usuario como se ha resuelto el escenario.

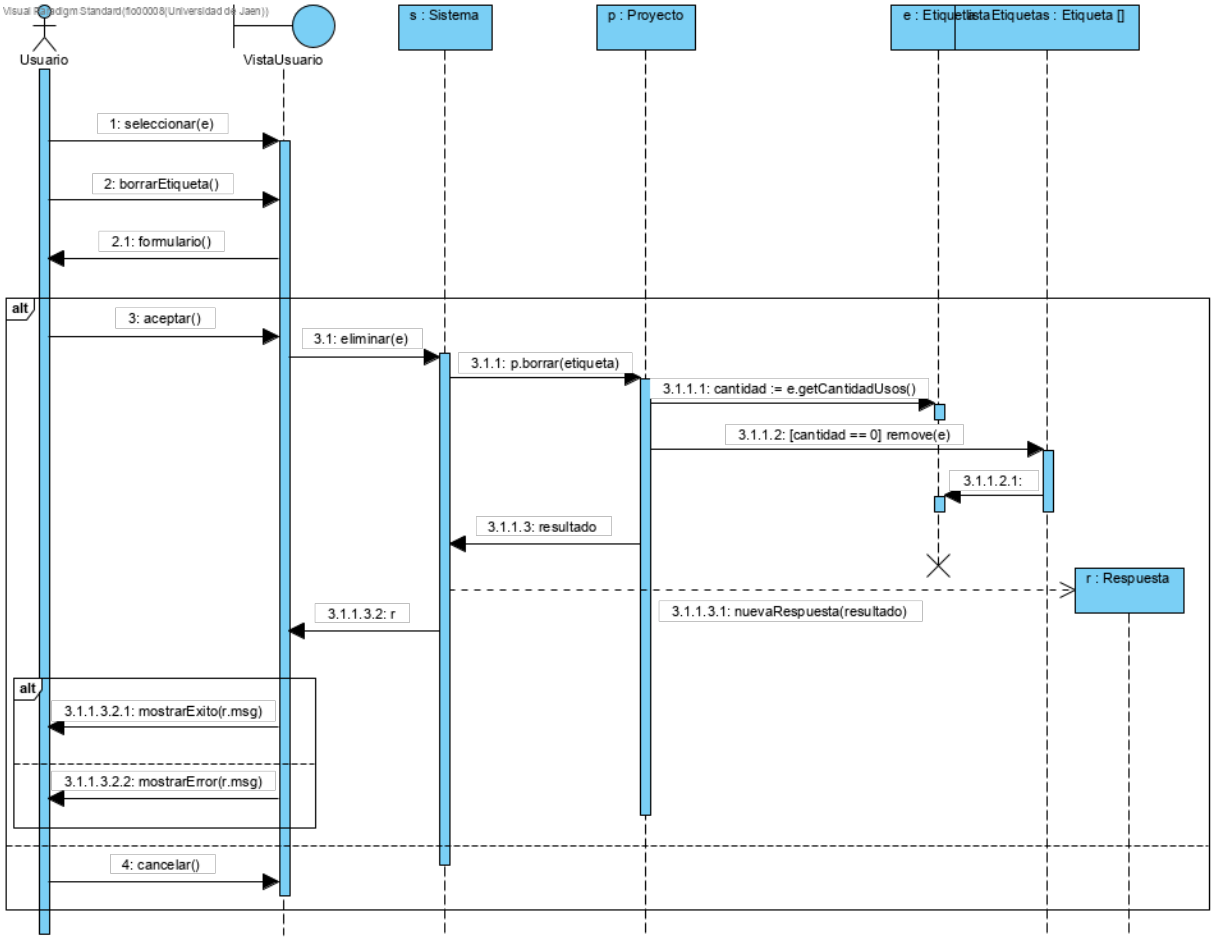

Figura 14: Diagrama de secuencia del caso de uso CU-12

Para asignar una etiqueta a una región debemos comprobar si no la tiene ya asignada. De ser etiquetas distintas se deben de recalcular el número de regiones a la que está asignada una etiqueta, y finalmente se reemplaza.

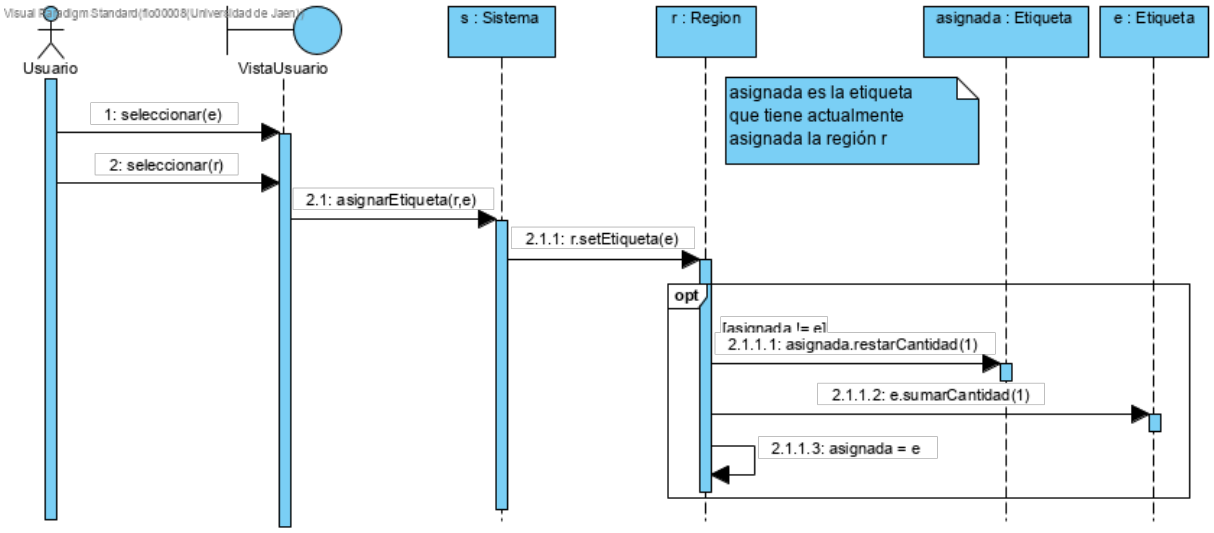

Figura 15: Diagrama de secuencia del caso de uso CU-13

Crear una región también es una operación común por lo que se han simplificado los pasos hasta completarla. En este caso el Usuario selecciona una región de la imagen cargada utilizando el ratón y el sistema se encargará de asociarla junto a la imagen.

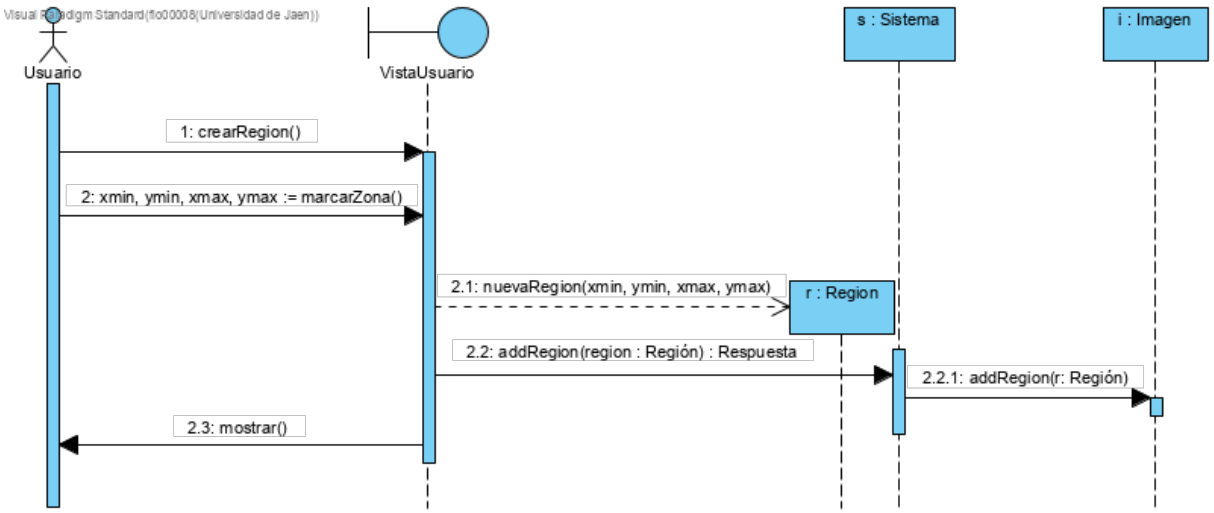

Figura 16: Diagrama de secuencia del caso de uso CU-14

Este es uno de los puntos más interesantes del proyecto ya que agiliza el proceso al encontrar objetos dentro de las imágenes. Para ello englobaremos la complejidad de la detección a través de una clase que al enviarle una imagen devuelve de manera transparente objetos que nuestro sistema ya gestiona de manera natural.

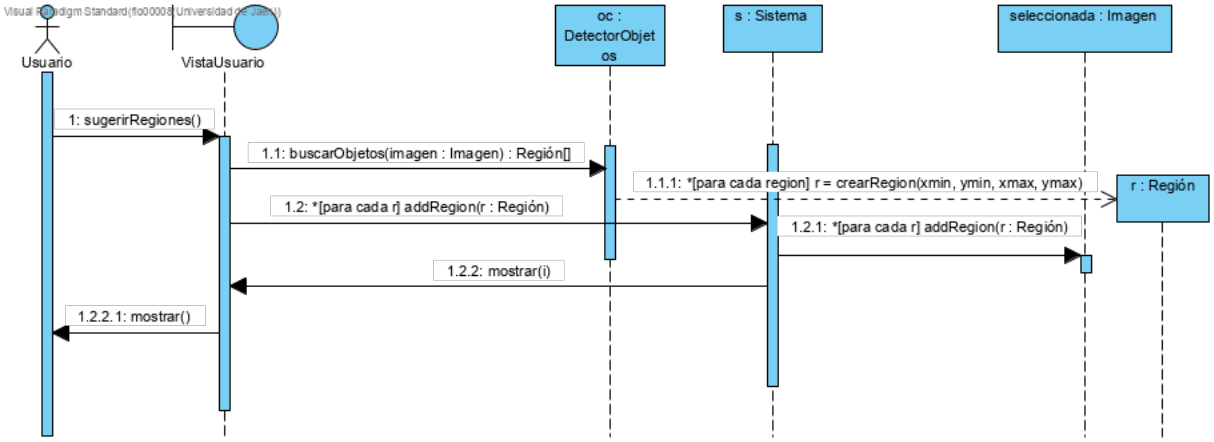

Figura 17: Diagrama de secuencia del caso de uso CU-15

Para seleccionar una región el Usuario decide con cuál quiere trabajar y el sistema internamente la asigna para operaciones como asignar etiqueta (tabla 23) o borrar región (tabla 27).

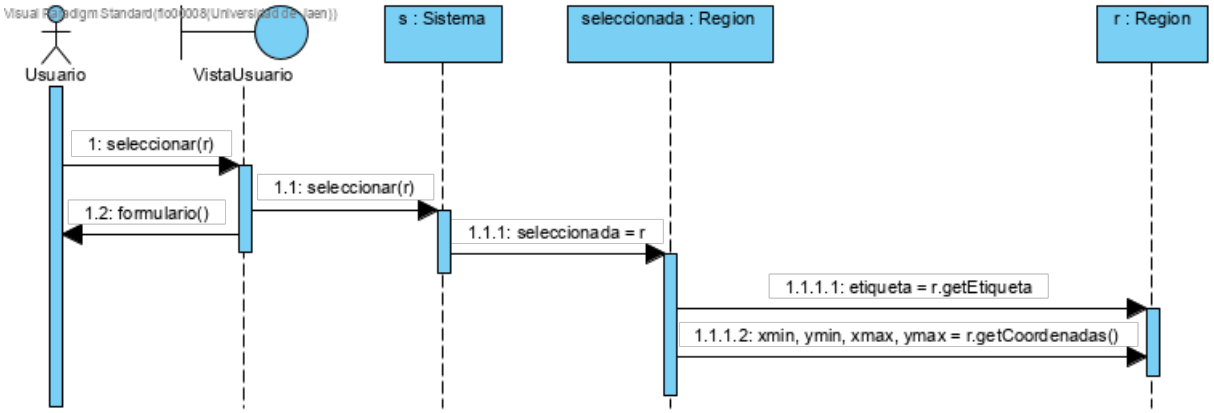

Figura 18: Diagrama de secuencia del caso de uso CU-16

Para borrar una región se hace uso de CU ya vistos como seleccionar imagen 19 y seleccionar región 26, de manera que para este DS sabemos qué región borrar y hacerlo de una manera más intuitiva.

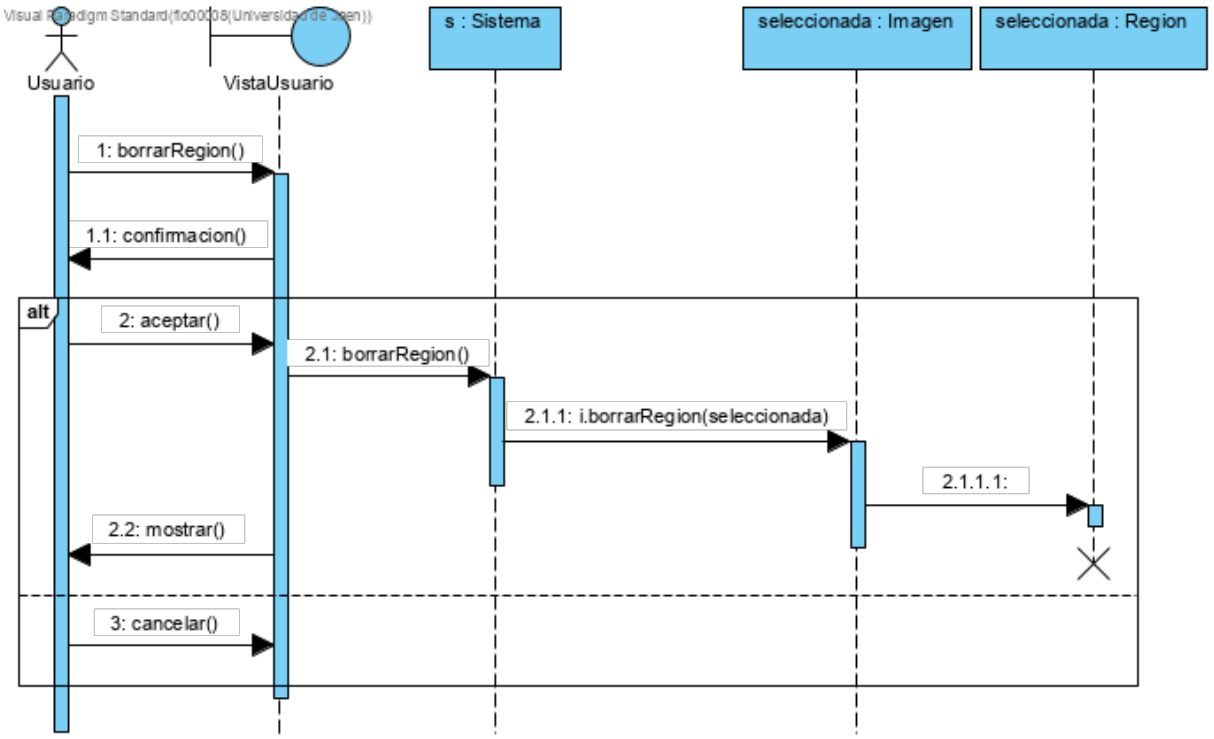

Figura 19: Diagrama de secuencia del caso de uso CU-17

Para modificar una imagen el método más simple es cambiar las coordenadas con esto se podrá modificar el tamaño y la posición. En el DS se especifica un orden para mover cada lado de la región, sin embargo en el diseño final este no será el caso. Una vez modificada la región se harán permanentes los cambios.

