UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

DETECCIÓN DE PATRONES ESCRITURALES EN IMÁGENES DE TEXTO MANUSCRITO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL E INGENIERO CIVIL EN COMPUTACIÓN

MAURICIO JOSÉ GIADACH PINOCHET

PROFESOR GUÍA:
SEBASTIÁN RÍOS PÉREZ

MIEMBROS DE LA COMISIÓN:
NELSON BALOIAN TATARYAN
BENJAMÍN BUSTOS CARDENAS

SANTIAGO DE CHILE
2014

Resumen

El análisis neuroescritural (o grafológico) corresponde al proceso de estudio mediante el cual un perito en la materia detecta una serie de características de un texto manuscrito y las interpreta para inferir características de la personalidad del autor del mismo.

El presente trabajo busca automatizar el proceso de un análisis neuroescritural para disminuir el tiempo que se requiere para llevarlo a cabo. Más concretamente, el objetivo general de este proyecto es: detectar automáticamente patrones grafológicos a partir de imágenes de texto manuscrito, mediante un sistema computacional usando técnicas de procesamiento de imágenes.

Para abordar este desafío, se propone una primera etapa de estudio de la situación actual y de análisis de los requisitos que debe satisfacer el sistema. Posteriormente, se realiza un diseño del mismo basado en tres componentes principales: un sistema de *backend* para realizar el procesamiento de imágenes; una plataforma web de interacción con el *backend*, para ejecutar el análisis e incluir información adicional; y otro sistema web para proveer acceso a clientes que soliciten la realización de un análisis.

A continuación, se construye un conjunto de algoritmos capaces de detectar características grafológicas y se implementan de acuerdo al diseño propuesto utilizando la biblioteca OpenCV en el *backend*. Mientras que para el *frontend* se utiliza un *framework* de desarrollo web siguiendo el paradigma MVC. Como resultado de esta fase, se logra construir un sistema funcional en sus tres componentes y se procede a evaluarlo de acuerdo a la calidad de las detecciones realizadas.

Con muestras analizadas manualmente por un experto, se evalúa la calidad de la detección automática de 15 patrones distintos, obteniéndose, para cada algoritmo, clasificaciones acertadas en al menos un 86 % de los casos analizados. Sin embargo, debido a sesgos detectados en los datos de las muestras analizadas, se requerirá obtener más datos para asegurar la efectividad de todos los algoritmos propuestos en cualquier escenario.

Por último, se puede concluir que el sistema genera una precisa clasificación agregada a nivel de competencias, coincidiendo totalmente con el análisis realizado por el experto neuroescritural en 14 de los 15 casos de evaluación.

A María Fresia, mi abuelita, que sé que me acompaña siempre.

Agradecimientos

La publicación de este trabajo constituye mucho más que el término de mis estudios universitarios, pues simboliza el cierre de una larga etapa de mi vida. Por este motivo, quiero agradecer a algunas de las personas que me han ayudado y acompañado a lo largo de este camino.

Quiero comenzar agradeciendo a mis padres, Fresia y Juan Ricardo, que han sido una fuente inagotable de apoyo, cariño, consejos e inspiración. Sin lugar a dudas, gran parte del mérito de haber llegado hasta este punto se lo debo a ellos, que siempre han buscado, con mucha entrega y dedicación, la mejor manera de potenciar mi desarrollo académico, valórico y personal. Quiero también agradecer a mi hermana Verónica, ya que su compañía, ideas y ayuda en todo tipo de labores, ha sido fundamental para mi crecimiento en la vida.

De forma especial, agradezco a Paula, mi querida polola, por haber estado a mi lado durante esta etapa universitaria, haciendo que sea una de las mejores de mi vida. Su amor incondicional, sus consejos y su infinita motivación han sido esenciales para mí en el andar de este camino. Gracias, Pau, por iluminar mi vida día a día.

No podría no mencionar a mis amigos Eduardo y Marco, a quienes agradezco por esta larga amistad que nos ha llevado a seguir caminos similares desde nuestra infancia hasta el día de hoy. Agradezco a las grandes personas que conocí en plan común y que me acompañaron en esta etapa: Antonio, Bayron, Nicolás, Ignacio, Camilo, Javier, Rodrigo y Sebastían. Así como también a quienes fueron parte de mi vida desde la especialidad: Andrea, Roberto, Alexander, Juan Luis, Gustavo y Víctor.

Destaco, además, a los profesores de mi comisión. A Sebastián Ríos, por haberme desafiado con este trabajo y haber contribuido a cerrar de manera exitosa mis estudios de ingeniería. A Nelson Baloian, por haber confiado en mí y haberme dado la posibilidad de trabajar con él desde mi tercer año en la escuela, cuando recién me conocía. A Benjamín Bustos, que con su conocimiento y comentarios hicieron posible mejorar y pulir este trabajo.

Por último, agradezco a todas las personas que de una u otra manera hicieron posible mis estudios y el desarrollo de este trabajo. Sepan que la felicidad que siento por estar cerrando esta etapa se la debo a todos ustedes. Muchas gracias.

Tabla de contenido

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

La escritura a mano es una habilidad que consiste en realizar marcas gráficas sobre una superficie con el objetivo de comunicar algo, utilizando convenciones propias del lenguaje de la persona que la ejecuta (Coulmas [4]). Por otro lado, esta ancestral tradición de escribir ha sido llevada a representaciones estándares, tanto desde el punto de vista gráfico (mediante la definición de alfabetos), como de su representación lógica computacional gracias al desarrollo de estándares internacionales como el Unicode Standard (Consortium [3]).

Frente a esta dualidad de la escritura en el mundo físico y en el mundo digital, numerosos esfuerzos han sido desarrollados con el fin de llevar el texto escrito (ya sea de origen manuscrito o impreso) a texto digital debido a las ventajas que éste posee, tales como su facilidad de conservación y simpleza de búsqueda. Este procedimiento es conocido como digitalización de texto.

Dentro del proceso de digitalización de texto manuscrito, se puede reconocer dos tipos de procesamiento: *online* y *offline* (Plamondon and Srihari [15]). El primero consiste en capturar, en tiempo real, la escritura del texto por medio de algún dispositivo electrónico, con lo cual se obtiene un conjunto de coordenadas de puntos en función del tiempo. Adicionalmente, dispositivos avanzados como tabletas digitalizadoras son capaces de medir otras características como presión e inclinación del lápiz. Por su parte, el procesamiento *offline* toma como entrada una imagen del texto escrito por una persona, por lo que no existe una forma directa ni exacta de recuperar información del proceso de escritura en sí.

Este trabajo se enmarca en el contexto del procesamiento *offline* de texto manuscrito. En particular, se propone desarrollar un sistema de detección automático de patrones gráficos en imágenes de múltiples textos, con el objetivo de entregar métricas que los identifiquen.

Dentro de una metodología convencional de procesamiento *offline* de texto manuscrito, el reconocimiento de patrones corresponde a una etapa previa a la identificación de caracteres, pues permite extraer características básicas de la imagen que pueden servir como información adicional durante el proceso de reconocimiento.

A modo de ejemplo, algunos patrones susceptibles de ser detectados son: espaciamientos

entre palabras, ángulos de inclinación de las líneas del texto, orientación de las letras dentro de una palabra y tamaño de separaciones entre caracteres al interior de una palabra.

Actualmente, la detección de éstos y otros patrones, así como el procesamiento posterior, constituyen un problema abierto de investigación, en donde numerosos estudios y desarrollos están siendo llevados a cabo con el objetivo de mejorar la efectividad y eficiencia de los algoritmos existentes. A modo de referencia, existen importantes conferencias internacionales que buscan abordar y reunir los avances logrados en torno a estos problemas, tales como la *International Conference On Document Analysis and Recognition (ICDAR)* y la *International Conference On Frontiers in Handwriting Recognition (ICFHR).*

Dentro de las aplicaciones que posee la detección de patrones escriturales se encuentran algunas ya mencionadas, como la optimización de la digitalización de textos, y otras menos convencionales, tales como

- la extracción de características propias de autores para la detección de falsificaciones de documentos y descubrimiento de anónimos en peritajes caligráficos (Santana et al. [17], Srihari and Shi [20]),
- las aplicaciones grafológicas en procesos de selección de personal, consistentes en relacionar características propias de la escritura con rasgos de la personalidad del autor (Pari [14]), y
- las aplicaciones en el ámbito de salud, donde investigaciones recientes sugieren que existe una relación entre algunos patrones de la escritura y ciertas enfermedades como el cáncer o el autismo (Colado [2], Langmaid et al. [12]).

1.1. Descripción del Proyecto y Justificación

A modo general, el presente proyecto busca desarrollar e implementar técnicas que permitan identificar patrones mediante métodos computacionales de procesamiento de imágenes, recibiendo como entrada una imagen de un texto manuscrito y entregando como salida un conjunto de métricas asociadas a los patrones y características detectadas.

Tal como se ha indicado en la sección anterior, una de las aplicaciones que tiene la detección de estos patrones es la relacionada con la grafología, proceso mediante el cual un experto en el tema -conocido como perito neuroescritural- estudia visualmente un texto escrito por una persona, identifica sus patrones característicos, mide ciertos atributos y finalmente saca conclusiones a partir de sus mediciones. Este procedimiento es lento y costoso de llevar a cabo debido a su naturaleza manual y al total grado de intervención humana que requiere.

Teniendo esto en consideración, se busca desarrollar un sistema que apoye la toma de decisiones de un perito neuroescritural, mediante la provisión de las mediciones que busca de manera automática gracias al procesamiento computacional. Estas decisiones corresponden, básicamente, a determinar si la persona que escribió el texto posee (o no) características de personalidad definidas. Sin embargo, para llegar a desarrollar un sistema de tal naturaleza, es necesario pasar previamente por una fase de investigación que evalúe la factibilidad de

detectar los patrones de escritura relevantes y la precisión de hacerlo computacionalmente.

Por otro lado, este trabajo forma parte de un proyecto de investigación multidisciplinario del profesor Sebastián A. Ríos que busca explorar científicamente relaciones entre patrones de escritura y características humanas. Por tanto, es fundamental contar con una herramienta que permita detectar estos atributos para, luego, poder establecer relaciones con las características de la persona en análisis.

De esta forma, la validación de este proyecto será empírica, contrastando la calidad de la detección computacional de los patrones con la detección realizada por un perito neuroescritural. Con este fin en consideración, se cuenta con una colección de imágenes de texto manuscrito ya analizadas por expertos de una empresa consultora en recursos humanos, lo que constituye una base para construir un conjunto de entrenamiento y de evaluación para el desarrollo de este trabajo.

No obstante lo anterior, el alcance de este proyecto no estará limitado a la aplicación ya mencionada. Adicionalmente, se busca que los algoritmos y técnicas a desarrollar puedan constituir un aporte al conocimiento dentro de la línea de investigación del procesamiento de texto manuscrito. A modo de ejemplo, uno de los problemas abiertos de esta área corresponde a la segmentación de imágenes de documentos manuscritos en líneas de texto y palabras (Gatos et al. [7]), lo cual corresponde a una fase previa a cualquier tipo de procesamiento posterior.

1.2. Objetivos

1.2.1. Objetivo General

Detectar automáticamente patrones grafológicos a partir de imágenes de textos manuscritos, mediante un sistema computacional usando técnicas de procesamiento de imágenes.

1.2.2. Objetivos Específicos

Establecer un conjunto de patrones grafológicos independiente de la factibilidad de detección. Estudiar cuáles de éstos son automatizables y generar una hoja de ruta a partir de esto.

Desarrollar y utilizar algoritmos computacionales para detectar de manera automática el subconjunto de patrones seleccionados. Evaluar cada uno de los algoritmos desarrollados mediante imágenes reales y conocimiento experto.

Diseñar e implementar una aplicación web que permita mostrar de manera sencilla los resultados generados en una o más muestras de texto manuscrito.

1.3. Resultados Esperados

Mediante el desarrollo de este trabajo, se espera lograr una serie de resultados intermedios que permitan llevar al cumplimiento del objetivo final. De esta forma, los productos concretos que se busca obtener son los que se indican a continuación:

1. Especificación de requisitos que detalle el conjunto de patrones escriturales que deben ser reconocidos, incluyendo una especificación de casos de uso, que será la base de la biblioteca que se construirá.
2. Documentación respecto de las técnicas existentes para detectar cada uno de los patrones de interés, especificando aquellos para los cuales no existen técnicas actualmente.
3. Especificación de algoritmos para la detección de patrones.
4. Código fuente correspondiente a la implementación de los algoritmos desarrollados y especificados.
5. Biblioteca o framework de acceso e integración a los distintos algoritmos desarrollados, proveyendo rutinas comunes para la interacción con el usuario.
6. Aplicación web que interactúe con la biblioteca desarrollada y que permita la carga de una imagen (o un conjunto de ellas), la selección de patrones a identificar y entregue como resultado una lista con las mediciones obtenidas del análisis de los patrones.

Naturalmente, se espera que la aplicación completa, y en particular los algoritmos de detección y clasificación, logren un grado de precisión *aceptable*. Sin embargo, debido a la naturaleza de investigación de este trabajo y a que no existen referencias de sistemas similares que hayan sido desarrollados, es necesario definir previamente ciertos rangos de aceptabilidad de la calidad de ellos.

Por este motivo, junto a los expertos de la empresa consultora que apoya este trabajo, se han definido los umbrales presentados en la Tabla 1.1, que servirán para realizar la evaluación final de este sistema.

Métrica de evaluación[1]	Rango	Calidad
Accuracy	[100 % - 90 %]	Muy bueno
Accuracy	]90 % - 80 %]	Bueno
Accuracy	]80 % - 65 %]	Regular
Accuracy	]65 % - 0 %]	Malo
Recall	[100 % - 90 %]	Muy bueno
Recall	]90 % - 80 %]	Bueno
Recall	]80 % - 65 %]	Regular
Recall	]65 % - 0 %]	Malo
False Positive Rate	[0 % - 10 %]	Muy bueno
False Positive Rate	]10 % - 20 %]	Bueno
False Positive Rate	]20 % - 35 %]	Regular
False Positive Rate	]35 % - 0 %]	Malo

Tabla 1.1: Criterios de evaluación para los algoritmos a desarrollar.

[1]La definición formal de estas métricas de evaluación puede consultarse en la Sección6.1.

1.4. Metodología

Para el desarrollo de este trabajo se utiliza una metodología basada en el modelo de Desarrollo Incremental de Software (Sommerville [18]).

Las tareas generales de esta metodología son esencialmente tres: especificación, desarrollo y validación. En la Figura 1 se presenta un diagrama que representa el uso de esta metodología.

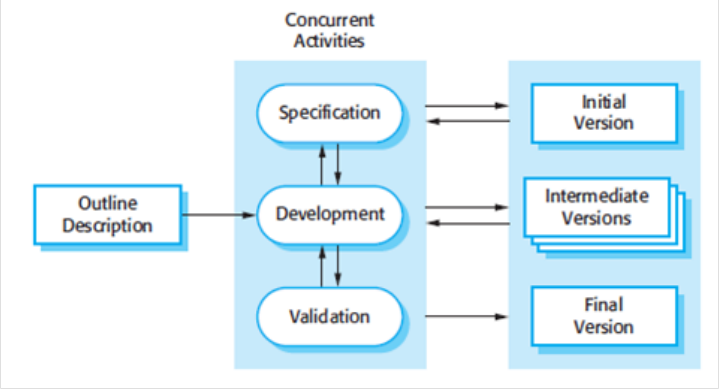


Figura 1.1: Metodología de Desarrollo Incremental (extraído de Sommerville [18]).

Para la etapa de descripción general (previa al comienzo del desarrollo en sí) se estudia el estado actual del análisis escritural y del procesamiento computacional de imágenes de texto manuscrito, definiendo además una lista de patrones candidatos a detectar. Todo esto, con el objetivo de conocer aquellas características que puedan tener poder predictivo en alguna de las aplicaciones antes mencionadas.

