Universidad de La Laguna
Escuela Superior de Ingeniería y Tecnología

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Catalogando colecciones de fotografía digital mediante etiquetado automático

Cataloging of digital photography collections by automatic labeling

La Laguna, 13 de marzo de 2019

D. **María Elena Sánchez Nielsen**, con N.I.F. 42.848.599-J profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

"Catalogando colecciones de fotografía digital mediante etiquetado automático"

ha sido realizada bajo su dirección por D. **Joaquín Sanchiz Navarro**,

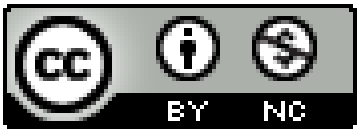
con N.I.F. 54.064.236-E.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 13 de marzo de 2019

Agradecimientos

A mi familia por hacer de este proyecto algo que pude llevar a cabo con calma, a mis amigos por apoyarme en todo momento, y a mi tutora por guiarme en este proyecto.

Licencia



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial 4.0 Internacional.

Resumen

El objetivo de este trabajo ha sido desarrollar un sistema que sea capaz de detectar e identificar personas previamente reconocidas en una base de datos. Para llevar a cabo dichos objetivos se utilizará la herramienta de catalogación de software libre digiKam, que presenta una serie tablas internas muy interesantes para el trabajo de reconocimiento facial. Además, utilizaremos y estudiaremos la aplicación de Amazon Rekognition como API en nuestro sistema, que hará la función de motor de comparación de rostros. Todo esto ha de funcionar bajo una interfaz de usuario sencilla, que cualquier persona pueda utilizar con facilidad, para así explotar al máximo las funcionalidades. El sistema además de reconocer diferentes rostros ha de ser capaz de determinar en qué imágenes ya guardadas aparece dicha persona seleccionada, y con qué otras personas puede aparecer en una colección fotográfica.

Como caso de uso para el proyecto, se propone identificar los miembros del Parlamento de Canarias.

Palabras clave: catalogación fotográfica, identificación facial, detección facial, marco facial, digiKam, Amazon Rekognition.

Abstract

The objective of this project has been to develop a system that is able to detect and identify people previously recognized in a database. To carry out these objectives, it will be use the open-source cataloging tool digiKam, which presents a series of very interesting internal tables for facial recognition work. In addition, we will use and study the Amazon Rekognition application as API in our system, which will be responsible for making the necessary facial comparisons in the system. All this infrastructure has to perform under a simple user interface, that anyone can use with ease, in order to fully exploit the functionalities. The system, besides recognizing different faces, must be able to detect which images already saved the selected person appears, and with which other members can appear on a photographic collection.

As case of study, it has been proposed to identify Parlamento de Canary Islands members.

Keywords: photographic cataloguing, facial identification, facial detection, bounding box, digiKam, Amazon Rekognition.

Índice general

Capítulo 1 Introducción 1
1.1 Elección del Trabajo de Fin de Grado 1
1.2 El problema que existe y estado actual 1
1.2.1 Aplicaciones 2
1.2.2 Mejoras recientes 2
1.2.3 Peligros que presenta 3
Pérdida de privacidad 3
Información abierta a todo el mundo 3
1.3 Estrategias de mejora y soluciones 4
1.4 Objetivo principal 4
1.5 Objetivos específicos 5
1.6 Caso de uso 5
Capítulo 2 Antecedentes 6
2.1 Elementos que se usarán en el proyecto 6
2.1.1 DigiKam 6
2.1.2 Amazon Rekognition 6
2.1.3 SQLite 7
2.1.4 draw.io 7
2.1.5 Adobe Photoshop CC 7
2.1.6 Imágenes *target* 7
2.1.7 Imágenes *source* 7
2.1.8 Imágenes *Unknown* 7
2.1.9 Threshold 7

2.2 Tecnologías 8
2.2.1 Aprendizaje profundo 8
2.2.2 Técnicas tradicionales de reconocimiento facial 8
2.2.3 Computación en la nube 8
Ventajas 9
Desventajas 9
2.3 Proyectos similares 9
2.3.1 Facebook Deep Face 9
2.3.2 Tecnología china para el reconocimiento facial y de voz para cerdos 10
2.3.3 Reconocimiento facial para entrar a ferias y festivales 10
Capítulo 3 Metodología 12
3.1 Fases del proyecto 12
3.2 Estudio de recursos utilizados 12
3.2.1 Eclipse y Java. 12
3.2.2 Amazon Rekognition 13
Primeros pasos en la integración 13
Instalación de la Command Line Interface de AWS 13
Configuración de la Command Line Interface de AWS 14
Integración de la API en la aplicación. 14
3.2.3 Base de datos de *digiKam* 16
Reconocimiento facial 16
Funcionamiento de tablas internas de reconocimiento facial 17
3.3 Primera aproximación al funcionamiento del sistema 18
3.3.1 Reconocimiento de rostro en imagen 18
3.3.2 Inserción de nueva imagen *source* 19
3.3.3 Reconocimiento de varios rostros en imagen 20
3.3.4 Inserción de caras no reconocidas en el sistema 21
3.4 Diseño de la interfaz gráfica de usuario 22
3.5 Objetos del sistema 24
3.5.1 Utilidades principales 24

DBContext 24
User 24
ImageData 24
Rekognition System 24
3.5.2 Interfaz gráfica de usuario 25
Controladores 25
Paneles 25
Capítulo 4 Resultados 34
4.1 Estudio de los resultados 34
4.2 Análisis de tiempos del sistema 36
Capítulo 5 Conclusiones y líneas futuras 38
digiKam 38
Amazon Rekognition 38
Cliente web 39
Visión del proyecto 39
Capítulo 6 Summary and Conclusions 41
digiKam 41
Amazon Rekognition 41
Web client 42
Project view 42
Capítulo 7 Presupuesto 44
7.1 Coste del proyecto 44
Capítulo 8 Algoritmos 45
8.1 Algoritmo Rekognition 45
Capítulo 9 Recursos 51
9.1 Código del proyecto 51

Índice de ilustraciones

Ilustración 1: Face Hallucination [12] 3
Ilustración 2: Reconocimiento facial 6
Ilustración 3: Atributos faciales [8] 15
Ilustración 4: Marco facial digiKam [13] 16
Ilustración 5: Marco facial Amazon Rekognition [13] 16
Ilustración 6: Fallo de reconocimiento digiKam [14] 16
Ilustración 7: Análisis facial [15] 19
Ilustración 8: Secuencia de inserción 19
Ilustración 9: Secuencia de reconocimiento 20
Ilustración 10: Inserción de Unknown 21
Ilustración 11: Inserción de imagen source teniendo en cuenta Unknown 22
Ilustración 12: Reasignación de Unknown a Source 23
Ilustración 13: Diseño Main Frame 22
Ilustración 14: Diseño Panel principal 23
Ilustración 15: Diseño Panel Consulta 23
Ilustración 16: Vista principal 25
Ilustración 17: Panel de confirmación de elección [16] 26
Ilustración 18: Panel que muestra las imágenes Source [16] 27
Ilustración 19: Vista Imágenes Unknown [13] 28
Ilustración 20: Cambio de nombre Unknown[13] 29
Ilustración 21: Panel de información de análisis [17] 30
Ilustración 22: Panel de Consulta 31
Ilustración 23: Información sobre donde aparece [18][19][20][21][13] 32
Ilustración 24: Información sobre con quien aparece [16] 33
Ilustración 25: Reconocimiento de rostro en cartel [22] 35
Ilustración 26: Gráfico de rendimiento 37

Índice de tablas

Tabla 1: Aplicaciones de reconocimiento facial [4] 2
Tabla 2: Tags 17
Tabla 3: Images 17
Tabla 4: Imagetags 17
Tabla 5: Imagetagproperties 17
Tabla 6: Tiempos de análisis 36
Tabla 7**:** Coste del proyecto 44

Capítulo 1
Introducción

1.1 Elección del Trabajo de Fin de Grado

Desde que comencé la carrera de Ingeniería Informática, he sentido mucho interés por el campo de la *Inteligencia Artificial* y el *Machine Learning*. Tuve muy claro que escogería la rama de *Computación* para poder seguir mis aprendizajes en esta dirección, y mis ganas de conocimiento han ido creciendo a medida que he ido superando los semestres.

Cuando cursé la primera asignatura de *Inteligencia Artificial*, indagué en las estructuras de datos que la componían y me dispuse a aprender por mi cuenta, mediante libros, los perceptrones y modelos neuronales que tanto se habían mencionado. Puesto que era material muy avanzado de informática y programación, no tuve la oportunidad de entenderlo hasta estos últimos cursos. Aunque estas estructuras de datos se encuentran abstraídas por el sistema de *Amazon Rekognition*, que se encarga de todos estos cálculos, poder trabajar en un proyecto de este tipo consiguió todo mi interés.

Además, el procesado de imágenes es un tema que está siendo acogido con gran interés dentro del mundo de la informática. Constantemente se están moviendo imágenes a través de la red entre usuarios, por lo que presentan una gran fuente de información, que, tratada de una manera adecuada, puede servir para numerosas aplicaciones. La comparación de rostros mediante el análisis facial de imágenes tan solo es uno de esos miles de aplicaciones.

1.2 El problema que existe y estado actual

El reconocimiento de rostros es algo que se ve desde hace varios años en aplicaciones que implican fotografías de personas con fondos determinados, como por ejemplo en redes sociales como *Facebook*. Dichos sistemas se alimentan automáticamente de la información que obtienen, permitiendo así, mediante un crecimiento de sus colecciones privadas, una mejor identificación.

También lo podemos encontrar en sistemas de vigilancia y defensa, donde se identifican usuarios con el motivo de proteger o controlar determinadas situaciones y

entornos, ya sea como identificación de personas dentro de una empresa mediante *biometrics*, o bien en sistemas de tráfico para la determinación de usuarios buscados.

Estos sistemas deberían avanzar a la misma velocidad que la tecnología que los rodea, por lo que siempre deberán ser actualizados y mejorados, con motivo de mejorar su funcionamiento y prestaciones. La mejora de dichos sistemas favorecerá los entornos en los que se trabaje con ellos, mejorando así no solo las aplicaciones vistas en primer plano, sino aquellas que utilizan estos sistemas como herramientas, provocando un beneficio a un sector muy amplio.

Hay infinidad de usos que se le pueden dar a los sistemas de reconocimiento de imagen, aumentando cada vez más su número, y lo seguirá haciendo el resto de los años hasta incorporarse a todo tipo de productos, desde domótica inteligente, IoT a otras cosas que aun ni podemos imaginar.

1.2.1 Aplicaciones

Áreas	Aplicaciones especificas
Biometría	Licencia para conducir, programas de derecho, inmigración, DNI, Pasaportes, Registro de Votantes, Fraude
Seguridad de la información	Inicio de sesión, seguridad en aplicaciones, seguridad en bases de datos, cifrado de información, seguridad en internet, acceso a internet, registros médicos, terminales de comercio seguro, cajeros automáticos
Cumplimiento de la ley y vigilancia	Video vigilancia avanzada, control CCTV, control portal, hurto, seguimiento de sospechosos, investigación
Tarjetas inteligentes	Valor almacenado, autenticación de usuarios
Control de acceso	Acceso a instalaciones, acceso a vehículos

Tabla 1: Aplicaciones de reconocimiento facial [4]

1.2.2 Mejoras recientes

A mediados del 2006 [4], se evaluaron los últimos algoritmos de reconocimiento facial. Escanners faciales 3D, imágenes de rostros de alta definición y del iris fueron utilizadas en las pruebas. Los resultados indicaron que estos nuevos algoritmos tienen 10 veces más exactitud que los algoritmos con fecha anterior a 2002, y 100 veces más que los de 1995. Estos algoritmos son tan precisos que son capaces de reconocer a dos gemelos idénticos.

Actualmente las imágenes de baja resolución ya no son un problema porque pueden ser tratadas a partir de la súper-resolución del rostro, conocido como *Face*

Hallucination. *Face Hallucination* [12] es una técnica de súper-resolución que permite obtener una imagen de alta resolución a partir de una imagen de entrada de baja resolución. Se aplica en sistemas de reconocimiento facial para que la identificación y análisis de un rostro sea más fácil y eficaz.

Debido a la importancia de las imágenes en los sistemas de reconocimiento facial, en los últimos años *Face Hallucination* se ha convertido en un área de interés para los investigadores.

Ilustración 1: Face Hallucination [12]

1.2.3 Peligros que presenta

Aunque estos sistemas [4] cuenten con numerosas ventajas, existen algunos riesgos:

Pérdida de privacidad

Algunos consideran que la privacidad en el siglo XXI se está perdiendo, especialmente si tenemos en cuenta los últimos sucesos del 2018 relacionados con la comercialización de información personal y los consecuentes cambios, en este mismo año, de la *Política de Privacidad*.

No obstante, sigue existiendo una mayoría que defiende el deseo de conservar la privacidad, algo que entorpecerá entre otras cosas el sistema de reconocimiento facial. Aunque la idea de utilizar el sistema de reconocimiento facial para reforzar los sistemas y hacer los países más seguros se acepta, toda esa información puede ser utilizada con otros fines.

Información abierta a todo el mundo

La posibilidad de acceder a las redes sociales a través del reconocimiento facial en una imagen permite poner a disposición de cualquiera, gran contenido de información personal. Aunque cuando nos registramos en una red social aceptamos las condiciones de privacidad, tenemos que tener en cuenta que este tipo de problemas los podemos tener nosotros, por lo que debemos tener mucho cuidado con la información personal que hacemos pública en las redes sociales, ya que podría ser usada en nuestra contra para chantajes u otros fines.

1.3 Estrategias de mejora y soluciones

Mediante el servicio recientemente lanzado y constantemente actualizado de Amazon, *Amazon Rekognition*, se puede realizar la identificación de rostros en imágenes con fondo fotográfico.

Al ser un sistema que computa la comparación de imágenes desde los propios servidores de Amazon, ahorra a las máquinas cliente la realización de complejos cálculos, y, por tanto, supone un paso en la eficiencia de este tipo de sistemas, ya que únicamente se tienen que enviar los datos a comparar a la red. Si los datos se tuvieran que realizar en local, necesitaríamos de máquinas muy potentes y con acceso a muchos recursos para poder efectuar los cálculos con la rapidez que los realizan los servidores de Amazon.

Amazon Rekognition [8] ofrece dos conjuntos de *API*. Estas *API* son *Amazon Rekognition Image* para el análisis de imágenes y *Amazon Rekognition Video* para el análisis de videos.

Ambas *API* realizan análisis de detección y reconocimiento de imágenes y videos para proporcionar información que puede utilizar en sus aplicaciones. Por ejemplo, se podría utilizar *Amazon Rekognition Image* con el fin de mejorar la experiencia de los clientes para una aplicación de administración de fotos. Cuando un cliente carga una foto, la aplicación puede utilizar *Amazon Rekognition Image* para detectar objetos del mundo real o rostros en la imagen. Después de que la aplicación almacene la información que devuelve *Amazon Rekognition Image*, el usuario podría consultar su colección de fotos para localizar fotos con un objeto o rostro concreto. Es posible una consulta más profunda. Por ejemplo, el usuario podría realizar consultas de rostros que sonrían o consultar rostros de una determinada edad.

Puede utilizar *Amazon Rekognition Video* para realizar un seguimiento del recorrido de las personas en un video almacenado. De forma alternativa, puede utilizar *Amazon Rekognition* Video para buscar un vídeo en *streaming* para localizar personas cuyas descripciones faciales coincidan con las descripciones faciales ya almacenadas por *Amazon Rekognition*.

1.4 Objetivo principal

Las diferentes actividades que se van a llevar a cabo están basadas en el estudio y evaluación de la *API* de *Amazon Rekognition* para detectar y reconocer rostros. Veremos las posibilidades que nos presenta esta herramienta y como la introduciremos en nuestro sistema.

También estudiaremos las diferentes posibilidades que ofrece *digiKam*, la herramienta de catalogación open-source que utilizaremos en el sistema. Esta herramienta ofrece también la posibilidad del reconocimiento facial, por lo que estudiaremos como funciona y la compararemos con la de *Amazon Rekognition*. El objetivo principal de esta herramienta es la de proporcionar las imágenes al sistema

para que posteriormente *Amazon Rekognition* ejecute la comparación.

1.5 Objetivos específicos

- Estudiar el funcionamiento de *Amazon Rekognition*
    - Obtener las credenciales para el acceso a la *API* de *Amazon Rekognition*.
    - Instalar las librerías de la *API* de *Amazon Rekognition*
    - Conocer el funcionamiento de la *API* para utilizarlo próximamente en la comparación de rostros.
    - Comprobar los resultados de la comparación y analizar los atributos de la respuesta.
- Estudiar el funcionamiento de *digiKam*
    - Conocer el funcionamiento de las tablas internas de reconocimiento facial.
    - Estudiar los resultados de la herramienta integrada orientada para el reconocimiento facial.
- Estudiar funcionamiento conjunto de *digiKam* junto a *Amazon Rekognition* en un sistema que sea capaz de identificar rostros de una base de datos y clasificarlos.
    - Carga de rostros a encontrar
    - Carga de imagen a analizar
- Diseñar una interfaz de usuario que sea capaz de controlar el funcionamiento del sistema de manera sencilla.

1.6 Caso de uso

Estudiaremos el comportamiento del sistema sobre una situación real como pueda ser el Parlamento de Canarias. La base de datos del sistema contendrá los perfiles faciales de los miembros del Parlamento, obtenidos mediante una imagen donde solo aparezca dicho miembro. El sistema será capaz de analizar y detectar los rostros sobre una imagen proporcionada, de las personas contenidas en la base de datos, y añadirá para una futura catalogación, aquellos rostros no reconocidos.

Capítulo 2
Antecedentes

En el capítulo anterior se han introducido los motivos de la elección del Trabajo de Fin de Grado, el problema que solventaría dicho proyecto junto con el estado actual del arte, las estrategias de mejora y los objetivos. En este capitulo veremos los antecedentes del proyecto, como los elementos para comprender este trabajo, las tecnologías que tienen relación directa con el proyecto, el estado actual del tema que se está tratando y la identificación de proyectos similares en desarrollo.

2.1 Elementos que se usarán en el proyecto

2.1.1 DigiKam

La herramienta *digiKam* es un servicio open-source de gestión de imágenes que presenta herramientas para la importación, gestión y edición de fotografías, así como de análisis. Se utilizará para proporcionar las colecciones de imágenes a la *API* de *Amazon Rekognition* para su futuro análisis. Más concretamente, se utilizarán las tablas internas del programa relativas al reconocimiento facial.