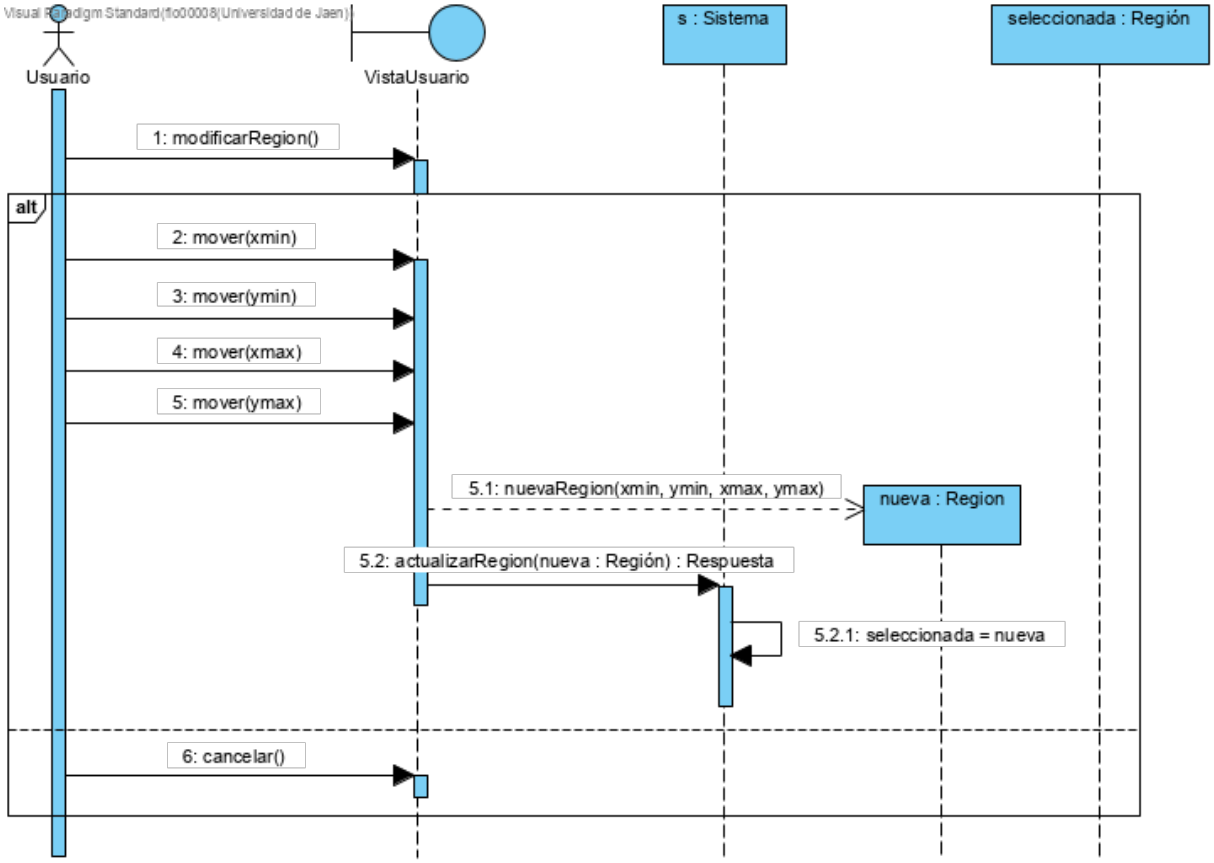

Figura 20: Diagrama de secuencia del caso de uso CU-18

Para borrar los proyectos el encargado final de borrar los datos será el objeto `ga`, ya que es el que gestiona la persistencia. Una vez borrado se le pueden mostrar los cambios al actor.

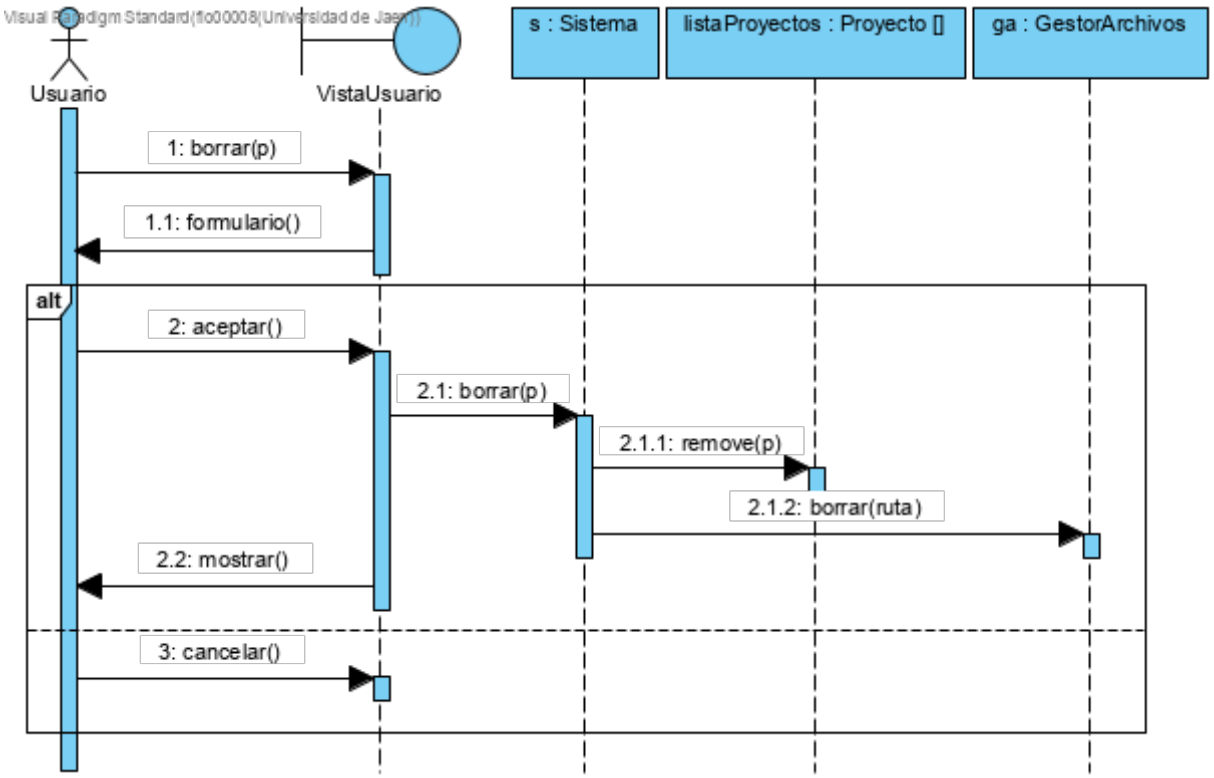

Figura 21: Diagrama de secuencia del caso de uso CU-19

Si necesitamos borrar un imagen el proceso es rápido ya que no tenemos que modificar ningún archivo, podemos directamente borrar el objeto asociado a esa imagen dentro del proyecto y seleccionar directamente la siguiente imagen disponible o notificar que no hay imágenes para etiquetar.

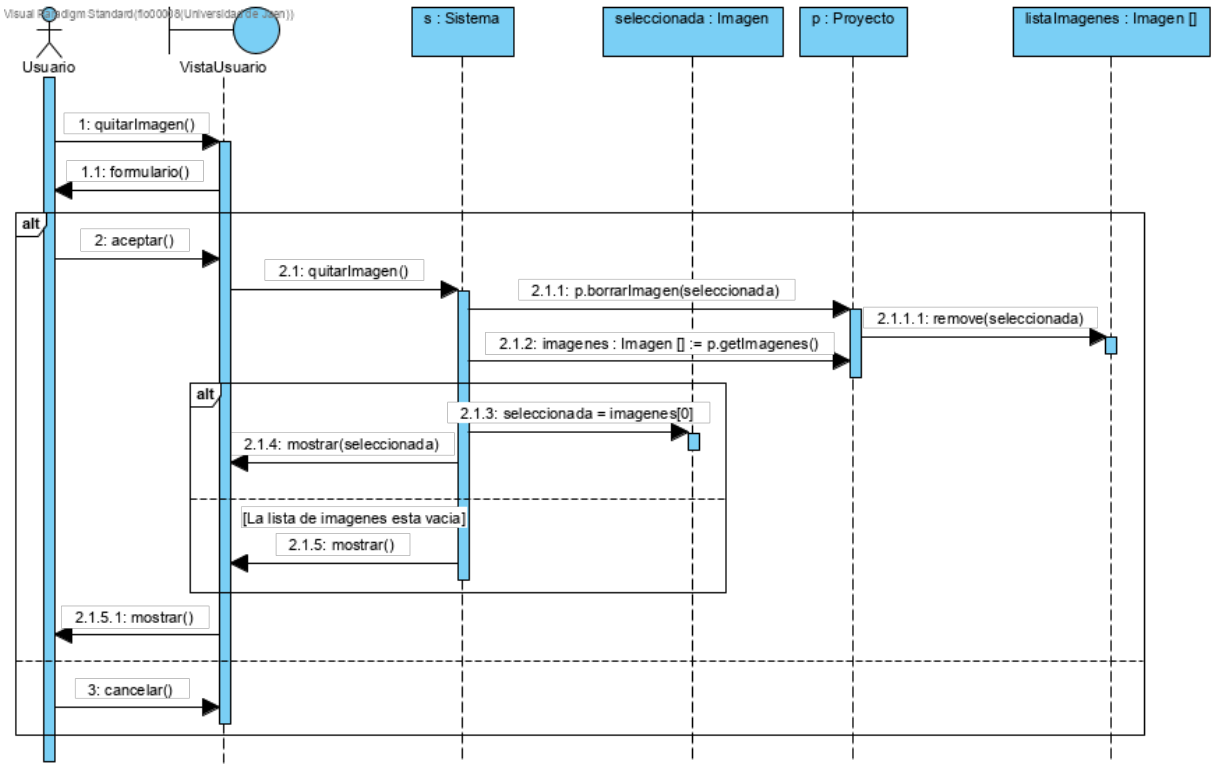

Figura 22: Diagrama de secuencia del caso de uso CU-20

4.2. Diagramas de clases

Una vez modelados los diagramas de secuencia generar un diagrama de clases (figura 23), forma parte de un proceso automático. Cada mensaje corresponde con una relación entre estas clases, el propio mensaje indica cómo debe realizarse esa comunicación. A lo largo de todos los diagramas de secuencia se menciona a VistaUsuario, lo cual hace referencia a las clases `interfaz usuario` y `controlador usuario` (como se explica en 4.3) , que representarán al *framework* el cual utilizará eventos para comunicarse, este es el motivo de que aparezcan métodos `on` y `send`, y que en la interfaz no se definan los componentes que la forman, como botones, entradas de usuario, imágenes, etc.

4.3. Diagrama de paquetes

Para el diagrama de paquetes (figura 24) se han agrupado las clases que tienen un objetivo similar como pueden ser los elementos que componen el modelo, estas clases representan la información con la que trabaja el sistema y por ese motivo se han unido en un mismo paquete. Por otro lado el paquete VistaUsuario y sus clases surge para resolver las interacciones con el Usuario, será el que representa las clases frontera[6] del sistema. Estas clases se comunicarán con el sistema el cual gestiona la información interna y resuelve las peticiones.

[6] `https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/process/modguide/md_bcls.htm`

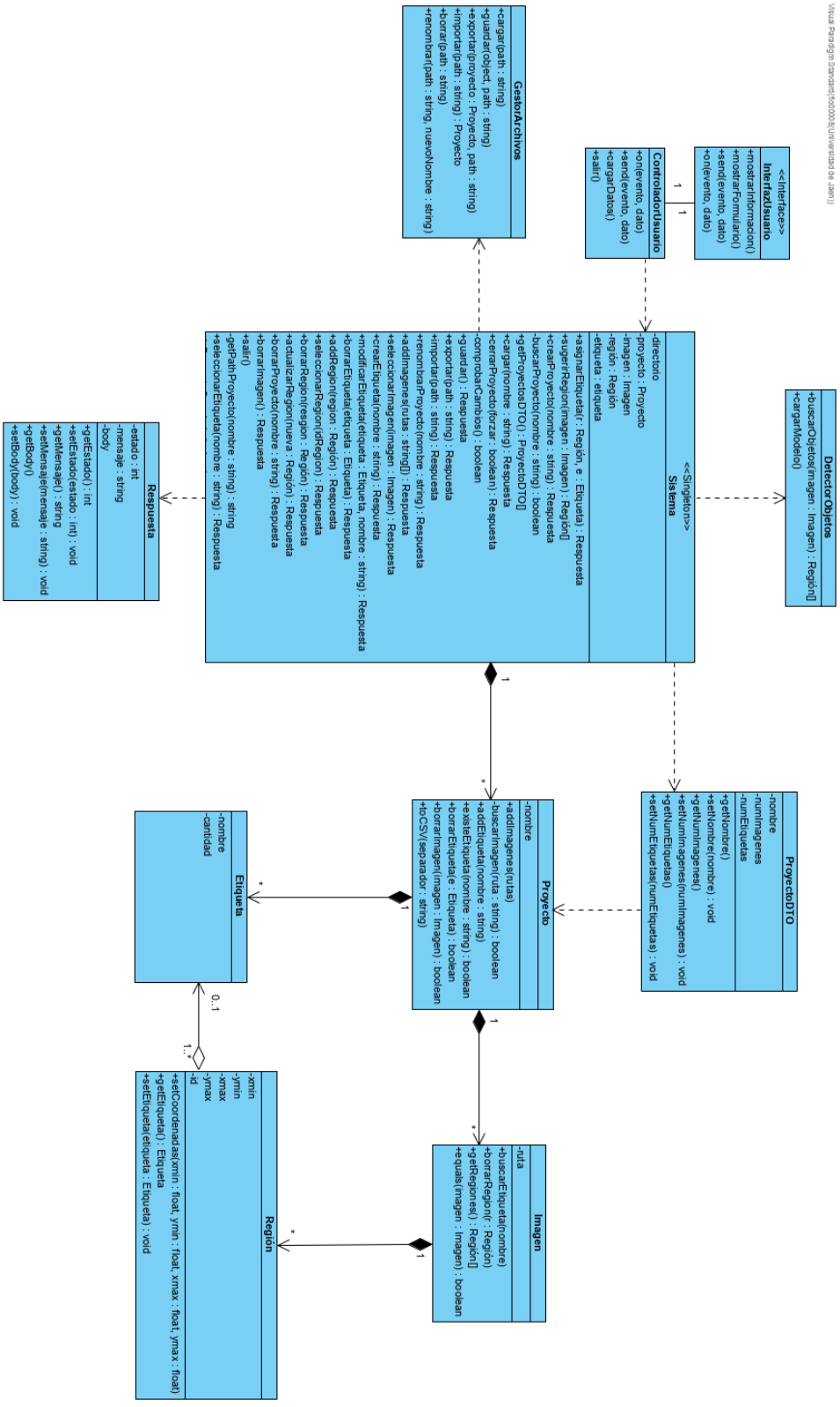

Figura 23: Diagrama de clases

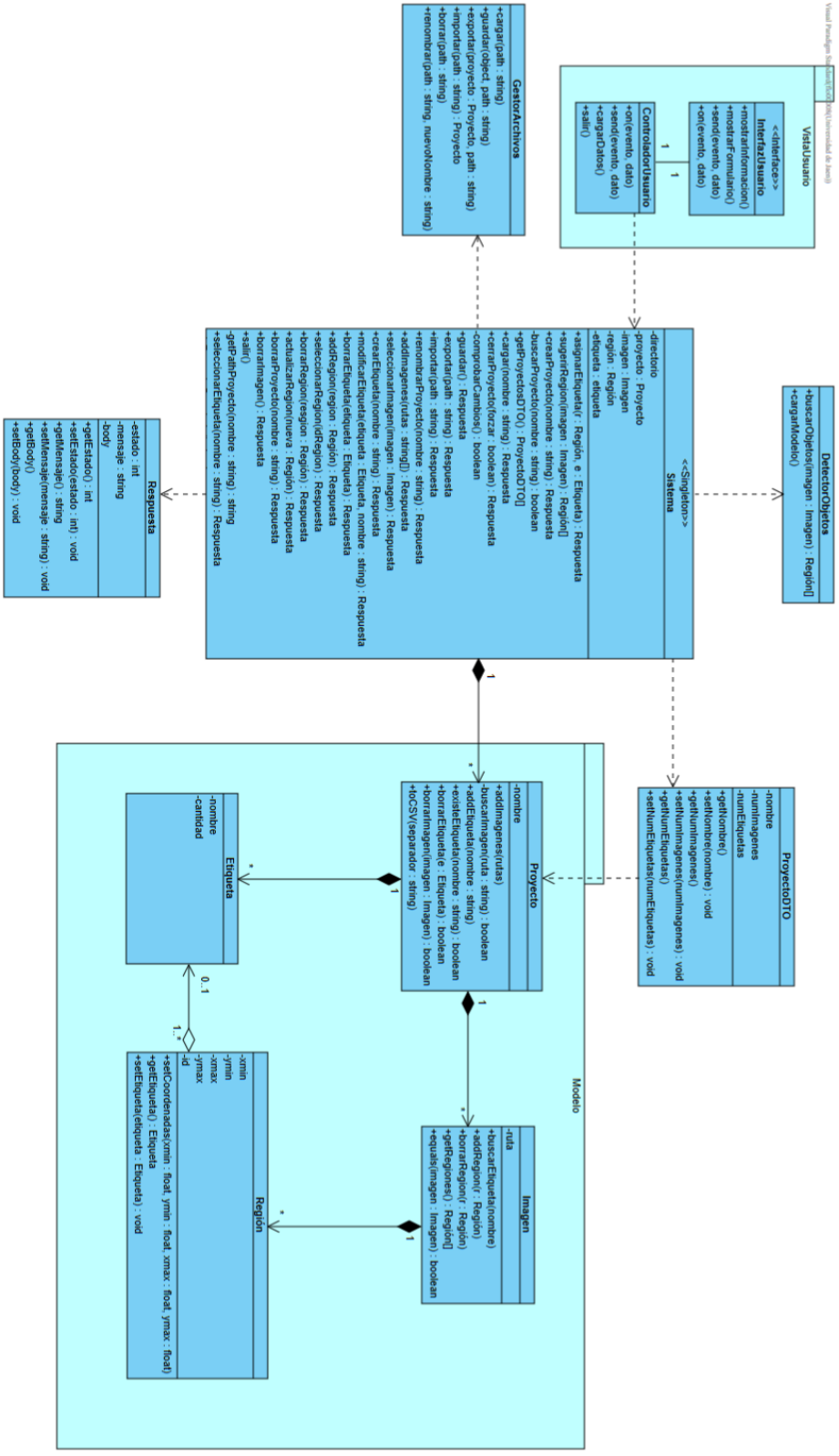

Figura 24: Diagrama de paquetes

5. Desarrollo

5.1. Métodos, herramientas, y prototipos

5.1.1. Métodos

Ya que durante la primera mitad del desarrollo era fácil y rápido escribir las pruebas, se desarrolló con TDD [8]. Sin embargo la segunda mitad, la cual integraba componentes nuevos y exóticos, sumado a que se desarrolló la parte gráfica y se pudo contar con el cliente, se optó por adaptarse y seguir un modelo incremental.