Por otro lado, y como una primera aproximación, se analiza un conjunto de imágenes de muestra de texto manuscrito para reconocer visualmente los patrones de escritura y explorar la presencia o ausencia de las distintas características en cada imagen. Asimismo, se aplica una variedad de técnicas de procesamiento de imágenes para extraer métricas que den origen a vectores representativos de ellas (Steinherz et al. [21]).

El realizar esta fase previa permite llevar a cabo una **especificación de requisitos** para el software a desarrollar, pues según el modelo incremental, esta etapa corresponde principalmente al análisis que se debe hacer para detectar las necesidades a las que debe responder el programa (Sommerville [18]).

En este punto se inicia la etapa de **desarrollo del software**, que da origen a la primera versión del mismo, implementando una aplicación base que sirva de plataforma para continuar evolucionando las capacidades del *software* en dos líneas distintas. Por un lado, permitir la inclusión de nuevos tipos de procesamiento automático y, por otro, presentar una interfaz de asistencia manual, en que un experto pueda ajustar y calibrar el resultado de ejecución de los algoritmos. Luego, en iteraciones sucesivas este primer núcleo central debe ir evolucionando y creciendo de acuerdo a los requisitos levantados en la fase previa.

Posteriormente, en la fase de **validación**, se aplica los algoritmos desarrollados sobre un conjunto de imágenes de texto manuscrito que ya han sido analizadas por un perito

neuroescritural. Más concretamente, para la evaluación particular de los algoritmos se utiliza las métricas clásicas de evaluación de calidad de clasificación, tales como *accuracy*, *recall* y *false positive rate*; etiquetando como correcta o incorrecta cada una de las detecciones realizadas por el procedimiento computacional, en contraste con las realizadas de manera manual por el experto.

A partir de este punto, y tal como la metodología lo señala, las fases anteriores se repiten de manera indefinida hasta lograr llegar a algoritmos debidamente implementados que permitan alcanzar los niveles explicativos deseados, esto es, que la calidad de la detección de los patrones en las imágenes sea aceptable según lo indicado en la Tabla 1.1.

Finalmente, en el **despliegue** se lleva a cabo la última iteración del Modelo Incremental, desarrollando la versión final del software, implementando los modelos seleccionados en un sistema que permita la carga de imágenes y procesamiento automático de ellas, junto al sistema de asistencia manual. Además, en esta etapa se realizará la puesta en producción del software sobre una plataforma web para que pueda ser utilizado por un usuario que desee realizar un análisis grafológico de manera automática.

1.5. Alcances

Este proyecto abarca hasta la fase de implementación de una biblioteca funcional básica de detección de patrones, compuesta por una serie de rutinas creadas para tal efecto. Adicionalmente, se considera la construcción de una aplicación sencilla sobre una plataforma web que interactúe con esta biblioteca y que sirva para demostrar sus capacidades.

Lo que se espera de esta biblioteca es lo siguiente:

- Permitir la carga de una imagen digital de texto manuscrito.
- Permitir seleccionar patrones de interés a partir de un conjunto dado.
- Procesar y entregar como resultado una serie de mediciones correspondientes a los patrones elegidos por el usuario.

Dentro del proyecto, también se analizan las técnicas desarrolladas en la literatura respecto a la etapa de procesamiento previo de las imágenes de texto manuscrito para implementar las que sean de utilidad para el proyecto y mejorarlas en caso de ser posible.

Asimismo, se lleva a cabo una validación de las métricas obtenidas a partir de las técnicas desarrolladas para detectar patrones que no han sido abordadas en la literatura, mediante una comparación con los resultados generados a partir del análisis manual de un experto neuroescritural, con el fin de poder estimar la precisión de las técnicas desarrolladas.

Adicionalmente, este trabajo se enfoca en imágenes de texto en español, pero también soporta su aplicación en imágenes de textos escritos en otros idiomas con alfabeto de base latina. Por tanto, quedan fuera del alcance de este proyecto todos los idiomas que utilicen un alfabeto distinto (como árabe, japonés, ruso, etc), debido a la complejidad que significaría dar soporte a todos ellos.

Capítulo 2

Marco Teórico

El foco central de esta investigación está puesto en el procesamiento de imágenes de texto manuscrito y de sus aplicaciones. Por este motivo, en este capítulo se abordan las tres bases teóricas sobre las que se funda este trabajo.

2.1. Procesamiento y Análisis de Imágenes

2.1.1. Conceptos Generales

Una imagen digital puede entenderse como una función de dos dimensiones f donde para cada coordenada (x, y), el valor de $f(x, y)$ corresponde a una o más cantidades finitas positivas, que indican la intensidad de color en dicha posición de la imagen. Dependiendo del sistema de representación utilizado, esta función puede representar un trío de distintas intensidades de color (tal como en el sistema RGB) o sólo uno (en un sistema de escala de grises). Cada uno de estos elementos, compuestos de una coordenada y un valor de intensidad, es denominado *pixel* (Jensen [10]).

El procesamiento de imágenes digitales consiste, a grandes rasgos, en el uso de un sistema computacional para transformar una imagen digital en otra. Sin embargo, en la literatura no existe consenso respecto a los límites de esta área del conocimiento, ya que muchas veces su ámbito de acción se cruza con las áreas de visión por computador y de inteligencia artificial, entre otras. Aún así, una postura comúnmente aceptada define al procesamiento de imágenes como el conjunto de procedimientos cuya entrada y salida son imágenes, así como, también, aquellos procedimientos que extraen atributos de una imagen o que reconocen objetos dentro de ella (González and Woods [8]).

En general, el procesamiento de imágenes consta de los siguientes pasos:

1. Adquisición de la imagen.
2. Mejora de la imagen.

3. Restauración de la imagen.
4. Procesamiento del color de la imagen.
5. Compresión de la imagen.
6. Procesamiento morfológico.
7. Segmentación de la imagen.
8. Representación de la imagen.
9. Reconocimiento de objetos o patrones.

Por otro lado, cabe destacar que el procesamiento de imágenes no se limita a trabajar con imágenes generadas por el espectro de luz visible, sino que abarca todo el espectro electromagnético. De hecho, algunas de las aplicaciones más reconocidas de esta disciplina se encuentran justamente en frecuencias fuera del rango de luz visible.

A modo de ejemplo, dentro del procesamiento de imágenes de rayos gamma, se puede mencionar la captura de imágenes en medicina nuclear gracias a la ingesta de un isótopo radiactivo por parte del paciente, que emite rayos gamma a medida que decae y que pueden ser capturados por sensores apropiados, midiendo su intensidad y transformándolo en imágenes. En el otro extremo del espectro, se puede encontrar aplicaciones de procesamiento de imágenes satelitales infrarrojas del planeta, en que se estiman niveles de consumo eléctrico a partir de la intensidad de emisión por zona geográfica (González and Woods [8]).

Dentro de las aplicaciones de procesamiento y análisis de imágenes del espectro visible, se puede nombrar algunas como la detección de rostros, muy popular en cámaras fotográficas digitales con sistemas de autoenfoque (Viola and Jones [23]); el conteo de personas, por ejemplo, usado para monitoreo de flujos en zonas turísticas (Sacchi et al. [16]); el conteo de células en imágenes microscópicas (Kothari et al. [11]) y el reconocimiento de texto en imágenes.

Esta última aplicación, de particular relevancia dentro de este trabajo, puede ser esencialmente de dos tipos: Optical Character Recognition (OCR), que consiste en la identificación y reconocimiento de texto impreso; e Intelligent Character Recognition (ICR), que consiste en la identificación y reconocimiento de texto manuscrito en imágenes.

2.1.2. Propiedades de una imagen digital

Tal como ya se especificó, una imagen digital consiste de un conjunto de pixeles que guardan información con grados de intensidad de luz en cada coordenada de la imagen. Por este motivo, la representación computacional más común de una imagen se realiza mediante una matriz bidimensional que simula una grilla de colores.

Teniendo en cuenta que esta representación no es más que una discretización de una imagen real, es necesario revisar algunas propiedades típicas utilizadas en el mundo continuo. En este sentido, la noción de distancia es, probablemente, la más importante de ellas en el contexto de este proyecto.

Formalmente, una distancia es cualquier función D que satisfaga las siguientes condiciones:

$$D(p,q) \geq 0, \quad con \ \ (D(p,q) = 0 \Leftrightarrow p = q) \tag{2.1}$$

$$D(p,q) = D(q,p) \tag{2.2}$$

$$D(p,r) \leq D(p,q) + D(q,r) \tag{2.3}$$

Si bien es posible definir muchas funciones que satisfagan dichas condiciones, en el ámbito de este trabajo es importante considerar la distancia de Minkowski L_p. En su caso general, se define la distancia de Minkowski de orden p entre dos puntos $p = (x_1, x_2, \ldots, x_n)$ y $q = (y_1, y_2, \ldots, y_n)$ como:

$$L_p = \left(\sum_{\mathrm{i}=1}^{n} |x_\mathrm{i} - y_\mathrm{i}|^p \right)^{1/p} \tag{2.4}$$

Llevada a su aplicación en dos dimensiones (lo cual ocurre en una imagen), es posible distinguir algunos casos particulares de interés. En primer lugar, para el caso $p = 1$, se conoce como distancia Manhattan o 'distancia de bloque' (ver Ecuación 2.5), que intuitivamente representa la cantidad mínima de posiciones que es necesario recorrer, mediante movimientos verticales y horizontales, para ir desde el punto $p = (\mathrm{i}, j)$ al punto $q = (h, k)$.

$$L_1 : D(p,q) = |\mathrm{i} - h| + |j - k| \tag{2.5}$$

Por otro lado, para el caso $p = 2$, se recupera la distancia Euclideana definida a partir de la noción geométrica clásica en que, para un par de puntos $p = (\mathrm{i}, j)$ y $q = (h, k)$, se satisface la Ecuación 2.6. En el contexto del procesamiento de imágenes digitales, su principal desventaja es que entrega valores no enteros, lo cual puede dificultar su utilización en algunos casos.

$$L_2 : D(p,q) = \sqrt{(\mathrm{i} - h)^2 + (j - k)^2} \tag{2.6}$$

Por último, para el caso $p = \infty$ se obtiene la Ecuación 2.7, conocida como 'distancia máximo' o 'distancia de tablero de ajedrez'. Intuitivamente, se puede interpretar como la cantidad de posiciones que es necesario recorrer para el ir desde el punto $p = (\mathrm{i}, j)$ al punto $q = (h, k)$, permitiendo movimientos verticales, horizontales y diagonales.

$$L_\infty : D(p,q) = max\{|\mathrm{i} - h|, |j - k|\} \tag{2.7}$$

Otra propiedad importante de tener en consideración al momento de realizar procesamiento de imágenes digitales es la de adyacencia. Dos pixeles p y q se denominan *4-adyacentes* si $L1 : D(p,q) = 1$ y se conocen como *8-adyacentes* si satisfacen $L_\infty : D(p,q) = 1$. Esta noción se puede entender gráficamente mediante la Figura 2.1.

Además, se conoce como región a un conjunto de pixeles tales que para cada pixel p y q dentro de la región, existe un camino de pixeles adyacentes que los conectan. Formalmente, una región también se conoce como componente conexa. Esta noción cobra sentido cuando se buscan pixeles de un cierto color (comúnmente negro) sobre un fondo plano (comúnmente blanco).

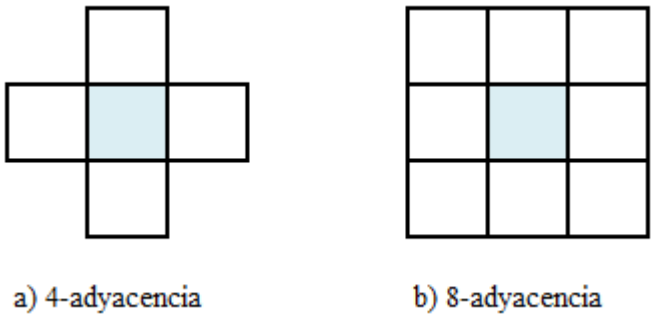


Figura 2.1: Tipos de adyacencia para un pixel central

Por último, otra de las propiedades de una imagen que se desea cubrir corresponde a su distribución de intensidad de color, usualmente conocida como histograma de brillo o de profundidad de color. Esta propiedad puede ser analizada para cualquier imagen (ya sea a color o en escala de grises). Sin embargo, sólo se presenta para el caso de escala de grises, puesto que el caso de colores es una generalización del mismo.

Dada una imagen en escala de grises, con L niveles, se definen $B \leq L$ rangos adyacentes de intensidad $(R_1, R_2, \ldots, R_B)$. Luego, el histograma de B niveles se calcula como la cantidad de pixeles que existen en cada uno de los B rangos. De esta forma, se puede caracterizar la composición de la imagen y, además, es posible determinar qué rangos de intensidad son los predominantes en ella.

2.1.3. Transformaciones de una imagen

La cantidad de posibles transformaciones de una imagen que pueden se realizadas de manera computacional es prácticamente infinita. Sin embargo, en este documento se pretende abordar las principales técnicas que sirven como base para el desarrollo del proyecto.

Transformaciones de color

Las transformaciones de color consisten, básicamente, en alterar los niveles de intensidad almacenados en cada pixel. Las razones de hacerlo pueden ser muy variadas, desde mejorar la visualización de la imagen, hasta segmentar distintas partes de la misma.

Una de las transformaciones de color más comunes es la de llevar una imagen a escala de grises. Este procedimiento se realiza midiendo las intensidades de color de cada una de las capas (típicamente rojo, azul y verde), promediándolas de manera ponderada (utilizando coeficientes definidos en función de lo que se quiera lograr) y obteniendo un valor único que representa la intensidad en una escala de grises.

En este trabajo se utiliza la fórmula de transformación definida por el estándar CCIR 601 y que corresponde a la implementación provista por OpenCV. En esta definición, el nivel de

intensidad Y del pixel en escala de grises se calcula a partir de las intensidades de color rojo R, verde G y azul B, de la manera indicada en la Ecuación 2.8.

$$Y = 0,299R + 0,587G + 0,114B \tag{2.8}$$

Otra de las transformaciones que entran en esta categoría es la de *thresholding* que permite reducir la cantidad de intensidades presentes en una imagen a partir de un umbral. La más común de este tipo de transformaciones es la de *thresholding binario* en que la intensidad int del pixel p se define como:

$$\text{i}nt(p) = \begin{cases} valor, & \text{si i}nt(p) > umbral \\ 0, & \text{si no} \end{cases} \tag{2.9}$$

Transformaciones morfológicas

Las transformaciones morfológicas son un conjunto de operaciones que utilizan las propiedades del álgebra lineal sobre matrices de imágenes en blanco y negro para lograr cambios que tienen algún significado desde el punto de vista visual. Las operaciones morfológicas básicas son dilatación y erosión, explicadas a continuación.

La dilatación $\oplus$ es la transformación que, a partir de una matriz M y un elemento estructural B, genera una matriz resultante definida por todos los puntos resultantes de todas las posibles adiciones de puntos entre un elemento de M y un elemento de B, usando la noción clásica de adición (es decir, $(\text{i}, j) + (h, k) = (\text{i} + h, j + k)$). Formalmente, se puede anotar como:

$$M \oplus B = \{p \in \varepsilon^2 : p = m + b, m \in M \text{ y } b \in B\} \tag{2.10}$$

En la Figura 2.2 se puede observar un ejemplo de dilatación (tomado de Sonka et al. [19]):

$$\begin{aligned} M &= \{(1,0),(1,1),(1,2),(2,2),(0,3),(0,4)\}, \\ B &= \{(0,0),(1,0)\}, \\ M \oplus B &= \{(1,0),(1,1),(1,2),(2,2),(0,3),(0,4),(2,0),(2,1),(2,2),(3,2),(1,3),(1,4)\}. \end{aligned} \tag{2.11}$$

El efecto concreto que tiene una operación de dilatación en una imagen de texto negro sobre fondo blanco es el de expandir los trazos, logrando conectar regiones que en la imagen original no lo están. El tamaño de esta expansión, así como la dirección de la misma, va a depender directamente de la forma utilizada para el elemento estructural B, siendo sus formas más comunes un cuadrado y una cruz.