2.1.2 Amazon Rekognition

Amazon Rekognition [1] facilita la incorporación del análisis de imágenes y videos. Con suministrar una imagen o video a la *API* de *Rekognition*, el servicio identificará objetos, personas, texto, escenas y actividades, además de detectar contenido inapropiado. *Amazon Rekognition* también ofrece reconocimiento y análisis facial con un alto nivel de precisión, que será en lo que se enfocará el proyecto en su mayoría. Puede detectar, analizar y comparar rostros para una amplia variedad de casos de uso de verificación de usuarios, catalogación, contabilización de personas y seguridad pública. Está basado en la misma tecnología de aprendizaje profundo sólida y de alta escalabilidad que desarrollaron los científicos de visión artificial de Amazon para analizar miles de millones de imágenes y videos diariamente.

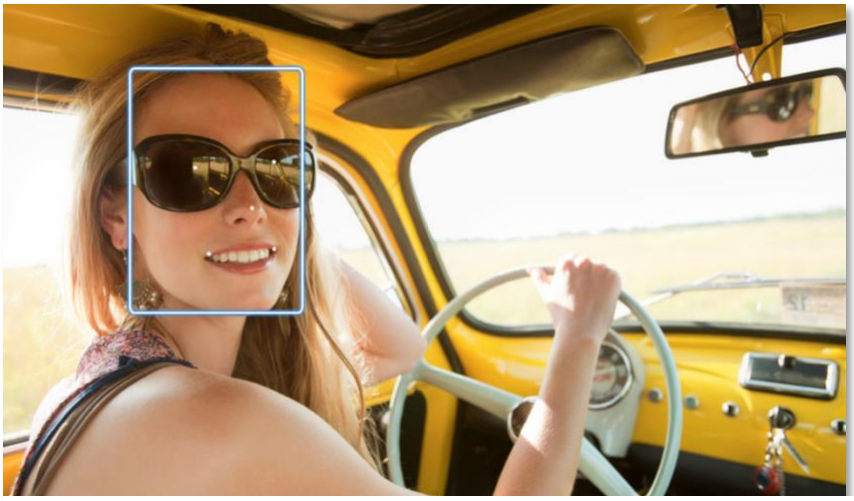


Ilustración 2: Reconocimiento facial Amazon Rekognition [8]

2.1.3 SQLite

El sistema de gestión de bases de datos relacional que se usará será el de SQLite, ya que es el promovido por *digiKam*. El programa utiliza las funcionalidades de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos [2]. El conjunto de la base de datos (definiciones, tablas, índices y los propios datos), son guardados como un solo fichero estándar en la máquina host (*digikam*4.db en el caso de nuestro proyecto).

2.1.4 draw.io

draw.io es una aplicación de diagramación gratis de *Google Drive* que le permite dibujar diagramas de flujo, lenguaje Unificado de Modelado, diagramas Entidad-Relación, diagramas de red, modelos de proceso de negocios, organigramas, circuitos electrónicos, wireframes y maquetas [3].

Se utilizará draw.io para la creación de las secuencias, algoritmos y tablas sobre las que se sostendrá nuestro proyecto.

2.1.5 Adobe Photoshop CC

Se utilizará Adobe Photoshop CC para el diseño de paneles de la interfaz visual.

2.1.6 Imágenes *target*

Las imágenes *target* son aquellas imágenes que introducimos en el sistema para analizar y buscar rostros.

2.1.7 Imágenes *source*

Definimos como imágenes *source* a las imágenes insertadas en la base de datos del sistema que se utilizarán como perfiles faciales para los futuros análisis de imágenes *target*. Cuando se analiza una imagen *target*, se realiza una comparación con los rostros de las caras de dicha imagen, y los perfiles faciales previamente insertados en el sistema, imágenes *source*.

2.1.8 Imágenes *Unknown*

Cuando aparece un rostro en una imagen *target* que no coincide con alguna de las imágenes *source*, dicho rostro se extrae y se inserta en la base de datos como imagen *Unknown* o desconocida. Estas imágenes son importantes ya que se tendrán en cuenta para futuros casos.

2.1.9 Threshold

Denominamos como *Threshold* al porcentaje mínimo de confianza que debe alcanzar una comparación para que se considere como fiable.

2.2 Tecnologías

Las tecnologías que se trabajarán en el Trabajo de Fin de Grado son las siguientes.

2.2.1 Aprendizaje profundo

El aprendizaje profundo [4] es parte de un conjunto más amplio de métodos de aprendizaje automático basados en asimilar representaciones de datos. Una observación (por ejemplo, una imagen) puede ser representada en muchas formas (por ejemplo, un vector de píxeles), pero algunas representaciones hacen más fácil aprender tareas de interés (por ejemplo, *"¿es esta imagen una cara humana?"*) sobre la base de ejemplos, y la investigación en esta área intenta definir qué representaciones son mejores y cómo crear modelos para reconocer estas representaciones

Varias arquitecturas de aprendizaje profundo, como redes neuronales profundas, redes neuronales profundas convolucionales, y redes de creencia profundas, han sido aplicadas a campos como visión por computador, reconocimiento automático del habla, y reconocimiento de señales de audio y música, y en nuestro caso, reconocimiento de caras en una imagen, han mostrado producir resultados de vanguardia en varias tareas.

2.2.2 Técnicas tradicionales de reconocimiento facial

Las técnicas tradiciones de reconocimiento facial se basan en métodos de correlación. El esquema de clasificación más simple, donde se utilizan modelos de comparación para el reconocimiento, es el *template matching*. El problema del *template matching* es que ha de comparar muchas características (para él, un píxel es una característica), y si tenemos en cuenta que en la base de datos encontramos *M* personas, con *N* imágenes por persona, observamos que este método no se puede implementar en tiempo real. Por lo tanto, se trabaja con otros métodos que decorrelacionan las características entre sí para conseguir reducir el espacio facial en un número menor de coeficientes, que tengan un alto poder discriminatorio entre las personas. Es lo que se denomina subespacio facial.

2.2.3 Computación en la nube

Dado que la *API* de *Amazon Rekognition* procesa los datos desde sus propios servidores, es algo que debemos comentar.

En este tipo de computación todo lo que puede ofrecer un sistema informático se ofrece como servicio, de modo que los usuarios puedan acceder a los servicios disponibles "*en la nube de Internet*" sin conocimientos en la gestión de los recursos que usan. En nuestro caso, suministraremos dos imágenes a comparar y un *threshold* de reconocimiento, y sin tener nosotros acceso a lo que ocurre internamente, obtendremos nuestro resultado.

La computación en la nube son servidores desde Internet encargados de atender

las peticiones en cualquier momento. Se puede tener acceso a su información o servicio, mediante una conexión a internet. Sirven a sus usuarios desde varios alojamientos repartidos frecuentemente por todo el mundo [11]. Es interesante en nuestro caso, elegir los servidores proporcionados por *AWS* de Amazon que estén más cerca de nosotros, para disminuir la latencia de datos. Esta medida reduce los costos, garantiza un mejor tiempo de actividad y unos mejores resultados.

Ventajas

Las principales ventajas de la computación en la nube son:

- Integración probada de servicios Red. Por su naturaleza, la tecnología de cloud computing se puede integrar con mucha mayor facilidad y rapidez con el resto de las aplicaciones empresariales.
- Prestación de servicios a nivel mundial. Las infraestructuras de cloud computing proporcionan mayor capacidad de adaptación, recuperación completa de pérdida de datos y reducción al mínimo de los tiempos de inactividad.
- Una infraestructura 100% de cloud computing permite también al proveedor de contenidos o servicios en la nube prescindir de instalar cualquier tipo de software, ya que este es provisto por el proveedor de la infraestructura o plataforma en la nube.

Desventajas

- La centralización de las aplicaciones y el almacenamiento de datos origina una independencia de los proveedores de servicios.
- La disponibilidad de las aplicaciones está sujeta a la disponibilidad de acceso a Internet.
- La confiabilidad de los servicios depende de la "salud" tecnológica y financiera de los proveedores de servicios en la nube.
- Seguridad. La información de la empresa debe recorrer diferentes modos para llegar a su destino, y cada uno de ellos (y sus canales) son un foco de inseguridad.

2.3 Proyectos similares

2.3.1 Facebook Deep Face

Facebook [5] ha presentado un nuevo informe sobre "***DeepFace***", un sistema de reconocimiento facial en el que asegura que lo han hecho aún más preciso, a sólo una décima de la efectividad humana. Está en fase de investigación.

En este nuevo informe apuntan que la forma moderna de reconocimiento de

rostros consiste en varios pasos convencionales: detectar, alinear, representar y clasificar. *Facebook* ha querido revisar ese proceso.

"*Hemos hecho una relectura del paso de alineación y el paso de representación, empleando modelado explícito en 3D para aplicar una transformación a trozos, que deriva en una representación compuesta por una red de nueve capas profundas*", explican en el documento.

Esa red de capas involucra más de 120 millones de parámetros. *Facebook* explica que han "*entrenado*" este modelo con "el mayor conjunto de datos facial hasta la fecha". Es decir, con los datos de los rostros de sus usuarios, pero no de todos. Para probar este sistema han utilizado cuatro millones de imágenes etiquetadas que pertenecían a 4.000 identidades.

"*Nuestro método alcanza una precisión de 97,25% en las caras marcadas en el conjunto de datos, reduciendo el error del estado actual de la técnica en más de un 25%, acercándose estrechamente el desempeño a nivel humano*", han dicho. Son sólo una décima de diferencia con la precisión humana. Las personas pueden hacer ese proceso con un 97,53% de efectividad.

La ambición por perfeccionar este proyecto por parte de *Facebook*, y la cantidad de dinero que han invertido en este propósito es un claro identificador de la importancia del tema en cuestión. Sistemas de reconocimiento facial con tanta precisión pueden servir para aplicarse a mil sistemas o para utilizarse como herramientas en nuevos entornos y aplicaciones.

2.3.2 Tecnología china para el reconocimiento facial y de voz para cerdos

Según la información de *The New York Times*, las grandes compañías tecnológicas chinas están impulsando el reconocimiento facial y de voz y otras tecnologías avanzadas como forma de proteger a los cerdos del país. El objetivo es crear nuevas tecnologías para resolver los problemas de seguridad alimentaria relacionados con los productos porcinos [9].

En este año, muchos de estos animales chinos están muriendo a consecuencia de una enfermedad porcina mortal, amenazando el suministro de cerdo del país, un alimento básico de las mesas de cena chinas.

La tecnología de reconocimiento facial permite detectar si el cerdo no está contento y no come bien, por lo que es capaz de predecir si el cerdo está enfermo en algunos de los casos.

2.3.3 Reconocimiento facial para entrar a ferias y festivales

El *Mobile World Congress* cuenta este año por primera vez con un sistema de reconocimiento biométrico para que los asistentes accedan al recinto. [10]

Los asistentes al *Mobile World Congress (MWC)* pueden usar este año para acceder al recinto su rostro como entrada. La feria internacional de móviles, que se celebra del

25 al 28 de febrero en Barcelona, utilizará por primera vez un sistema de reconocimiento biométrico que permitirá a los usuarios acceder a las instalaciones sin necesidad de portar las acreditaciones físicas tradiciones. Esta nueva herramienta, llamada *Breez (Biometric Recognition Easy Entry Zone),* es solo uno de los ejemplos que muestran como la tecnología puede cambiar por completo el modo de acceder a eventos o festivales.

Para usar este sistema, es necesario subir antes del comienzo de la feria una fotografía a la web del *MWC*. Los organizadores explican en la página de preguntas frecuentes que esta imagen "*se utilizará para crear un perfil biométrico en una fecha cercana al evento*". De esta forma, los asistentes que elijan esta opción podrán acceder al recinto solo con acercar su cara a una cámara.

Capítulo 3
Metodología

3.1 Fases del proyecto

El proyecto lo comenzaremos con un estudio del funcionamiento de *Amazon Rekognition*. Conoceremos el sistema, aprenderemos a ponerlo en funcionamiento en nuestro ordenador, y como conectarlo con nuestro proyecto. Trabajaremos el funcionamiento de sus diferentes funciones para decidir cual de ellas será optima para nuestro caso en particular.

Posteriormente, estudiaremos como funciona *digiKam*: investigaremos el funcionamiento de sus tablas internas para el reconocimiento facial de tal manera que podremos interactuar con ellas en el futuro para carga y descarga de información del sistema.

Una vez conocidas las herramientas a utilizar, nos dispondremos a hacerlas interactuar entre sí, de tal manera que podamos cargar imágenes a analizar desde *digiKam* a nuestro sistema conectado con *Amazon Rekognition*, y posteriormente cargar datos sobre el reconocimiento facial de vuelta a la base de datos.

Ya estructurada la ingeniería de nuestro sistema, trabajaremos en la interfaz gráfica de usuario (*GUI*) para que se pueda utilizar esta herramienta de una manera sencilla y sin conocimientos previos de ningún tipo. Para eso mantendremos un sistema de vistas sencilla e intuitiva que sea capaz de cargar nuevas imágenes, analizar y consultar sobre el resultado de dicho análisis.

3.2 Estudio de recursos utilizados

3.2.1 Eclipse y Java.

Para llevar a cabo este proyecto, usaremos el lenguaje de programación Java, mediante el entorno de programación Eclipse. Hemos decidido utilizar este entorno y lenguaje dada la familiaridad debida a los proyectos de universidad, ya que, en la gran mayoría de trabajos computacionales, los hemos trabajado utilizando estas herramientas.

Java [6] es un lenguaje de programación de propósito general, concurrente orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir

de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos diez millones de usuarios reportados.

Dado el alto nivel, y lo más importante, la existencia de *APIs* de los servicios que vamos a usar, disponibles para este lenguaje, hemos considerado que ha sido la mejor opción para este proyecto.

Eclipse [7] es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "*Aplicaciones de Cliente Enriquecido*", opuesto a las aplicaciones "*Cliente-liviano*" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

3.2.2 Amazon Rekognition

Como ya hemos adelantado anteriormente, utilizaremos *Amazon Rekognition* como cliente de reconocimiento facial. La integración de la *API* a nuestro sistema ha requerido varios pasos, que explicaremos a continuación [8].

Primeros pasos en la integración

Para poder utilizar este producto, ha sido necesario registrarse en *AWS*, servicio de Amazon que se encarga de la gestión de este entorno.

Una vez registrados en *AWS*, se ha creado un usuario administrador que se encargará de manejar las herramientas de *AWS*, dejando una huella rastreable que podremos utilizar para comprobar violaciones de seguridad o mal uso del sistema, dándole así los permisos que sean necesarios para cada una de las herramientas que tengamos que realizar.

Instalación de la Command Line Interface de AWS

Para poder utilizar este servicio, necesitaremos instalar el cliente de *CLI* de *AWS*, para el que necesitamos disponer de Python 3. Esta instalación la hemos hecho mediante **pip**, que facilita mucho la situación.

Pip es un sistema de gestión de paquetes de Python, que normalmente viene ya instalado por defecto.

```
$ pip install awscli --upgrade --user
```

Necesitaremos verificar que se ha instalado correctamente, por lo que insertaremos el comando aws:

```
$ aws --version
```

La salida deberá ser algo similar a esto:

```
aws-cli/1.11.84 Python/3.6.2 Linux/4.4.0-59-generic botocore/1.5.47
```

Configuración de la Command Line Interface de AWS

La mejor manera de configurar la *CLI* de *AWS* es mediante el siguiente comando:

```
$ aws configure
```

Este comando desplegará una serie de entradas de teclado que servirán para rellenar estos datos:

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

AWS Access Key ID y AWS Secret Access Key hacen referencia a los tokens de acceso desde el sistema que utilizaremos al servicio de Amazon en la red. Estos tokens son como una puerta de acceso al servicio, y son personales e intransferibles para cada usuario.

La **región** por defecto hace referencia a donde se conectará el sistema para computar el servicio, puesto que hay varias sedes donde realizarlo.

La **salida** por defecto hace referencia a como traerá de vuelta la información computada por el sistema.

Estas claves se pueden introducir de esta manera o definiéndolos posteriormente en el programa (como es nuestro caso) en la declaración del objeto que hace referencia a las credenciales de *AWS*.

Integración de la API en la aplicación.

Una vez realizado todo esto, es necesario descargarse los *SDK* para poder importar las librerías de *AWS* a nuestro proyecto. Esto lo haremos desde la web de aws.amazon.com y cargaremos los archivos de la carpeta "*lib*" en nuestro proyecto mediante *Build Path > Configure Build Path > Add External JARs*.

Estas librerías las utilizaremos para crear un objeto Usuario que contendrá las credenciales de nuestro usuario de Amazon *AWS*, que se utilizarán en el cliente de *Amazon Rekognition*. El sistema de Amazon está diseñado para que sea necesario llamar al cliente de reconocimiento a la hora de hacer una petición de comparación de caras.

El objeto Usuario será utilizado por nuestro objeto *Rekognition System*, que se encargará de computar la comparación de imágenes y de convertir el resultado en una serie de entradas a la base de datos de *digiKam*.

Vemos que con la *API* de *Amazon Rekognition* podemos obtener varios datos sobre la comparación de caras.

- *Cuadro delimitador*: Indica las coordenadas del cuadro delimitador que rodea

el rostro.

- *Confianza*: Indica el grado de confianza de la comparación.
- *Referencias faciales*: Una matriz de referencias faciales. Para cada referencia (como el ojo izquierdo, el ojo derecho y la boca), la respuesta proporciona las coordenadas x, y.
- *Atributos faciales*: Un conjunto de atributos faciales, incluidos el sexo o si el rostro tiene barba. Para cada atributo, la respuesta proporciona un valor. El valor puede ser de diferentes tipos, como un tipo booleano (si una persona lleva gafas), una cadena (si la persona es un hombre o una mujer), etc. Además, para la mayoría de los atributos la respuesta proporciona también una confianza en el valor detectado para el atributo.
- *Calidad*: Describe el brillo y la nitidez del rostro.
- *Postura*: Describe la rotación del rostro dentro de la imagen.
- *Emociones*: Un conjunto de emociones con confianza en el análisis.
    - *Felicidad*: 23,3%
    - *Neutro*: 13,6%
    - *Sorpresa*: 11,4%...