5.1.2. Herramientas

Durante el diseño se ha usado la herramienta Visual Paradigm para crear los diagramas y Axure Cloud para el prototipo, podemos usar la versión gratuita de 30 días o importar desde otras aplicaciones como Figma. En mi caso voy a utilizar la versión de 30 días ya que es más que suficiente para un primer prototipo.

Como se explica en 5.2 se usará ElectronJS como framework para el desarrollo con lo cual es necesario instalar NodeJS v14.7.0, la versión más reciente el día que comenzó la implementación, además como entorno de desarrollo integrado se usará VScode, el cual tiene integrado desarrollo con Git. Para el repositorio remoto se ha utilizado la plataforma Gitlab proporcionada por la Universidad.

5.1.3. Prototipos

Los prototipos creados están basados en herramientas existentes como SuperAnnotate y Cloud Annotations. Existe un enlace [7] a Axure cloud con el prototipo. Como alternativa, las imágenes 25 y 26 son equivalentes.

[7] `https://pj1kvz.axshare.com`

Figura 25: Vista inicial del proyecto

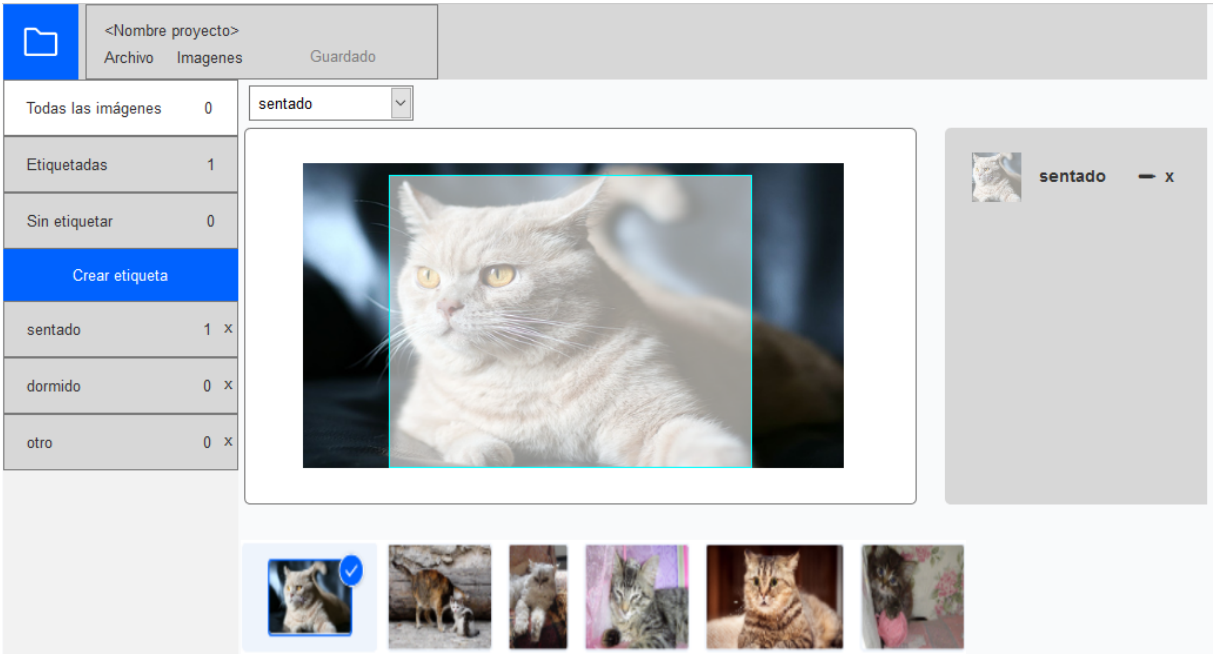

Figura 26: Vista etiquetando

5.2. Estudio de alternativas y viabilidad

En este apartado se discuten las posibles alternativas dentro del entorno de desarrollo, acerca de qué lenguaje usar,qué plataforma y qué *frameworks* utilizar.

Antes de saber el lenguaje de programación es importante decidir el tipo de aplicación

a desarrollar. En la tabla 31 se muestran las ventajas de desarrollar una aplicación de escritorio frente a una web [8], para más información se puede consultar el artículo [13]. A partir de la cual se podrá decidir.

Escritorio	**Web**
Las aplicaciones de escritorio no están forzadas a usar conexión	Las aplicaciones web no necesitan instalación
Son más seguras al tener acceso limitado	Se pueden usar en múltiples ordenadores
Son más baratas a largo plazo	Son más baratas a corto plazo
Su velocidad está limitada a la del ordenador donde se ejecutan	No suponen tanta carga para el ordenador
Te permiten usar versiones antiguas del programa	Mayor accesibilidad
No se necesita soporte de terceros para hacer copias de seguridad	Gran documentación y ejemplos ya desarrollados

Tabla 31: Comparativa entre plataformas de desarrollo

Como se expone es preferible crear una aplicación de escritorio pero también es interesante una web. Por lo que podríamos sacarle el máximo partido si creamos una aplicación de escritorio de la cual se pueda reutilizar código en el caso de que en un futuro se necesite renovar la aplicación o que ambas versiones coexistan. Con el fin de lograr este objetivo nos centraremos en frameworks de aplicaciones de escritorio que puedan satisfacer nuestras necesidades.

Partiendo del problema anteriormente mencionado lo mejor sería usar un framework que utilice HTML para crear la aplicación de escritorio. En nuestro caso se va a usar ElectronJS[9] el cual ha sido usado para crear cientos de aplicaciones conocidas[10] como Facebook Messenger, Twitch, Microsoft Teams, etc. Además presenta, entre otras, la ventaja de que permite generar aplicaciones multiplataforma.

Ya que hemos decido desarrollar en JavaScript necesitamos escoger un modelo de detección de objetos que se base en este lenguaje. Existe un framework popular para el

[8] `https://brandongaille.com/15-desktop-vs-web-application-pros-and-cons/`
[9] `https://www.electronjs.org/`
[10] `https://www.electronjs.org/apps`

desarrollo de aprendizaje automático, llamado Tensorflow[11] el cual también está disponible para JavaScript. Esto permitiría que en el futuro se pudiera usar modelos entrenados localmente gracias al uso de la aplicación.

Para la persistencia de datos se ha buscado la simplicidad y por ello los datos se vuelcan en ficheros, pero gracias a que esta lógica la implementa una sola clase independiente del resto, se podría reemplazar por una base de datos si se utiliza un mapeo objeto-relacional (ORM)[12], y adaptando las clases del modelo.

5.3. Descripción de la solución propuesta

La solución propuesta será una aplicación de escritorio donde cada usuario trabajará con proyectos en los que añadirán imágenes que podrán contener múltiples regiones cada una con una etiqueta, esta es la interacción básica descrita antes en el sistema.

Por último necesitaremos los siguientes paquetes básicos en nuestro proyecto:

- @tensorflow-models/coco-ssd: modelo que usaremos en la detección de objetos.
- @tensorflow/tfjs: necesario para trabajar con el modelo.
- husky: para realizar operaciones como ejecutar las pruebas antes de cada *push*.
- mocha y spectron: herramientas para realizar las pruebas.
- electron-builder: para compilar el proyecto en un ejecutable.

Ya que se está utilizando el gestor de paquetes de NodeJS no es necesario instalarlas manualmente, cabe decir que existen otras dependencias pero esas han sido las principales al comienzo del desarrollo. Además existe una librería no importada como paquete, llamada FabricJS[13] la cual nos ayudará con la interacción de las regiones.

[11] `https://www.tensorflow.org/`
[12] `http://www.agiledata.org/essays/mappingObjects.html`
[13] `http://fabricjs.com/`

5.4. Implementación

El primer paso será empezar a implementar la parte del modelo descrita en el diagrama de clases mostrado en la figura 23, con ello podremos preparar una base donde ejecutar las pruebas. Gracias a husky a partir de ahora con cada *commit* o con cada *push* se le dará formato al código y ejecutarán las pruebas. De este modo el único problema será crear las mejores pruebas posibles.

Una vez sentadas las bases se puede empezar a implementar la funcionalidad del sistema, para ello se ha desarrollado guiándose a partir de los diagramas de secuencia y el diagrama de clases, a lo largo de esta implementación necesitaremos desarrollar clases relacionadas como **GestorArchivos**, a medida que el proyecto crece y se termina la implementación del sistema con sus relaciones, podríamos estimar que se alcanzó el 50 %. A lo largo de este proceso también se han creado las pruebas que permiten validar el funcionamiento del sistema, con lo que podemos asegurar en gran medida que nuestro código estará libre de errores.

Hasta ese momento no se había integrado la parte gráfica en ningún punto, esto ayudará a separar las dos partes del proyecto, lo que reducirá el acoplamiento y aumentará la cohesión [14]. Como la parte gráfica está gestionada por ElectronJS al separar las dos partes aporta ventajas si en se plantea cambiar de *framework* o se busca transformar la aplicación en un servicio web. Es en esta última etapa donde se desarrolla la funcionalidad con el usuario y se generan prototipos, para discutir con el cliente cual debería ser el aspecto de la aplicación.

Con la ventaja de que se está utilizando HTML para modelar las vistas se utilizarán *frameworks* de CSS para asegurar usabilidad y accesibilidad. En este caso se ha utilizado una versión avanzada de un *framework* CSS conocido llamado Bootstrap[15]. El que se ha utilizado se llama *Material Design for Bootstrap* (MDB)[16], en vez de usar la versión

[14] `https://www.disrupciontecnologica.com/acoplamiento-y-cohesion`
[15] `https://getbootstrap.com/`
[16] `https://mdbootstrap.com/`

original. Al ser una versión extendida, existe una versión de pago sin embargo la parte gratuita es más que suficiente para la aplicación, pero los iconos serán de la página original ya que estos son más versátiles.

5.5. Pruebas

En el caso de este proyecto se han realizado las pruebas de manera continua inspirándose en el desarrollo guiado por pruebas, donde el desarrollo comienza por escribir las pruebas, el siguiente paso es conseguir que pasen las pruebas y finalmente se refactoriza el código. Más información acerca de como se entrelazan TDD con JavaScript en [8].

5.5.1. Pruebas de verificación del sistema

Las pruebas de verificación se basan en comprobar que estamos desarrollando lo que ha pedido el cliente. En esta parte se comprobó que se cumplen los requisitos funcionales y no funcionales de la especificación inicial 3.1.

5.5.2. Pruebas de validación del sistema

La implantación del sistema se ha desarrollado principalmente a la par con pruebas del sistema de manera que cada parte se prueba en detalle, para asegurar el correcto funcionamiento por separado y de manera conjunta. Mencionar que se ha recurrido a las pruebas unitarias para verificar funciones complejas. Además para las pruebas del sistema lanzando la aplicación solo se implementaron las básicas como comprobar que solo se abre una ventana o que no se muestra el panel de desarrollador, esto se debe a la excesiva complejidad y extensión de las pruebas automáticas lo cual resultaría en pruebas que tardarían demasiado en ejecutarse y ser desarrolladas, por este motivo en esta etapa se optó por validar manualmente.

5.5.3. Validación de la usabilidad del sistema

La usabilidad es una parte crucial en las aplicaciones ya que aportan utilidad, efectividad, eficiencia a la hora de utilizar la aplicación de una manera intuitiva y simple en

la cual usuarios que nunca han visto la aplicación previamente saben como utilizarla. Con este objetivo se podrían utilizar herramientas integradas que realizan este tipo de pruebas[17], sin embargo al no estar en un navegador esa funcionalidad no nos sirve. Esto no supone un problema para nosotros ya que al utilizar estilos aportados por un framework cuestiones acerca de la accesibilidad están resueltas.

[17] `https://developers.google.com/web/tools/chrome-devtools/accessibility/reference`

6. Conclusión y trabajos futuros

Como conclusión final para este proyecto he de resaltar que para la organización de las tareas (tabla 3) ha sido de mucha utilidad considerar el tiempo estimado para cada una de ellas, y que junto con las prioridades ha ayudado mucho para decidir cuáles desarrollar primero, aunque no se mencione durante la implementación. También por la limitación de tiempo algunas de las funcionalidades no se han llegado ha desarrollar todo lo que me gustaría, sin embargo estoy muy contento y motivado con el resultado final, y ver como todas las tecnologías se unen junto con algoritmos de aprendizaje automático para hacer más interesante la aplicación final.

En cuanto a propuestas para siguientes trabajos, sería interesante una nueva versión de la aplicación donde cada proyecto tenga un modelo independiente, para que a medida que se etiquetan nuevas imágenes el proyecto se especialice gradualmente de manera que se dé un nuevo paso en la automatización y ayude aún más al usuario con detecciones más precisas.

7. Apéndices

A continuación se introduce documentación adicional de utilidad para comprender correctamente el proyecto.

7.1. Guía original del Trabajo Fin de Titulo

En el caso de que no se pueda consultar online [18] la guía era la siguiente: (Cód.: 19/20-2642) Asistente para el etiquetado de objetos presentes en imágenes
Tutor del TFG: FRANCISCO CHARTE OJEDA
Modalidad: Proyecto de Ingeniería — Tipo: TFG Específico
Número máximo de estudiantes: 1 (1 asignados)
Idioma: Castellano
Asignado al alumno con DNI:53914392T
Segundo tutor del TFG: MARÍA DOLORES PÉREZ GODOY

Los sistemas de inteligencia artificial capaces de identificar diferentes tipos de elementos presentes en imágenes han de aprender de ejemplos que han sido previamente etiquetados de forma manual. El presente TFG tiene por objetivo el desarrollo de un sistema que sirva de asistencia a esta tarea, identificando las zonas de la imagen donde hay presentes objetos e indicando de qué clase de objetos se trata.

Durante el desarrollo de este proyecto el estudiante pondrá en práctica los conocimientos adquiridos en asignaturas como Procesamiento de información visual o Minería de datos (en caso de haberlas cursado), así como las herramientas de ingeniería del software necesarias para gestionar un proyecto software de principio a fin.

Conocimientos Previos

El TFG demandará del estudiante los conocimientos propios de la titulación

[18] `http://eps-anterior.ujaen.es/TFGtemporal/mostrarTFG.php?id=2642`

Objetivos del TFG

- Obtención de una visión general del campo de la Minería de datos y, en particular, del aprendizaje supervisado a partir de datos etiquetados.
- Estudio de aplicaciones prácticas de las técnicas de visión artificial, tanto clásicas como basadas en los modernos métodos de aprendizaje profundo.
- Definición a priori de un conjunto de categorías/clases que se emplearán para proceder al etiquetado de lotes de imágenes.
- Diseño de un procedimiento que facilite la detección de objetos presentes en imágenes, estableciendo el área exacta que ocupan y la etiqueta que les corresponde.

Metodología a Desarrollar

- Revisión bibliográfica sobre técnicas de Minería de datos y detección de objetos.
- Análisis de requisitos del sistema a desarrollar.
- Especificación y diseño del asistente para el etiquetado de objetos en imágenes.
- Implementación del sistema empleando la tecnología más apropiada.
- Realización de pruebas.
- Redacción de documentación: manuales de instalación y uso.
- Redacción de la memoria del TFG.

Documentos y Formatos de Entrega

- Prototipo del software desarrollado a lo largo del proyecto.
- Memoria del TFG en formato digital.

8. Manuales

8.1. Instalación

8.1.1. Requisitos mínimos

- SO: Windows de 64 bit o distribución GNU/Linux de 64 bits.
- Procesador: Core i3-370M o equivalente.
- Memoria: 1 GB de RAM.
- Gráficos: Intel HD Graphics 630.
- Almacenamiento: 250 MB de espacio disponible.

Antes de empezar la instalación es importante saber que ya que el programa almacena las rutas absolutas a las imágenes pero no las clona, una vez creado un proyecto. Si se exporta a otra máquina sin las imágenes incluidas, dará fallos al cargarse.

8.1.2. Windows

Para realizar la instalación en windows, solo es necesario extraer el contenido del fichero .zip del proyecto.

Es aconsejable descomprimir el fichero en una carpeta individual, al haber múltiples archivos. En mi caso voy a descomprimirlo en el escritorio para tener un acceso más rápido.