Por otro lado, la erosión $\ominus$ es una operación análoga (pero no inversa), que se basa en la sustracción. Su funcionamiento se puede describir diciendo que dada una matriz inicial M y un elemento estructural B, se genera una matriz resultante $M \ominus B$ compuesta por aquellos puntos p para los cuales todos los posibles puntos $p+b$ ($b \in B$) pertenecen a M. Formalmente, esta operación se puede definir como:

$$M \ominus B = \{p \in \varepsilon^2 : p = m + b \in M \text{ para cada } b \in B\} \tag{2.12}$$

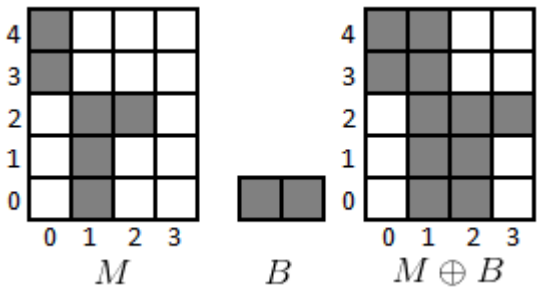


Figura 2.2: Ejemplo sencillo de dilatación

Un ejemplo gráfico de cómo esta operación ocurre, se puede observar en la Figura 2.3:

$$\begin{aligned} M &= \{(1,0),(1,1),(1,2),(0,3),(1,3),(2,3),(3,3),(1,4)\}, \\ B &= \{(0,0),(1,0)\}, \\ M \oplus B &= \{(0,3),(1,3),(2,3)\}. \end{aligned} \tag{2.13}$$

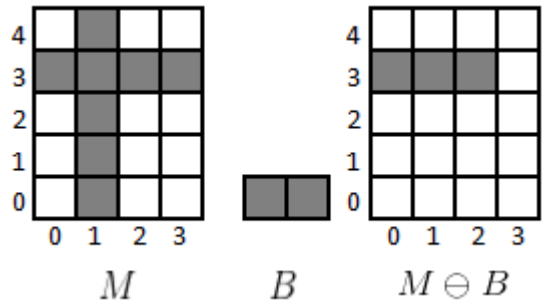


Figura 2.3: Ejemplo sencillo de erosión

El efecto visual que produce la aplicación de erosión en una imagen de texto negro sobre fondo blanco es el de adelgazar los trazos que la componen y eliminar aquellos muy delgados. La magnitud de este adelgazamiento, así como la selección de cuáles trazos desaparecen, depende directamente del elemento estructural utilizado.

Una observación importante es que aplicar dilatación y luego erosión sobre una imagen utilizando un elemento estructural adecuado, permite realizar una extracción de contornos de la misma.

2.2. Análisis de Imágenes de Texto

De acuerdo a lo presentado, las principales aplicaciones del análisis de imágenes de texto apuntan al reconocimiento de caracteres, ya sea a partir de texto manuscrito o de texto impreso digitalizado. Sin embargo, otra importante aplicación que ha sido abordada en la literatura es la del reconocimiento e identificación de firmas, para su validación automática.

En cualquiera de estas aplicaciones, una metodología de procesamiento consiste en extraer una serie de características de la imagen (de una palabra, letra, firma, etc.) que sean representativas y que sirvan para discriminar entre imágenes similares y distintas. De esta forma, se puede crear un conjunto de variables que permitan entrenar un algoritmo clasificador.

Por ejemplo, en el problema del reconocimiento de firmas, algunos de las características que se busca calcular, de acuerdo a lo planteado en Frias-Martinez et al. [5], son:

1. **Proyecciones horizontal (P_H) y vertical (P_V)**: Corresponden a la distribución de pixeles negros (o distintos de blanco) a lo largo de los ejes X e Y dentro de una imagen de dimensiones $x_{max} \times y_{max}$. Formalmente pueden definirse como:

$$P_H[y] = \sum_{x=1}^{x_{max}} b(x,y), \quad y = 1, \ldots, y_{max} \tag{2.14}$$

$$P_V[x] = \sum_{y=1}^{y_{max}} b(x,y), \quad x = 1, \ldots, x_{max} \tag{2.15}$$

   donde $b(x,y) \in \{0,1\}$ indica si existe un pixel negro (o no blanco) en la coordenada (x,y) de la imagen.

2. **Centros de gravedad horizontal (C_H) y vertical (C_V)**: Corresponden a la noción clásica de centro de gravedad en función de la distribución de pixeles negros sobre el fondo blanco. Formalmente puede calcularse en función de las proyecciones horizontal y vertical, respectivamente:

$$C_H = \frac{\sum_{x=1}^{x_{max}} x P_V[x]}{\sum_{x=1}^{x_{max}} P_V[x]} \tag{2.16}$$

$$C_V = \frac{\sum_{y=1}^{y_{max}} y P_H[y]}{\sum_{y=1}^{y_{max}} P_H[y]} \tag{2.17}$$

3. **Líneas basales horizontal (B_H) y vertical (B_V)**: Corresponden a las rectas horizontal y vertical que concentran la mayor cantidad de pixeles en cada eje. Formalmente pueden definirse como:

$$B_H = max\{P_H[y]\}, \quad y = 1, \ldots, y_{max} \tag{2.18}$$

$$B_V = max\{P_V[x]\}, \quad x = 1, \ldots, x_{max} \tag{2.19}$$

4. **Recta de mínimos cuadrados**: Corresponde a la recta que marca la tendencia de la firma, en el sentido de la dirección en que está escrita. Esta recta se define por el conjunto de pixeles que satisfacen $y = mx + b$, donde b y m satisfacen, respectivamente:

$$b = \frac{\left(\sum_{i=1}^{N} x_i^2\right)\left(\sum_{i=1}^{N} y_i\right) - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} x_i y_i\right)}{N\left(\sum_{i=1}^{N} x_i^2\right) - \left(\sum_{i=1}^{N} x_i\right)} \tag{2.20}$$

$$m = \frac{N\left(\sum_{i=1}^{N} x_i y_i\right) - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} y_i\right)}{N\left(\sum_{i=1}^{N} x_i^2\right) - \left(\sum_{i=1}^{N} x_i\right)^2} \tag{2.21}$$

5. **Otras medidas geométricas**: Algunas otras medidas geométricas buscadas son el área de la envoltura convexa de los puntos de la firma, su perímetro y el ratio $\frac{área}{perímetro}$.

Por otro lado, en la literatura se ha estudiado diversas distorsiones del texto manuscrito que buscan ser corregidas para mejorar la calidad de los sistemas de reconocimiento de texto (Marti and Bunke [13]). Si bien, para el desarrollo de este trabajo, no se busca lograr reconocimiento de caracteres, sí es de interés detectar estas distorsiones propias de la escritura, pues algunas de ellas constituyen patrones escriturales específicos.

En este sentido, diversos autores coinciden en que hay tres características principales que generan distorsión en un texto: ángulo de inclinación de la palabra (*skew angle*), ángulo de inclinación de las letras (*slant angle*) y relación entre la línea basal inferior y la línea basal superior (Marti and Bunke [13]). En la Figura 2.4 se presenta una imagen que ejemplifica la presencia de estas características.

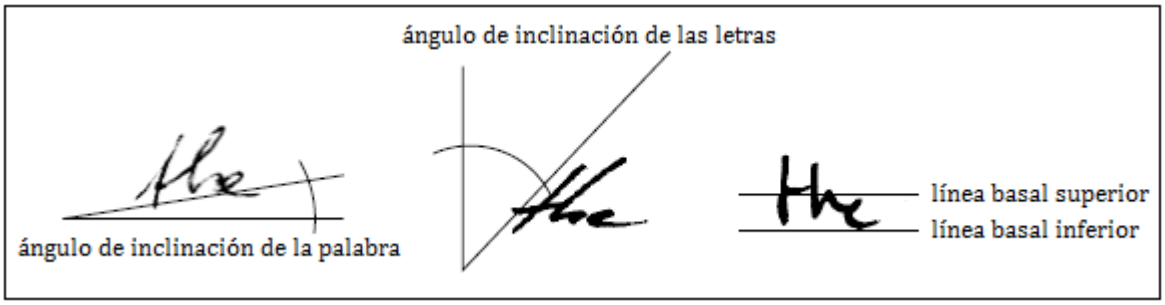


Figura 2.4: Características propias de distorsión de un texto

Para poder detectar estos patrones y corregirlos, se han planteado distintas técnicas en Plamondon and Srihari [15] y en Marti and Bunke [13], pero su explicación queda fuera del alcance de este trabajo.

2.3. Grafología

La grafología es una disciplina dedicada al estudio de la forma de la escritura de un individuo con el fin de conocer características de su personalidad. Si bien esta disciplina ha sido cuestionada en cuanto a su nivel de validez científica, hoy en día es usada activamente

en procesos de selección y evaluación de personal para conocer el perfil de los postulantes a distintos cargos dentro una organización.

En el contexto de este trabajo, la grafología es usada en la etapa de validación de la calidad de los algoritmos desarrollados. En particular, lo que se busca es detectar de manera automática los patrones que un experto en la materia analiza de manera manual y contrastar los resultados obtenidos por la herramienta computacional con aquellos extraídos por el experto. Por tanto, es importante conocer los fundamentos en que se basan estas técnicas.

El acercamiento que se ha realizado a esta materia está basado en el libro *Grafología: Teoría y Práctica* de los autores Honroth y Ribera [9]. En él se plantea que la grafología científica es el estudio analítico de los movimientos de expresión gráfica del individuo, es decir, el análisis de las causas o motivos psicosomáticos circunstanciales que inducen a fijar el gesto gráfico de un modo peculiar y personal sobre el papel.

Adicionalmente, se plantea que dicha expresión gráfica es la resultante de dos procesos psicomotrices: un movimiento imitador, voluntario y consciente; y un movimiento modificador, involuntario e inconsciente. Este último movimiento sería el dominante y el responsable de plasmar características propias del individuo en la manera como escribe.

Fisiológicamente, esta influencia se explicaría por la conexión de los distintos centros nerviosos ubicados en el cerebro, que posibilitan la percepción a través de los distintos sentidos. Estos centros nerviosos son el centro de la visión, que preside la percepción visual; el centro motor del lenguaje, que rige la acción de los músculos de la laringe, los labios, la lengua, el velo del paladar y del tórax, relacionados con la emisión de la voz; el centro sensorial del lenguaje, que regula la comprensión de los sonidos y del lenguaje hablado; el centro de la lectura, que preside la comprensión del lenguaje escrito; y el centro motor de la escritura, que controla la acción de los músculos de la mano al momento de escribir.

Por otro lado, los autores plantean que toda expresión gráfica del ser humano involucra un cierto nivel de simbolismo. De esta forma, analizando las tendencias de la escritura es posible desentrañar características propias de la persona. Si bien en Honroth and Ribera [9] se presenta un detalle del significado de cada uno de los estilos de escritura, a modo de ejemplo se puede mencionar que si una persona escribe otorgando predominancia a las zonas altas del texto (con letras *t*, *l*, *b* y *d* alargadas hacia arriba y muy altas), se puede interpretar como una tendencia hacia el idealismo y la grandeza.

En dicho texto, además, el autor introduce una rama de la grafología denominada grafopatología, que estudia los trastornos de la escritura producidos por alteraciones en el estado de salud de la persona, distinguiendo entre anomalías de orden somático, anomalías de orden psíquico y anomalías de orden psicosomático. Esto muestra que las aplicaciones de la grafología pueden ser muy amplias y van más allá del estudio de la personalidad.

Más concretamente, algunos patrones y características que se busca medir en un análisis grafológico son los detallados a continuación:

1. **Espaciamientos inter-línea**: Corresponde a la distancia entre líneas consecutivas de texto, calculando una relación entre el espaciamiento y la altura del cuerpo central de

cada línea.

2. **Medidas de zonas de texto o tamaño de cuerpos**: Corresponde a las medidas de las zonas superior, media e inferior de cada línea del texto. La zona media corresponde a aquella delimitada por las letras minúsculas que no tienen proyecciones hacia arriba o hacia abajo. La zona superior es aquella comprendida entre el límite superior de la zona central y la línea tope demarcada por las letras que poseen proyecciones hacia arriba (como las letras *t*, *l*, *b* y *d*). Por último, la zona inferior es aquella delimitada entre el límite inferior de la zona central y la línea tope demarcada por las letras que poseen proyecciones hacia abajo (como las letras *p* y *g*).
3. **Grado de conexión intra-palabras**: Corresponde a la relación entre la cantidad de desconexiones que hay entre letras de una misma palabra y el número máximo que podría haber (equivalente al número de letras menos uno).
4. **Grado de conexión intra-letras**: Corresponde a la cantidad de desconexiones (o veces que se levantó el lápiz del papel) dentro del trazo de una misma letra. Este fenómeno es común en letras cerradas con partes circulares, tales como la *a* y la *d*.
5. **Espaciamiento entre palabras**: Corresponde a la distancia horizontal promedio entre palabras de una misma línea, calculado como una relación respecto al ancho promedio de las letras que componen la línea.
6. **Inclinación de líneas**: Corresponde al ángulo de inclinación de las líneas del texto respecto a la horizontal. En el caso de escrituras que no poseen una inclinación única, se calcula un ángulo por cuadrantes en la hoja.
7. **Tamaño de márgenes**: Tal como su nombre lo indica, corresponde a la distancia dejada desde el borde de la hoja hasta el área de texto escrito por la persona. Se calcula un margen promedio superior, inferior, lateral izquierdo y lateral derecho.

Finalmente, a partir de éstas y otras numerosas características, un grafólogo es capaz de sintetizar las mediciones obtenidas y sacar conclusiones a partir de ellas, pudiendo crear un perfil psicológico del autor del texto.

Capítulo 3

Análisis y Diseño de la Solución

En este capítulo se realiza una descripción de los requerimientos que satisface la solución que aborda el problema y se presenta un diseño integral (incluyendo dimensiones arquitectónica y lógica) que es capaz de cumplir con dichos requerimientos, sirviendo como base para la implementación de la solución.

3.1. Requerimientos de la solución

Uno de los objetivos prácticos de este proyecto es el de abordar el análisis neuroescritural desde un punto de vista tecnológico, automatizando el proceso de detección de características de texto y apoyando la toma de decisión del experto. De esta forma, se busca estandarizar parte del procedimiento y disminuir la carga de trabajo manual que implica realizar un análisis.

Por dicho motivo, las necesidades que satisface la solución deben estar alineadas con la realidad de un análisis de esta índole. Teniendo esto en consideración, se realiza un estudio de cuál es el procedimiento concreto que realiza un perito escritural al llevar a cabo un análisis, para poder determinar cuáles etapas de este proceso son susceptibles de ser automatizadas. Este estudio se puede sintetizar en los pasos mostrados en la Figura 3.1.

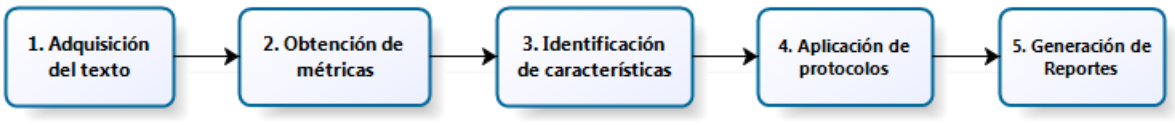


Figura 3.1: Etapas de un análisis neuroescritural manual

1. **Adquisición del texto:** Consiste en conseguir una muestra de un texto manuscrito de al menos diez líneas sobre un fondo blanco, firmado al final (ya sea físicamente o en formato digital).