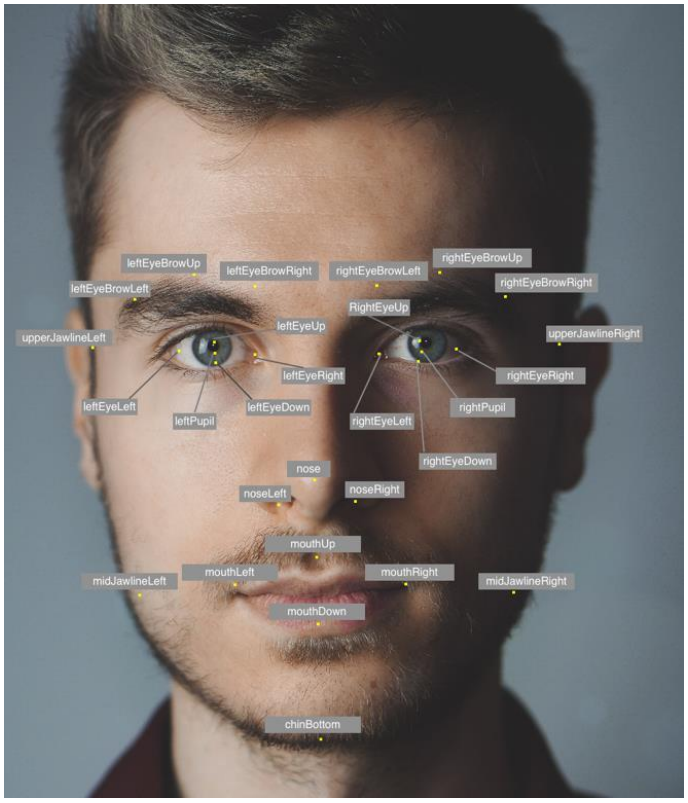


Ilustración 3: Atributos faciales [8]

3.2.3 Base de datos de *digiKam*

Reconocimiento facial

DigiKam también presenta un sistema de reconocimiento facial. Es capaz de analizar la colección y buscar rostros conocidos para luego relacionarlos entre ellos. Las características que ofrece principalmente son muy parecidas a las de *Amazon Rekognition*, pero no presentan todos los atributos de reconocimiento que nos ofrece Amazon, como el análisis de emociones y otros datos comentados anteriormente. Con *digiKam* podríamos reconocer rostros, pero no va más allá de la comparación facial. Encontramos que el marco de reconocimiento facial pierde un poco en comparación con el de Amazon.

Ilustración 4: Marco facial digiKam [13]

Ilustración 5: Marco facial Amazon Rekognition [13]

En esta comparativa se puede ver que el marco facial de *digiKam* (situado a la izquierda) es un poco más pequeño que el de *Amazon Rekognition*, y pierde un rasgo facial como es la barbilla. También hemos podido comprobar gracias a la documentación de *digiKam* que su algoritmo aún es un poco rudimentario. Nos enseñan como se puede confundir un rostro con un cactus.

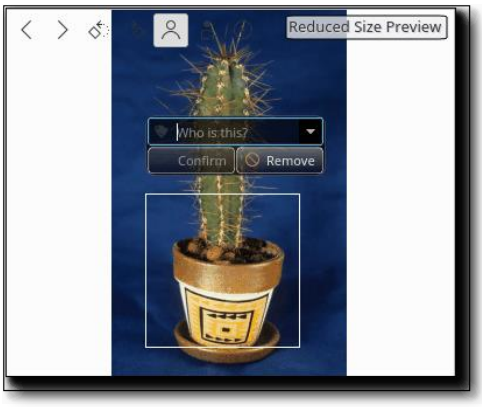


Ilustración 6: Fallo de reconocimiento digiKam [14]

Funcionamiento de tablas internas de reconocimiento facial

Para el almacenamiento de la información sobre el reconocimiento facial, utilizaremos el sistema de tablas orientadas a reconocimiento facial que nos proporciona *digiKam*. La base de datos utilizada es de *SQLite*, y proporciona las siguientes tablas, entre otras:

Tabla 3: Images

Images	
Id	Proporciona el identificador de la imagen, proporcionado automáticamente por *digiKam*.
Álbum	Identificador del álbum, según la posición de la carpeta en el sistema. (/*Source* > Álbum=3, /*Target* > Álbum=4).
Name	Nombre de la imagen.

Tabla 2: Tags

Tags	
Id	Identificador de la etiqueta.
PID	
Name	Nombre de la etiqueta.

Tabla 4: Imagetags

Imagetags	
ImageId	Id de la imagen a la que referencia.
TagId	Id de la etiqueta a la que referencia

Imagetagproperties	
Imageid	Id de la imagen a la que referencia
Tagid	Etiqueta que aparece en dicha imagen
Property	tagRegion
Value	Marco facial <x, y, width, height>

Tabla 5: Imagetagproperties

Se utilizarán estas tablas de tal manera que las imágenes *source* en la tabla *IMAGES* tendrán asignadas una etiqueta de reconocimiento en la tabla TAGS, que asigna dicha imagen con el nombre de la persona que aparece en ella. Esta relación ocurre mediante la tabla *IMAGETAGS*. La tabla *IMAGETAGPROPERTIES* se utilizará para cargar la información de los marcos faciales del reconocimiento de la imagen *target*.

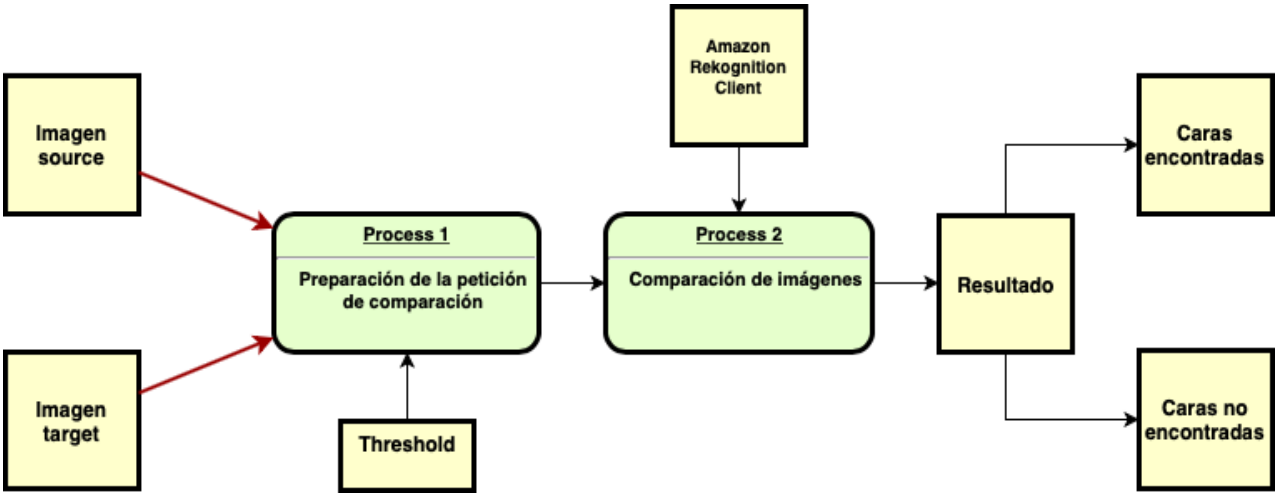
Algo que presenta la base de datos de *digiKam*, es que realiza los "*commits*" de manera automática cuando se modifica la tabla, con una nueva inserción, o bien con una actualización. En cambio, cuando se realiza una consulta y se termina de utilizar la conexión, el sistema no cierra automáticamente dicha conexión, por lo que es conveniente cerrarla manualmente al final de su uso. En caso contrario ocurrirá una ralentización del sistema que hará que falle. Es necesario que *digiKam* esté abierto mediante el uso de la aplicación, ya que, si no, no actualiza la tabla *IMAGE*, encargada de gestionar las imágenes de la biblioteca.

3.3 Primera aproximación al funcionamiento del sistema

3.3.1 Reconocimiento de rostro en imagen

Para poder llevar acabo el estudio del funcionamiento de *Amazon Rekognition*, utilizaremos los ejemplos proporcionados por la guía de usuario de *AWS*. Contiene unos códigos que explican el funcionamiento y la secuencia a seguir para poder llevar a cabo la comparación de imágenes y el reconocimiento facial, así como la respuesta dada.

La secuencia dada es la siguiente:



Al sistema se le carga una imagen a analizar (imagen *target*), una imagen con la cara que se quiere buscar (imagen *source*), y un *threshold* que determina el porcentaje mínimo de confianza del reconocimiento de dicha cara. Cuando el programa realiza la búsqueda del retrato de la imagen *source*, el resultado es un **conjunto** α de tamaño 1 de caras encontradas, y un **conjunto** β de N-1 de caras no encontradas, siendo "N" el número de caras en la imagen *target*. Dicho conjunto α contiene el porcentaje de confianza de la comparación. Por ejemplo, si el rostro no está del todo claro en la imagen, y solo consigue identificar en el marco unos pocos elementos, el sistema tendrá menos elementos con los que realizar la comparación, por lo que en caso de resultar afirmativo el análisis, se presentará un porcentaje de confianza más bajo.

- Esto suele pasar con las imágenes tomadas desde arriba, o de espaldas mirando hacia detrás, según hemos podido comprobar. En muchos de los casos, no se llega a reconocer la cara.

Ilustración 7: Análisis facial [15]

Estos conjuntos α y β, contienen información sobre cada rostro de la imagen. Lo que más nos interesa es el *Bounding Box*, un atributo que contiene el marco facial del rostro. Esta información es la que se introducirá en la tabla *IMAGETAGPROPERTIES* de la base de datos de *digiKam*, junto con el identificador de la imagen analizada y la etiqueta asignada al perfil del rostro reconocido.

3.3.2 Inserción de nueva imagen *source*

Para que el sistema funcione bien, necesitaremos una secuencia que inserte correctamente nuevas imágenes *source* a la base de datos:

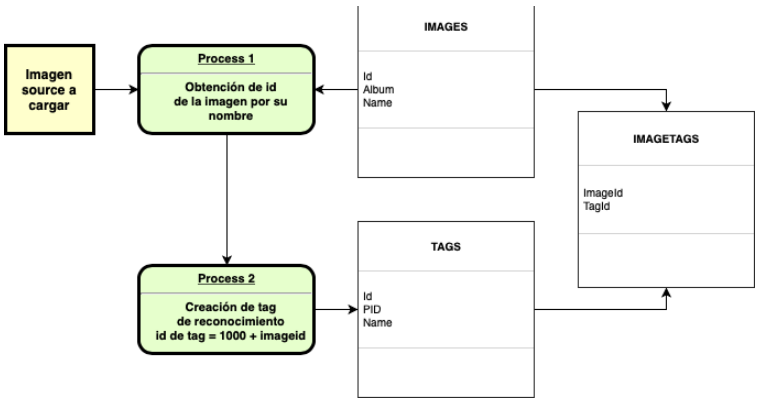


Ilustración 8: Secuencia de inserción

Cuando se carga una imagen *source*, se buscará el id asignado por *digiKam* a dicha imagen en la biblioteca. Es necesario que, en el momento de la inserción, la aplicación de *digiKam* esté abierta, para que así, se le asigne un identificador automáticamente. Una vez obtenido este Id, se creará una etiqueta que relacione dicha imagen *source*. Esta relación se hará mediante la tabla *IMAGETAGS*. De esta manera, se habrá introducido correctamente la imagen *source* para la futura asignación en caso de que se encuentre dicho rostro en alguna imagen *target*.

3.3.3 Reconocimiento de varios rostros en imagen

Lo siguiente será buscar un algoritmo que sea capaz de buscar varios rostros en una imagen a analizar, ya que el sistema de *Amazon Rekognition* solo nos permite comparar una imagen *source* con un retrato con una imagen *target* con varios retratos, y decirnos si dicha imagen *source* se encuentra en dicha imagen *target*, y los marcos faciales de los rostros no reconocidos.

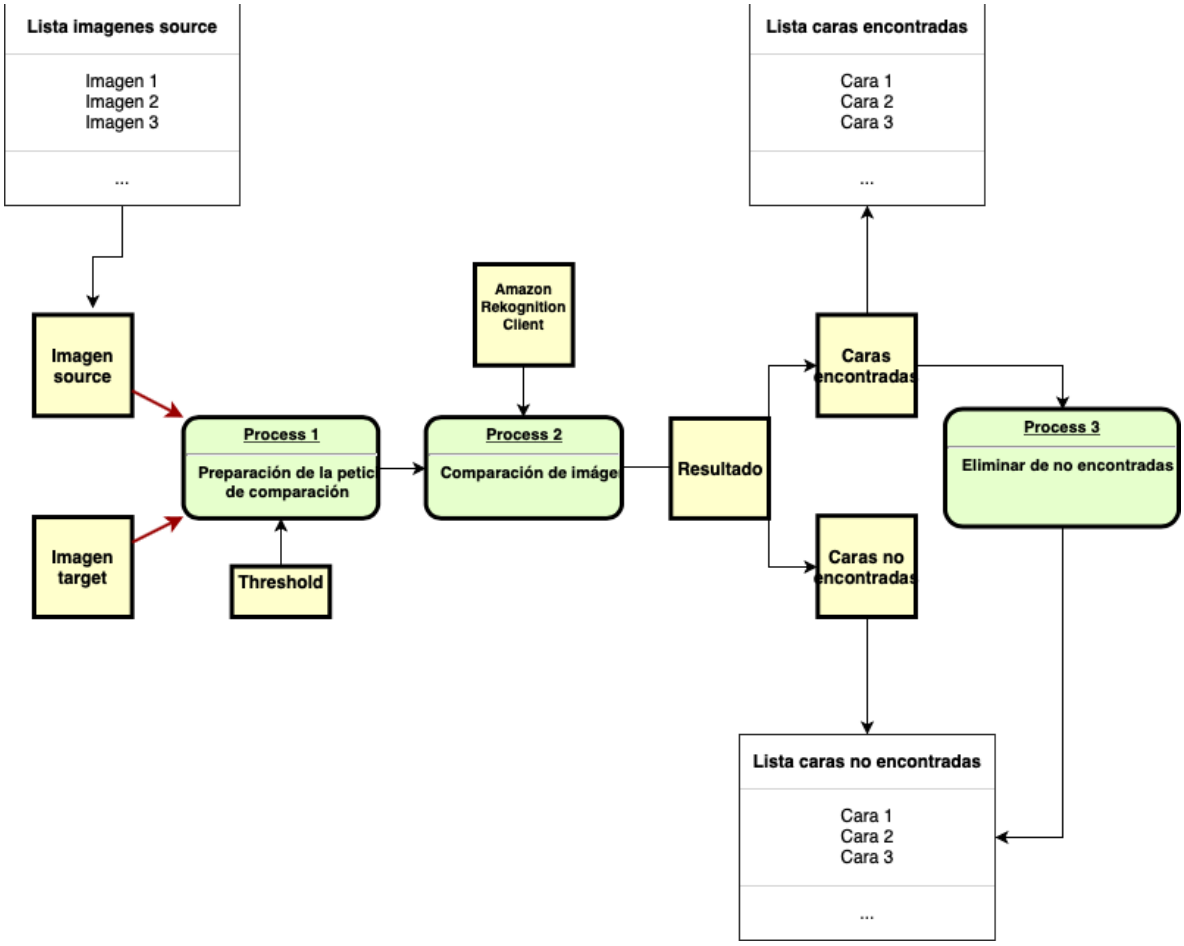


Ilustración 9: Secuencia de reconocimiento

La forma con la que hemos solucionado este problema es bastante sencilla. Realizaremos la comparación de rostros con todas las imágenes *source* que haya en la

base de datos hasta que se recorra toda la lista o bien hasta que se hayan reconocido todas las caras, esto es, que el conjunto β esté vacío. Cuando aparezca una coincidencia, se insertará dicho rostro en una lista que contendrá las caras reconocidas, y se eliminará de la lista de rostros no encontrados dicha cara, en caso de que estuviera contenida en dicho conjunto.

Con estas secuencias, ya tenemos un sistema que es capaz de cargar nuevas imágenes *source* y imágenes *target* a analizar. Lo siguiente es pensar que haremos con las caras que no han sido reconocidas por el sistema.

3.3.4 Inserción de caras no reconocidas en el sistema

Cuando tenemos que el conjunto β no está vacío, crearemos nuevas imágenes que denotaremos como *Unknown* mediante el recortado de la imagen *target* con el marco de caras obtenido por el sistema de reconocimiento. Cuando se crea esta imagen, se mueve a la carpeta *Unknown* y se hace un proceso similar en la base de datos a cuando se introduce una nueva imagen *source*, solo que además se relaciona dicho rostro *Unknown* con la imagen *target* analizada. Para que se produzca este cambio en la base de datos es necesario que *digiKam* esté abierto.

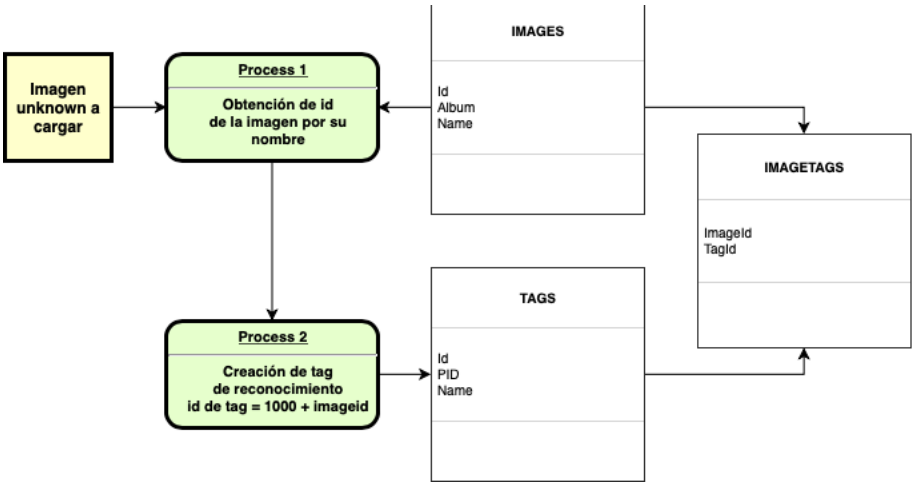


Ilustración 10: Inserción de Unknown

Para evitar que se produzca un duplicado en las imágenes *Unknown*, cuando una imagen *Unknown* nueva vaya a ser insertada, se producirá una secuencia de búsqueda de match de reconocimiento facial entre dicha imagen *Unknown* y las otras del conjunto. Si dicha cara *Unknown* nueva coincide con alguno de los rostros *Unknown* encontrados anteriormente, se obtendrá el id de la imagen *Unknown* existente en la base de datos y se consultará sobre la etiqueta del perfil facial, para usar esta etiqueta como etiqueta de reconocimiento sobre la imagen *target* previamente analizada.