Una vez finalizada la descompresión se mostrarán los mismos ficheros y archivos que en la imagen 29.

En este caso para lanzar la aplicación solo es necesario ejecutar el archivo `app-tfg`. La carpeta `data` es donde se almacenarán todos los datos de la aplicación, por lo que en el caso de trabajar con un gran volumen de datos es importante asegurarse de que el proyecto tendrá espacio suficiente. Dentro de `settings.json` se pueden configurar comportamiento básico del sistema, el cual se explica en el apartado 8.2 Configuración.

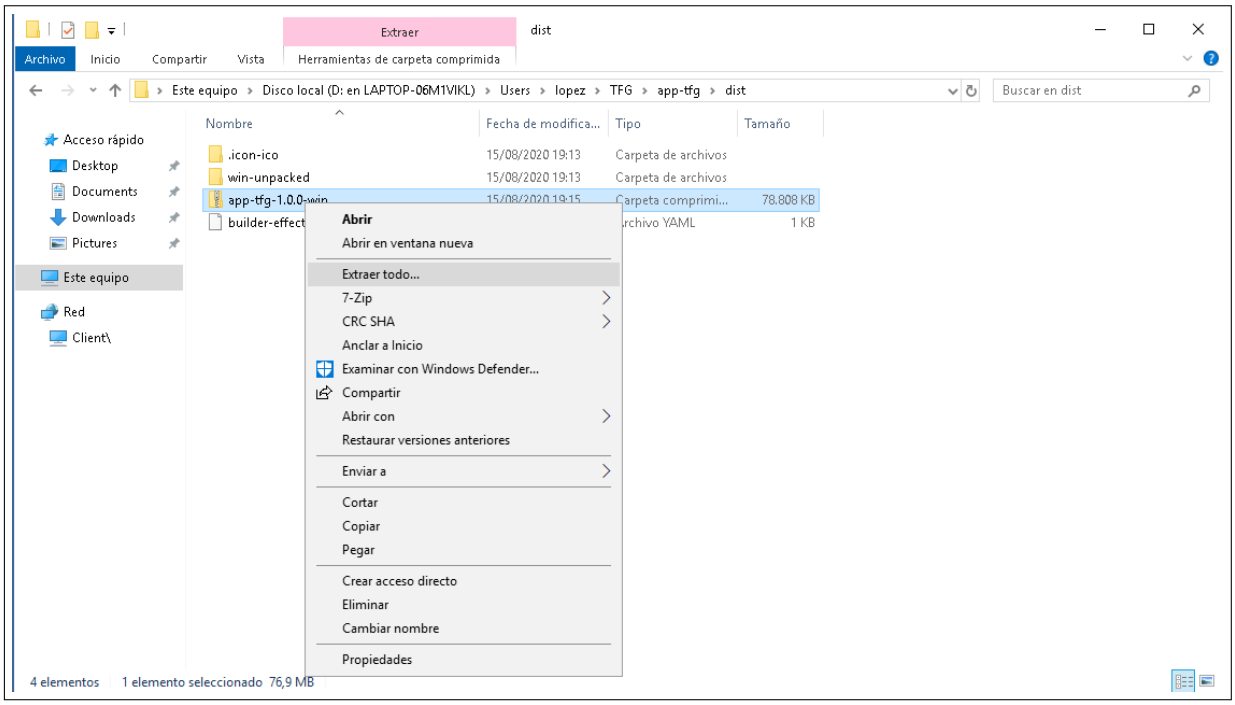

Figura 27: Descomprimiendo proyecto - instalación Windows

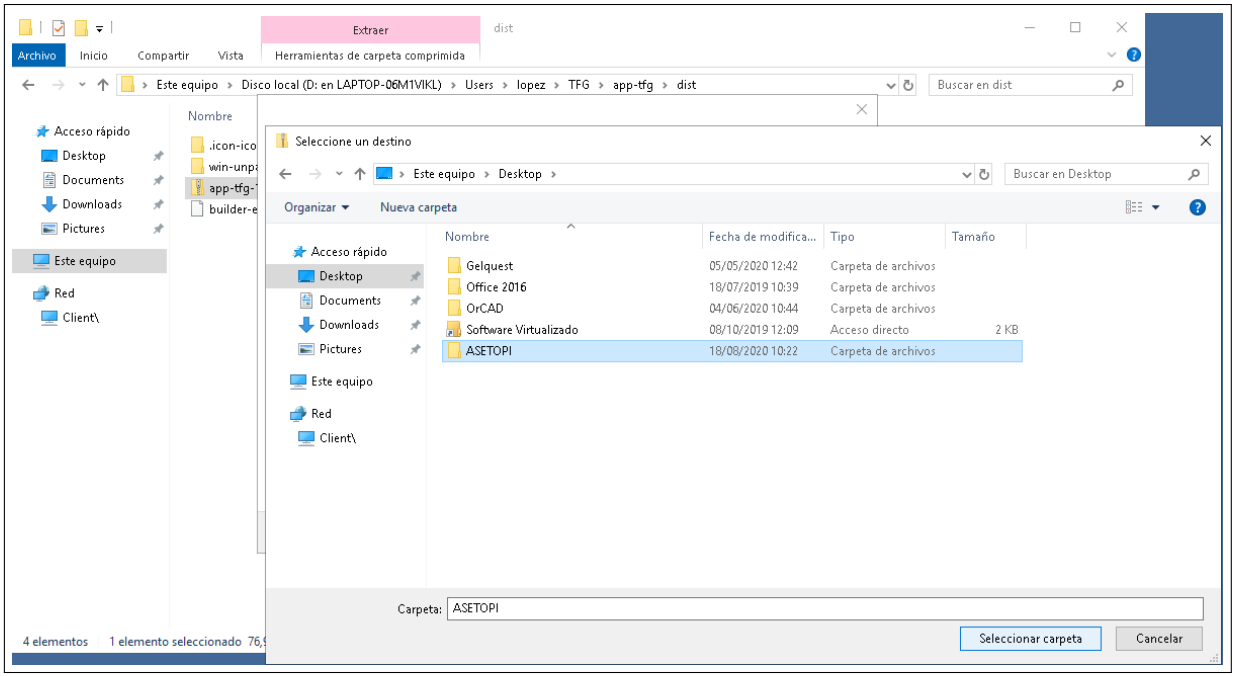

Figura 28: Seleccionando carpeta destino - instalación Windows

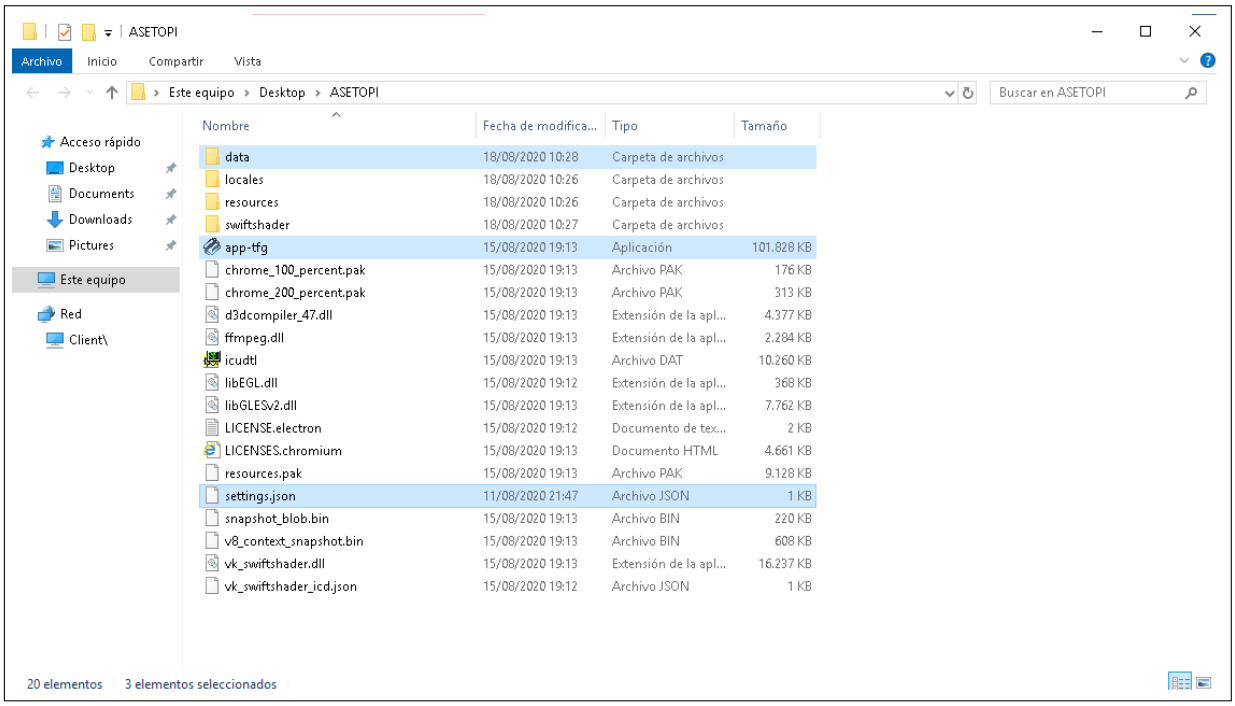

Figura 29: Instalación completa - instalación Windows

8.1.3. Linux

Para la instalación se usará Ubuntu 20.04 LTS. Los pasos a seguir son los mismos pero con diferente interfaz.

El primer paso es extraer los datos del fichero

En mi caso voy a volver a descomprimirlo en el escritorio, pero cualquier carpeta sirve.

Una vez terminado, se mostrarán los siguientes archivos.

En este punto necesitamos darle permisos de ejecución al programa con el comando

```
$ chmod a+x app-tfg.AppImage
```

De modo que quede como en la figura 33.

Ahora se puede ejecutar el programa.

Una vez terminada la instalación al ejecutar la aplicación se mostrará una pantalla de carga tras la cual se verán los proyectos creados. Al ser una instalación limpia se mostrará como aparecen en la imagen 36.