2. **Obtención de métricas:** Consiste en realizar mediciones sobre atributos del texto, tales como tamaño de márgenes, espacios entre palabras y ángulos de inclinación de líneas.
3. **Identificación de características:** A partir de las métricas obtenidas anteriormente y de observaciones generales sobre el texto (usando el juicio del experto) se identifican características del mismo, tales como *tipo de distribución del texto*, *nivel de regularidad en ritmo* y *grado de cohesión*.
4. **Aplicación de protocolos:** Utilizando las características extraídas en el paso previo, se calcula niveles de logro (denominados protocolos) en cada uno de los atributos buscados, tales como *responsabilidad*, *trabajo bajo presión* y *sentido del deber*. El cómo se traducen las características del texto en protocolos, está basado en la teoría de la grafología y es parte del conocimiento del perito neuroescritural.
5. **Generación de reportes:** Finalmente, se construye un reporte que sintetiza los niveles de logro calculados y presenta las características de la personalidad del autor del texto. Este documento es el que se provee como resultado a quien haya solicitado el análisis.

Estudiando las etapas 2, 3 y 4 del análisis, aplicado a un proceso de evaluación de personal, es posible identificar un total de 20 métricas, 51 características y 5 protocolos. La razón de que el número de métricas sea menor a la cantidad de características se justifica debido a que algunas de estas últimas, son reconocidas por el ojo experto del perito y no son identificadas a partir de un valor medido del texto. Sin embargo, se presume que en un futuro, y con una cantidad de información considerable, sería posible construir un sistema que emule este juicio experto a partir de otras métricas, utilizando técnicas de minería de datos.

Idealmente, la solución propuesta debería automatizar cada una de las etapas anteriormente descritas. Sin embargo, al tener una fuerte componente de análisis experto, se propone construir una plataforma que asista al perito durante el proceso de análisis, mediante un sistema semiautomático.

De esta forma, se requiere construir un sistema de dos componentes. Por una parte, se debe contar con una plataforma de *backoffice* que le permita al perito crear un sujeto de análisis dentro del sistema, cargar la imagen asociada al texto escrito por él, ejecutar un procesamiento automático preliminar, corregir los errores de este procesamiento de manera manual, obtener las métricas susceptibles de ser calculadas de manera automática, ingresar la identificación manual de las características basadas en juicio experto, calcular los niveles de cada uno de los protocolos y generar un reporte final.

Por otro lado, se requiere también una solución que automatice la Etapa 1 del análisis. Para ello es necesario contar con una plataforma de *frontoffice* en la que un cliente que desee someterse a un estudio (o una empresa que desee evaluar a su personal), pueda cargar un sujeto de análisis con su respectiva imagen dentro del sistema y, una vez que el perito complete el procesamiento, pueda consultar el informe generado para él.

Por último, se requiere que el sistema a desarrollar sea capaz de lograr un grado de efectividad tal que no altere el resultado final del análisis, comparado con el resultado que se habría obtenido al realizar un análisis manual. Por este motivo, y con el fin de evaluar y validar la propuesta, se cuenta con un conjunto de 15 imágenes correspondientes a 15

sujetos distintos que ya han sido de analizados de manera manual, para las cuales se tiene su identificación de características, su construcción de protocolos y su informe respectivo.

3.2. Diseño de la solución

Pensando en satisfacer los requerimientos ya expuestos, se diseña una solución compuesta de tres grandes componentes:

1. Programa de procesamiento *backend* que cumpla la función de implementar toda la lógica del procesamiento de imágenes.
2. Plataforma de *backoffice* para el experto, que actúe como una interfaz hacia el primer componente.
3. Plataforma de *frontoffice*, para los clientes que deseen cargar sus imágenes al sistema, sirviendo como canal nutridor de información para la plataforma de *backoffice*.

Abstrayendóse de la implementación y de las tecnologías utilizadas para desarrollar y conectar los distintos componentes, esta solución puede sintetizarse en el esquema de la Figura 3.2.

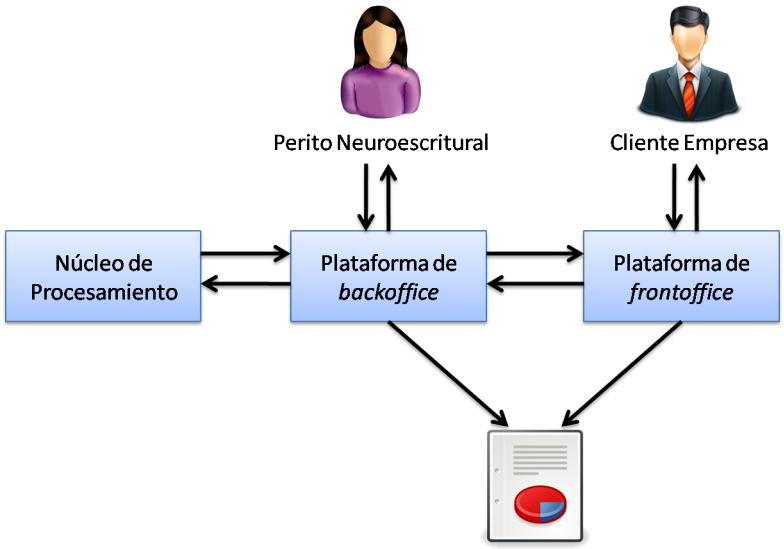


Figura 3.2: Diseño de alto nivel de la solución

3.2.1. Arquitectura

El diseño de alto nivel debe ser traducido en una arquitectura que permita su implementación. Por este motivo, se establece un diseño de arquitectura de 3 capas anidado, presentado en la Figura 3.3. El primer nivel se compone de una capa de procesamiento o lógica de negocio, correspondiente a la plataforma de *backend*; una capa de datos, correspondiente a la forma de almacenamiento de las imágenes, de los datos asociados y de las métricas calculadas;

y una capa de visualización, correspondiente a los sistemas de *backoffice* y *frontoffice*, puesto que actúan como interfaz hacia la capa de procesamiento.

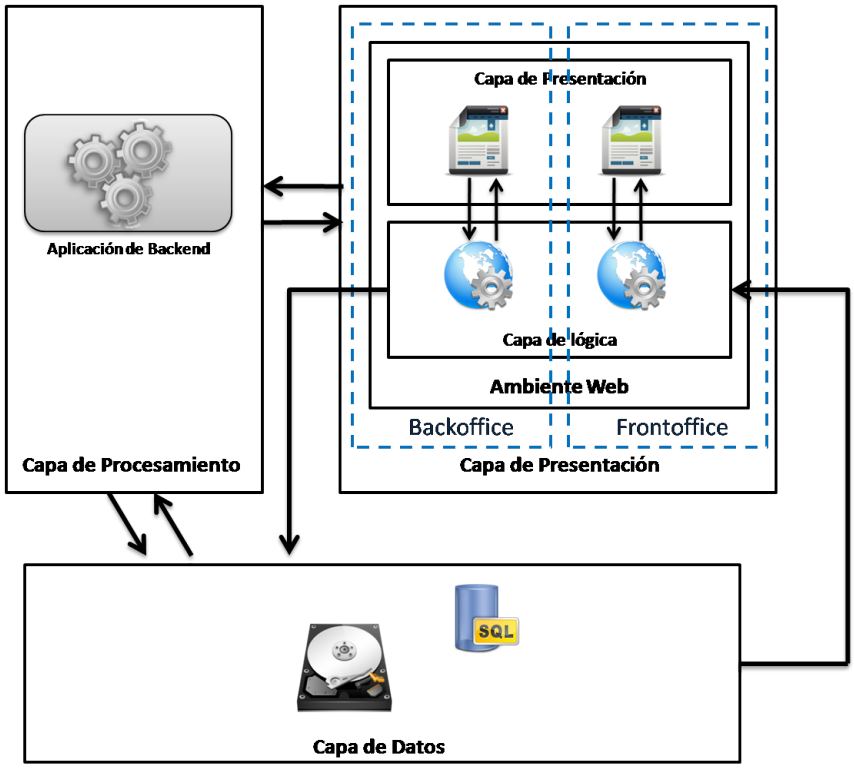


Figura 3.3: Arquitectura de la solución propuesta

Teniendo en consideración la naturaleza de la información que se debe almacenar en la capa de datos (principalmente imágenes y especificaciones de las mismas), se utiliza almacenamiento en disco tradicional y en una base de datos relacional, se almacenan las asociaciones entre los distintos archivos necesarios para un análisis.

En la misma línea, debido a la eficiencia requerida para un trabajo con este tipo de archivos, se propone que la aplicación *backend* sea desarrollada nativamente sobre el sistema operativo. De esta forma se puede lograr una herramienta más robusta y estable frente a otras alternativas.

Por otro lado, debido a la necesidad de contar con las plataformas de *backoffice* y *frontoffice* disponibles desde cualquier lugar y con un fácil acceso para todo tipo de usuario, se requiere que estén basadas en una arquitectura web, accesibles desde un navegador tradicional o desde un dispositivo móvil. Por este motivo, sobre cada uno de estos sistemas se utiliza un modelo de 3 capas usando el paradigma MVC. Más aún, considerando que en la plataforma de *backoffice* el usuario debe ser capaz de asistir manualmente el funcionamiento del *backend*, cobra aún mayor sentido el hecho de que la plataforma web cuente con una pequeña capa de procesamiento, en la que además se realice la generación de los reportes para ser entregados a los respectivos usuarios.

Respecto a la comunicación de cada uno de estos módulos se decide usar protocolos estándares para facilitar su implementación. De esta forma, en el primer nivel se plantea que la

comunicación entre la capa de procesamiento y la capa de datos se realice mediante la API del sistema operativo para escritura y lectura en disco, y a través de una API de conexión y manipulación de la base de datos relacional. En el otro nivel, se propone que la comunicación entre la capa de procesamiento y la capa de presentación, se realice mediante archivos de tipo *JSON*, pues este formato de especificación facilitará la visualización en la capa correspondiente. Por último, al ser una arquitectura web, el método de comunicación entre la capa de presentación y el navegador del usuario, se llevará a cabo mediante el protocolo *HTTP*.

3.2.2. Diseño Lógico de Aplicación Backend

Debido a la naturaleza incremental de la metodología de desarrollo, el diseño lógico que se propone debe ser fácilmente extensible para incluir nuevos módulos que permitan ir incluyendo las nuevas funcionalidades requeridas. Asimismo, el diseño incluye la posibilidad de que módulos existentes puedan ser reemplazados por versiones actualizadas o mejoradas para cada una de las funcionalidades propuestas.

El diseño lógico se presenta en las Figuras ubicadas en sección Anexos A (el diagrama ha sido fraccionado en numerosas partes para su mejor visualización). En él, se puede apreciar la aplicación del paradigma de orientación a objetos, incluyendo el uso de diversos patrones de diseño que permiten dar una solución extensible y escalable al problema, de acuerdo a los requerimientos planteados.

Debido a la naturaleza de los análisis que se busca realizar con esta herramienta, es posible distinguir dos tipos de procesamiento sobre una imagen. En primer lugar, se tiene aquellos procesos que alteran la representación de la imagen original, es decir, que cambian su estructura interna, alterando sus características y su visualización. Ejemplos de este tipo de procesamiento son técnicas de reducción de ruido, operaciones morfológicas y otras transformaciones abordadas en el Capítulo 2.1. En segunda instancia, es posible distinguir procesos que identifican características de la imagen o que extraen métricas de la misma sin alterar su composición.

Con estos antecedentes en consideración, y como parte del diseño lógico, se busca representar una imagen (o una parte de ésta), mediante un objeto de imagen abstracto (*ImageObject*). Entre otras, puede concretizarse a través de:

- La imagen misma (objeto *Image*).
- Una caja contenedora de una palabra detectada dentro de la imagen (objeto *WordBox*).
- Una lista de cajas contenedoras de palabras (objeto *WordBoxes*)
- Una línea de texto (objeto *TextLine*).

Si bien cada uno de estos objetos concretos tiene atributos que le son propios e implementan funcionalidades únicas para ellos, todos comparten una misma interfaz mediante herencia, por lo que pueden ser manejados de manera transparente por otros objetos.

Dentro de la interfaz que implementan, todos estos objetos son capaces de almacenar una colección de metadatos representados por una clase abstracta denominada *MetaData*. Pero

adicionalmente, cada uno de estos objetos implementa también dicha interfaz, logrando una dualidad en el comportamiento de un objeto como metadato y viceversa. La idea de fondo es que cada objeto de tipo imagen puede tener metadatos asociados (como por ejemplo el resultado de la extracción de una métrica), pero que cada uno de estos metadatos a su vez pueda contener otros objetos de imagen y así sucesivamente, logrando una estructura de datos recursiva.

La justificación de esta decisión de diseño radica en la secuencialidad del tipo de procesamiento particular que se quiere realizar para esta aplicación. A modo de ejemplo, a partir de la imagen de un texto (encapsulada en un *ImageObject*) es posible detectar las palabras que la componen (encapsuladas también en una concretización de un *ImageObject*, pero simultáneamente enlazadas a la imagen como un *MetaData*) y, a su vez, sobre cada palabra es posible detectar la línea basal de su cuerpo central (nuevamente representada por una concretización de un *ImageObject*, pero enlazada a la palabra como un *MetaData*). De acuerdo a la estructura propuesta, este procedimiento puede repetirse indefinidamente hasta lograr la profundidad requerida según sea la situación.

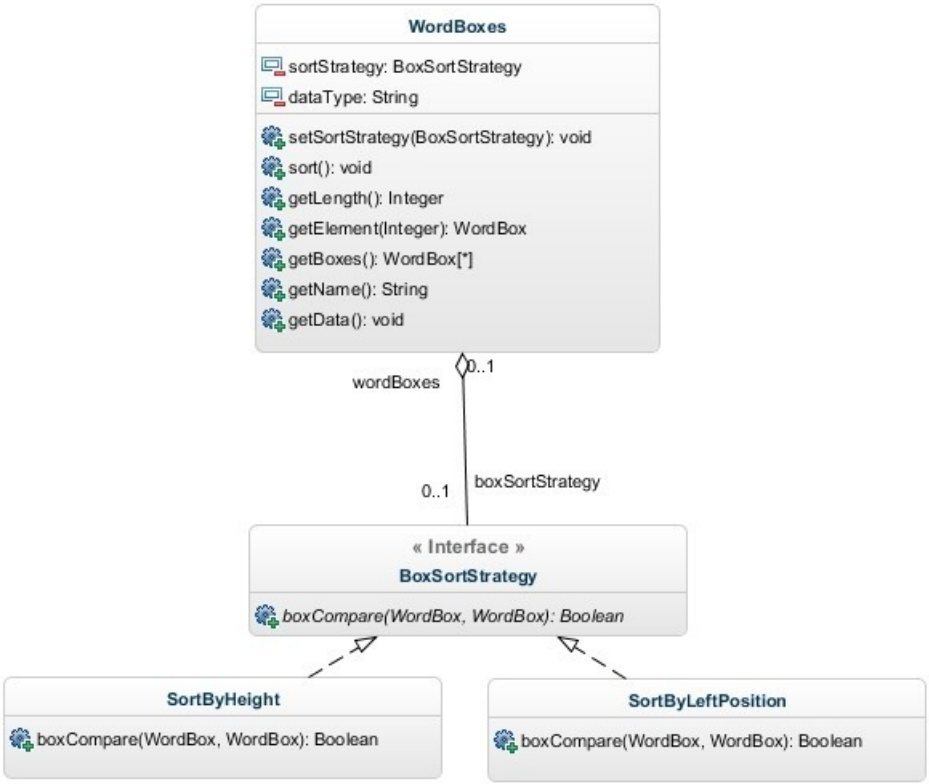


Figura 3.4: Ejemplo de uso del patrón *Strategy* en el ordenamiento de palabras.

Por otro lado, para implementar ciertas funcionalidades en algunos de los objetos que representan imágenes se utiliza el patrón de diseño *Strategy* (Gamma et al. [6, cap. 5]). Un ejemplo de su aplicación se puede observar en la Figura 3.4; en ella se presenta el caso de ordenar la colección de palabras de una imagen (*WordBoxes*), procedimiento que puede ser llevado a cabo según distintos criterios. Utilizando el patrón de diseño mencionado, es posible definir una interfaz única que resuelve el problema de ordenamiento, mediante la

implementación de cada una de las estrategias (por ejemplo ordenando según tamaño, según posición del vértice superior izquierdo, etc.).