En caso de que no coincida ninguna, se realizará el proceso de inserción de *Unknown* en la base de datos como si fuera una *Unknown* nueva.

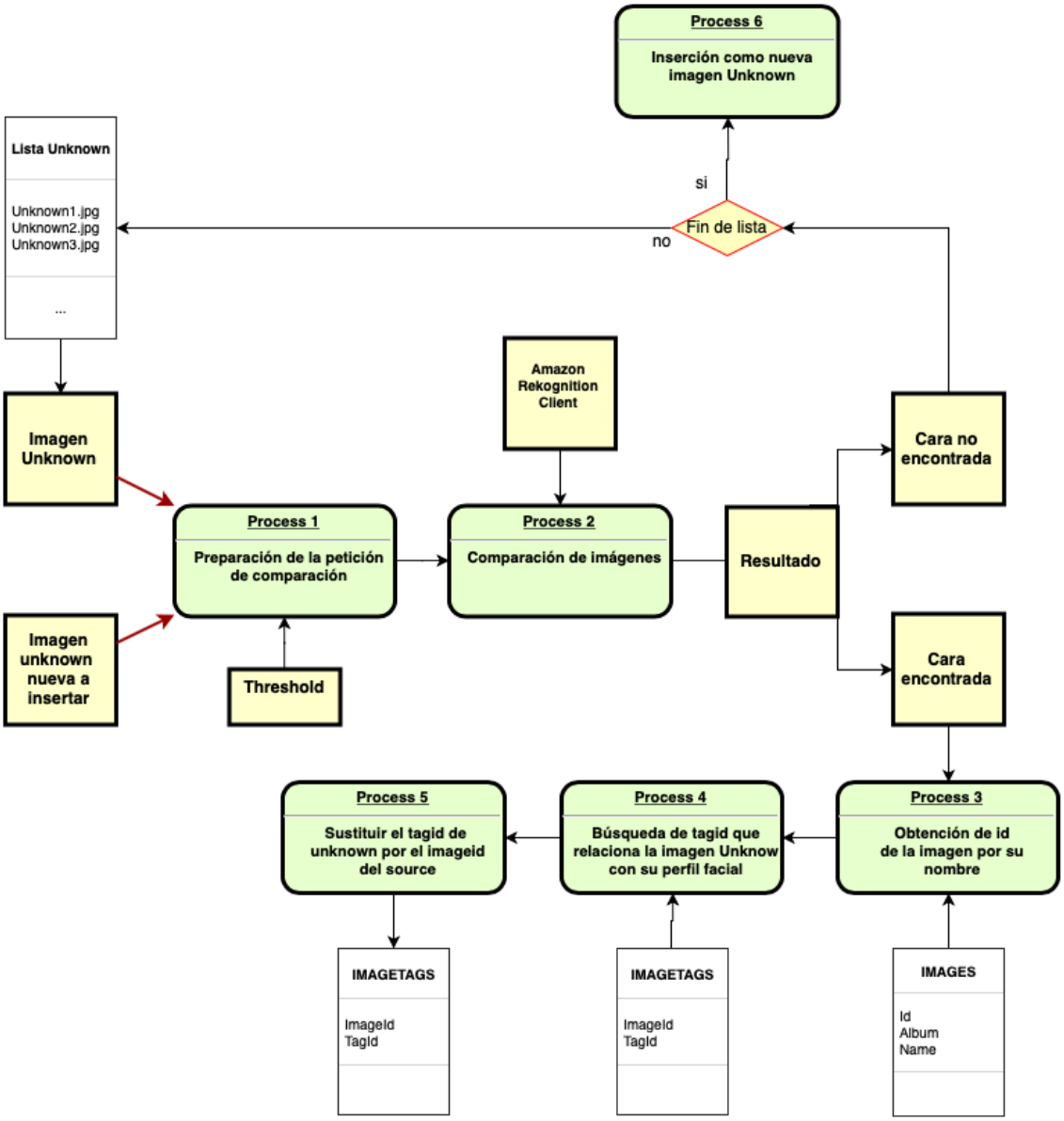


Ilustración 11: Inserción de imagen source teniendo en cuenta Unknown

Para mantener el sistema consolidado, cada vez que se cargue una nueva imagen *Source*, el sistema buscará si dicho rostro que se va a cargar existe ya en la base de datos como *Unknown*.

Se irán introduciendo las imágenes *Unknown* en el sistema comparándolas con dicha imagen *source* a insertar, de tal manera que en el momento en el que se produzca un match, se obtendrá el id de la imagen *Unknown* para intercambiar en *IMAGETAGS* la relación entre la etiqueta de reconocimiento *Unknown* con su nueva imagen *source*.

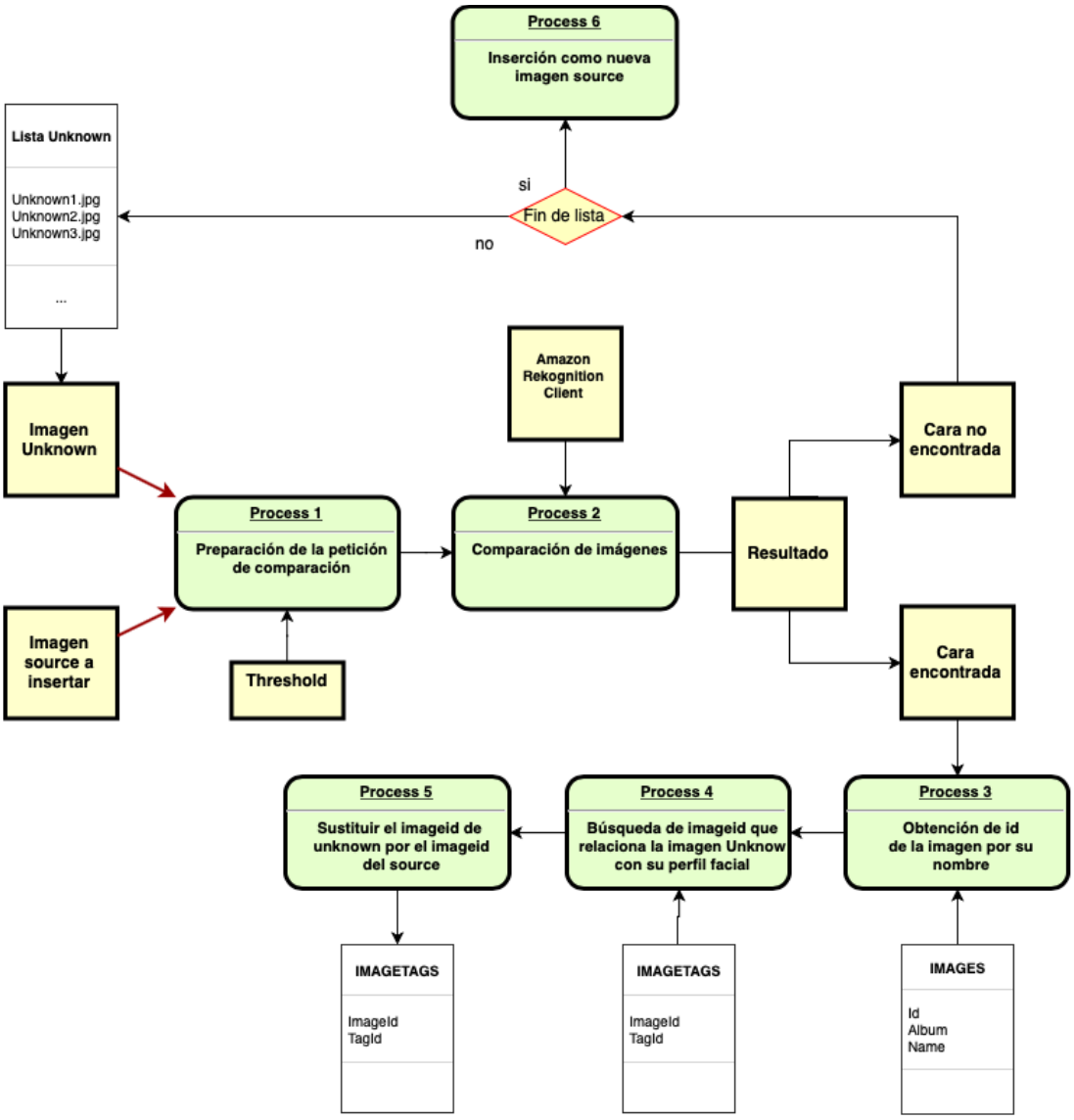
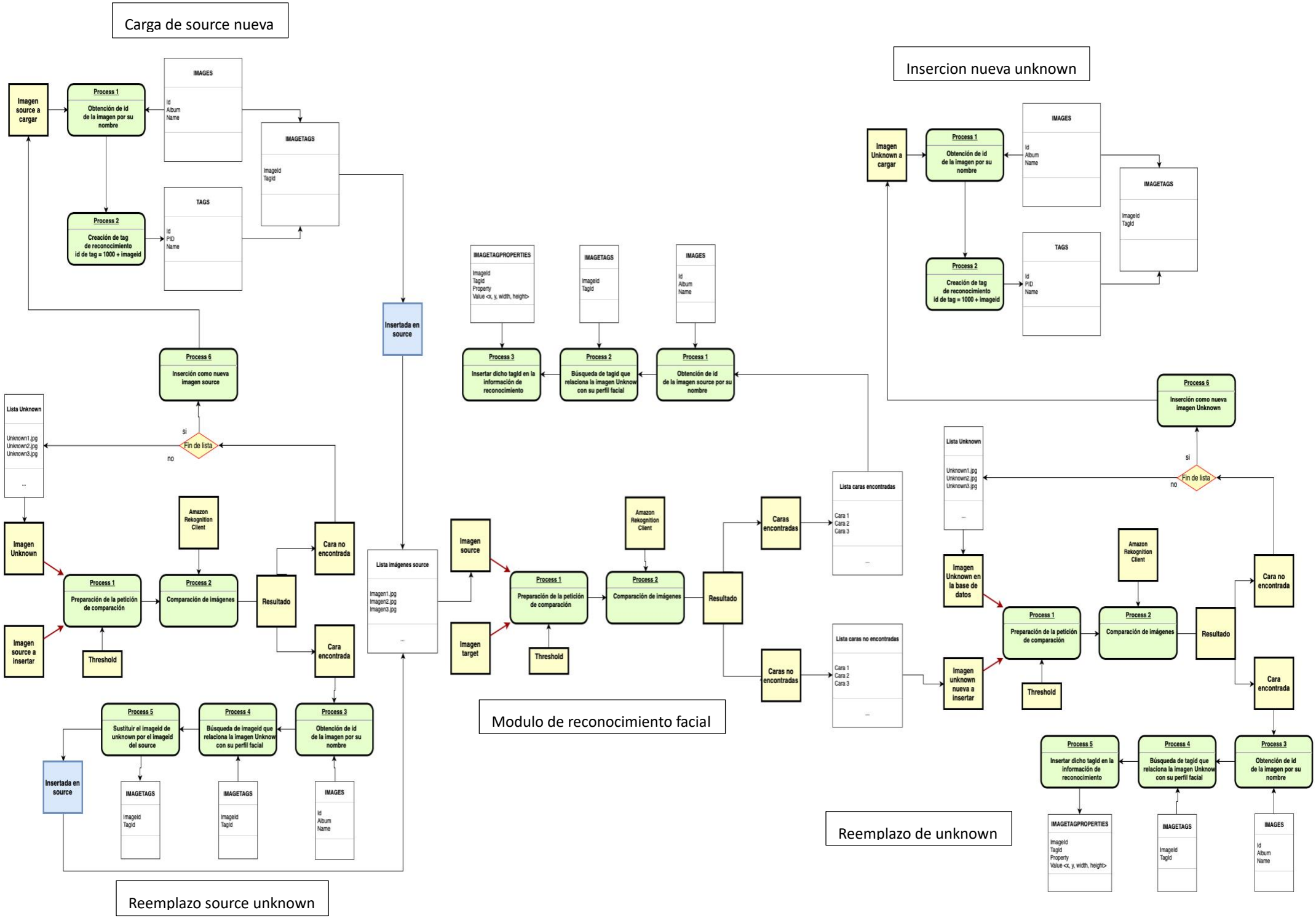


Ilustración 12: Reasignación de Unknown a Source

Con esto tenemos un sistema que se va enriqueciendo constantemente de una manera eficiente y consolidada. Si juntáramos todos estos módulos mencionados anteriormente, el resultado de nuestro sistema sería el siguiente:


Carga de source nueva
Imagen source a cargar
Process 1
Obtención de id de la imagen por su nombre
IMAGES
Id
Album
Name
IMAGETAGS
ImageId
TagId
Process 2
Creación de tag de reconocimiento id de tag = 1000 + imageid
TAGS
Id
PID
Name
Insertada en source
Process 6
Inserción como nueva imagen source
Lista Unknown
Unknown1.jpg
Unknown2.jpg
Unknown3.jpg
si
Fin de lista
no
Imagen Unknown
Amazon Rekognition Client
Process 1
Preparación de la petición de comparación
Process 2
Comparación de imágenes
Resultado
Cara no encontrada
Cara encontrada
Imagen source a insertar
Threshold
Process 5
Sustituir el imageid de unknown por el imageid del source
Process 4
Búsqueda de imageid que relaciona la imagen Unknow con su perfil facial
Process 3
Obtención de id de la imagen por su nombre
Insertada en source
IMAGETAGS
ImageId
TagId
IMAGETAGS
ImageId
TagId
IMAGES
Id
Album
Name
Reemplazo source unknown
Lista imágenes source
Imagen1.jpg
Imagen2.jpg
Imagen3.jpg
Imagen source
Imagen target
Process 1
Preparación de la petición de comparación
Threshold
Amazon Rekognition Client
Process 2
Comparación de imágenes
Resultado
Caras encontradas
Caras no encontradas
Lista caras encontradas
Cara 1
Cara 2
Cara 3
Lista caras no encontradas
Cara 1
Cara 2
Cara 3
Modulo de reconocimiento facial
IMAGETAGPROPERTIES
ImageId
TagId
Property
Value <x, y, width, height>
IMAGETAGS
ImageId
TagId
IMAGES
Id
Album
Name
Process 3
Insertar dicho tagId en la información de reconocimiento
Process 2
Búsqueda de tagId que relaciona la imagen Unknow con su perfil facial
Process 1
Obtención de id de la imagen source por su nombre
Insercion nueva unknown
Imagen Unknown a cargar
Process 1
Obtención de id de la imagen por su nombre
IMAGES
Id
Album
Name
IMAGETAGS
ImageId
TagId
Process 2
Creación de tag de reconocimiento id de tag = 1000 + imageid
TAGS
Id
PID
Name
Process 6
Inserción como nueva imagen Unknown
Lista Unknown
Unknown1.jpg
Unknown2.jpg
Unknown3.jpg
si
Fin de lista
no
Imagen Unknown en la base de datos
Amazon Rekognition Client
Process 1
Preparación de la petición de comparación
Process 2
Comparación de imágenes
Resultado
Cara no encontrada
Cara encontrada
Imagen unknown nueva a insertar
Threshold
Process 5
Insertar dicho tagId en la información de reconocimiento
Process 4
Búsqueda de tagId que relaciona la imagen Unknow con su perfil facial
Process 3
Obtención de id de la imagen por su nombre
IMAGETAGPROPERTIES
ImageId
TagId
Property
Value <x, y, width, height>
IMAGETAGS
ImageId
TagId
IMAGES
Id
Album
Name
Reemplazo de unknown

3.4 Diseño de la interfaz gráfica de usuario

Una vez realizado el esqueleto de nuestro sistema, será necesario diseñar una interfaz gráfica de usuario que sea capaz de interactuar de manera sencilla con el usuario. Si se realiza una observación a rasgos generales del funcionamiento del sistema, podemos distinguir que hay 3 maneras con las que podemos interactuar con él:

1. Trabajo con imágenes *source*
2. Trabajo con imágenes *target*
3. Consultas sobre el sistema

Estas tres funcionalidades además tendrán sus subfunciones o funciones de segundo nivel:

1. Trabajo con imágenes *source*
    a. Inserción de imagen *source* al sistema
    b. Mostrar imágenes *source*
    c. Imágenes Unknown
2. Trabajo con imágenes target
    a. Inserción de imágenes target a analizar
    b. Mostrar historial de imágenes analizadas
3. Consultas sobre el sistema

Basándose en este esquema, se montará un sistema de paneles que se adapten a nuestro requerimiento:

- El *Main Frame* contendrá 2 paneles principales: El *Panel Principal* (Panel 1) y la *Botonera Principal*. La botonera principal se encargará de cambiar el *Panel Principal* en función del lugar del menú principal en el que se esté localizado en la aplicación. Este menú se cambiará mediante los botones contenidos en la *Botonera Principal*. Tendrán las funciones de *Source*, *Target* y *Consulta*. Cada uno de estos lugares tendrá su propia alteración del *Panel Principal*.

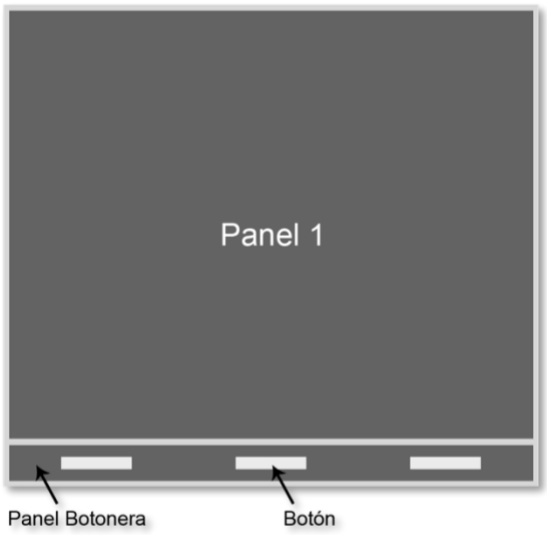


Ilustración 13: Diseño Main Frame

- En el menú de *Source*, el *Panel Principal* insertará la *Botonera Secundaria* y el *Panel Secundario*. Dependiendo del lugar en el menú secundario que se encuentra en la *Botonera Secundaria*, el *Panel Secundario* tendrá una funcionalidad determinada. Este esquema lo encontraremos solo en las secciones de *Source y Target*.