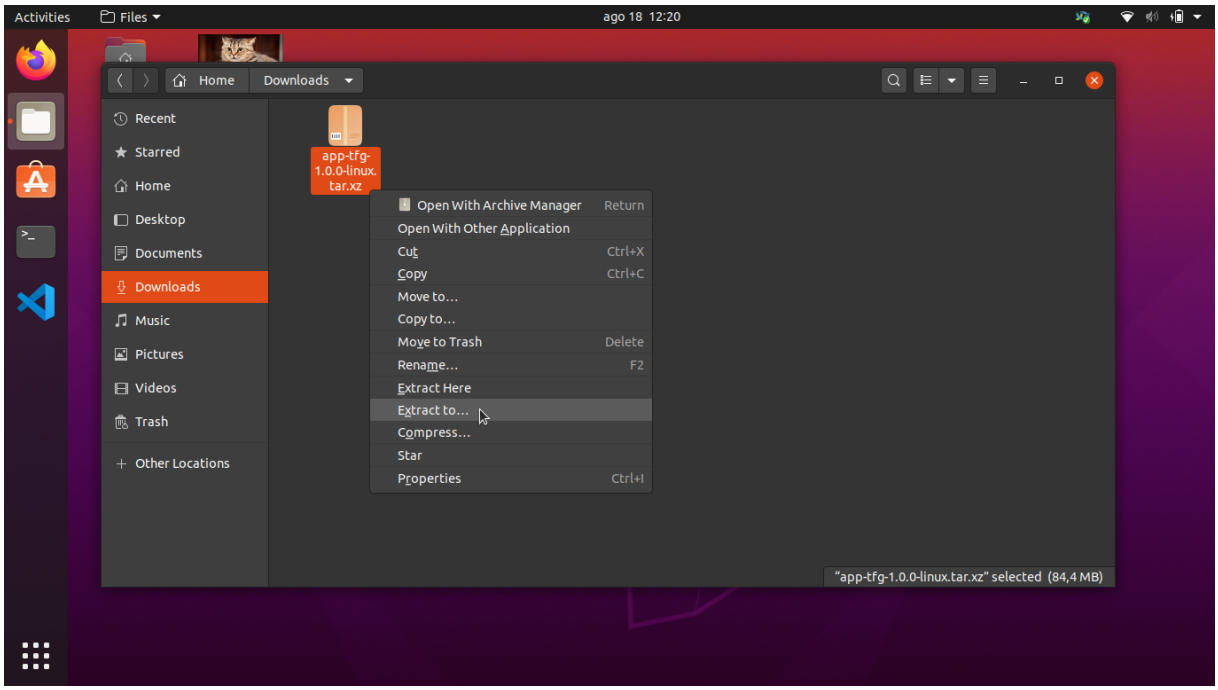

Figura 30: Descomprimiendo proyecto - instalación Linux

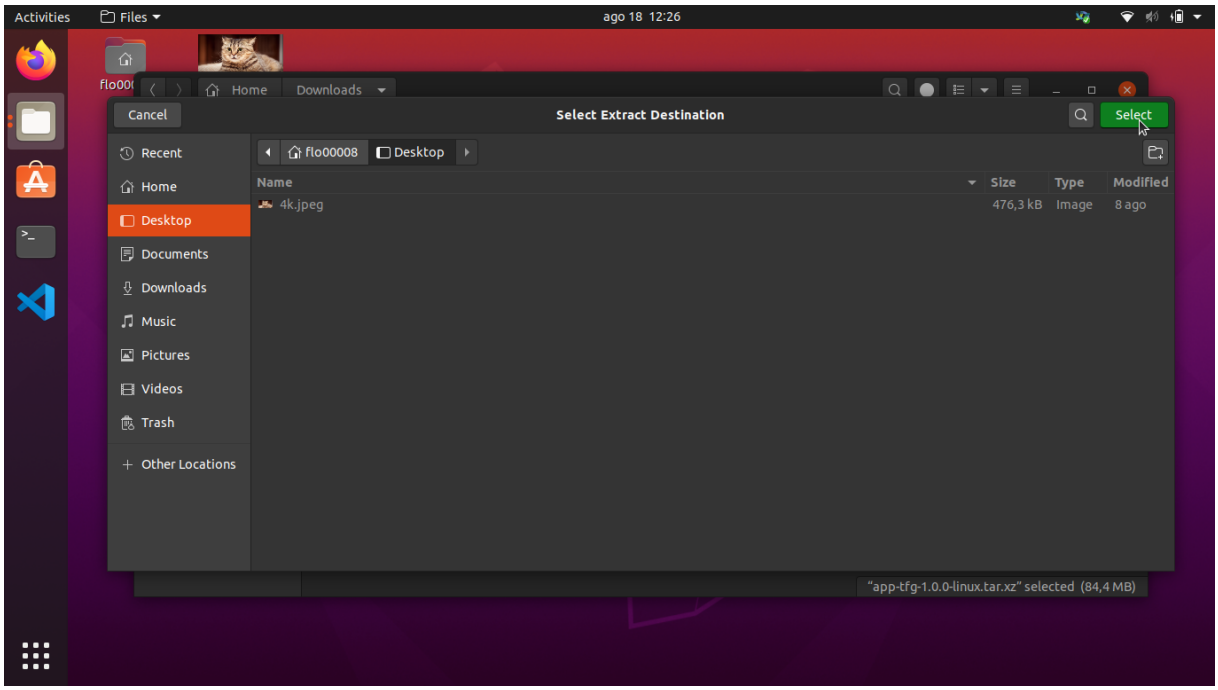

Figura 31: Seleccionando carpeta destino - instalación Linux

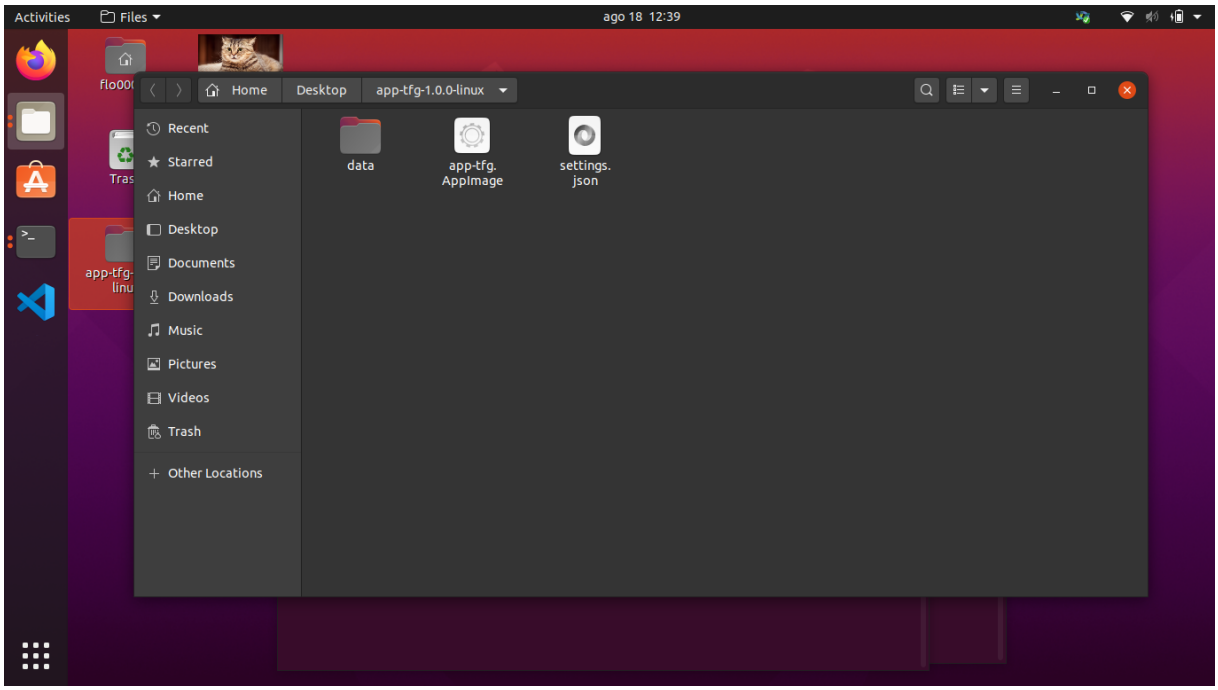

Figura 32: Instalación completa - instalación Linux

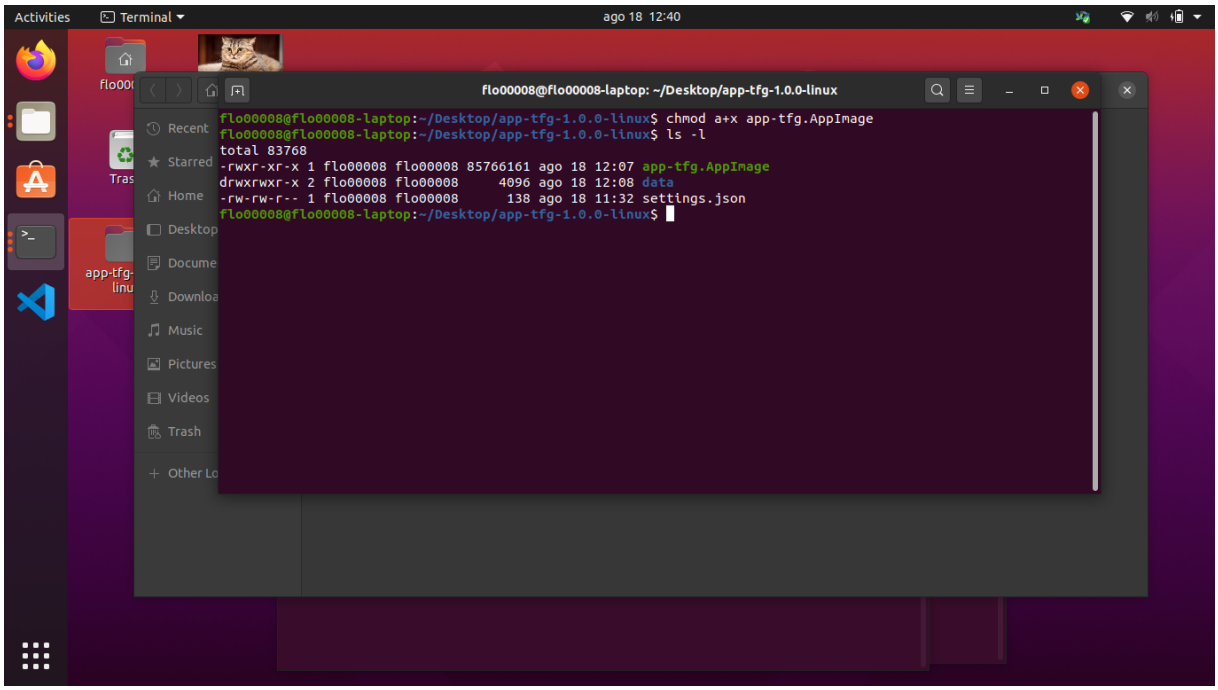

Figura 33: Configurando instalación - instalación Linux

8.2. Configuración

Para cambiar la configuración de la aplicación, es necesario modificar el fichero `settings.json` que acompaña a cada ejecutable. Una vez modificados los datos de este

fichero para que tengan efecto deberá de reiniciarse la aplicación.

Al abrir el fichero se cargará la configuración por defecto como se muestra en la imagen 34.

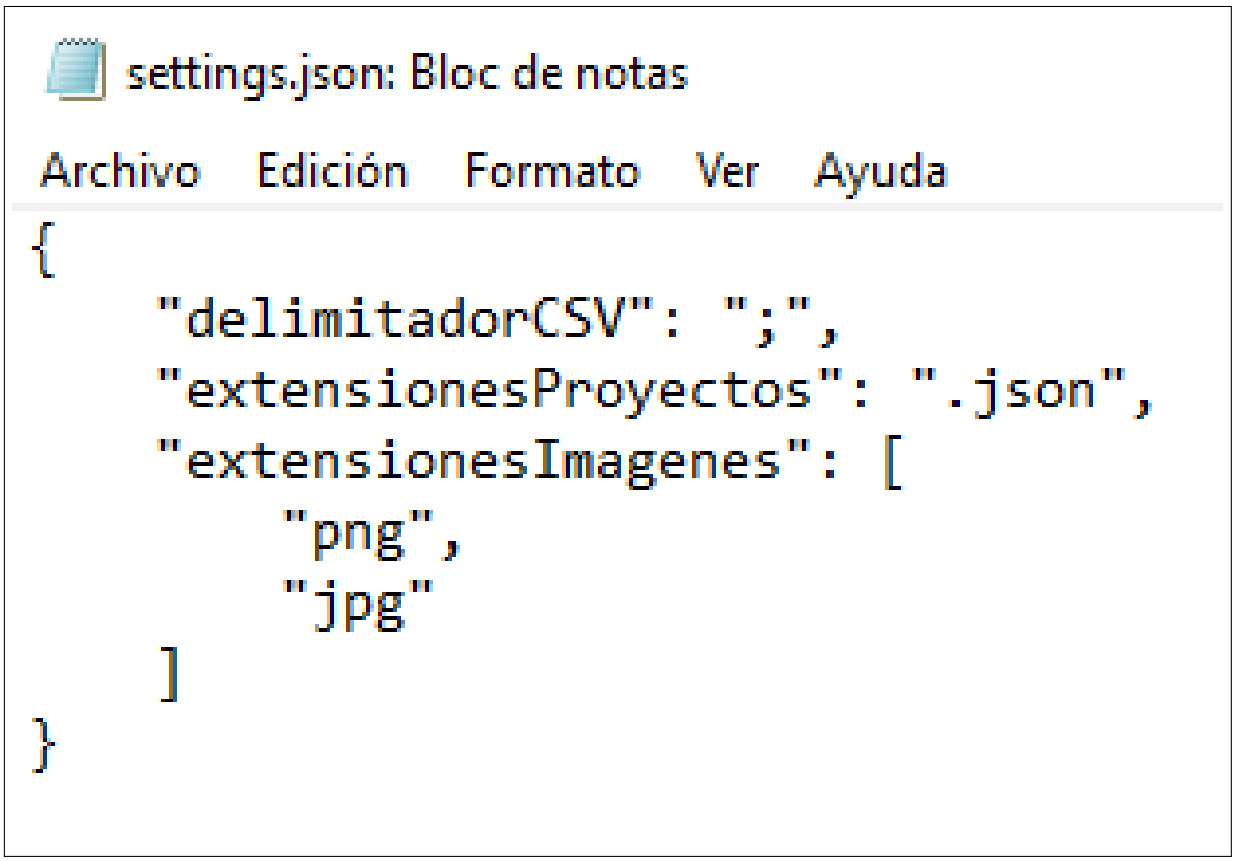

Figura 34: Fichero configuración

- `delimitadorCSV`: será el separador utilizado al exportar el proyecto en formato CSV. Es aconsejable que este valor solo sea un carácter aunque se permita usar varios, ya que al generar el archivo CSV [19] si se encuentra este valor se escapará con \.

- `extensionesProyectos`: el sistema almacena los datos de los proyectos de manera persistente por lo que es necesario generar archivos. Por defecto estos archivos tendrán formato JSON [20] JSON

[19] Documentación acerca de como generar CSV `https://tools.ietf.org/html/rfc4180`
[20] Documentación acerca de JSON `https://www.json.org/json-es.html`

- `extensionesImagenes`: esta opción es el filtro que limita que tipo de imágenes se pueden añadir al sistema. Los posibles valores están definidos en la documentación de Mozilla. Es importante recordar que se deben usar minúsculas, ya que se trata de extensiones de archivos, y que cambiar la extensión no hará que una imagen no soportada se muestre.

8.3. Manual de usuario

El manual de uso se documentará sobre la instalación de Windows por comodidad, ya que la única diferencia es el formato de las ventanas emergentes.

8.3.1. Inicio de la aplicación

Al ejecutar la aplicación se mostrará la siguiente pantalla de carga. Esto se debe a que la aplicación está preparando el modelo de detección de modelos y necesita un tiempo.

Figura 35: Iniciando aplicación

Una vez cargado, al usuario se le mostrará la vista inicial vacía. Como se muestra en la imagen 36. En este punto el usuario solo puede crear un nuevo proyecto.

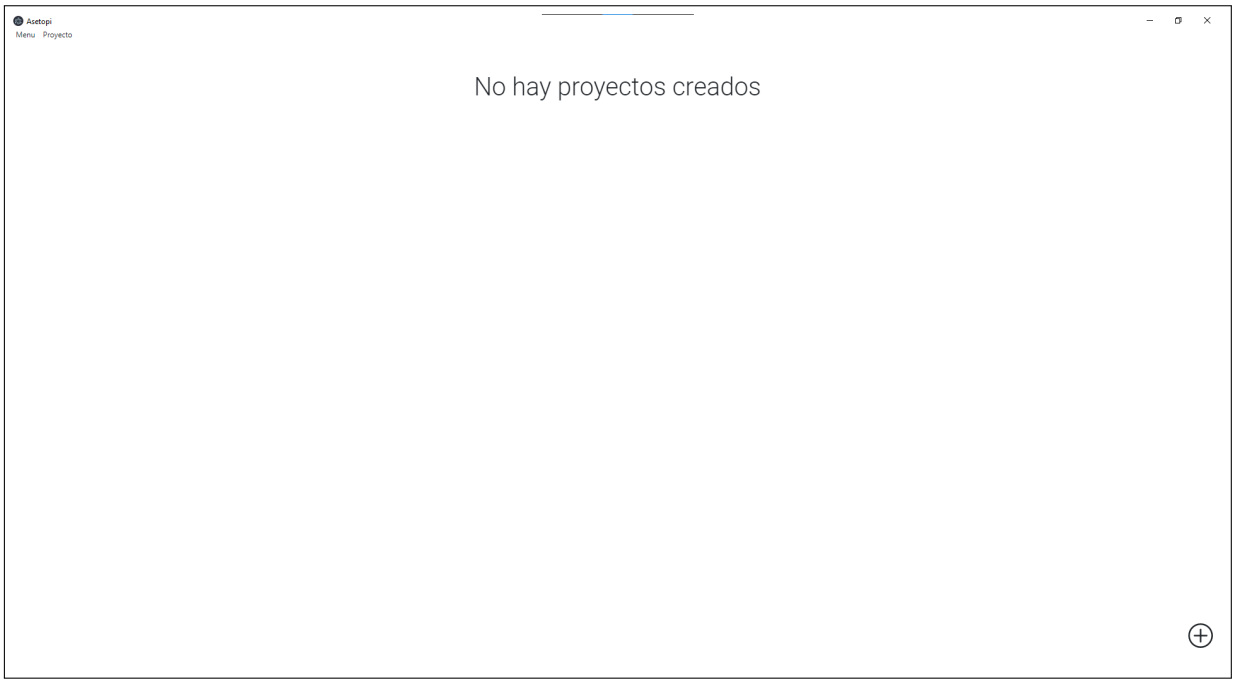

Figura 36: Aplicación sin proyectos

8.3.2. Creación de un proyecto

Crear un proyecto puede hacerse a través del icono abajo a la derecha o del menú en la sección `proyecto`. Al realizar cualquiera de estas dos acciones se mostrará un formulario. En este caso vamos a hacer un ejemplo con imágenes de gatos por lo que llamaré al proyecto **Gatos**.

Una vez creado un proyecto, se mostrará y se indicará al usuario que la operación ha tenido éxito.

8.3.3. Gestionar proyecto

Con proyectos creados, se pueden borrar o renombrar a través de los botones a la derecha de proyecto. Una vez tengamos el proyecto creado ya podemos cargarlo haciendo clic en él. Si el proyecto contiene imágenes tardará unos segundos más en cargar como en mi caso está vacío se mostrará la siguiente pantalla.

Al cargar un proyecto se muestra una vista dividida en tres columnas, la parte de la izquierda donde se mostrarán estadísticas como el número de imágenes o el número de imágenes etiquetadas, además de todas las etiquetas creadas. En el centro la imagen

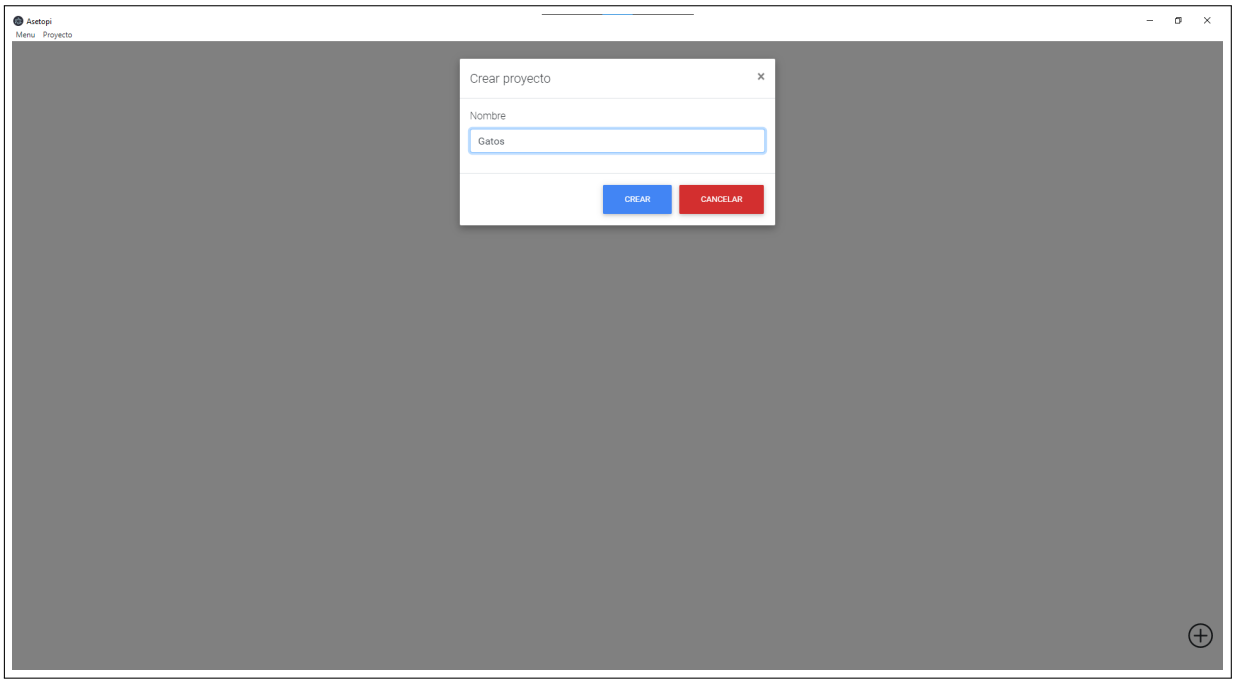

Figura 37: Creando nuevo proyecto

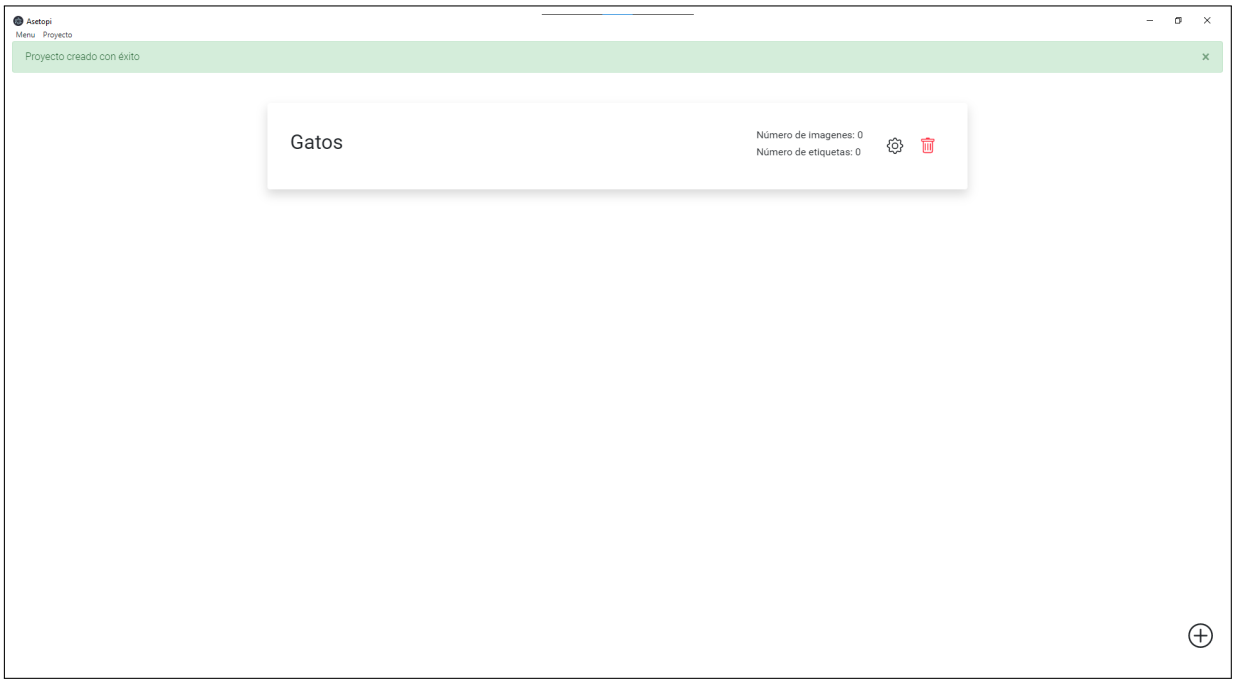

Figura 38: Nuevo proyecto creado

seleccionada y debajo de ella todas la imágenes del proyecto. Finalmente en la parte derecha se podrán ver las regiones de la imagen seleccionada y la etiqueta que tienen