Dentro de las aplicaciones de procesamiento de imágenes en general, la visualización del resultado de las transformaciones corresponde a una etapa crucial. Sin embargo, debido a la estructura jerárquica recursiva propuesta anteriormente, este paso no es directo, ya que la información base de una imagen, así como la generada a partir de las distintas etapas de procesamiento, se encuentran almacenadas en niveles inferiores de la jerarquía. Por este motivo, es necesario proponer un algoritmo que sea capaz de manejar esta estructura y que sea capaz de renderizar todos los niveles de la imagen en una sola.

Con dicho objetivo en consideración, se utiliza el patrón de diseño *Decorator* (Gamma et al. [6, cap. 4]), de manera de tener un conjunto de objetos decoradores, donde cada uno de ellos encapsula el conocimiento de cómo renderizar un tipo de objeto de imagen. De esta forma, se tiene un conjunto de objetos cuyas responsabilidades son renderizar la imagen base, renderizar las cajas que encierran a las palabras detectadas, marcar las líneas de los cuerpos de cada palabra y así sucesivamente. Por lo tanto, a partir de este diseño, y aprovechando el hecho de que todos los objetos de imagen comparten una misma interfaz, el renderizar una imagen es posible mediante la composición de los decoradores que sean necesarios. La estructura de esta solución se presenta en la Figura 3.5.

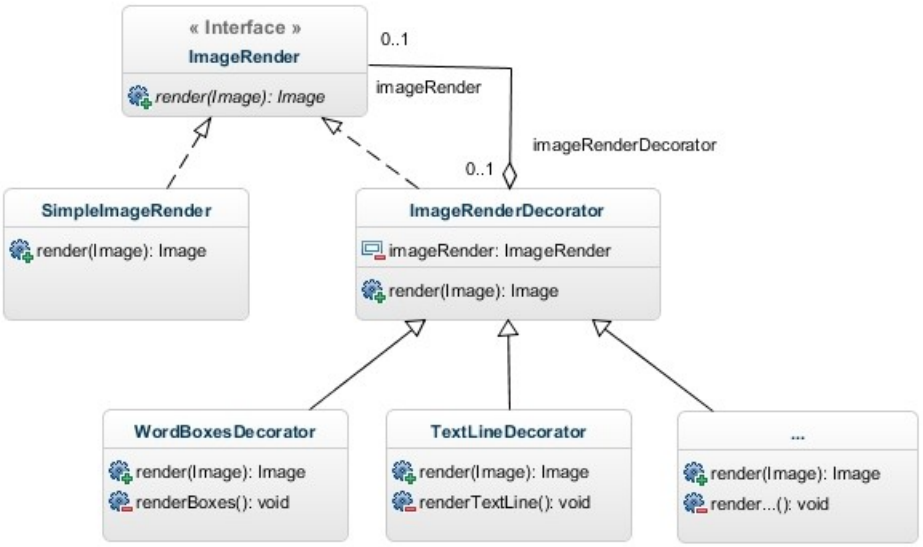


Figura 3.5: Uso del patrón *Decorator* para la renderización de la imagen.

Otro aspecto importante que debe considerar la solución planteada es el de permitir que los distintos algoritmos a utilizar para concretar el procesamiento de una imagen sean cargados en tiempo de ejecución. Por este motivo, se propone un diseño en el que cada algoritmo se implementa en un objeto distinto (tal como se presenta en el diagrama de la Figura 3.6, pero todos bajo una interfaz común (*ImageProcessor*) con un único método publico que permite ejecutar el procesamiento (método *run*), tomando como entrada un objeto *Image* y retornando uno del mismo tipo. Notar que debido a que los objetos de tipo *Image* implementan la interfaz de *ImageObject*, éstos son capaces de almacenar una lista de metadatos. Por lo tanto, la funcionalidad ejecutada por los objetos que encapsulan a los algoritmos de proce-

samiento, puede consistir en agregar nuevos metadatos a esta lista (que a su vez pueden ser otros objetos de tipo *ImageObject*).

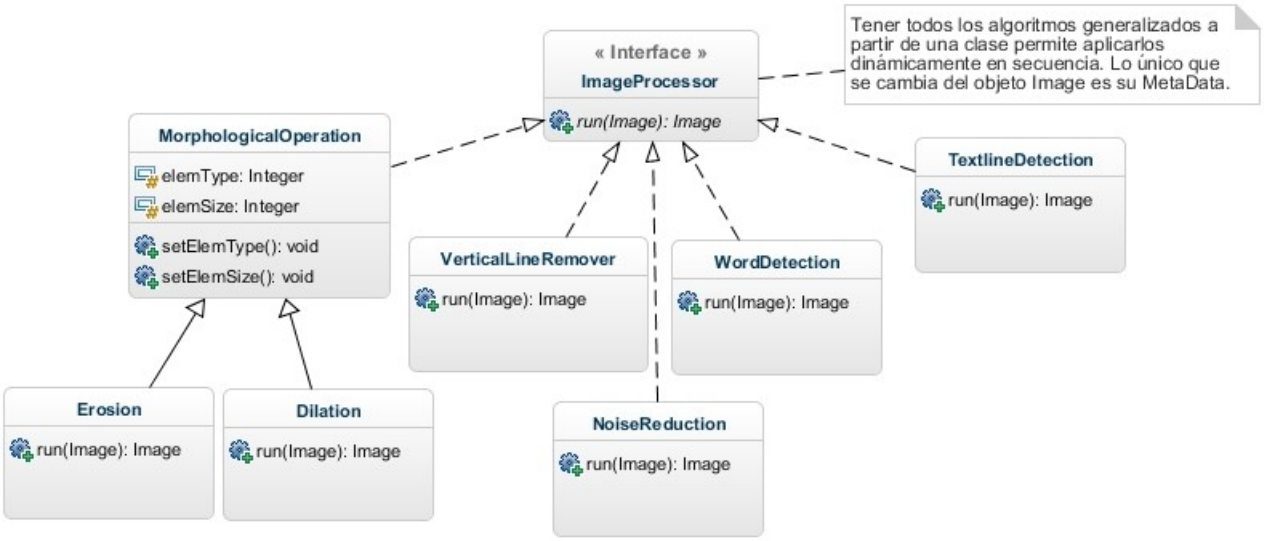


Figura 3.6: Algoritmos de procesamientos encapsulados en una interfaz común.

En línea con lo anterior, y considerando que esta aplicación corresponde al *backend* de la solución, es necesario contar con una manera sencilla de recibir una especificación de los objetos que se quiere procesar y de los algoritmos que se desea utilizar, para construirlos y ejecutarlos. Esto se logra mediante la utilización del patrón de diseño *Factory Method* (Gamma et al. [6, cap. 3]), donde se aprovecha el hecho de que, tanto objetos de imágenes como algoritmos tienen, cada uno, una interfaz común. De esta forma, es posible recibir una especificación, interpretarla y, en tiempo de ejecución, construir los objetos necesarios para responder a la solicitud.

Además, el proceso inverso debe ser también posible. Es decir, que a partir del resultado del procesamiento de una imagen (que queda encapsulado en un objeto de imagen con una lista de metadatos), pueda construirse una especificación que es transmitida a la aplicación que actúa en el *frontend*. Para ello, cada uno de los objetos de imagen necesita implementar obligatoriamente un método de serialización que cumpla dicha labor, pudiendo ser invocado de manera genérica desde la interfaz común.

Por otro lado, en el diseño lógico también se considera la construcción de clases para leer distintos tipos de imágenes y para almacenarlas en diversos formatos, así como otras clases que buscan otorgar flexibilidad al proceso iterativo de construcción de la solución. Sin embargo, el núcleo funcional es el que ya se ha presentado en esta sección.

Por último, cabe destacar que existe una clase principal única (*Main*) encargada de utilizar cada uno de los recursos diseñados, permitiendo comunicarse con la plataforma de *frontend*, recibir las especificaciones, construir los objetos representativos necesarios, ejecutar el procesamiento requerido, serializar su resultado y entregarlo mediante una especificación que de respuesta a la solicitud.

3.2.3. Diseño Lógico de Plataforma de Backoffice

La plataforma de *Backoffice* es la parte del sistema que es visible para el perito neuroescritural y es, justamente, la única interfaz de acceso que tiene hacia el procesamiento en *backend*. Por este motivo, debe proveerle de todas las funcionalidades necesarias para poder realizar un análisis neuroescritural, concretando todas las etapas introducidas en la Sección 3.1.

Por tal motivo, se realiza una traducción de los requisitos del usuario hacia un diagrama de casos de uso (presentado en la Figura 3.7) que pueda ser luego transformado en un diseño lógico que permita su implementación.

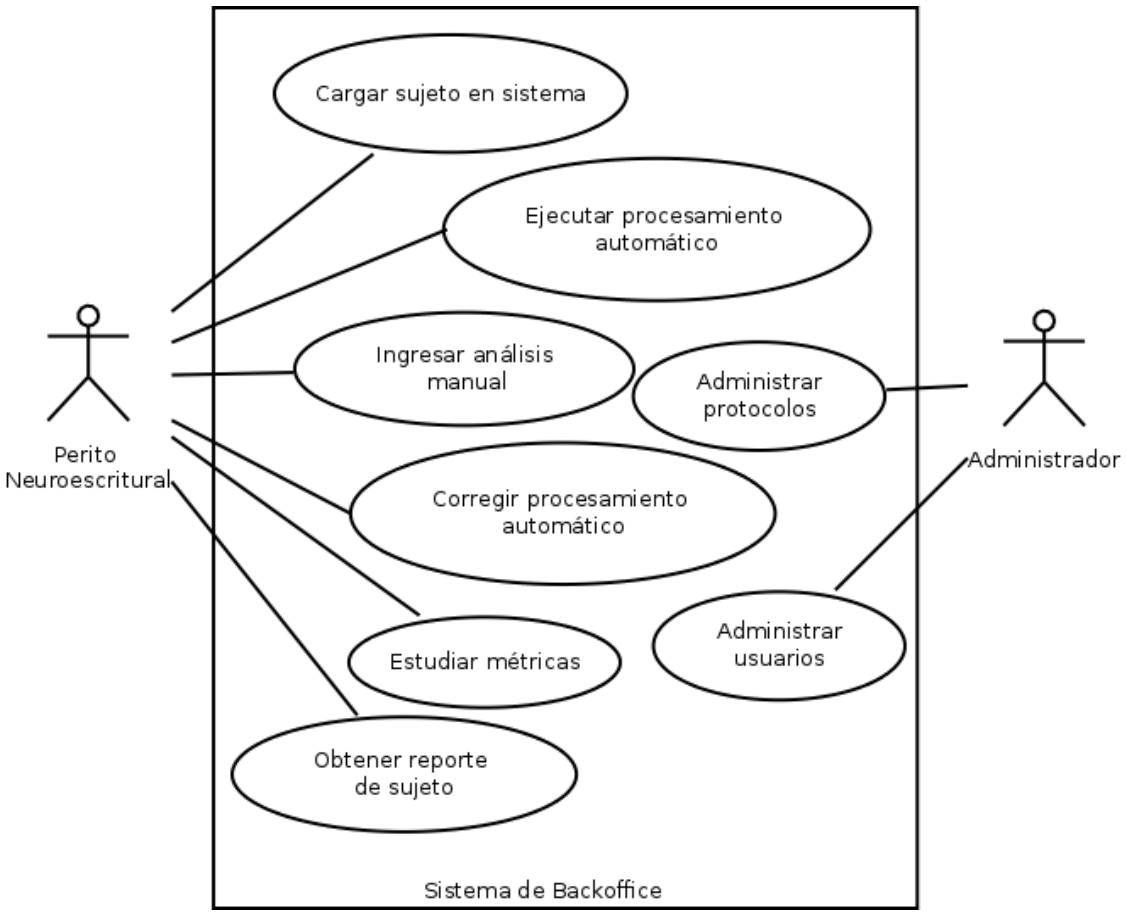


Figura 3.7: Diagrama de casos de uso para la plataforma de *backoffice*.

El caso de uso **Cargar sujeto en sistema** hace referencia a las funcionalidades relacionadas con la posibilidad de crear un nuevo sujeto en el sistema, ingresando su información básica (nombre, edad, sexo, empresa, cargo y país) junto a una imagen digitalizada de un texto escrito por el sujeto. Esta información debe quedar registrada de manera permanente en el sistema, para servir como base del procesamiento y análisis neuroescritural.

El caso de uso **Ejecutar procesamiento automático** se relaciona con la capacidad de invocar la ejecución de una cadena de algoritmos sobre la imagen de un sujeto, de manera de extraer las métricas de valor para el análisis. Cabe destacar que el actor involucrado en este caso de uso es el perito neuroescritural, que no tiene conocimiento técnico de los algoritmos

utilizados para obtener dichas métricas. Por este motivo, su ejecución debe ser sencilla y transparente, permitiéndole obtener los resultados que busca sin necesidad de entender el funcionamiento algorítmico que lo respalda.

El caso de uso **Corregir procesamiento automático** apunta a que el actor en cuestión, debe ser capaz de intervenir los resultados entregados por el sistema de procesamiento *backend* de manera de corregir las imperfecciones que hayan podido ocurrir durante la etapa de procesamiento. Para ello se debe proveer una interfaz usable que facilite la realización de esta tarea.

El caso de uso **Ingresar análisis manual** corresponde a la posibilidad que debe tener el perito de ingresar las características detectadas según su juicio experto y que no pueden ser detectadas automáticamente por el sistema. La finalidad de esta etapa es la de contar con un conjunto de datos completo que permita la aplicación de los protocolos y la generación automática de los respectivos reportes.

El caso de uso **Estudiar métricas**, como resultado de este proceso, se refiere a la capacidad que debe tener el usuario de observar cuáles son los resultados obtenidos por la herramienta, para su inspección y validación. Por este motivo, se propone construir una interfaz que muestre de manera gráfica, y en tablas, los valores obtenidos en cada una de las métricas de interés.

Como último caso de uso relevante para el rol del perito, se tiene *Obtener reporte de sujeto* que consiste en la capacidad de obtener un informe generado automáticamente por el sistema, que sintetice los niveles de logro alcanzados por el sujeto en cada uno de los protocolos, a partir de la mezcla del procesamiento automático y el análisis asistido realizado con anterioridad.

Por otro lado, se tiene un rol de administrador, el cual tiene dos funciones principales. Por una parte, posee el caso de uso **Administrar usuarios**, el cual le permite mantener y controlar el acceso a la plataforma de *frontoffice*. Por otra, tiene la posibilidad de **Administrar protocolos**, es decir, definir qué métricas influyen en el cálculo de cada una de las características medidas al momento de realizar un análisis escritural.

Teniendo en consideración el planteamiento de casos de uso ya descrito, y considerando que se busca realizar una implementación de acuerdo al paradigma MVC sobre una arquitectura web, se plantea un diseño de componentes de alto nivel como el presentado en la Figura 3.8, en donde la línea continua indica interacción directa y la línea punteada simboliza interacción indirecta.