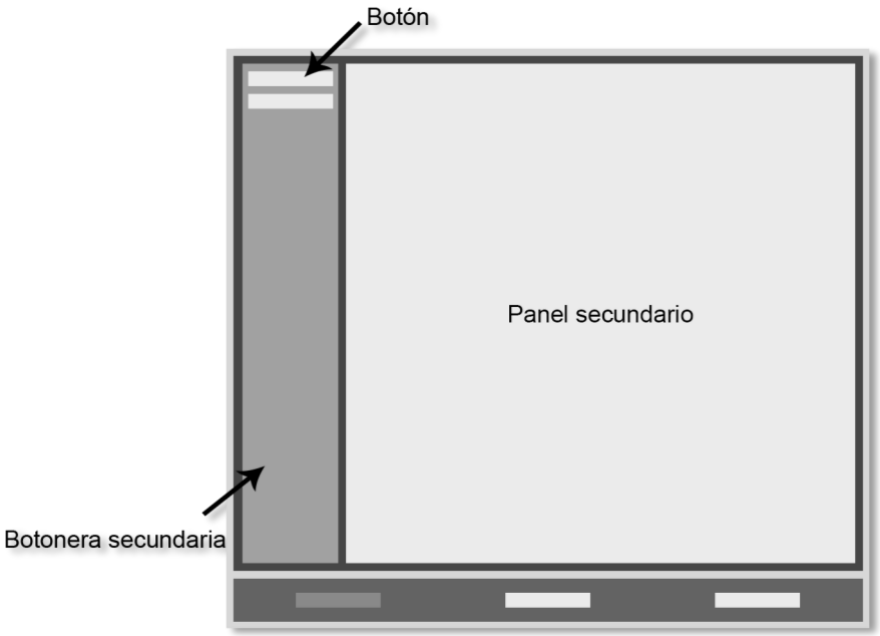


Ilustración 14: Diseño Panel principal

- En el menú de *Consulta*, insertaremos un nuevo *Panel Consulta* que se encargará de activar la consulta seleccionada. El resultado de dicha consulta se mostrará en el *Panel Resultado*. De esta manera se podrá alternar entre consultas de manera sencilla y intuitiva, y sin cambiar mucho la estructura de la interfaz.

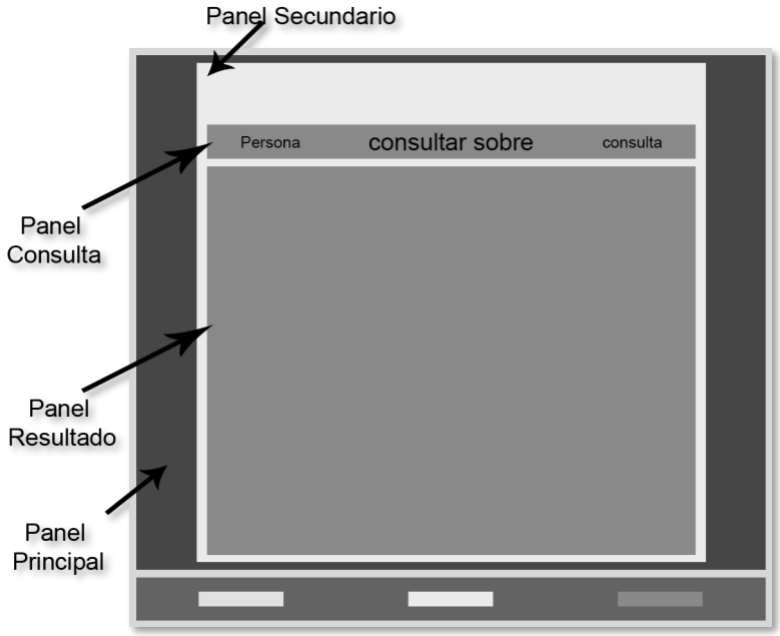


Ilustración 15: Diseño Panel Consulta

3.5 Objetos del sistema

3.5.1 Utilidades principales

DBContext

Las llamadas a la base de datos serán gestionadas mediante el objeto "*DBContext*", que contiene la conexión a la base de datos, así como las rutas a las carpetas de "*Source*", "*Target*", "*Historial*" y "*Unknown*"; y la dirección de la base de datos en el sistema. Este objeto se encarga de funcionalidades como obtener los nombres de las imágenes *source* y target, obtener el identificador de las imágenes, añadir las propiedades de los marcos faciales a la base de datos, mover imágenes en el sistema de archivos…

User

El objeto "*User*" contiene las credenciales para el acceso al sistema de AWS, así como el cliente de reconocimiento de *Amazon Rekognition*. Cada vez que sea necesario hacer una comprobación entre varias imágenes se realizará una llamada a este objeto. Se almacenan las claves privadas y publicas, así como la región donde se computarán los cálculos, *US_EAST_1* en nuestro caso.

ImageData

El objeto "*ImageData*" lo usaremos como un objeto que contendrá información sobre la imagen que se está trabajando, como su identificador, su nombre, y las propiedades del marco facial (*x, y, width, height*). También se encarga del trabajo con las imágenes, como la obtención de la orientación y su correspondiente corrección, o del recorte de los marcos faciales de las caras que no se encuentran en el análisis, para su futura consideración en el álbum de imágenes *Unknown*.

Rekognition System

El objeto *Rekognition System* se encargará de realizar el análisis facial de la imagen proporcionada como target sobre los retratos proporcionados como *source*, aceptando solamente aquellas con un porcentaje de confianza superior al proporcionado por el *threshold*. El arranque de esta función dejará como resultado unos atributos que contendrán una lista de *ImageData* que contienen la información de los rostros encontrados, y otra lista que contendrá la de los rostros no encontrados.

3.5.2 Interfaz gráfica de usuario

Controladores

Para la funcionalidad de la interfaz gráfica de usuario, utilizaremos 2 objetos controladores principales. El primer controlador se encargará de gestionar el *Panel Principal* mediante las distintas acciones de los botones de la *Botonera Principal*. El segundo controlador se utilizará para gestionar el *Panel Secundario* mediante las funciones de la *Botonera Secundaria*. Dependiendo de las funciones del menú, se realizarán los cálculos correspondientes a dicha funcionalidad antes de mostrar los paneles. Para gestionar las consultas existe otro controlador que realizará los cálculos.

Paneles

Sección *Source*

Será necesario de un panel que permita realizar el proceso de inserción de imágenes *source*. Se denominará *PanelCargaImagen*. Este panel contendrá un botón

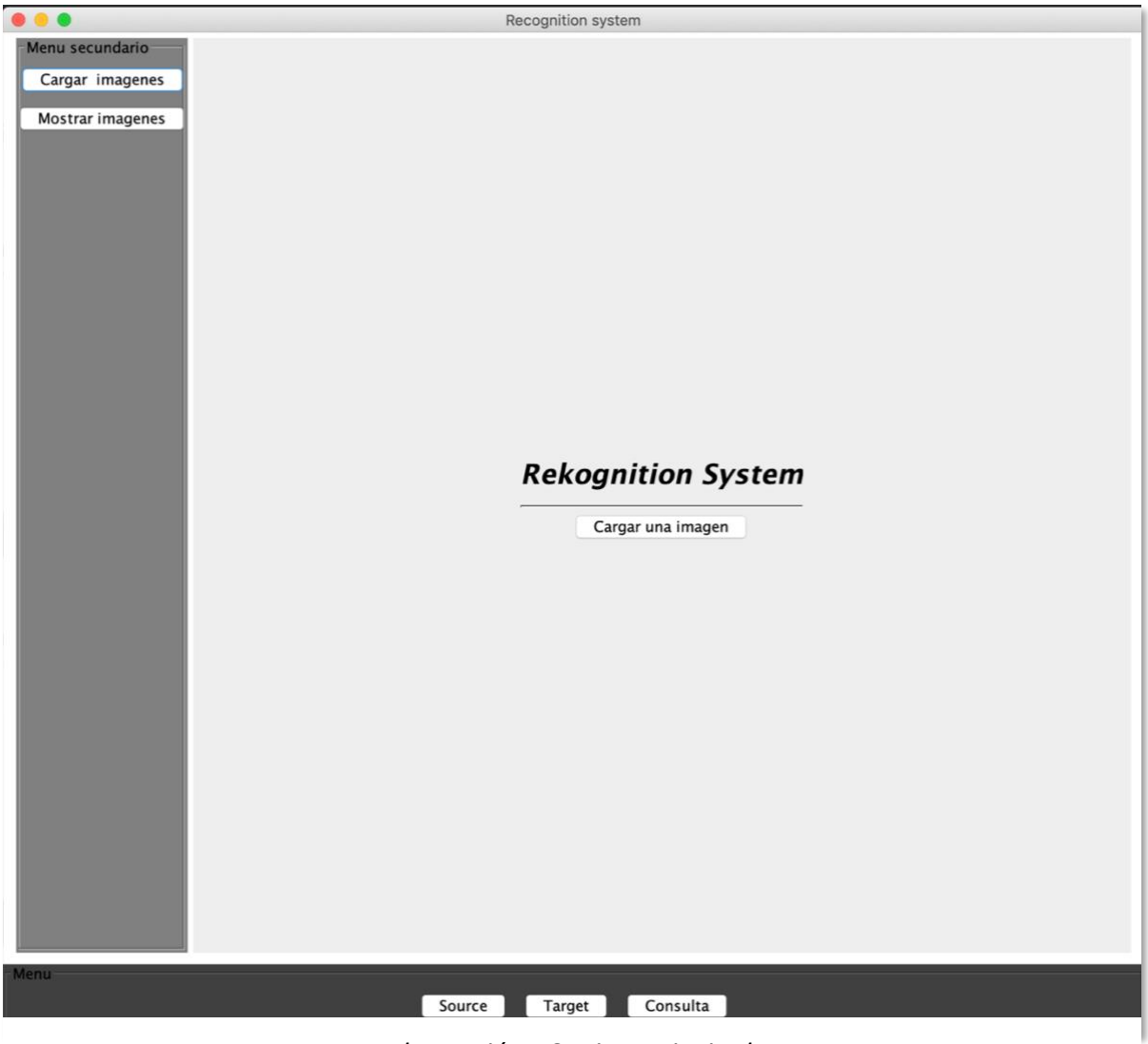


Ilustración 16: Vista principal

que permitirá cargar una imagen al sistema. Desplegará un navegador del sistema (*FileChooser*) que permitirá seleccionar de manera intuitiva la imagen que se quiere cargar.

Posteriormente, se necesitará un panel con un botón que confirme la selección de la imagen. Será necesario re escalar la imagen a un tamaño que sea capaz de caber en el panel. En este caso, se re escalará la altura de la imagen para que sea proporcional a un ancho de 300px, mediante la función `getScaledInstance` del objeto *BufferedImage* de Java. También se añadirá un botón para cambiar la imagen en caso de que se haya producido un cambio de opinión.

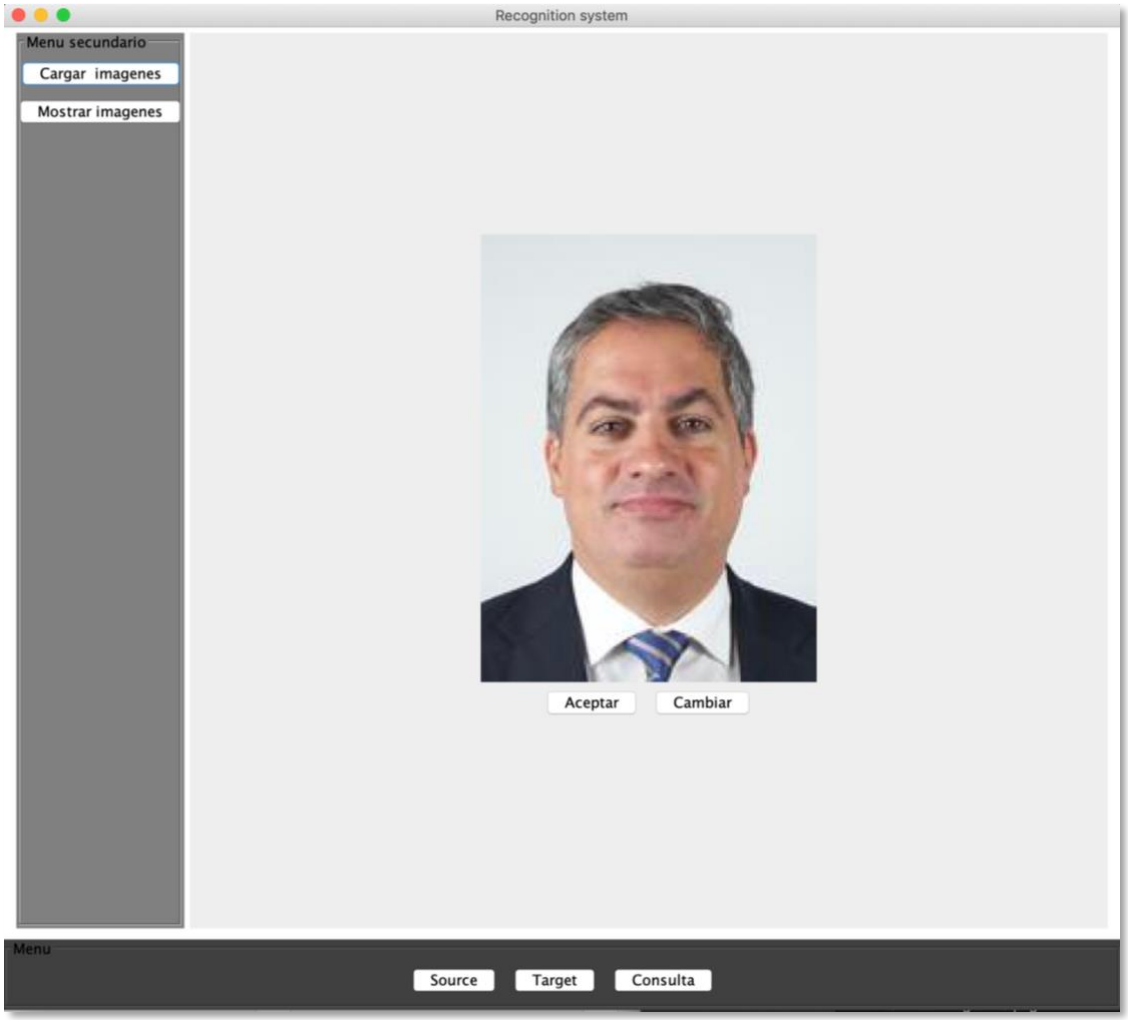


Ilustración 17: Panel de confirmación de elección [16]

Una vez realizadas estas acciones, se computarán los datos necesarios para realizar la inserción de la imagen *source* que hemos comentado anteriormente, y se mostrarán las imágenes *source* añadidas anteriormente en un nuevo panel, que denotaremos como *PanelMostrarImagenesSource*. Este panel hereda de PanelGridImagenes, que muestra una cuadrícula de imágenes, todos ellos re escalados a 200px de ancho en una cuadrícula que tendrá 3 columnas, y *"N/3 + (N%3)*" columnas, siendo "N" el número de imágenes. Dicho panel se insertará en un ScrollPane para poder recorrer

todas las imágenes.

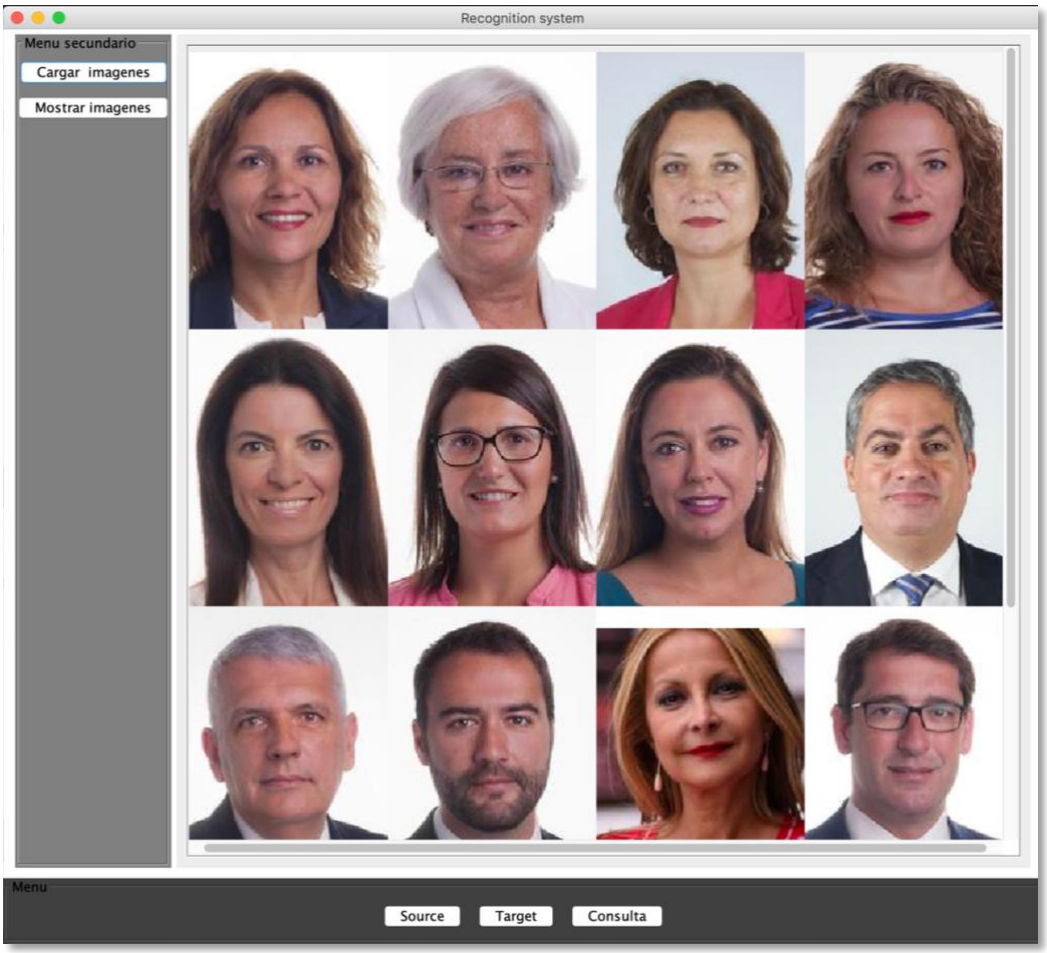


Ilustración 18: Panel que muestra las imágenes Source [16]

Se han añadido también una opción en el menú para poder catalogar manualmente rostros que el usuario conozca y que estén como imágenes *Unknown*, para que la base de datos crezca lo mínimo posible.