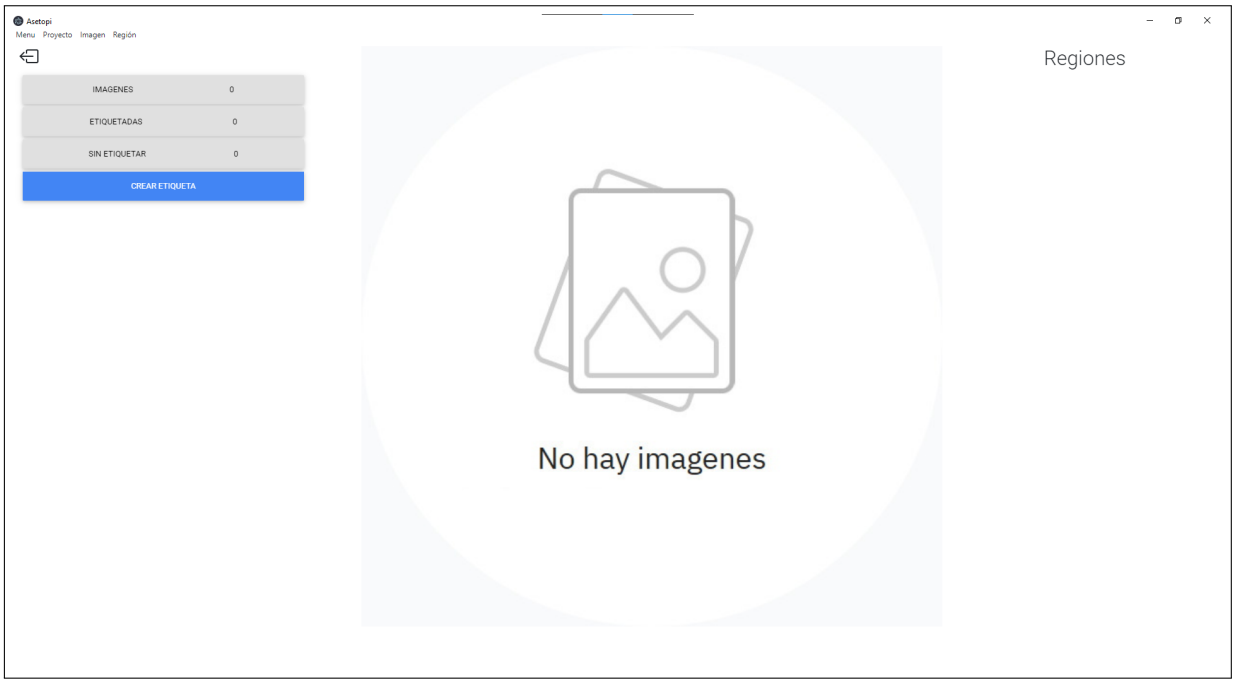

Figura 39: Cargado nuevo proyecto

asignada.

En este punto solo hay dos opciones, añadir imágenes o crear etiquetas. Empezaremos añadiendo imágenes ya que es el paso más lógico, pero no habría problema si se decide empezar creando etiquetas.

8.3.4. Añadir imágenes

Para añadir imágenes al proyecto se debe seleccionar esa opción en el menú en la sección `Imagen`.

Para este ejemplo vamos a necesitar varias imágenes, en mi caso he utilizado imágenes de un conjunto de imágenes gratuitas [21] distribuidas por Kaggle. En este ejemplo usaremos sólo los elementos de la carpeta **CAT_00**.

Una vez añadidas y cargadas las imágenes se mostrará un mensaje indicando el número de imágenes que fueron añadidas y que no existían ya en el proyecto.

[21] `https://www.kaggle.com/crawford/cat-dataset`

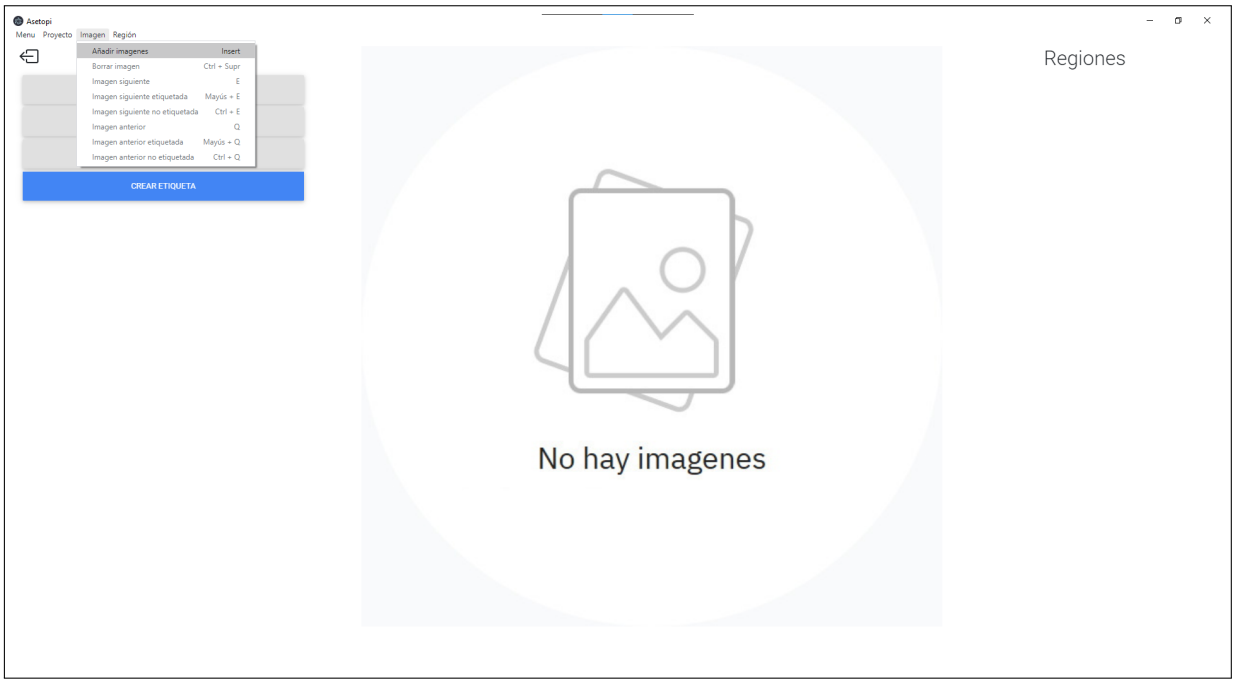

Figura 40: Añadiendo imágenes al proyecto

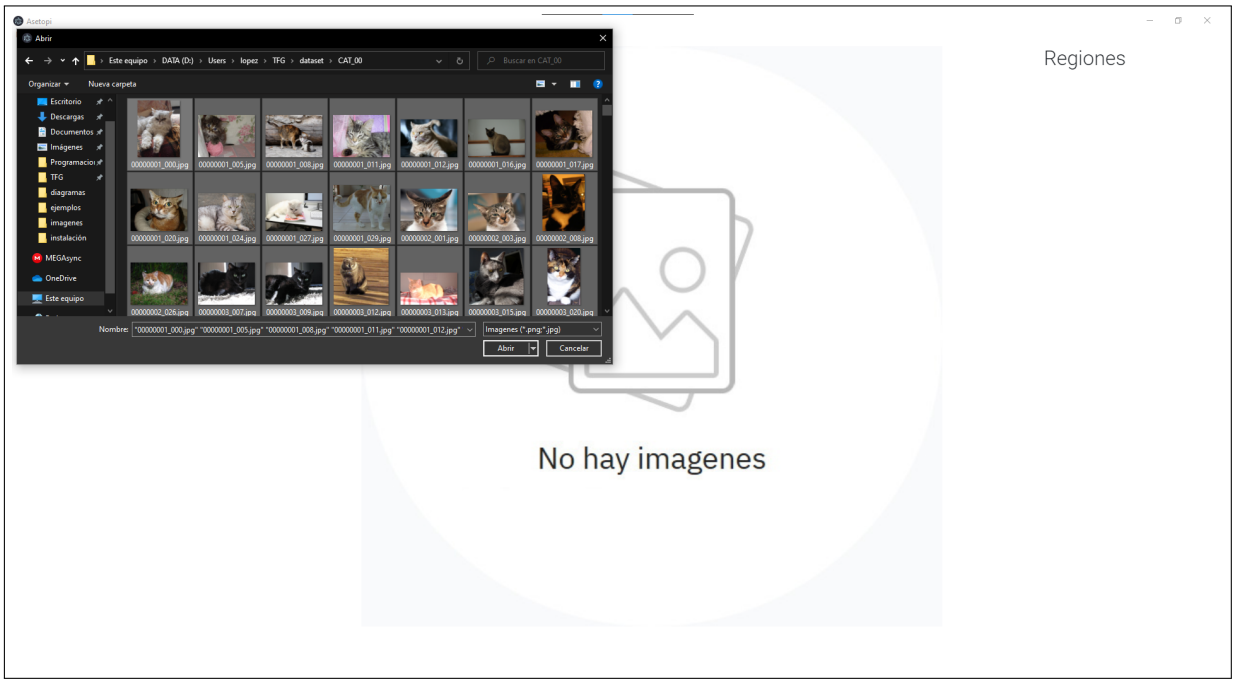

Figura 41: Seleccionando imágenes

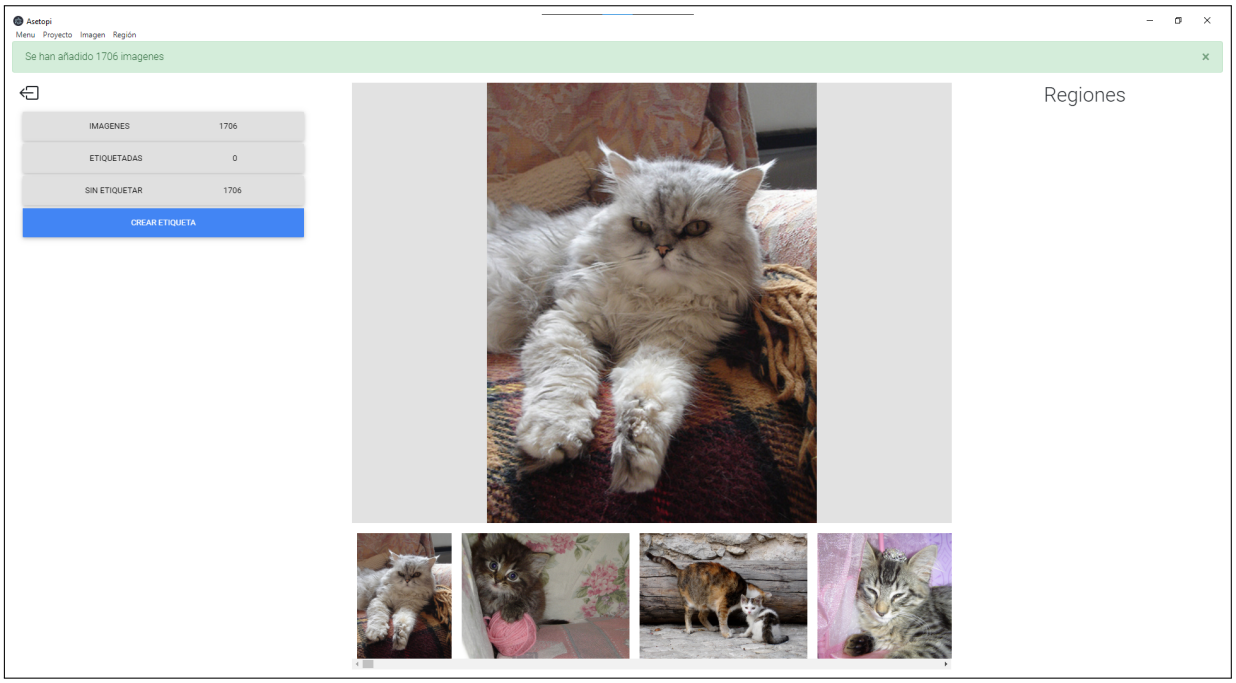

Figura 42: Imágenes añadidas

8.3.5. Crear etiquetas

Tras este paso ya estamos listos para crear nuestras etiquetas. Como es un proyecto genérico solo vamos a crear una etiqueta con el nombre gato, pero con este conjunto de datos podríamos aumentar el detalle y usar por ejemplo etiquetas como orejas, ojos o boca.

Una vez creada una etiqueta se mostrará debajo de la última etiqueta creada. Una etiqueta está formada de tres partes, el nombre de la etiqueta, el número de regiones a la que está asignada y la opción para borrar la etiqueta. Es importante saber que no se puede borrar una etiqueta si está asignada a una región.

Para renombrar una etiqueta solo hay que hacer doble clic en el nombre de la etiqueta, escribir el nuevo nombre y hacer clic fuera para confirmar, si este nombre nuevo está vacío entonces esta modificación no tendrá efecto.

8.3.6. Crear regiones

Una vez creadas todas las etiquetas, solo necesitamos seleccionar la que vamos a usar, haciendo clic en ella. Ahora estamos listos para empezar a crear regiones y etiquetarlas.