Como se puede observar, es posible hacer una relación prácticamente uno a uno entre los componentes propuestos y los casos de uso detectados. De esta forma, se requiere construir un componente enfocado en el mantenimiento de sujetos, incluyendo su carga, edición de información y eliminación. Además, se contempla la construcción de un módulo de procesamiento, cuyo rol sería el de recuperar la imagen asociada al sujeto que se quiere analizar desde la capa de almacenamiento, generar una especificación y ejecutar el procesamiento. Adicionalmente, este módulo es el responsable de permitir el ingreso del análisis manual y de la edición de los resultados del procesamiento automático. Por otro lado, se plantea la

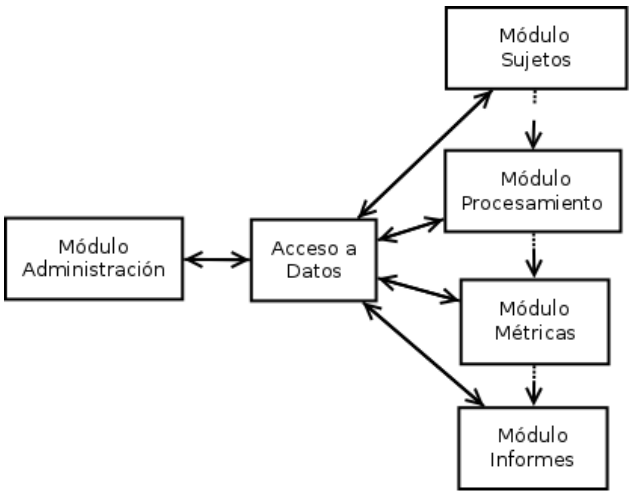


Figura 3.8: Diagrama de componentes de alto nivel para plataforma de *backoffice*.

creación de un módulo para la recuperación de las métricas obtenidas como salida de un proceso de análisis. Este componente sería el encargado de generar las estructuras adecuadas para lograr una presentación de los valores acorde a las necesidades del usuario. Por último, tal como su nombre lo dice, el módulo de reportes sería el encargado de realizar la aplicación de los protocolos para la obtención de los niveles de logro finales y la generación automática del informe del sujeto.

En paralelo, se propone la construcción de un módulo de administración que sirva como mantenedor de una lista de usuarios que tienen acceso a la plataforma de *frontoffice*.

3.2.4. Diseño Lógico de Plataforma de Frontoffice

Siguiendo la misma línea del diseño propuesto para la plataforma de *backoffice*, se presenta un diagrama de casos de uso (en Figura 3.9) que traducen los requerimientos del usuario en acciones concretas que deben ser permitidas por la plataforma.

Al observar este diagrama, es posible detectar dos aspectos relevantes. El primero de ellos guarda relación con el actor involucrado, que en este caso corresponde a un cliente externo, que en el ámbito de negocio sería una empresa que busca realizar el estudio de desempeño de sus empleados (siendo directamente extensible hacia personas naturales). El segundo aspecto de relevancia consiste en que los casos de uso presentados corresponden prácticamente a un subconjunto de aquellos disponibles en la plataforma de *backoffice*.

Nuevamente, a partir de este diagrama es posible traducir estas necesidades en un diseño por componentes de alto nivel que sea capaz de darles respuesta. Este diseño se presenta en la Figura 3.10 y las funcionalidades que cada uno de estos módulos debe satisfacer es análoga a los presentados en la Sección 3.2.3, pero para el caso en que el usuario es un cliente externo.

Cabe destacar que, en este caso, existe forzosamente una ventana temporal entre las distintas acciones que ejecuta el actor, ya que al solo tener la capacidad de cargar un sujeto

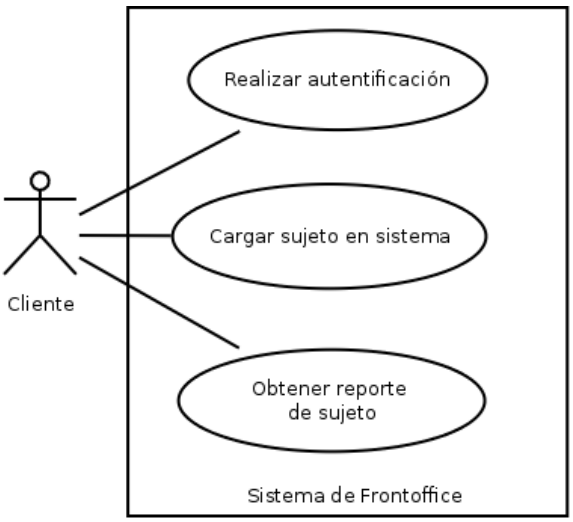


Figura 3.9: Diagrama de casos de uso para la plataforma de *frontoffice*

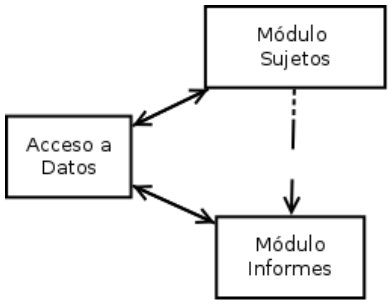


Figura 3.10: Diagrama de componentes de alto nivel para plataforma de *frontoffice*.

en el sistema, debe esperar que el perito neuroescritural desencadene el proceso automático, realice la asistencia manual y autorice la generación del informe.

3.2.5. Modelo de Datos

Si bien ya se ha explicado la necesidad que debe satisfacer la capa de datos de la aplicación, es necesario definir y presentar una estructura de almacenamiento coherente con los requerimientos detectados. Por este motivo, en esta sección se presenta el modelo de datos a utilizar en un motor de bases de datos relacional. En la Figura 3.11 se muestra un diagrama entidad relación del modelo de datos propuesto y a continuación se explica la lógica detrás de la definición de cada entidad y sus relaciones.

De acuerdo al diseño planteado, el objeto central de análisis es el **sujeto**, el que se identifica por una serie de características personales y que, además, posee una **imagen** de texto asociada. Ahora bien, debido a la naturaleza del procesamiento (con diversas etapas y distintos grados de automatización), un sujeto puede (y en la mayoría de los casos ocurre) tener más de una imagen relacionada, donde una de ellas es la base original y el resto corresponde al resultado de procesamientos intermedios. Por este motivo, la información del tipo de **hoja**

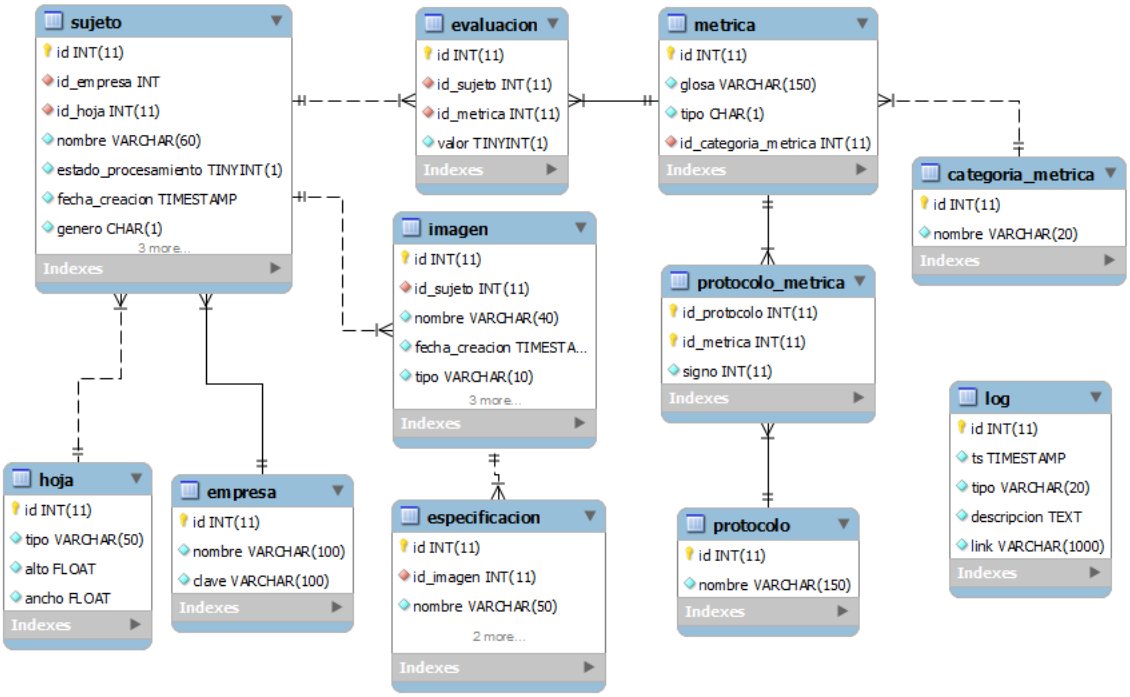


Figura 3.11: Diagrama entidad relación para el modelo de datos propuesto.

(carta, oficio, etc) usada en el análisis está asociada al sujeto y no a la imagen (para evitar duplicidad de información).

Por otro lado, una imagen está caracterizada por atributos que le son propios (formato, nombre de archivo, tipo, etc) y además posee una especificación asociada. Una **especificación** corresponde a un archivo de texto en formato JSON que contiene la serialización de la imagen junto a sus metadatos (provenientes del procesamiento en *backend*). Cabe destacar que tanto estos archivos de especificación, así como los archivos de imagen, no son almacenados dentro de la base de datos relacional, sino que lo que se guarda es su información para ser accedidos directamente en disco.

Ahora bien, en el diseño también se presenta una entidad **métrica**, que tal como su nombre lo sugiere, busca representar cada una de las métricas que deben ser extraídas de una imagen durante el proceso de análisis. Cada una de estas métricas puede ser clasificada de dos formas. Por una parte, cada una posee una **categorización** de acuerdo a aquello que mide en la imagen. Por ejemplo, la métrica **cohesión ligada** se puede categorizar dentro del conjunto de métricas referidas a **medición de espaciamientos**. Por otro lado, cada una de estas características medibles, se pueden categorizar según el o los **protocolos** que puede integrar. A modo de ejemplo, la métrica ya mencionada puede servir para reconocer simultáneamente el protocolo *Responsabilidad* y *Tolerancia a la frustración*.

Pensando en la extensibilidad de la solución, el hecho de tener entidades separadas para representar métricas y protocolos, permite una fácil integración de nuevas métricas y nuevos protocolos, así como también la posibilidad de modificar la influencia de ciertas medidas en determinadas características.

Adicionalmente, considerando la necesidad de asociar resultados del análisis de una imagen sobre un sujeto, se cuenta con una entidad **evaluación** cuyo objetivo es representar los valores obtenidos por un sujeto en cada una de las métricas de análisis, de manera de centralizar en una sola tabla toda la información necesaria para computar los niveles de logro según protocolos y poder generar el reporte final.

Por último, cabe destacar la existencia de una entidad **empresa** que además de servir como identificación de uno de los atributos de un sujeto, cumple el rol de almacenar la información de los usuarios del sistema de *frontoffice*, ya que el diseño asume que los clientes de dicho sistema serán justamente clientes empresa que carguen sujetos de su entorno en la plataforma.

3.3. Elección de tecnologías

Debido a la complejidad de la solución y a la gran cantidad de componentes que posee, la elección de tecnologías que faciliten su construcción es crucial. Por este motivo, se decide utilizar una serie de bibliotecas, *frameworks* y lenguajes detallados a continuación.

Para la plataforma de *backend* se considera utilizar la biblioteca OpenCV en su implementación en C++ debido a tres razones principales. En primer lugar, contiene una gran cantidad de funcionalidades implementadas para la representación y el tratamiento computacional de imágenes. En segundo término, la cantidad de documentación disponible es muy superior al de otras iniciativas similares (como por ejemplo sci-kit) gracias a la gran comunidad que la respalda. Por último, cabe destacar que su publicación es bajo licenciamiento BSD, lo que se traduce en un uso libre, tanto para fines académicos, como comerciales (en contraposición a *toolkits* de MATLAB, por ejemplo).

En línea con lo anterior, se elige desarrollar dicha plataforma utilizando el lenguaje de programación C++, principalmente por motivos de compatibilidad con la biblioteca, por la alta eficiencia que permite alcanzar y por soportar el paradigma de orientación a objetos, lo cual permite plasmar el diseño propuesto en una implementación concreta.

Para la implementación del modelo de datos, es posible utilizar cualquier motor de bases de datos relacionales disponible en el mercado. Sin embargo, por temas de licenciamiento y simpleza de uso, se decide utilizar MySQL, pudiendo ser reemplazado de manera transparente por otro motor de la misma línea como PostgreSQL.

Para el desarrollo de los dos sistemas de *frontend*, basados en una arquitectura web, se plantea utilizar el *framework* de desarrollo CodeIgniter debido a su facilidad de uso, portabilidad e integración del paradigma MVC, en línea con el diseño planteado. En consecuencia, el lenguaje utilizado para la implementación de la capa lógica de los sistemas web es PHP. Para la capa de presentación, y pensando en lograr una experiencia de usuario fluida, se escoge utilizar el *framework* Bootstrap en conjunto con la biblioteca de JavaScript jQuery.

Capítulo 4

Implementación de Backend

El núcleo del sistema *backend* radica en los algoritmos utilizados para procesar las imágenes y obtener las métricas requeridas. El resto de funcionalidades, tales como escritura y lectura de archivos de disco, serialización e interpretación de archivos de especificación, son abordadas utilizando técnicas y procedimientos convencionales, por lo que no son detalladas en el presente documento.

Por lo tanto, a continuación se presenta una descripción de los principales algoritmos implementados dentro de la aplicación, identificando entre paréntesis el nombre de la clase que lo encapsula, de acuerdo al diseño propuesto en la Sección 3.2.2.

4.1. Erosión (ImageErosion) y Dilatación (ImageDilation)

Los algoritmos de erosión y dilatación implementan las operaciones morfológicas conocidas por sus mismos nombres, según lo especificado en la Sección 2.1.3. Para ello, se usan las funciones provistas por OpenCV para tal efecto, conocidas por los nombres *erode()* y *dilate()*, respectivamente.

El elemento estructural de la operación morfológica puede ser especificado como un argumento del algoritmo. Sin embargo, en forma estándar se utiliza un elemento rectangular de dimensiones 3×3, en línea con lo propuesto en Gatos et al. [7].

4.2. Remoción de líneas de ruido (VerticalLineRemover)

Mediante la observación de numerosas hojas de texto manuscrito escaneadas, es posible notar que en muchas de ellas aparecen líneas verticales u horizontales en los límites de la

hoja, que introducen ruido al momento de efectuar el procesamiento de la imagen. Por este motivo, se propone un método de eliminar dichas líneas de la imagen.

En consecuencia, para implementar este algoritmo se realiza un *thresholding* binario para obtener una imagen en blanco y negro y luego se utiliza una implementación de la transformada de Hough para la detección de líneas en ángulos rectos. Esta última es provista por OpenCV mediante la función *HoughLinesP()* y entrega como resultado una colección de puntos pertenecientes a las líneas que cumplen las características mencionadas.

Para finalizar, se colorean de blanco todos aquellos puntos pertenecientes a líneas de largo superior a 100 pixeles. De esta manera se evita eliminar trazos rectos que no sean ruido, sino que son trazos genuinamente realizados por el autor del texto.

4.3. Detección de palabras (WordDetectionByContours)

Tal como se destaca en Gatos et al. [7], la segmentación de una imagen de texto en las palabras que lo componen es un problema abierto de investigación. En este trabajo se propone utilizar un enfoque basado en la detección de contornos.

La idea general de esta aproximación se resume en el diagrama presentado en la Figura 4.1. Ésta consiste en la composición de distintos procedimientos intermedios que dan como resultado la detección buscada.

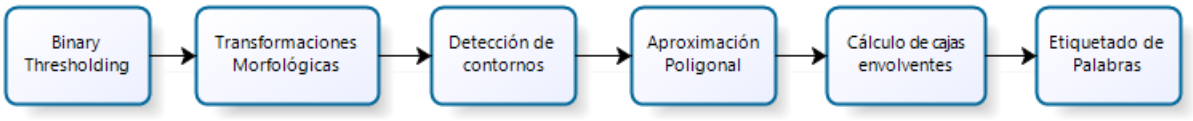


Figura 4.1: Proceso de identificación de palabras mediante detección de contornos.

En primer lugar, se realiza un *thresholding* binario sobre la imagen para simplificar la composición de la imagen y facilitar el procesamiento siguiente. Posteriormente se aplican sucesivas transformaciones de erosión y dilatación, con el objetivo de conectar los trazos al interior de las palabras y de separar los trazos de palabras muy cercanas dentro de la imagen. Luego se aplica un algoritmo de detección de contornos, para ello se utiliza la función *findContours()* provista por la librería, que implementa el algoritmo descrito en Suzuki and Abe [22] para estos fines. En la siguiente etapa, se realiza una aproximación poligonal de tercer orden, usando la función *approxPolyDP()* para reducir la cantidad de puntos por contorno. Finalmente, para cada nube de puntos aproximada, se encuentra su caja envolvente más pequeña y se etiqueta como una palabra.