Esta opción mostrará los rostros que aun no han sido catalogados para que se seleccione el que se quiere identificar.

Una vez se clique la imagen, se mostrará un panel nuevo que nos permitirá cambiar el nombre del rostro y una vez hecho esto, movido como nueva imagen *source*.

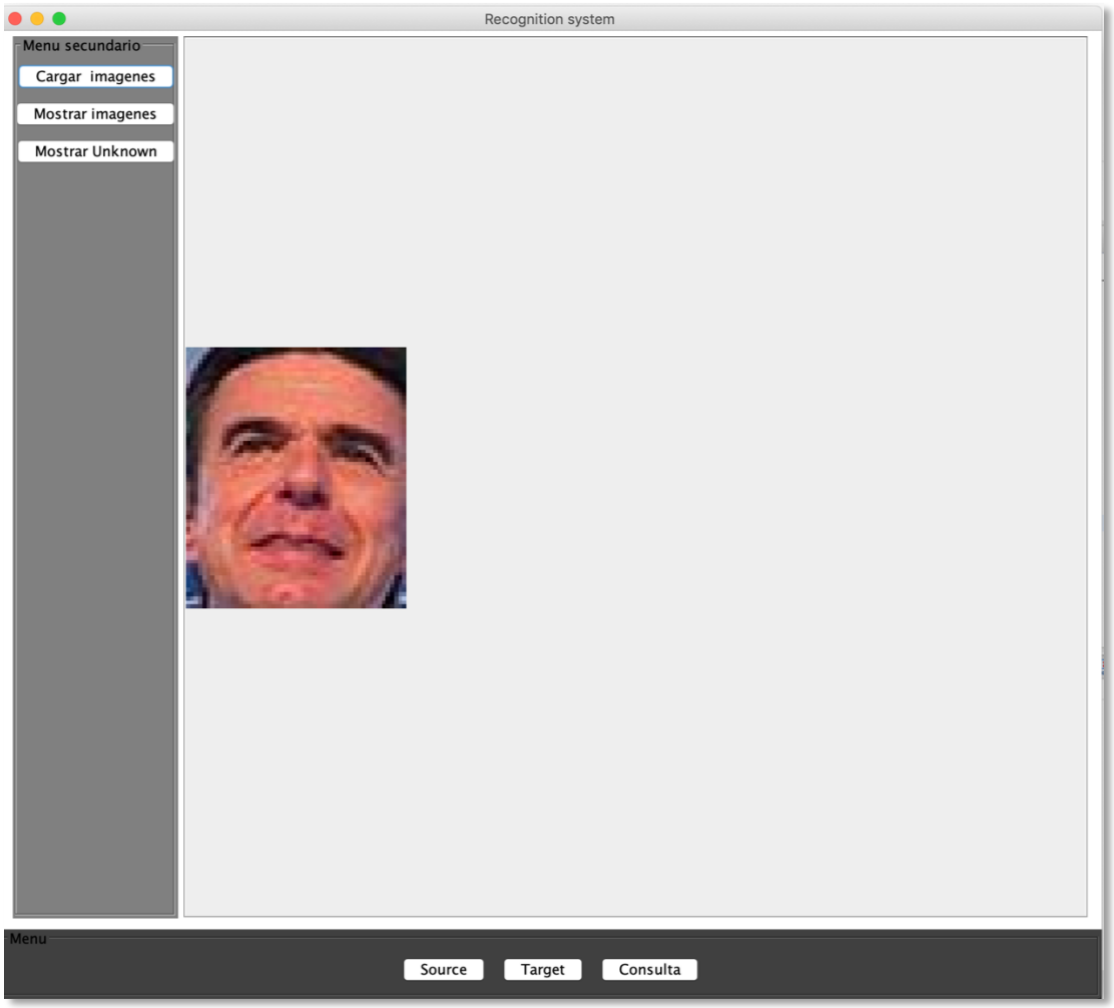


Ilustración 19: Vista Imágenes Unknown [13]

Mantener la base de datos organizada es muy importante, por eso se debe escoger el nombre que identifique esta imagen con precisión.

Se ha añadido la opción de *Cambiar*, en caso de que se requiera volver a la ventana de selección a modificar la elección.

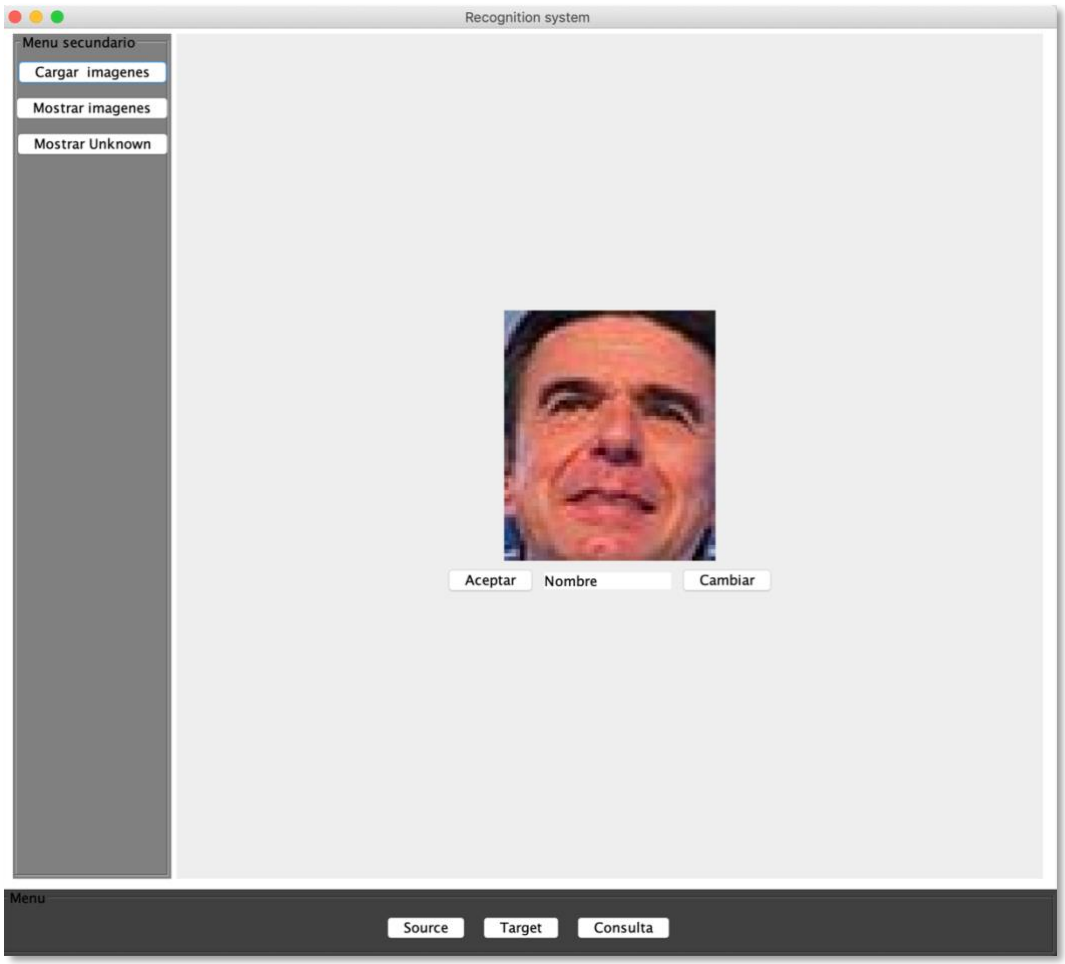


Ilustración 20: Cambio de nombre Unknown[13]

Sección Target.

Para la sección de *target*, se necesitará un panel que sea capaz de cargar imágenes en el sistema, tal y como se explicó en la sección de *Source*. Para esta función, heredaremos el panel utilizado anteriormente, así como el panel de confirmación de la imagen.

Una vez confirmada la imagen, será necesario computar la secuencia de análisis y mostrar los resultados. Para esto, el sistema creará un panel que muestre la imagen con la correspondiente información sobre el reconocimiento y los marcos faciales. Se pintará de color verde los marcos faciales de las caras que se hayan reconocido desde la función *paintComponent* del panel, así como los marcos faciales de las caras no reconocidas, que contendrán un marco de color rojo.

Este panel lo será introducido en un *ScrollPane*, ya que hacer esto en lugar de re escalar la imagen facilitará que se pueda ver detalladamente todos los aspectos de la imagen y de su correspondiente reconocimiento.

Ilustración 21: Panel de información de análisis [17]

Sección Consulta.

Para la sección de consultas, es necesario un panel que se encargara de seleccionar que consulta se realizará sobre que persona, y otro panel que muestre dicha información. En el Panel Consultas, aparecerá un *PanelAccionesConsulta* que contendrá un *Combo Box* para la selección del retrato a consultar, y otro *Combo Box* que mostrará las distintas opciones. Se utilizará el *Combo Box* ya que un objeto de tipo DropDownList permitirá realizar estas selecciones de una manera muy intuitiva. Desde el *Panel Consulta*, que hará de controlador en este caso, se pasarán los distintos nombres de retratos y las consultas previamente programadas al *PanelAccionesConsulta* y se crearán los distintos controladores de los botones, así como sus funcionalidades.

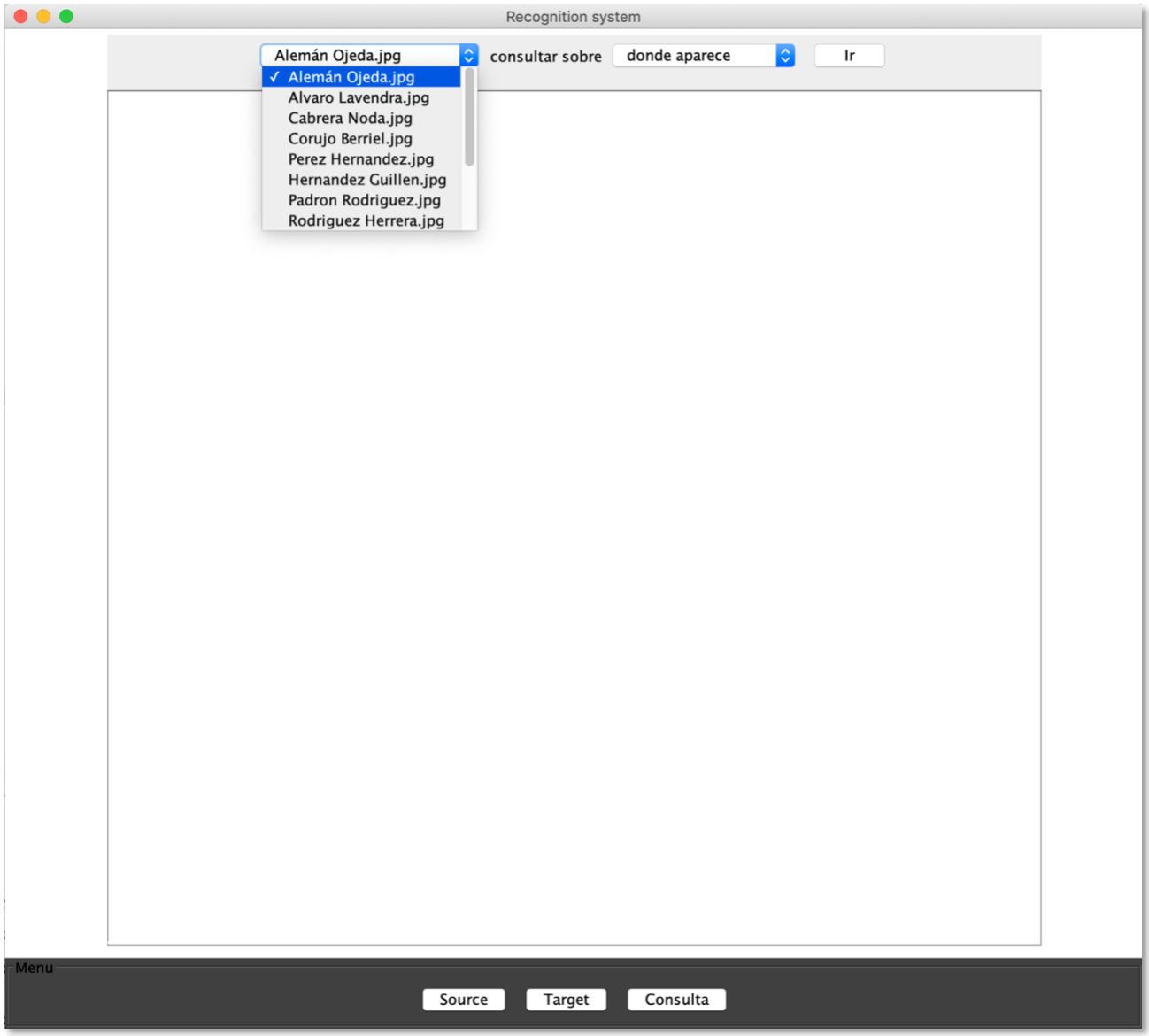


Ilustración 22: Panel de Consulta

1. **Consulta "Donde aparece":** una de las consultas más básicas que se pueden realizar en un sistema de reconocimiento es la de donde aparece una persona. Para conseguir esta información, es necesario obtener el identificador de *IMAGES* para posteriormente obtener su tag de perfil facial. Con esta etiqueta, se podrá consultar en la tabla *IMAGETAGPROPERTIES* para saber en que ImageId aparece dicha etiqueta. Obteniendo el nombre de la imagen asignado a ese ImageId, el sistema conocerá el nombre de las imágenes que hay que buscar en Historial.

En este caso, se muestran las imágenes en una cuadrícula como ya se ha hecho en el panel de muestra de imágenes *Source* y el panel de muestra del historial de análisis, heredando de *PanelGridImagenes*.

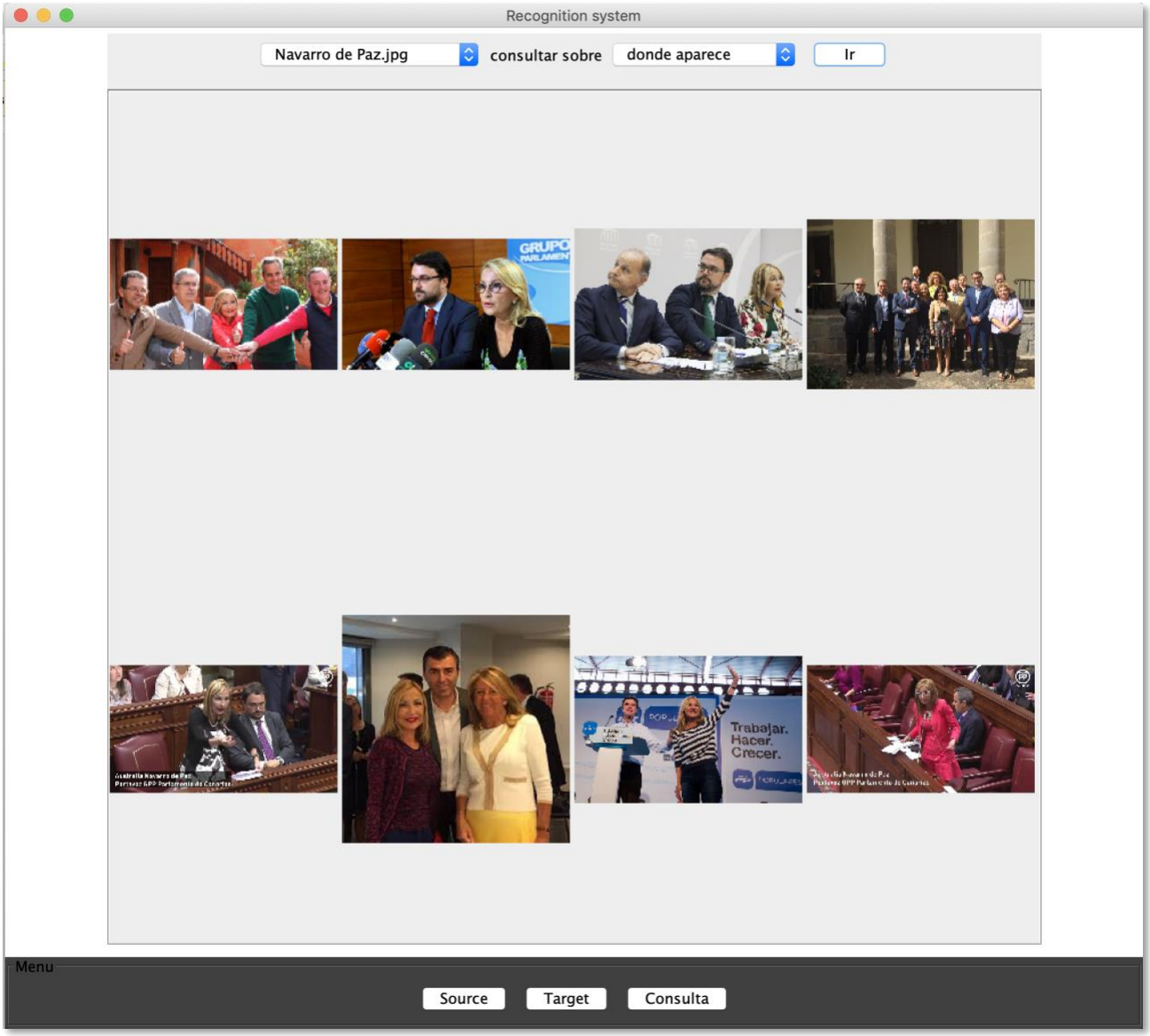


Ilustración 23: Información sobre donde aparece [18][19][20][21][13]

2. **Consulta "Con quien aparece"**: otro dato que sería interesante conocer, es con quien aparece dicha persona en otras imágenes analizadas. Para obtener esta información, es necesario saber el ImageId del retrato a consultar, para posteriormente obtener su etiqueta de perfil facial. Con este tag, se podrán obtener los identificadores de las imágenes en las que aparece este rostro. Conociendo estas imágenes, mediante una consulta a la tabla *IMAGETAGPROPERTIES* podremos obtener los identificadores de los perfiles sociales que sean distintos al que estamos consultando, para

posteriormente obtener la imagen del retrato asignado a dicho perfil social. Esta información se computará desde el controlador, y se le pasará al panel *PanelConQuien* las imágenes que se deberán mostrar, siguiendo el esquema para mostrar imágenes en los paneles mediante la herencia de *PanelGridImagenes*.