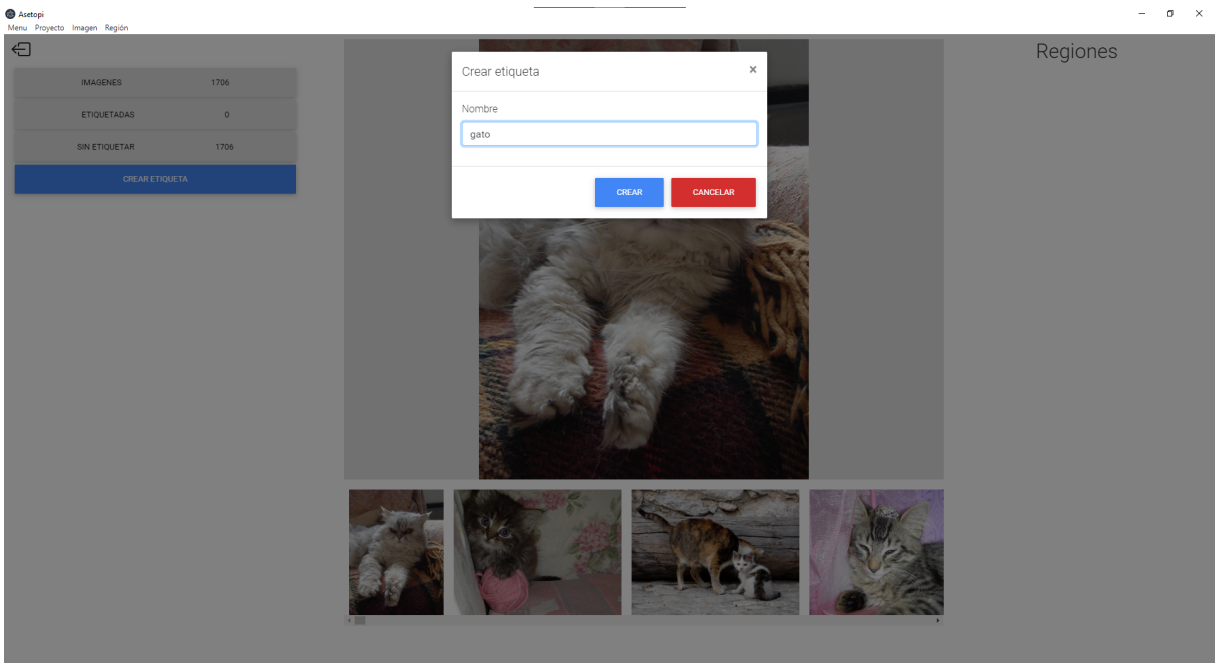

Figura 43: Creando etiqueta

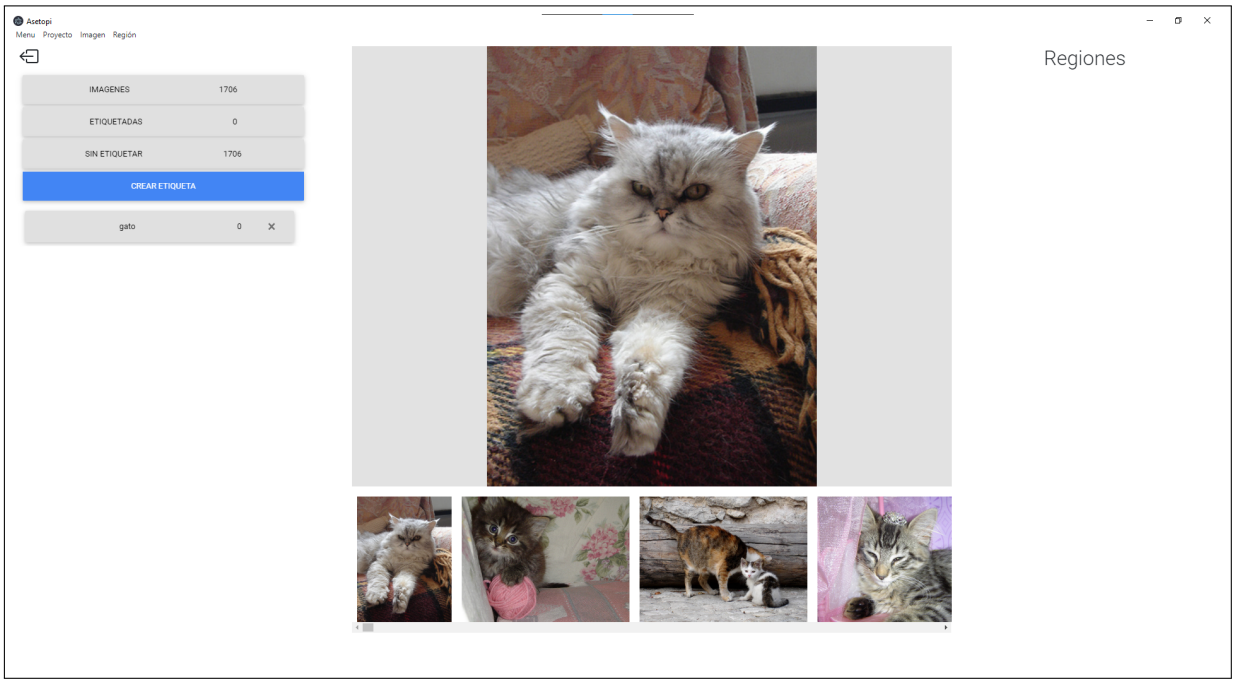

Figura 44: Etiqueta creada

Para crear una región simplemente hay que seleccionar un rectángulo dentro de la imagen.

Figura 45: Creando región

8.3.7. Buscar regiones

Otra alternativa para crear una región es usando el modelo de detección de objetos, simplemente selecciona la opción en el menú en la sección `Región`. Para seleccionar otra imagen solo hay que hacer clic en la imagen que queramos cargar o usando el menú dentro de la sección `imagen`, la cual nos dará un mayor control para cambiar a otra imagen.

Una vez seleccionada esta opción ya sea con el menú o con atajos de teclado se mostrará una nueva región. Al buscar una región se mostrará como aparece en la figura 47.

Al buscar regiones la región creada no tendrá ninguna etiqueta asignada. Esto se debe a que aunque el modelo es capaz de buscar objetos, no puede marcar solo los objetos que nosotros necesitemos, si por ejemplo nosotros estamos etiquetando solo la del gato, esta opción marcará el objeto completo. Por lo que para asignarle una etiqueta a esa región necesitamos hacer clic en la nueva región manteniendo la tecla **Ctrl**, esto asignará la etiqueta seleccionada a la región marcada.

Otro caso de lo anteriormente mencionado, es que puede que no se encuentren objetos, por ello se notificará si no se han encontrado objetos. La advertencia será como en la figura 49.

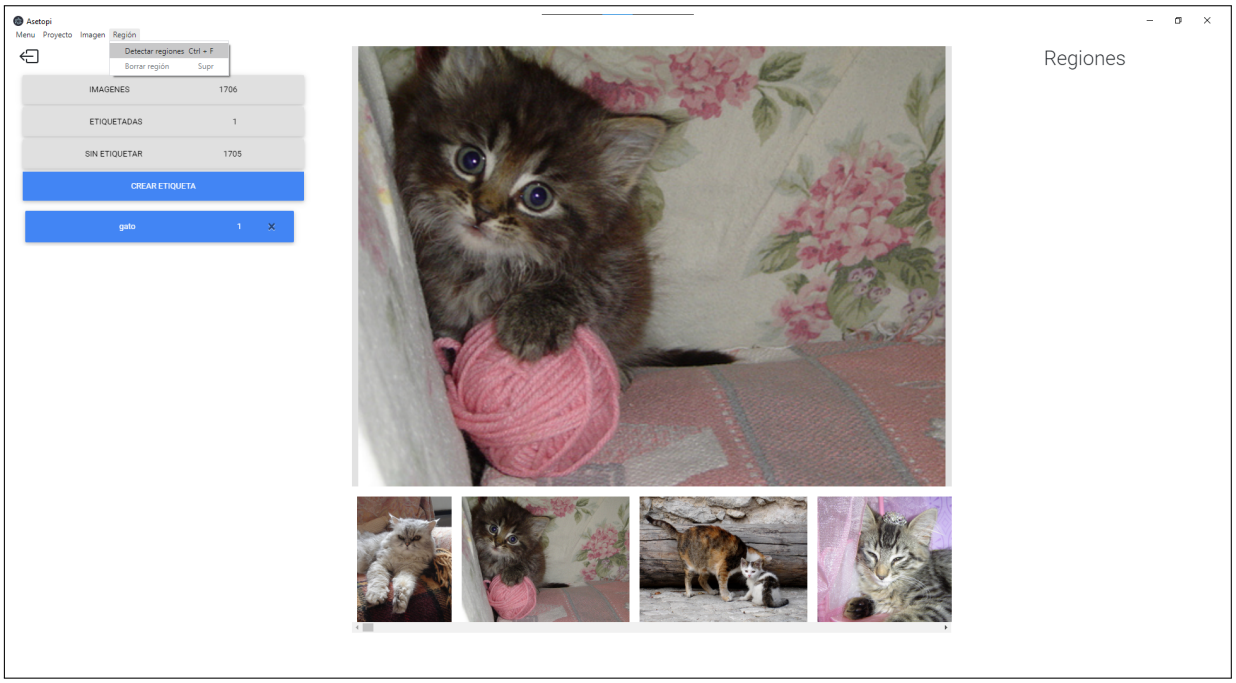

Figura 46: Busca objetos

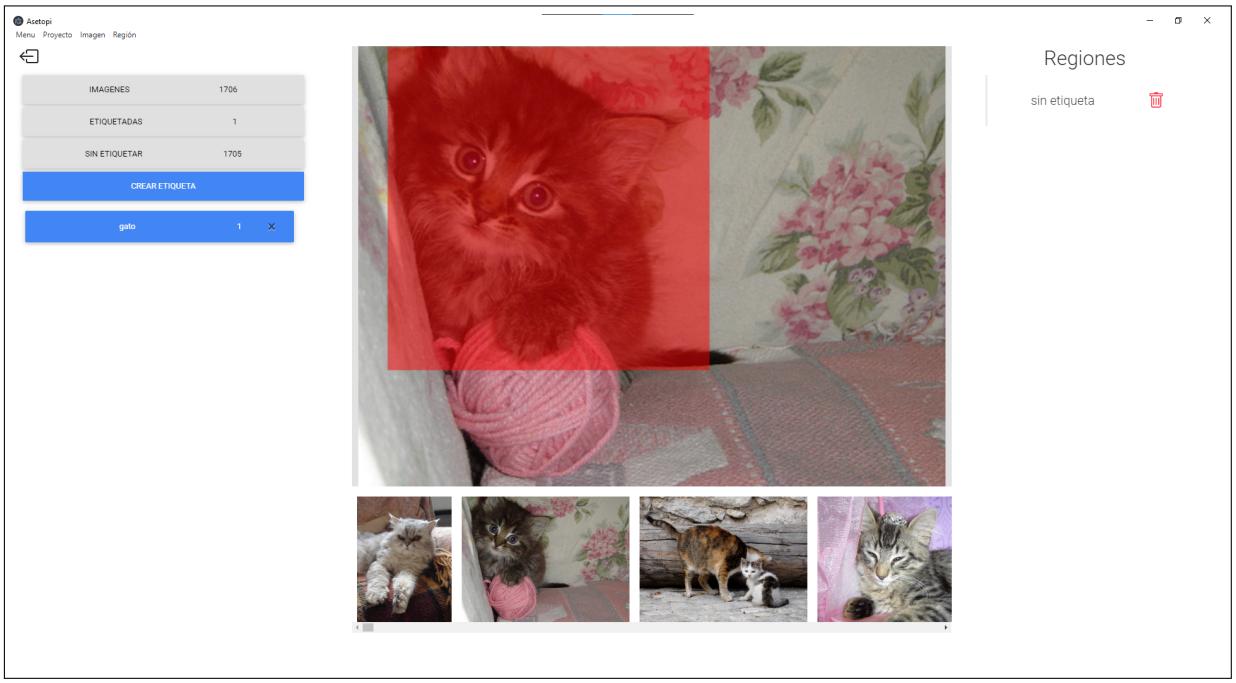

Figura 47: Objetos encontrados

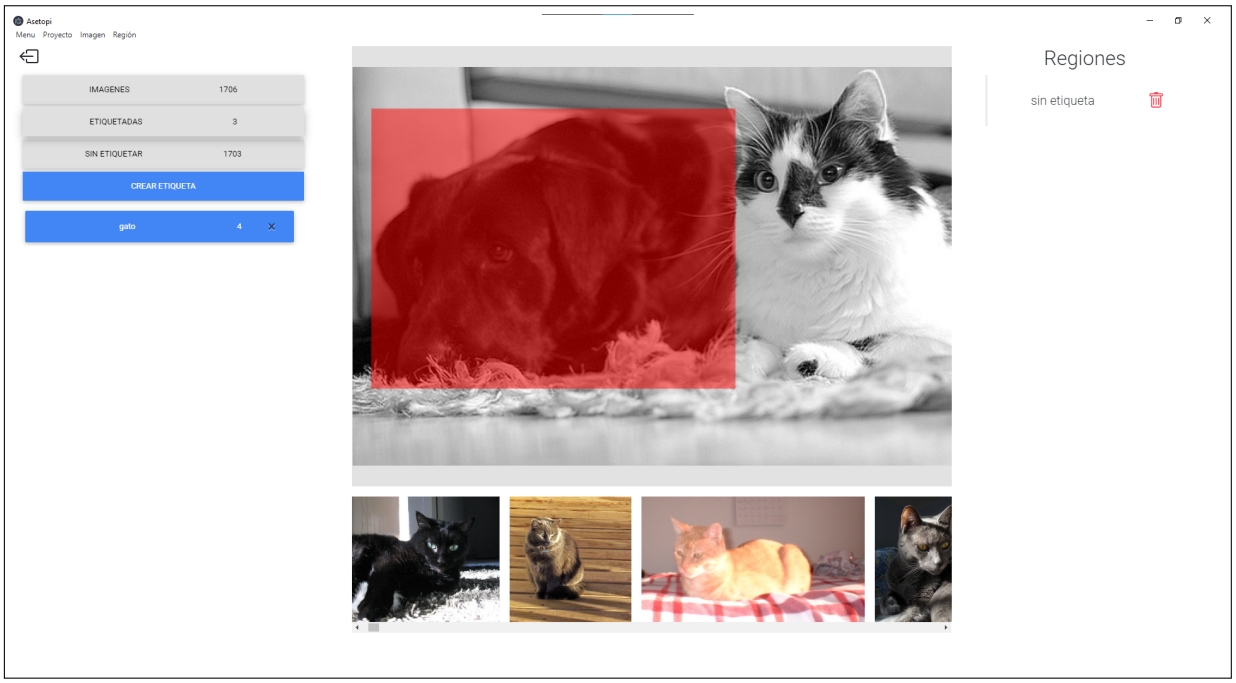

Figura 48: Detección errónea

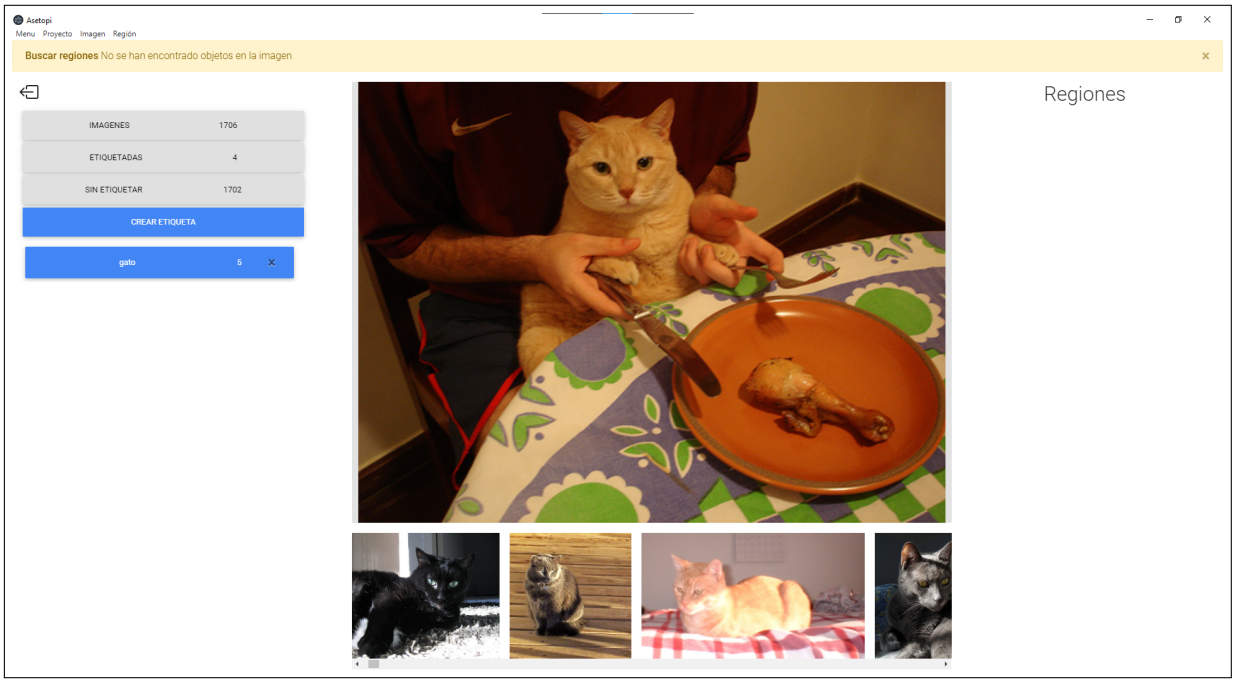

Figura 49: No encuentra objetos

8.3.8. Modificar región

Por último también podría darse el caso en el que las regiones creadas se creen sobre objetos no relacionados con el etiquetado, como es el caso anterior. Para modificar una

región simplemente hay que arrastrarla y modificarla hasta que cubra la zona deseada, otra opción es borrar por completo la región y crear una nueva.

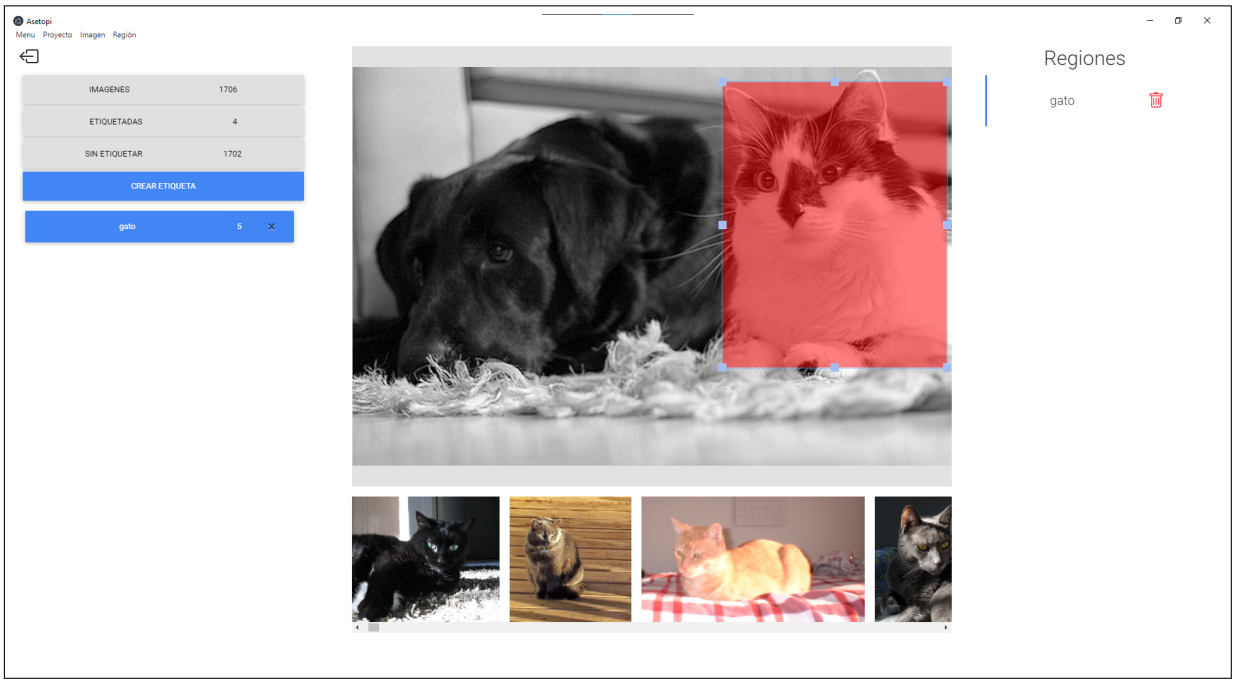

Figura 50: Modificando región

Si se necesita crear más de una región para una imagen, puede hacerse con total libertad, dos regiones pueden solaparse e incluso superponerse. Por ello en la zona de las derecha donde se muestran las regiones con sus opciones, pueden seleccionarse las regiones haciendo clic y se permitirá modificarlas en la zona de la imagen.