En la Figura 4.2 se presenta un fragmento de imagen de texto y las transformaciones que sufre en cada una de las etapas. La zona A corresponde a la imagen original tras haber aplicado el proceso de binarización; en B se observa la imagen tras un proceso de erosión y uno de dilatación; en C se muestran los contornos detectados a partir del paso anterior; y en D se muestran los rectángulos envolventes de los contornos, dibujados sobre la imagen

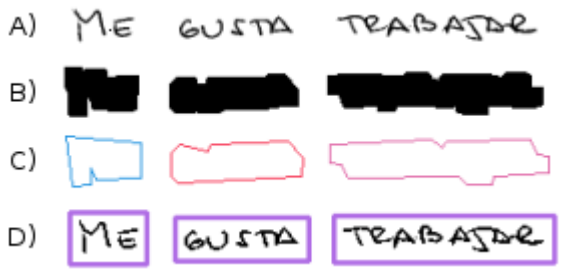


Figura 4.2: Visualización de las fases del proceso de detección de palabras.

original.

4.4. Identificación de líneas de texto (TextLineDetection)

En primer lugar, se debe entender como *línea de texto* al conjunto de palabras que componen una línea dentro de la escritura realizada por un sujeto. La identificación de una línea de texto puede ser abordada desde diversas perspectivas, siendo una de las más comunes en la literatura la generación de histogramas de proyección vertical y detección de *peaks* de frecuencia, de manera similar a la técnica presentada en la Sección 2.2. El problema de este enfoque es que requiere corregir las tendencias de inclinación del texto previamente o, de manera análoga, la generación del histograma debe ser proyectada en un cierto ángulo en vez de ser vertical.

Para hacer frente a este problema, y considerando que se tiene una detección de palabras satisfactoria, se propone un algoritmo de detección de líneas de texto que emule el proceso de identificación visual que realiza un humano. En general, una persona es capaz de identificar cuáles palabras pertenecen a una línea de texto gracias a dos principios. El primero de ellos tiene que ver con la distribución espacial que ocupan las palabras en el texto; un lector es capaz de reconocer la palabra superior ubicada más a la izquierda y continuar leyendo por las palabras que se encuentran más próximas hacia la derecha. El segundo, guarda relación con la semántica de las palabras y apunta al grado de coherencia que el lector es capaz de detectar entre dos palabras consecutivas. El algoritmo propuesto, se basa en el primero de estos principios.

Con el objetivo de evitar el problema de identificar la palabra superior ubicada más a la izquierda, se propone un enfoque levemente distinto, pero siguiendo la misma lógica. La idea es encontrar la palabra ubicada más arriba en el texto y luego encontrar la más cercana hacia la derecha y hacia la izquierda, tales que su diferencia en posición vertical sea menor a la altura de la palabra. Este proceso se repite recursivamente en cada palabra encontrada hasta que no existan más palabras que cumplan dicha condición, en ninguna de las dos direcciones. Una vez que esto ocurre, el conjunto de palabras es etiquetado como pertenecientes a la misma línea y se repite el procedimiento hasta que no queden elementos sin etiquetar.

Algoritmo 1 Identificación de líneas de texto

```
1: nl ← 1                                                    ▷ nl: Número de línea
2: while ∃ Palabras sin clasificar do
3:     p ← Palabra sin clasificar ubicada más arriba
4:     p.nl ← nl                                             ▷ Etiquetar palabra
5:     l ← Palabra más cercana ubicada a la izquierda de p
6:     while ∃ Palabras a la izquierda de l donde distancia vertical menor a altura(l)/2 do
7:         l.nl ← nl
8:         l ← Palabra más cercana a la izquierda de l que cumple condición
9:     r ← Palabra más cercana ubicada a la izquierda de p
10:    while ∃ Palabras a la derecha de r donde distancia vertical menor a altura(r)/2 do
11:        r.nl ← nl
12:        r ← Palabra más cercana a la derecha de r que cumple condición
13:    ln ← ln + 1
```

Concretamente, esta idea se puede generalizar en el pseudocódigo presentado en Algoritmo 1. Notar que este algoritmo funciona bajo el supuesto de que no existen dos palabras adyacentes pertenecientes a la misma línea de texto en que el límite superior de la palabra de la izquierda esté ubicado más abajo que el límite inferior izquierdo de la palabra siguiente.

4.5. Optimización en la identificación de líneas de texto (TextLine DetectionOptimization)

Empíricamente, es posible comprobar que el Algoritmo 1 falla en los casos de escritura con pendientes fuertes (sobre 5°) y en casos de textos cuyas palabras están escritas con una separación grande entre ellas (considerando grande como el tamaño de una palabra de más de dos letras).

Para hacer frente a este problema y mejorar el algoritmo anterior, se propone mejorar la clasificación ya realizada mediante un algoritmo inspirado en *k-Means*. Haciendo la analogía con dicho algoritmo, cada palabra corresponde a un elemento a clasificar, k corresponde a la cantidad de líneas de texto, los centroides son entendidos como las rectas de regresión de cada línea de texto y la distancia a los centroides se calcula como la distancia en línea recta desde el punto medio de una palabra hacia la recta de regressión.

En concreto, esta idea se presenta como pseudocódigo del Algoritmo 2.

4.6. Cálculo de márgenes (MarginsCalculator)

Una de las métricas que deben ser calculadas guarda relación con el tamaño de los márgenes. Por este motivo, es necesario implementar un algoritmo que permita satisfacer esta

Algoritmo 2 Optimización en la identificación de líneas de texto

```
1: n ← Cantidad de líneas de texto identificadas
2: repeat
3:     for i ← 1 to n do
4:         reg[i] ← Recta resultante de la regresión lineal de los puntos medios de las envolturas de las palabras pertenecientes a la línea i
5:     for all Palabras (w) en el texto do
6:         for j ← 1 to n do
7:             d[j] ← Distancia de w a reg[j]
8:         w.ln ← argmin_k d[k]
9: until No existen reasignaciones de palabras
```

necesidad. Éste tiene el objetivo de determinar los márgenes izquierdo y derecho de cada línea de texto, por lo tanto, espera recibir como entrada un conjunto de palabras ya clasificadas según la línea a la que pertenece. Su implementación se puede encontrar en forma de pseudocódigo en el Algoritmo 3.

Algoritmo 3 Cálculo de márgenes

```
1: for all Líneas de texto (tl) en la imagen (img) do
2:     wl ← Palabra ubicada más a la izquierda en tl
3:     rl ← Palabra ubicada más a la derecha en tl
4:     tl.lm ← wl.x                    ▷ tl.lm: Margen izquierdo de línea tl
5:     tl.rm ← img.w − (wl.x + wl.w)   ▷ img.w: Ancho de la imagen
```

4.7. Cuenta de palabras por línea (WordsPerLineCounter)

Otra de las métricas que deben ser calculadas de manera automática corresponde a la cantidad de palabras escritas por línea. El funcionamiento de este algoritmo es muy sencillo, pues toma como entrada una imagen con las palabras ya identificadas junto a su respectiva clasificación en líneas de texto y realiza el conteo requerido. Por lo tanto, su ejecución solo agrega un nuevo metadato a nivel de línea de texto, indicando la cantidad de palabras que la componen.

4.8. Cálculo de ángulos por línea (AngleLineCalculator)

La determinación de la inclinación de la escritura es otra de las características buscadas por un perito neuroescritural y que puede ser determinada de manera automática. En este

caso, el algoritmo también requiere como entrada la imagen con la identificación de palabras y de líneas de texto ya realizada.

Su funcionamiento consiste en que, para cada línea de texto, se calcula una regresión lineal sobre todos los puntos de las palabras que la componen. De esta forma, se logra capturar la tendencia de la escritura y luego se calcula el ángulo de esta recta respecto a la horizontal. Finalmente, este valor obtenido se agrega como un metadato adicional a la imagen.

4.9. Detección de líneas basales del texto (BaseLineDetection)

Se denomina línea basal del texto a aquella línea que marca la tendencia de los puntos inferiores del cuerpo central de cada palabra. Es decir, corresponde a una línea que puede ser trazada bajo las letras minúsculas que no tienen proyecciones inferiores (Ver Figura 2.4). Su detección es importante dentro del contexto de este proyecto, ya que permite calcular métricas a partir de ésta, tales como el tamaño de separación entre líneas y el tamaño del cuerpo central.

El algoritmo que implementa esta detección se compone de dos partes. En primer lugar, busca todos los puntos que cumplen con las restricciones mencionadas y que los hacen ser candidatos a componer la línea basal; luego, el algoritmo calcula una regresión lineal sobre los puntos detectados, de forma de obtener una recta que los aproxime.

Dada una palabra de dimensiones $w \times h$, con el sistema de referencia en el extremo superior izquierdo, creciente hacia la derecha y hacia abajo; en la primera fase del algoritmo, para cada palabra, se busca el conjunto de puntos $P = \{p_1, \ldots, p_N\}, N \leq w$ tales que cada $p \in P$, de coordenadas (i, j), satisface que p es un pixel negro y que $j \geq k$ para cualquier pixel negro q de coordenadas $(\mathrm{i}, k), k \in \{0, 1, \ldots, h\}$.

En la segunda fase, se calcula una regresión lineal sobre todos los puntos $p \in \cup P$ en todos los conjuntos encontrados de las palabras de cada línea de texto; obteniendo como resultado las coordenadas de inicio y término de cada una de estas rectas por línea de texto.

Formalmente, siguiendo la notación recién introducida, el pseudocódigo que implementa esta funcionalidad se puede observar en el Algoritmo 4.

4.10. Cómputo de proyección vertical (VerticalHistogramCalculator)

Siguiendo lo planteado en la Sección 2.1.3, se implementa un algoritmo capaz de cálcular la proyección vertical de la imagen, es decir, su histograma de distribución. Este procedimiento consiste en calcular, para cada fila de la imagen, la cantidad de pixeles negros presentes en

Algoritmo 4 Detección de líneas basales

```
1: for all Líneas de texto (tl) en imagen (img) do
2:     P ← {}                                          ▷ Lista de puntos
3:     for all Palabras (b) en tl do
4:         for i ← 0 to w do
5:             for j ← h to 0 do
6:                 if p(i, j).isBlack() then
7:                     P.push(p)
8:                     break
9:     r ← Regresión lineal sobre P
10:    tl.addData(r)
```

esa línea. Debido a la sencillez de su implementación, no se profundiza en los pasos necesarios para lograrlo.

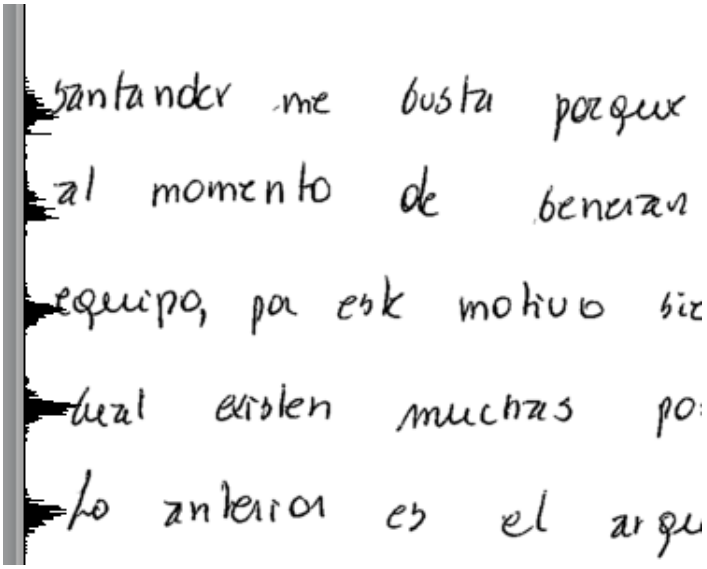


Figura 4.3: Visualización de la proyección vertical sobre un fragmento de texto.

En la Figura 4.3 se presenta el resultado de la visualización del cómputo de la proyección vertical sobre un fragmento de imagen de texto. En su zona izquierda se puede ver graficado el histograma de acumulación de pixeles negros por fila.

4.11. Cálculo de densidad por área (DensityCalculator)

Una cantidad considerable de las métricas que deben ser obtenidas guarda relación con el uso del espacio de la hoja que hace el sujeto. Por este motivo, calcular la densidad de pixeles negros dentro de la imagen de la hoja es relevante.

De acuerdo a lo expresado por los expertos, dividir la hoja en 6 partes (por la mitad de manera vertical y en tercios de forma horizontal), permite tener una granularidad adecuada

para realizar el análisis. Por este motivo, el algoritmo que implementa esta funcionalidad actúa contando la cantidad de pixeles negros existentes en cada una de las seis partes de la imagen. Luego, para normalizar de acuerdo a la cantidad escrita por la persona, cada uno de estos valores es dividido por el total de pixeles negros en la hoja. Finalmente, estos valores porcentuales son agregados a la imagen en forma de metadatos.

4.12. Detección de cuerpos por palabra (WordBodies-Detection)

La detección de los cuerpos de cada palabra es, junto a la detección de palabras y su clasificación, uno de los algoritmos de mayor complejidad en cuanto a desarrollo e implementación propuestos en este proyecto. En particular, la métrica que se quiere obtener a partir de esta detección es la descrita en la Sección 2.3, bajo el nombre de "Medidas de zonas de texto o tamaño de cuerpos".

La lógica subyacente consiste en realizar un procedimiento similar al de la detección de líneas basales, pero en el sentido inverso, para encontrar la línea superior. Sin embargo, este método presenta varias falencias que deben ser corregidas.

En primer lugar, la mayoría de las letras imprenta solo tienen un trazo en una misma columna, por lo que se identificaría el mismo punto tanto para la línea basal, como para la línea superior. En segundo lugar, no bastaría con limitar la búsqueda a una sección de la palabra (por ejemplo mitad inferior para la línea basal y mitad superior para la línea de dicha zona), puesto que en escritos con pendientes muy pronunciadas, habría secciones de la palabra para las cuales no se identificarían puntos y, por tanto, se perdería justamente la tendencia que se quiere capturar. Por último, otro problema surge en palabras que poseen una gran cantidad de letras con proyecciones inferiores y es que la detección de estos puntos "empujarían" la línea del límite inferior del cuerpo central hacia abajo.

Teniendo todos estos puntos en consideración, se propone una modificación del algoritmo de detección de líneas basales de manera de incluir la posibilidad de adaptarse a las características de cada palabra.

A grandes rasgos, la solución propuesta consiste en hacer dos procedimientos análogos para detectar el límite superior del cuerpo central y el límite inferior del mismo, pero por simplicidad solo se explica este último. En concreto, lo que se busca es dividir la palabra en zonas de tamaño l y en cada una de ellas encontrar al menos n puntos inferiores. Sin embargo, esta búsqueda de puntos inferiores debe estar acotada a un área de las h/r filas de más abajo, siendo h la altura de la palabra.

Ahora bien, en caso de no encontrar una cantidad suficiente de puntos tal que se cumpla la restricción, el algoritmo iterativamente relaja la restricción del área de búsqueda, permitiendo la búsqueda en la zona de las $\mathrm{i} \times h/r$ filas, aumentando el valor de i en cada iteración. Luego, al concluir este procedimiento en cada una de las l zonas, se procede a eliminar una fracción x de los puntos más cercanos al extremo inferior en cada región, de manera de eliminar posibles

distorsiones.

Finalmente, se calcula una regresión lineal sobre los puntos resultantes y se determina que la recta obtenida corresponde a la línea inferior del cuerpo central de la palabra. Luego, para delimitar el cuerpo inferior, se encuentra el punto más bajo de toda la palabra y se proyecta una recta con la misma pendiente que la línea ya encontrada. Cabe destacar que este argumento supone que la pendiente se mantiene constante a lo largo de la escritura de una palabra.