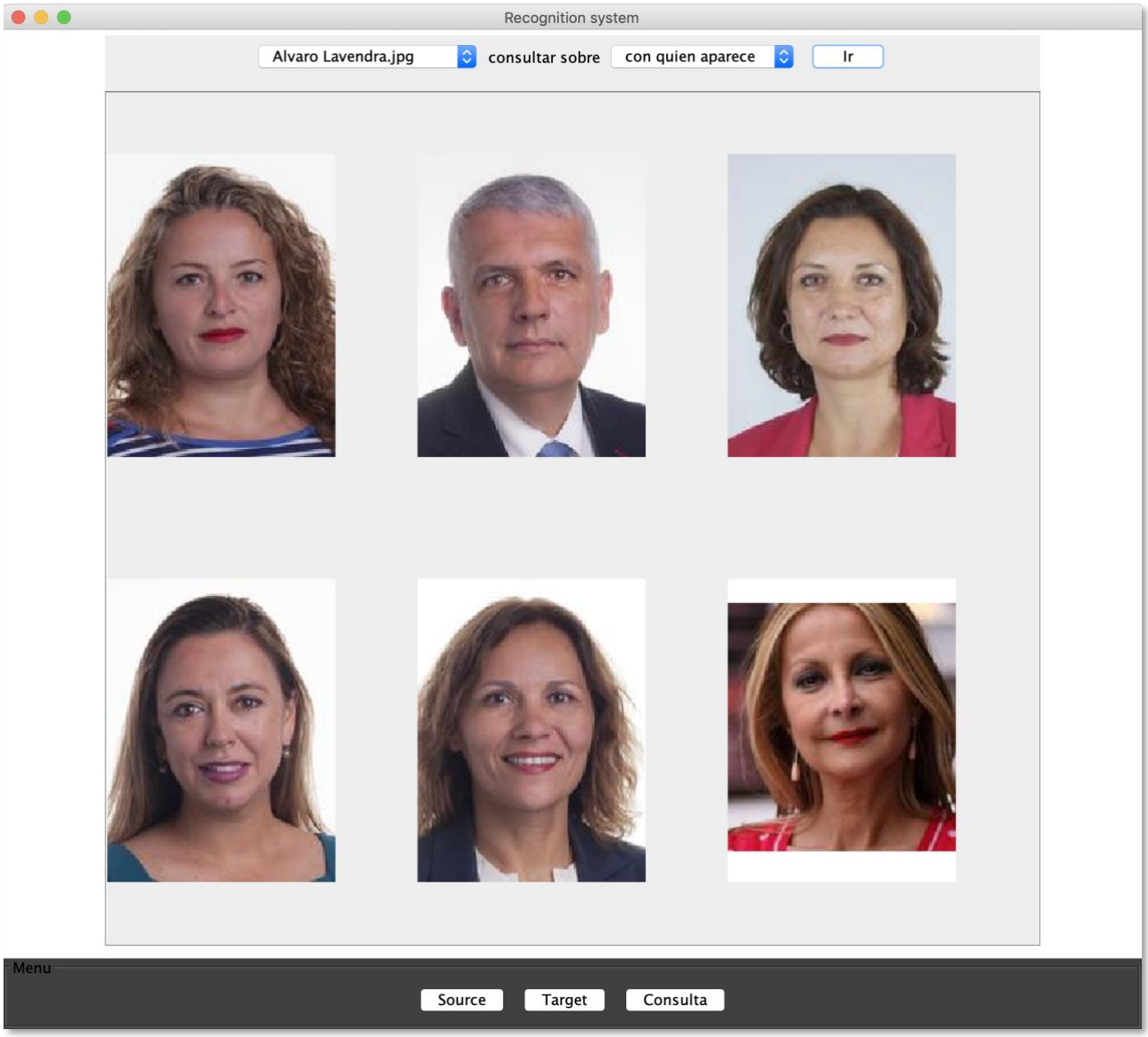


Ilustración 24: Información sobre con quien aparece [16]

Capítulo 4
Resultados

4.1 Estudio de los resultados

El sistema funciona de manera satisfactoria en la situación utilizada como caso de uso. Es capaz de reconocer a los miembros del *Parlamento de Canarias* almacenados en la base de datos, y de seguir suministrando rostros a reconocer como *Unknown*. Hemos podido observar lo siguiente en el comportamiento del programa:

- A mayores rostros en la base de datos, mayor es el tiempo de espera de resultados en el peor caso. Es lógico, ya que cuantos más rostros existan en la comparación secuencial, más iteraciones serán necesarias en el peor caso.
- El sistema de comparación es más lento en el análisis de la imagen con el fondo que realizando la comparación con los rostros *Unknown*. Esto también es lógico, ya que la imagen a analizar tiene mucho más detalle y mayor cantidad de elementos para tener en cuenta que una imagen *Unknown* de baja resolución debido al recorte sobre otra imagen y con un único rostro, siendo la imagen un marco facial.
- Es capaz de encontrar la gran mayoría de rostros proporcionados. Presenta dificultades a la hora de encontrar un rostro cuando este no está bien definido, ya sea por desenfoque o por falta de resolución. También hemos podido observar que tiene dificultad con imágenes donde las fotos están tomadas desde arriba.
- El manejo de excepciones ha sido crucial para el correcto funcionamiento del proyecto. Como el sistema cuenta con muchas peticiones a diferentes aplicaciones, ya sea *Amazon Rekognition* o las consultas a la base de datos, había que preparar el programa para cuando las consultas fallaran o no fuera todo como era necesario, para al menos continuar con la secuencia del programa.
- Es necesario que *digiKam* se encuentre abierto mientras el sistema está funcionando, ya que las tablas internas no se actualizan si *digiKam* está abierto. Hemos podido observar que *digiKam* presenta una pequeña latencia de actualización, que varía dependiendo de la cantidad de recursos que se usen en el sistema, por lo que ha sido necesario preparar al sistema frente a dicho retardo.
- Es una buena costumbre insertar en el sistema imágenes con un nombre

completo que ayude a identificar la imagen.

- El sistema de *Amazon Rekognition* para reconocimiento facial presenta un mejor funcionamiento que el de *digiKam*. Hemos podido comprobar que el marco facial de *Amazon Rekognition* es más preciso y delimita con mucho acierto el perfil de la persona a reconocer.
- *Amazon Rekognition* es capaz de identificar rostros donde no aparece la cara al completo, como por ejemplo rostros cubiertos por otro cuerpo donde tan solo se ve la boca, eso si, con un porcentaje de confianza menor.
- Es capaz de identificar rostros de hasta 14x21 pixeles. Este ha sido el rostro más pequeño que se ha podido identificar.

- Detecta rostros en carteles y otros lugares en 2 dimensiones, no sabemos según que criterio, ya que detecta algunos, pero otros no, como podemos comprobar en la siguiente imagen en la parte superior.

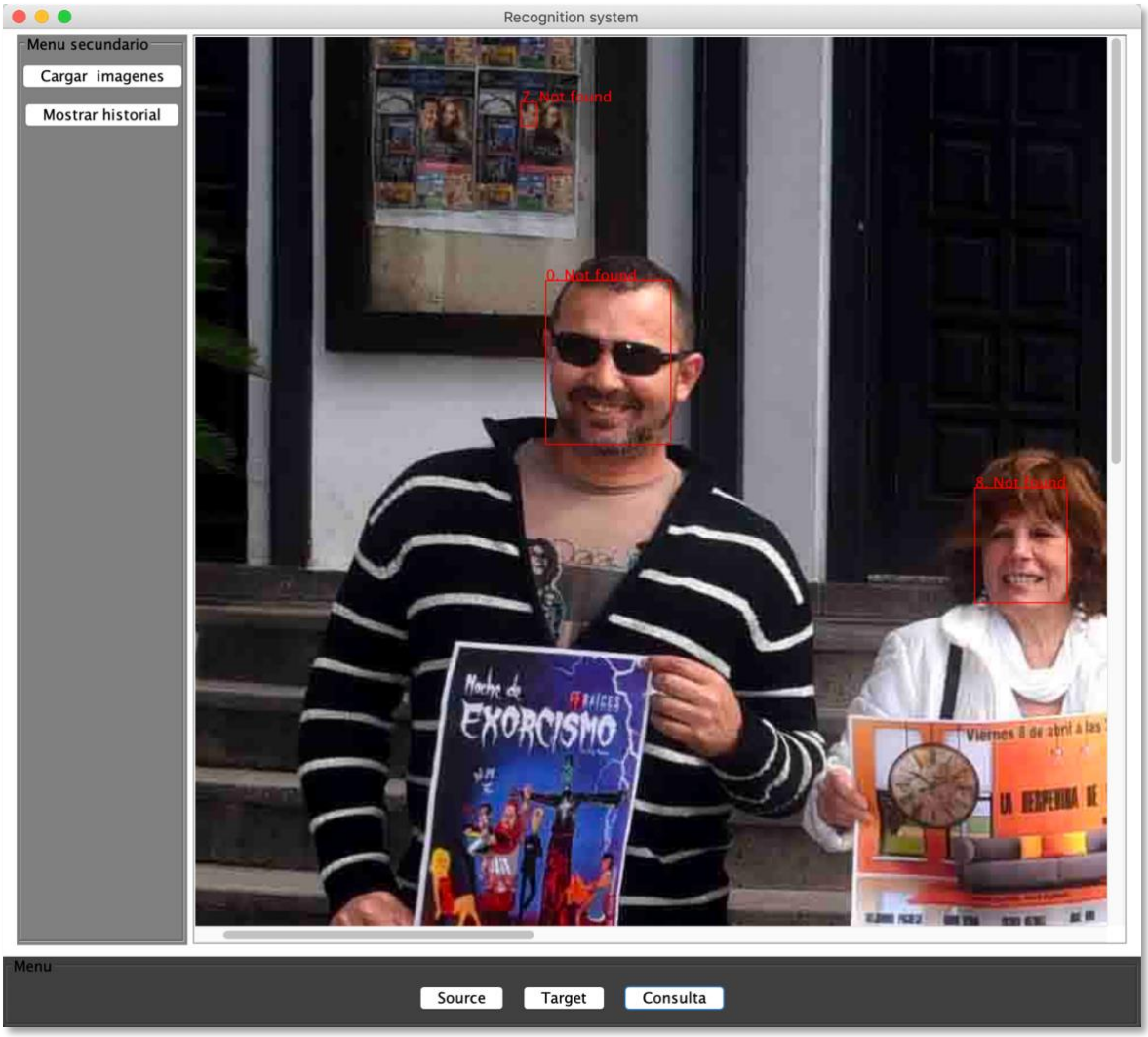


Ilustración 25: Reconocimiento de rostro en cartel [22]

Ha sido posible comprobar que el incremento tan grande que se provocaba en el sistema después de varias comprobaciones con un largo número de caras ralentizaba el funcionamiento. Es por ello, por lo que ha sido necesario añadir una nueva comprobación para evitar que se inserten perfiles duplicados en la sección de rostros no reconocidos, por lo que en cada análisis de producen al menos 2 búsquedas secuenciales de α (el conjunto de rostros conocidos) y β (el conjunto de rostros no conocidos). Hablamos entonces de que el algoritmo de reconocimiento es de complejidad $O(n^2)$.

4.2 Análisis de tiempos del sistema

Estos datos explican el comportamiento del programa en función del número de rostros en la imagen a analizar:

Tabla 6: Tiempos de análisis

Caras en la imagen	Tiempo de análisis 1er rostro	Tiempo de análisis 2º rostro	Tiempo de análisis 3er rostro	Tiempo de análisis 4º rostro
1	3019 (milisegundos)	-	-	-
2	2967	-	-	-
3	2871	1652	-	-
5	3241	1642	1579	-
6	2931	1185	1067	1061
7	4594	2903	3815	3239
8	2538	1050	1264	1089
9	4109	1358	1224	1163
10	5551	3560	3311	3526
11	4275	3262	2177	1822
14	4772	2973	2776	4114

Sabiendo que el tiempo medio de comparación de un marco facial de *Unknown* es 919 milisegundos, podemos interpretar esta tabla de resultados de la siguiente manera:

- Cuanto menor sea el número de rostros en la imagen más rápido se analiza.
- Cuanto menor sea el tamaño de la imagen a analizar más rápido se analiza.
- Los mayores tiempos de comparación se producen al principio de la petición, por lo que imaginamos que lo que más tiempo lleva es el envío de datos y el establecimiento de conexión.
- Cuanto mayor es la velocidad de conexión, más rápido lleva a cabo la comparación, algo que tiene sentido, ya que la computación de los datos se realiza de cara al servidor de Amazon *AWS*.

En este gráfico podemos apreciar claramente los resultados. Vemos que la línea azul, que corresponde al análisis del primer rostro *source*, es siempre la que más tarda en computarse, y en cambio, las demás van relativamente conjuntas. También podemos comprobar que a más rostros en la imagen (eje x), aumenta el tiempo de ejecución. Son interesantes también los picos en 7 rostros y 10 rostros ya que son las fotos con mayor resolución de las insertadas.

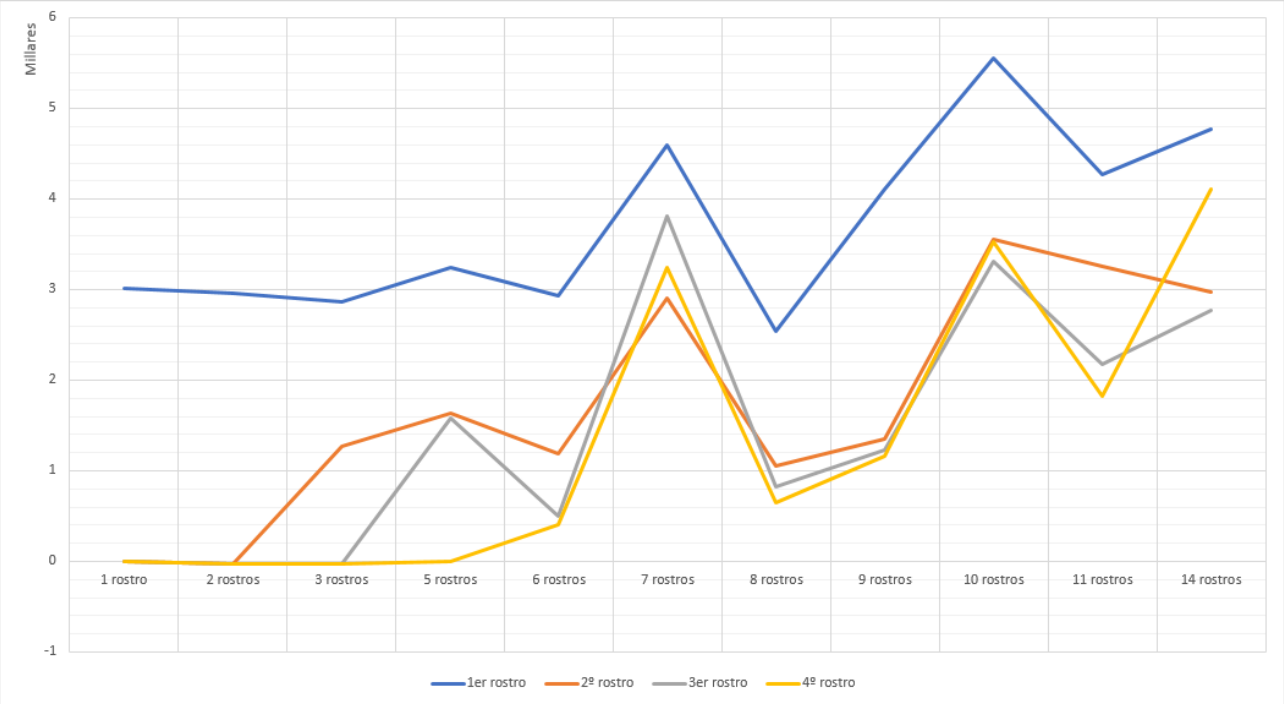


Ilustración 26: Gráfico de rendimiento

Capítulo 5
Conclusiones y líneas futuras

Ahora que ha sido estudiado el funcionamiento del sistema en su totalidad, se puede obtener una vista global de todos los aspectos que se han trabajado. En el caso de que fuera necesario empezar el proyecto desde cero, hay algunas cosas que serían enfocadas de otra manera:

digiKam

- No existe mucha documentación sobre el funcionamiento interno de la aplicación, y lo que resulta más necesario, en relación con las bases de datos internas. Se ha invertido tiempo buscando información que concretara la manera de plantear el proyecto, pero toda la información que hay acerca de este programa es escasa. El diseño de una documentación más precisa habría facilitado mucho las labores de programación y diseño del proyecto.
- Las tablas de *digiKam* están diseñadas de manera interesante para el trabajo de reconocimiento facial, pero han sido un problema a lo largo de todo el desarrollo del programa. Dado que presentan unos tiempos de actualización de la base de datos largos, ha sido necesario colocar paradas del sistema que esperen a que ésta se actualice. Además, muchos de los datos en la tabla no eran relativos al proyecto, por lo que habría sido más interesante diseñar la base de datos en función de nuestras necesidades, y no depender de una ya prediseñada. Esto aumentaría la velocidad del sistema.
- Por otra parte, es una aplicación de uso gratuito, por lo que es normal que exista alguna parte que no funcione del todo a la perfección.

Amazon Rekognition

- La respuesta de *Amazon Rekognition* ha resultado muy satisfactoria. Es una aplicación cuyo precio varía en función de la cantidad de llamadas a la API que se realicen, y en este caso el precio es prácticamente insignificante. El resultado es muy satisfactorio y el costo de instalación es relativamente bajo.
- Presenta una documentación completa y detallada en la gran parte de los aspectos del sistema.
- La cantidad de información que puedes obtener del reconocimiento es muy amplia, y que el procesado de datos se realice de cara al servidor acelera mucho el proceso.
- No se pueden comparar varias imágenes a la misma vez, devolviendo las diferentes imágenes encontradas. Esto habría facilitado mucho el sistema de

reconocimiento.