8.3.9. Guardar proyecto

En cualquier punto de la interacción puede guardarse el estado de todo el proyecto, dentro de la sección **Proyecto** del menú. Al guardar se mostrará al usuario si la acción ha tenido éxito o no.

Si por algún motivo el usuario intenta salir de la aplicación o cerrar el proyecto sin guardarlo, se mostrará una ventana de confirmación.

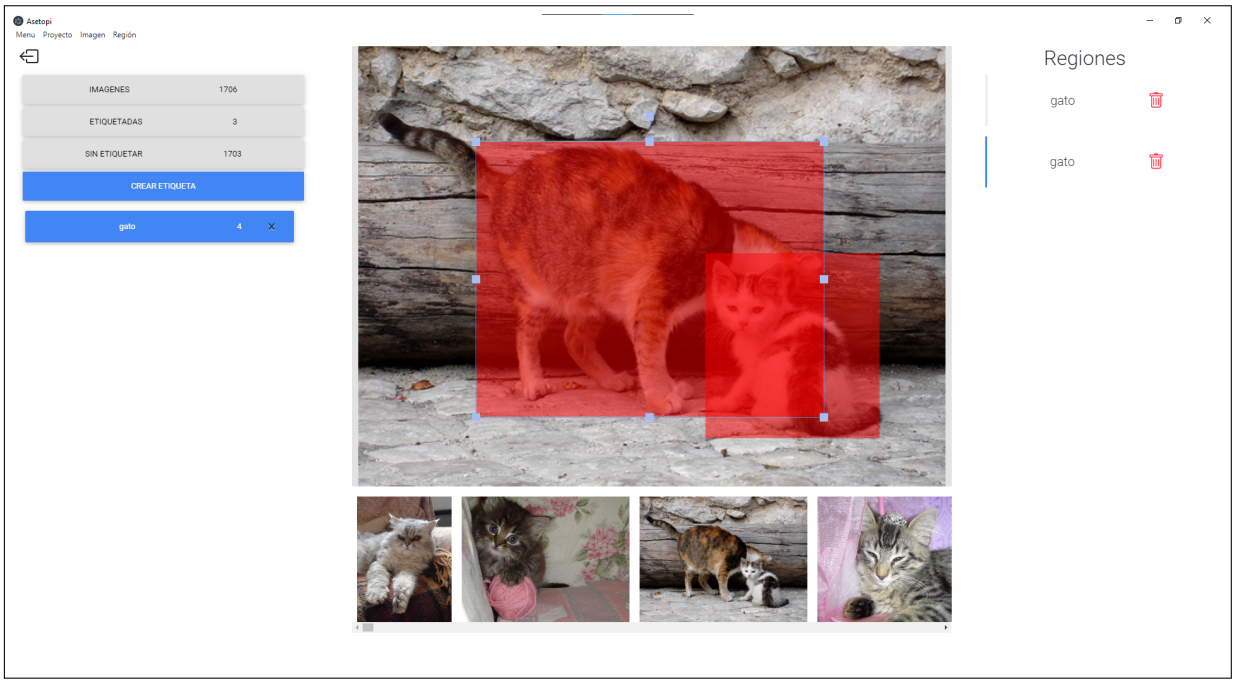

Figura 51: Múltiples regiones

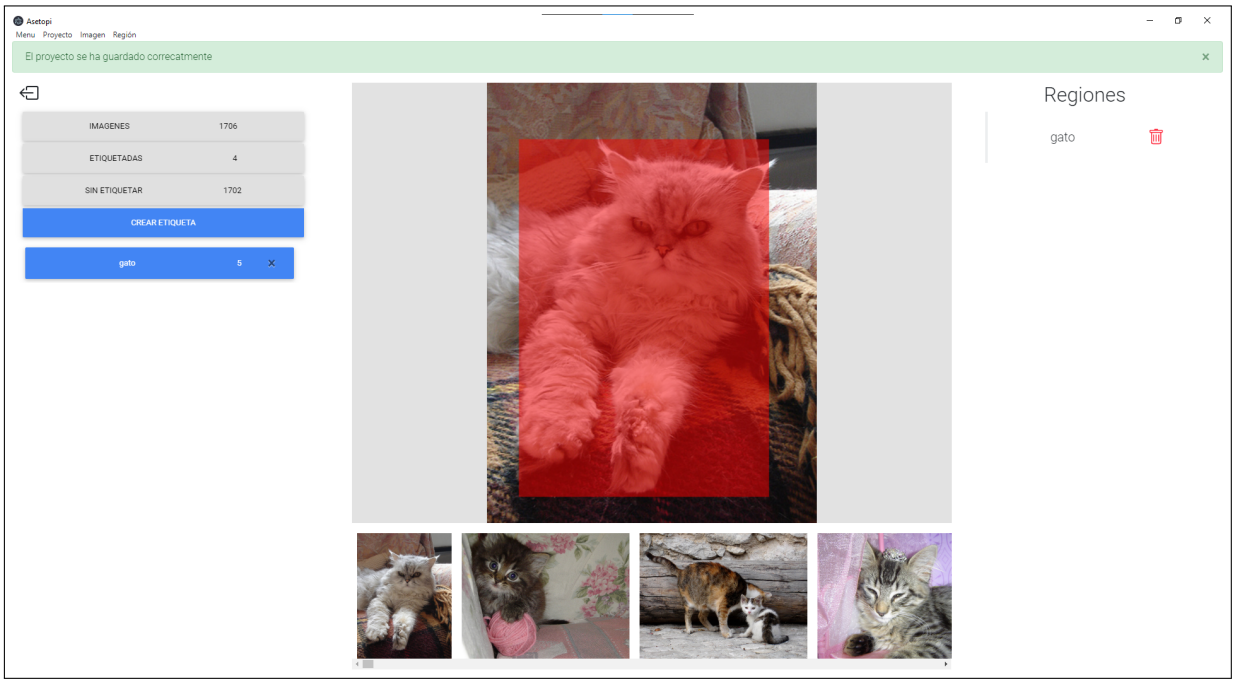

Figura 52: Proyecto guardado

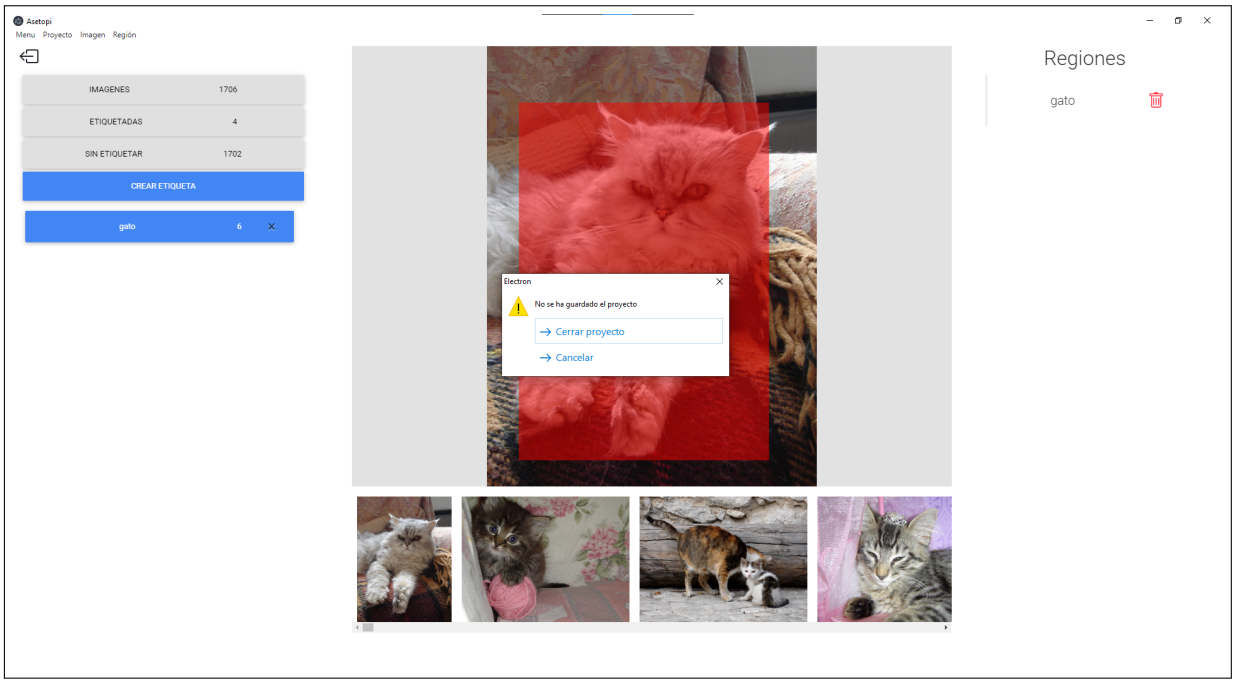

Figura 53: Advertencia al salir

8.3.10. Cerrar proyecto

Para cerrar el proyecto solo hay que hacer clic en el icono de arriba a la izquierda, encima de la zona donde se muestran las imágenes del proyecto. Esto devolverá al usuario a la pantalla que se vio inicialmente.

8.3.11. Importar proyecto

Para poder importar un proyecto, debe hacerse desde la vista de los proyectos. Una vez seleccionada la opción solo podrán importarse proyectos que fueron exportados previamente con formato **Proyecto asetopi**. Este tipo de operaciones de importar/exportar deben de estar enfocadas a hacer copias de seguridad para el proyecto, ya que si se importa el proyecto en una máquina donde no se encuentran las imágenes del proyecto, se producirán errores.

8.3.12. Exportar proyecto

Para exportar un proyecto, es necesario tenerlo cargado previamente. Una vez cargado se puede seleccionar la opción `exportar`, los proyectos se podrán exportar en formato

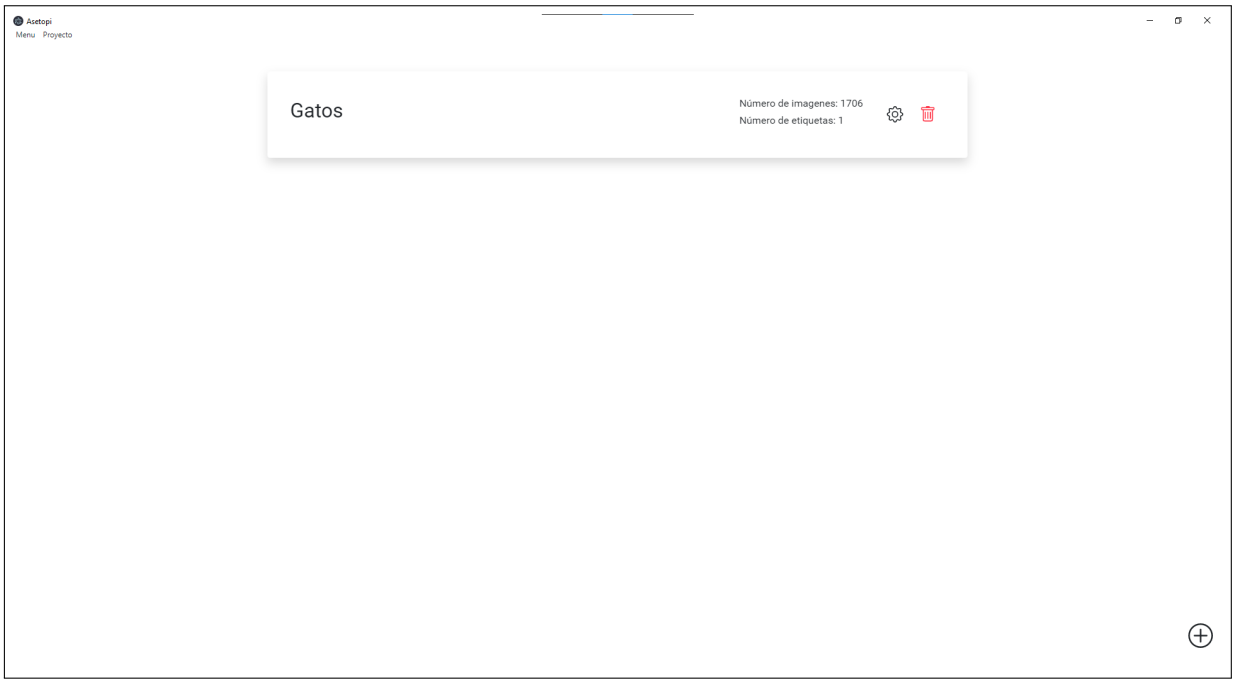

Figura 54: Cambios realizados

CSV, **JSON** o **Proyecto asetopi**. Los usuarios de distribuciones linux deberán de añadir la extensión del archivo manualmente ya sea .csv, .json o .gz respectivamente. Al exportar se generará un fichero donde el usuario lo establezca. Para el caso de seleccionar **Proyecto asetopi** se generará un fichero comprimido.

9. Definiciones y abreviaturas

Siglas

ASETOPI Asistente para el etiquetado de objetos presentes en imágenes. 11

CSS Hoja de estilos en cascada. 68

CSV Valores separados por comas. 79, 93

CU Caso de uso. 16, 17, 48, 49, 57

DRA Desarrollo Rápido de Aplicaciones. 14

DS Diagrama de secuencia. 43, 44, 50, 52, 54, 57, 58

EF Factores ambientales. 16–19

HTML Lenguaje de marcas de hipertexto. 68

HTTP Protocolo de transferencia de hipertexto. 10

JSON Notación de objeto de JavaScript. 79, 93

LOPD Ley Orgánica de Protección de Datos de Carácter Personal. 21

TCF Factores técnicos. 16–18

TDD Desarrollo guiado por pruebas. 15, 64, 69

TFG Trabajo de fin de grado. 13, 16, 72, 73

UAW Factor de peso de los actores sin ajustar. 17

UCP Puntos de caso de uso. 16–19

UUCP Puntos de caso de uso sin ajustar. 16, 17

UUCW Factor de peso de los casos de uso sin ajustar. 17

Referencias

[1] A Anooja and Shalini Rajawat. Analyzing agile estimation techniques and software development. 2018.

[2] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168, 2006.

[3] Mike Cohn. Estimating with use case points. *Methods & Tools*, 13(3):3–13, 2005.

[4] Junta de Andalucia. Guía para la redacción de casos de uso `http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416.html`.

[5] Boletín Oficial del estado. Convenio Estatal de las TIC `https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf`.

[6] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387, 2014.

[7] Zoubin Ghahramani. Unsupervised learning. In *Summer School on Machine Learning*, pages 72–112. Springer, 2003.

[8] Ravi Kumar Gupta, Hetal Prajapati, and Harmeet Singh. *Test-Driven JavaScript Development*. Packt Publishing Ltd, 2015.

[9] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[10] Gurpreet Singh Matharu, Anju Mishra, Harmeet Singh, and Priyanka Upadhyay. Empirical study of agile software development methodologies: A comparative analysis. *ACM SIGSOFT Software Engineering Notes*, 40(1):1–6, 2015.

[11] Mark Nixon and Alberto Aguado. *Feature extraction and image processing for computer vision*. Academic press, 2019.

[12] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Science and Information Conference*, pages 128–144. Springer, 2019.

[13] Paul Pop. Comparing web applications with desktop applications: An empirical study. *Linköping University, Linköping*, 2002.

[14] Marian Stoica, Marinela Mircea, and Bogdan Ghilic-Micu. Software development: Agile vs. traditional. *Informatica Economica*, 17(4), 2013.

[15] Kelly Waters. Prioritization using moscow. *Agile Planning*, 12:31, 2009.

[16] Jure Zupan. Basics of artificial neural network. *Nature-inspired Methods in Chemometrics: Genetic Algorithms and Artificial Neural Networks (Leardi, R., ed.)*, pages 199–229, 2003.