- Sería interesante cambiar la manera en la que se realiza la búsqueda de un rostro *Unknown* que haga matching con alguno de la base de datos. Se crearía una matriz de rostros *Unknown* de la base de datos y el sistema realizaría la comparación del interesa hacer matching con la matriz de rostros, realizando la comparación de 1 a N. Esto aumentaría la velocidad con la que se ejecuta el proceso de análisis total, ya que es una de las etapas de este.
- El funcionamiento de *Amazon Rekognition* bajo el debugger es muy lento a la hora de realizar la comparación, por lo que estudiar la secuencia del programa paso a paso, ha sido tedioso. El debugger es una herramienta esencial en un proyecto de esta magnitud, ya que conocer paso a paso las acciones que realiza el programa y cuanto vale cada variable en ese punto, es algo que ayuda a evitar muchos errores.
- Los mejores valores para el threshold han sido 0.8 para el análisis de imágenes nuevas, 0.6 para la comparación de imagen *source* con imagen *Unknown* y 0.6 también para la comparación de *Unknown* con *Unknown.* Se ha discutido variar el valor del umbral en función de la resolución de la imagen (bajar el valor cuando la imagen tuviera una resolución más baja y subir el valor cuando fuera más alta), pero esto podría dar lugar a asociaciones erróneas dentro del sistema. Es preferible que el sistema deje sin reconocer algún rostro, que hacerlo de manera errónea.

Cliente web

- Si fuera necesario rediseñar el sistema y prescindir del uso de *digiKam* en local, lo más interesante habría sido un cliente web desarrollado en *C#* utilizando *MVC* en *Visual Studio*. Es un entorno familiar debido a las prácticas de empresa, y he podido comprobar que funciona de una manera interesante. Mover el cliente a la web supondría poder conectarse desde cualquier dispositivo, algo que no se puede lograr con el cliente de Java. Además, el diseño de la parte visual del programa, que en este caso se ha hecho con swing, se podría realizar utilizando *HTML* de una manera sencilla y lograr una mejor respuesta visual más completa y intuitiva. Otro dato interesante es la manera que tiene Visual Studio de trabajar con las bases de datos. En nuestro caso, ha sido necesario trabajar con la base de datos mediante consultas en *SQL*. En cambio, en *C#*, es posible acceder a los atributos de las tablas como si estos fueran atributos de un objeto, evitando así la comunicación en *SQL*.

Visión del proyecto

- Una vez que se ha estudiado el funcionamiento de la herramienta de software libre *digiKam*, llegamos a la conclusión de que sería más

interesante diseñar una herramienta que asemeje el funcionamiento de *digiKam* desde cero adecuada a nuestro sistema, que adaptar la herramienta ya existente y tener que depender de sus limitaciones. Debido a la falta de documentación sobre dicho software, muy posiblemente se habría tardado menos diseñando una herramienta nueva que adaptando la ya existente.

- He aprendido mucho en el desarrollo de este proyecto. Cuando comencé con el proyecto, era aparentemente algo muy denso con muchas capas a desarrollar.
- Se han trabajado muchos aspectos de la informática en el diseño del sistema, desde el planteamiento previo al trabajo, pasando por el aprendizaje autónomo y la investigación de las distintas herramientas que lo forman en cuanto a su funcionamiento y su utilidad, hasta el diseño gráfico de la interfaz para lograr un funcionamiento intuitivo y sencillo de la aplicación.
- Se han aplicado muchos conocimientos y se han aprendido otros muchos, pero el aspecto más importante y a destacar de este proyecto, es el trabajo de, juntando todos los conocimientos que se han ganado en la carrera, buscar el camino en un proyecto de tal magnitud, confiando en mi criterio y en los pasos que se iban dando, ir pasando cada una de las diferentes etapas con éxito e ir uniéndolas con la siguiente hasta finalmente tener un proyecto consolidado y estable, que cumple con los requisitos solicitados.
- En muchos aspectos, el desarrollo del TFG en sí se asemeja a lo que sería una experiencia de trabajo: un proyecto asignado que se tiene que completar en base al criterio de un profesional y una serie de requisitos, mediante los conocimientos actuales y los que se irán aprendiendo buscando aquello que no se conoce.

Capítulo 6
Summary and Conclusions

Now that the system has been studied, a global view of all the aspects that have been worked on can be obtained. If it was necessary to start the project from scratch, there are some things that would be done in another way:

digiKam

- There is not much documentation about the internal operation of the application and related to internal databases. Time has been spent looking for information that will specify the way to present the project, but all the information about this program is scarce. The design of a more precise documentation would have facilitated much the programming and designing work of the project.
- *digiKam* tables are designed in an interesting way for facial recognition, but they have been a problem in every part of the development of the program. Given that they have long update times for the database, it has been necessary to place system stops waiting for it to be updated. In addition, many of the table data were not relative to the project, so it would have been more interesting to design the database based on our needs, and not dependending on a predesigned one. This would increase the speed of the system.
- On the other hand, it is a free application, so it is normal that there is a part that does not work perfectly.

Amazon Rekognition

- The response of *Amazon Rekognition* has been very satisfactory. It is an application whose price varies depending on the number API calls made, and in this case the price is insignificant. The result is very satisfactory, and the installation is not time consuming.
- It presents a complete and detailed documentation in most aspects of the system.
- The amount of information that you can obtain from the recognition is very large, and that the data processing is done server-side, so accelerates the process a lot.
- You can not compare several images at the same time, returning the different images found. This would have greatly facilitated the recognition system.
- It would be interesting to change the way in which the search of a *Unknown*

face is made for matching any of the database. It would create a matrix of *Unknown* faces of the database and the system would make the comparison to make matching with the face matrix, making the comparison 1 to N. This would increase the speed of the total analysis process.

- *Amazon Rekognition*'s performance under the debugger is very slow when making the comparison, so studying the sequence of the program step by step has been tedious. Debugging is an essential tool in a project of this magnitude.
- The best values for the threshold were 0.8 for new image analysis, 0.6 for the comparison between source image with Unknown image and 0.6 also for the comparison between Unknown with Unknown. It has been discussed to vary the value of the threshold depending on the resolution of the image (lower the value when the image had a lower resolution and upper value when it was higher), but this could lead to erroneous associations within the system. It is better to leave a face unrecognized, than doing it in a wrong way.

Web client

- If it were necessary to redesign the system and dispense with the use of *digiKam* locally, the most interesting thing would have been to design a web client developed in C # using MVC in Visual Studio. It is a family environment due to business practices, and I have seen the way it works. Moving the client to the web would be able to connect from any device, something that can not be achieved with the Java client. In addition, the design of the visual part of the program, which in this case has been done by Swing, could be done using HTML in a simple way and achieving a better visual response completer and more intuitive. Another interesting fact is the way Visual Studio works with databases. In our case, it has been necessary to work with the database through SQL queries. In contrast, in C #, it is possible to access the attributes of the tables as if they were attributes of an object, thus avoiding communication in SQL.

Project view

- Once the operation of the open-source tool *digiKam* has been studied, we conclude that it would be more interesting to design a tool that resembles the *digiKam* function from scratch suitable to our system, than adapting the existing tool and having to depend on its limitations. Due to the lack of documentation on such software, it would very likely have taken less time to design a new tool than adapting the existing one.
- I have learned a lot in the development of this project. When I started with the project, it was something very dense with many layers to develop.
- Many aspects of computer science have been worked on in the design of the system, from the study approach, through autonomous learning and research

into the different tools that comprise it in terms of its operation and its usefulness, to graphic design of the interface to achieve an intuitive and simple operation of the application.

- A lot of knowledge has been applied and many others have been learned, but the most important aspect to highlight in this project is the work of, gathering all the knowledge that has been gained in the career, looking for the way in a project of such magnitude , trusting in my criteria and in the steps that were being taken, going through each of the different stages successfully and joining them until finally having a consolidated and stable project, which meets the requirements requested.
- In many aspects, the development of the Final Degree Project itself resembles what would be a work experience: an assigned project that has to be completed based on the criteria of a professional and a series of requirements, through the current knowledge and those that will be learnd by looking for what is not known.

Capítulo 7
Presupuesto

7.1 Coste del proyecto

Utilizando como referencia el saldo de un ingeniero de 10€ la hora, estos serían los presupuestos del proyecto.

Tarea	Horas empleadas	Coste
Estudio *Amazon Rekognition*	30	300
Estudio *digiKam*	20	200
Diseño infraestructura interna	90	900
Diseño de la aplicación	60	600
Total:	**200**	**2000**

Tabla 7**:** Coste del proyecto

Capítulo 8
Algoritmos

8.1 Algoritmo Rekognition

```
/*****************************************************************************
*
* RekognitionSystem.java
*
*****************************************************************************
*
* AUTORES
*   Joaquin Sanchiz Navarro
*
* FECHA
*   7 de mar. de 19
*
* DESCRIPCION
*   Fichero encargado de ejecutar la comparación facial en los diferentes apartados del proyecto
*
*****************************************************************************/
package TFG;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.CompareFacesMatch;
import com.amazonaws.services.rekognition.model.CompareFacesRequest;
import com.amazonaws.services.rekognition.model.CompareFacesResult;
import com.amazonaws.services.rekognition.model.ComparedFace;

public class RecognitionSystem {

	private ArrayList<ImageData> carasEncontradas;
	private ArrayList<ImageData> carasNoEncontradas;

	public ArrayList<ImageData> getCarasEncontradas() {
		return carasEncontradas;
	}
```

```
	public void setCarasEncontradas(ArrayList<ImageData> carasEncontradas) {
		this.carasEncontradas = carasEncontradas;
	}

	public ArrayList<ImageData> getCarasNoEncontradas() {
		return carasNoEncontradas;
	}

	public void setCarasNoEncontradas(ArrayList<ImageData> carasNoEncontradas) {
		this.carasNoEncontradas = carasNoEncontradas;
	}

	public RecognitionSystem()
	{

	}

	public void analizarSource(DBContext context, User admin, ArrayList<ImageData> sources,
ImageData target, Float similarityThreshold) throws SQLException
	{
		this.carasEncontradas = new ArrayList<ImageData>();
		this.carasNoEncontradas = new ArrayList<ImageData>();
		String url = context.getSourceRoute() + target.getName();

		Image targetImage = new Image().withBytes(ImageData.getImageBuffer(url));

		float t = System.currentTimeMillis();
		for(int i = 0; i < sources.size(); i++)
		{
			System.out.println("COMPARANDO con " + sources.get(i).getName());
			String sourcerl = null;

			System.out.println("Es analisis source");
			sourcerl = DBContext.unknownRoute + sources.get(i).getName();

			Image sourceImage = new Image().withBytes(ImageData.getImageBuffer(sourcerl));
			CompareFacesRequest request = new
CompareFacesRequest().withSourceImage(sourceImage).withTargetImage(targetImage).withSimilarityThres
hold(similarityThreshold);
			CompareFacesResult result = null;
			try
			{
```

```
                        result = admin.getRekognitionClient().compareFaces(request);
                    }
                    catch(Exception e)
                    {
                        System.err.println("Imposible comparar rostros.");
                    }

                    if(result != null)
                    {
                        List<CompareFacesMatch> matches = result.getFaceMatches();

                        if(matches.size() == 1)
                        {
                            System.out.println("Encontrado " + sources.get(i).getName());

                        ImageData dummy = new ImageData(target);
                            dummy.setName(sources.get(i).getName());
                        this.getCarasEncontradas().add(dummy);
                            break;

                        }

                    }

                }

                float t2 = System.currentTimeMillis();
                System.err.println("RENDIMIENTO DE RECONOCIMIENTO UNKNOWN PARA " + sources.size()
+ ": " + (t2-t) + " MILISEGUNDOS." );

        }

        public void analizarTarget(DBContext context, User admin, ArrayList<ImageData> sources,
ImageData target, Float similarityThreshold) throws SQLException
        {
            this.carasEncontradas = new ArrayList<ImageData>();
            this.carasNoEncontradas = new ArrayList<ImageData>();
            String url = context.getTargetRoute() + target.getName();

            Image targetImage = new Image().withBytes(ImageData.getImageBuffer(url));

            for(int i = 0; i < sources.size(); i++)
```

```
		{
			System.out.println("COMPARANDO con " + sources.get(i).getName());
			String sourcerl = null;

			System.out.println("Es analisis target");
			sourcerl = context.getSourceRoute() + sources.get(i).getName();

			Image sourceImage = new Image().withBytes(ImageData.getImageBuffer(sourcerl));
			CompareFacesRequest request = new
CompareFacesRequest().withSourceImage(sourceImage).withTargetImage(targetImage).withSimilarityThres
hold(similarityThreshold);
			CompareFacesResult result = null;
			long t = System.currentTimeMillis();
			try
			{
				result = admin.getRekognitionClient().compareFaces(request);
			}
			catch(Exception e)
			{
				System.err.println("Imposible comparar rostros.");
			}
			long t2 = System.currentTimeMillis();

			System.err.println("RENDIMIENTO DE " + (t2 - t) + " MILISEGUNDOS");

			if(result != null)
			{
				List<CompareFacesMatch> matches = result.getFaceMatches();
				List<ComparedFace> unmatches = result.getUnmatchedFaces();

				if(matches.size() == 1)
				{
					System.out.println("Encontrado " + sources.get(i).getName());

					target.setFaceMark(targetImage,
matches.get(0).getFace().getBoundingBox(), result.getTargetImageOrientationCorrection());

					ImageData dummy = new ImageData(target);
					this.getCarasEncontradas().add(dummy);

					if(this.getCarasNoEncontradas().contains(dummy))
					{
						this.getCarasNoEncontradas().remove(dummy);
```

```
                }

                context.asignarIdentificacionEnSource(sources.get(i), dummy);

                        dummy.setName(sources.get(i).getName());

                }


                for(int j = 0; j < unmatches.size(); j++)
                {
                        target.setFaceMark(targetImage,
unmatches.get(j).getBoundingBox(), result.getTargetImageOrientationCorrection());
                        ImageData dummy = new ImageData(target);

                        if( (!this.getCarasNoEncontradas().contains(dummy)) &&
(!this.getCarasEncontradas().contains(dummy)))
                        {
                                this.getCarasNoEncontradas().add(dummy);
                        }
                }


                //Cuando se hayan reconocido todas las caras, salir del programa
                if(this.getCarasNoEncontradas().size() == 0)
                {
                        i = sources.size();
                }
            }

        }
    }

    public void analizarUnknown(DBContext context, User admin, ArrayList<ImageData> sources,
ImageData target, Float similarityThreshold) throws SQLException
    {
        this.carasEncontradas = new ArrayList<ImageData>();
        this.carasNoEncontradas = new ArrayList<ImageData>();
        String url = context.validarRoute + target.getName();

        Image targetImage = new Image().withBytes(ImageData.getImageBuffer(url));

        for(int i = 0; i < sources.size(); i++)
```

```
		{
			long t = System.currentTimeMillis();
			System.out.println("COMPARANDO con " + sources.get(i).getName());
			String sourcerl = null;

			System.out.println("Es analisis unknown");
			sourcerl = context.unknownRoute + sources.get(i).getName();

			Image sourceImage = new Image().withBytes(ImageData.getImageBuffer(sourcerl));
			CompareFacesRequest request = new
CompareFacesRequest().withSourceImage(sourceImage).withTargetImage(targetImage).withSimilarityThres
hold(similarityThreshold);
			CompareFacesResult result = null;
			try
			{
				result = admin.getRekognitionClient().compareFaces(request);
			}
			catch(Exception e)
			{
				System.err.println("Imposible comparar rostros.");
			}

			if(result != null)
			{
				List<CompareFacesMatch> matches = result.getFaceMatches();

				if(matches.size() == 1)
				{
					System.out.println("Encontrado " + sources.get(i).getName());


				ImageData dummy = new ImageData(target);
					dummy.setName(sources.get(i).getName());
				this.getCarasEncontradas().add(dummy);
					break;
				}
			}
			long t2 = System.currentTimeMillis();
			System.err.println("RENDIMIENTO UNKNOWN DE " + (t2-t) + " MILISEGUNDOS");
		}
	}

}
```

Capítulo 9
Recursos

9.1 Código del proyecto

Se puede encontrar el código del proyecto en mi perfil de GitHub a continuación: https://github.com/joaquinsanchiz/TFG

Bibliografía

1. *Amazon Rekognition*: https://docs.aws.amazon.com/es_es/rekognition/latest/dg/what-is.html
2. SQLite: https://es.wikipedia.org/wiki/SQLite
3. Draw.io: https://chrome.google.com/webstore/detail/drawio-diagrams/onlkggianjhjenigcpigpjehhpplldkc?hl=es-419
4. Sistema de reconocimiento facial: https://es.wikipedia.org/wiki/Sistema_de_reconocimiento_facial
5. El reconocimiento facial de Facebook: https://www.abc.es/tecnologia/redes/20140321/abci-facebook-reconocimiento-facial-preciso-201403211737.html
6. Java: https://es.wikipedia.org/wiki/Java_(lenguaje_de_programación)
7. Eclipse: https://es.wikipedia.org/wiki/Eclipse_(software)
8. Documentación AWS: https://docs.aws.amazon.com/es_es/rekognition/latest/dg/rekognition-dg.pdf
9. Tecnología china para el reconocimiento facial y de voz para cerdos: https://www.20minutos.es/noticia/3572906/0/tecnologia-china-reconocimiento-facial-voz-cerdos/
10. Reconocimiento facial para entrar a ferias y festivales: https://www.bbva.com/es/reconocimiento-facial-para-entrar-a-ferias-y-festivales/
11. Computación en la nube: https://es.wikipedia.org/wiki/Computación_en_la_nube
12. Face Hallucination: https://es.wikipedia.org/wiki/Face_hallucination
13. Ilustración Navarro de Paz y Soria: http://www.pp.es/buscador?c=7434&content=imagenes
14. Documentación de digiKam: https://www.digikam.org/documentation/
15. Ilustración Cabrera Noda: https://www.youtube.com/watch?v=COiB6R1ZKlI
16. Diputados Parlamento de Canarias: https://www.parcan.es/composicion/con_fotos.py?LEGIS=9
17. Ilustración Hernández Miranda: https://twitter.com/ppdecanarias/status/840175883793096705
18. Ilustración Navarro de Paz: https://www.laprovincia.es/tags/maria-australia-navarro-de-paz.html
19. Ilustración Navarro de Paz: http://www.eldentistamoderno.com/2015/11/el-colegio-de-dentistas-de-

las-palmas-lidera-el-primer-paso-hacia-la-regulacion-de-la-publicidad-sanitaria-en-canarias/junta-gobierno-coelp/
20. Ilustración Navarro de Paz: http://eldia.es/canarias/2016-12-20/40-Antona-espera-epoca-navidena-lleve-amor-paz-seno-Gobierno.htm
21. Ilustración Navarro de Paz: https://www.youtube.com/watch?v=EyM1tfbxG-0
22. Ilustración teatro San Bartolomé: http://www.masscultura.com/mass/7-obras-teatrales-apuesta-municipal-de-la-ix-semana-cultural-de-teatro-de-san-bartolome/