El algoritmo luego debe iterar sobre cada una de las palabras de la imagen del texto, repitiendo el mismo procedimiento.

Teóricamente, este algoritmo tiene sentido y puede llegar a detecciones acertadas. Sin embargo, los valores de los parámetros n, l, h, r y x deben ser definidos y calibrados de manera adecuada. Para lograrlo en primera instancia, se realiza una calibración empírica, quedando abierta la posibilidad de utilizar métodos formales más sofisticados que permitan plantear y resolver un problema de optimización que maximice la calidad de la detección.

Para concluir, en la Figura 4.4 se muestra el resultado de aplicar este algoritmo ya calibrado sobre un fragmento de una imagen de texto. Las líneas rojas (las interiores) delimitan el cuerpo central y las azules (las exteriores) delimitan los cuerpos inferior y superior.

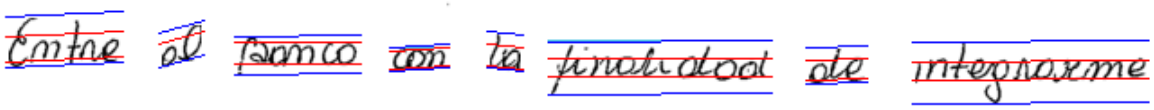


Figura 4.4: Aplicación de algoritmo de detección de cuerpos sobre una línea de texto.

4.13. Detección de desconexiones al interior de palabras (IntraWordsDisconnectionsCounter)

En teoría, una palabra escrita en letra manuscrita debería tener todas sus letras conectadas. Sin embargo, en la práctica esto suele no ser así debido a que la gente usualmente levanta el lápiz al escribir. Considerando este antecedente, el nivel de grado de conexión de las letras de una palabra, también constituye una métrica que debe ser calculada.

Si se mira este problema de manera generalizada, se puede notar que es análogo al problema de detección de palabras en imagen de texto, pero llevado a detección de componentes dentro de una imagen de una palabra.

Formalmente, y utilizando la terminología introducida en la Sección 2.1.1, el problema consiste en encontrar las componentes conexas de tamaño significativo dentro de la región de interés (la palabra) y obtener la cantidad de desconexiones como el número de componentes menos uno.

El algoritmo propuesto para resolver este problema es análogo al propuesto para la detección de palabras, pero obviando la etapa de las transformaciones morfológicas, para evitar conectar las componentes que se quiere identificar.

4.14. Número de letras en el texto (LettersCalculator)

Una de las métricas que se busca extraer, y que sirve como base para el cálculo de otras, corresponde al número de letras escritas en todas las palabras del texto. Debido a que para hacerlo de manera exacta se necesitaría detectar cada una de las letras, lo cual significa un alto grado de complejidad, se propone utilizar una aproximación.

Dado que mediante el algoritmo de detección de palabras se puede conocer con exactitud la cantidad de ellas, y considerando que en un texto del estilo de los analizados se encuentran más de 50 palabras, se propone calcular el número de letras como la cantidad de palabras multiplicado por la cantidad promedio de letras por palabra en español. Este promedio fue calculado de manera experimental sobre una novela en español, obteniendo un valor de $5,1$ letras por palabras (con una desviación estándar de $2,63$ letras).

Por lo tanto, el algoritmo propuesto solo debe contar la cantidad de palabras detectadas y hacer la multiplicación ya mencionada por el factor encontrado.

4.15. Cálculo de métricas compuestas

Además de los algoritmos previamente descritos, existe un conjunto de ellos enfocados en realizar cálculos sobre las detecciones realizadas y computar valores en otras métricas. Sin embargo, debido a que su funcionamiento es similar, no se explicará cada uno en detalle.

A modo de ejemplo, existe una métrica denominada *Espacio Inter-Línea (EIL)* que representa la relación entre el espacio de dos líneas consecutivas de texto y el tamaño promedio de los cuerpos centrales de éstas. Por lo tanto, el algoritmo que la implementa debe medir la distancia entre el límite de la zona inferior de una línea de texto y el límite de la zona superior de la siguiente a partir de la detección de cuerpos de las palabras. A continuación, debe calcular el promedio de tamaño de los cuerpos centrales a lo largo de ambas zonas centrales. Y, finalmente, debe computar la relación que describe la métrica, almacenando el resultado como un metadato asociado a la línea de texto.

Capítulo 5

Implementación de Frontend

De acuerdo al diseño propuesto en el Capítulo 3, el *frontend* de la aplicación está compuesto de dos sistemas interconectados: un sistema de *backoffice* a ser utilizado por el perito neuroescritural y administrador del mismo, y una plataforma de *frontoffice* a ser utilizada por un cliente del servicio. En esta sección se presenta el resultado de la implementación de ambas partes.

5.1. Implementación módulo Base Backoffice

Si bien en el diseño planteado no se explicita la construcción de un módulo base, éste corresponde al núcleo encargado de conectar los diversos componentes y de entregar funcionalidades adicionales para el funcionamiento del sistema, relacionadas con procedimientos de autentificación y de auditoría.

La primera vista que provee este módulo corresponde a un formulario sencillo de inicio de sesión (ver Figura 5.1) en que se solicita las credenciales para autorizar el ingreso al sistema. Una vez que el usuario completa este paso de manera satisfactoria, es redirigido a la pantalla de inicio de la aplicación.

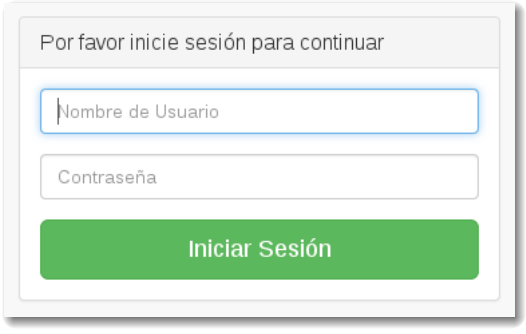


Figura 5.1: Formulario de inicio de sesión.

En la vista de inicio (ver Figura 5.2) es posible distinguir tres elementos principales. El

primero de ellos es el menú ubicado en el sector izquierdo, que da acceso a cada una de las secciones que componen la aplicación y que se detalla a continuación. El segundo es el área central se muestra un gráfico estadístico que representa la cantidad de procesamientos realizados sobre sujetos cargados en el sistema en los últimos meses. Por último, en el lado derecho de la región central, se muestra una tabla de las últimas acciones realizadas en el sistema, ya sean de tipo carga de usuario, ejecución de procesamiento, edición de resultados o generación de reportes. Este panel provee un acceso directo a cada una de las acciones realizadas y sirve para llevar un control de éstas.

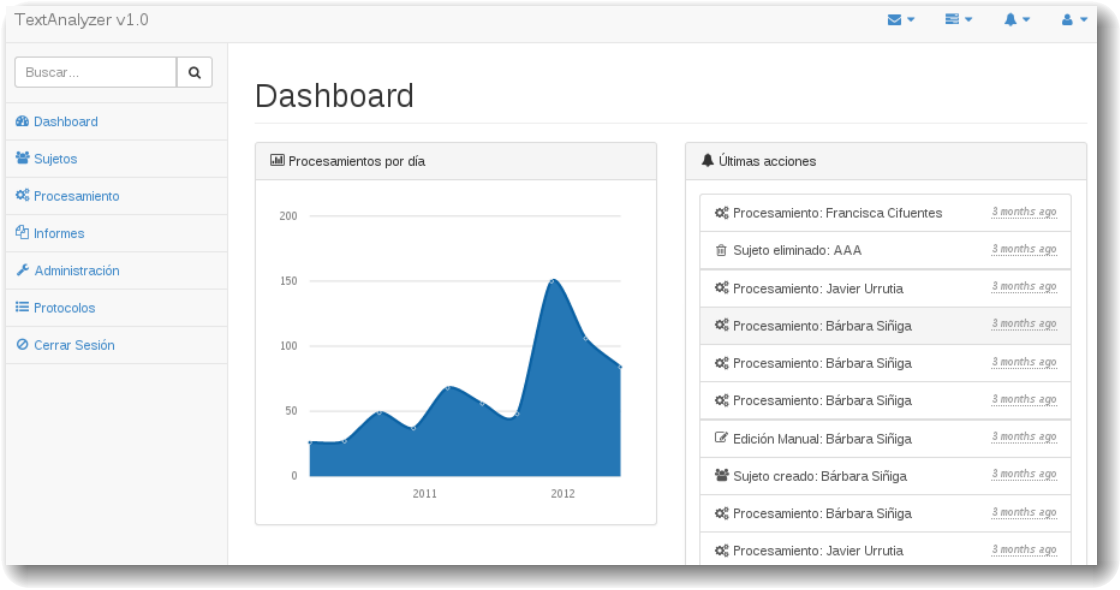


Figura 5.2: Vista de inicio de la aplicación de backoffice.

Cabe destacar que el menú lateral se mantiene a lo largo de toda la aplicación, por lo que en lo sucesivo solo se presenta imágenes de la región central del sitio, pues es la única zona que cambia según la sección que se esté visitando.

5.2. Implementación módulo Sujetos

Tal como se propone en el diseño del sistema, debe existir un módulo dedicado a la administración y mantención de los sujetos de estudio.

En la Figura 5.3 se presenta la vista principal que sirve este módulo. En ella se puede observar una lista tabulada de los sujetos cargados en el sistema, así como diversas acciones para cada uno de ellos. Mediante esta interfaz es posible eliminar un sujeto, consultar la imagen que tiene asociada y pasar directamente a la etapa de procesamiento de ella.

Además, en esta misma vista se provee un enlace de acceso hacia la funcionalidad de ingresar un sujeto en el sistema (mediante un botón ubicado en la parte inferior de la lista). La interfaz que da acceso a esta acción se presenta en la Figura 5.4, en el que se puede observar que se presenta un formulario con todos los datos requeridos para la carga del

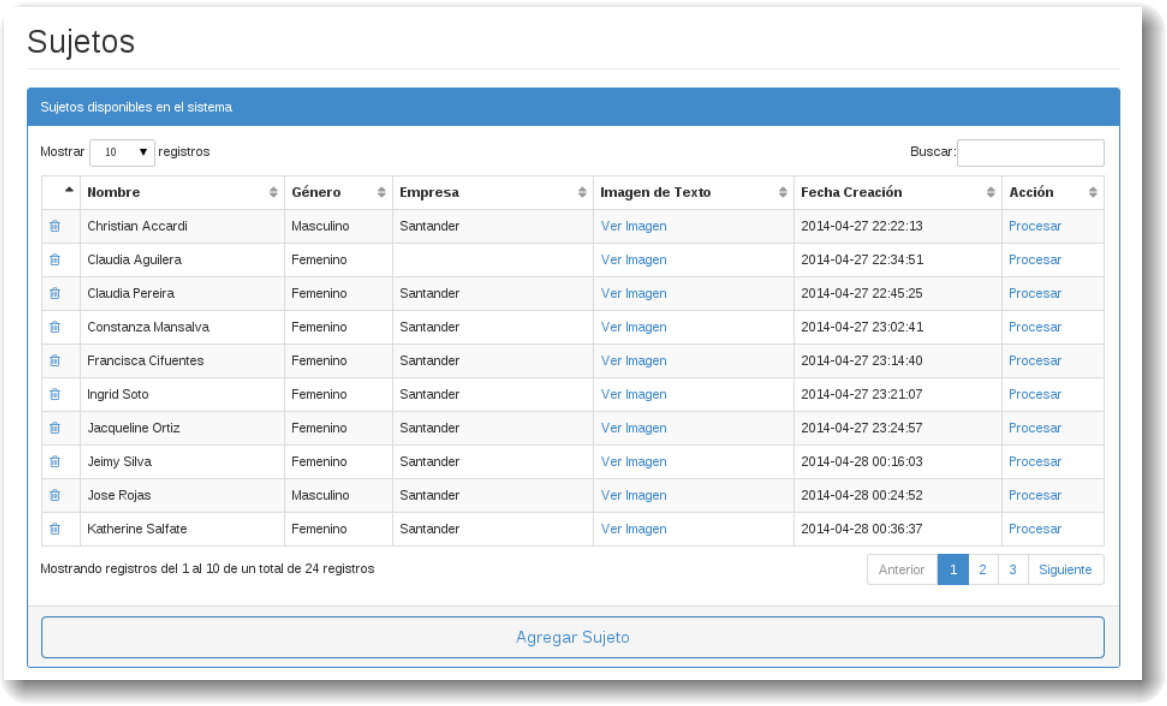


Figura 5.3: Vista principal del módulo Sujetos.

sujeto, incluyendo la imagen escaneada del texto escrito por la persona y la especificación de la hoja en que fue escrito.

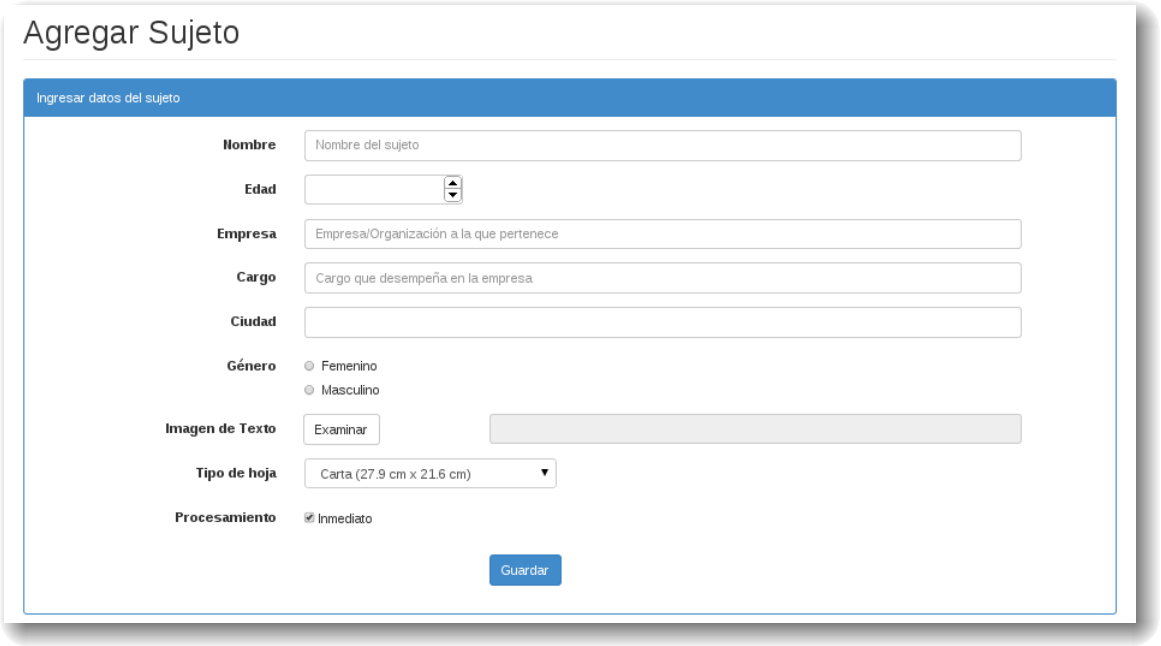


Figura 5.4: Vista con formulario para cargar un sujeto en el sistema.

Adicionalmente, al final de dicho formulario se presenta la opción de ejecutar el procesamiento de manera inmediata o de posponerlo para darle inicio de forma manual en otro momento.

5.3. Implementación módulo Procesamiento

El módulo de procesamiento es el que implementa la lógica de negocio relacionada con la ejecución de los análisis, proveyendo una interfaz de interacción con la plataforma de *backend*.

La vista de entrada que entrega este módulo se puede observar en la Figura 5.5, la cual tiene una estructura muy similar a la presentada por el módulo encargado de la administración de sujetos. En ella se despliega una lista de los sujetos cargados en el sistema junto a un enlace de acceso a la interfaz de procesamiento.

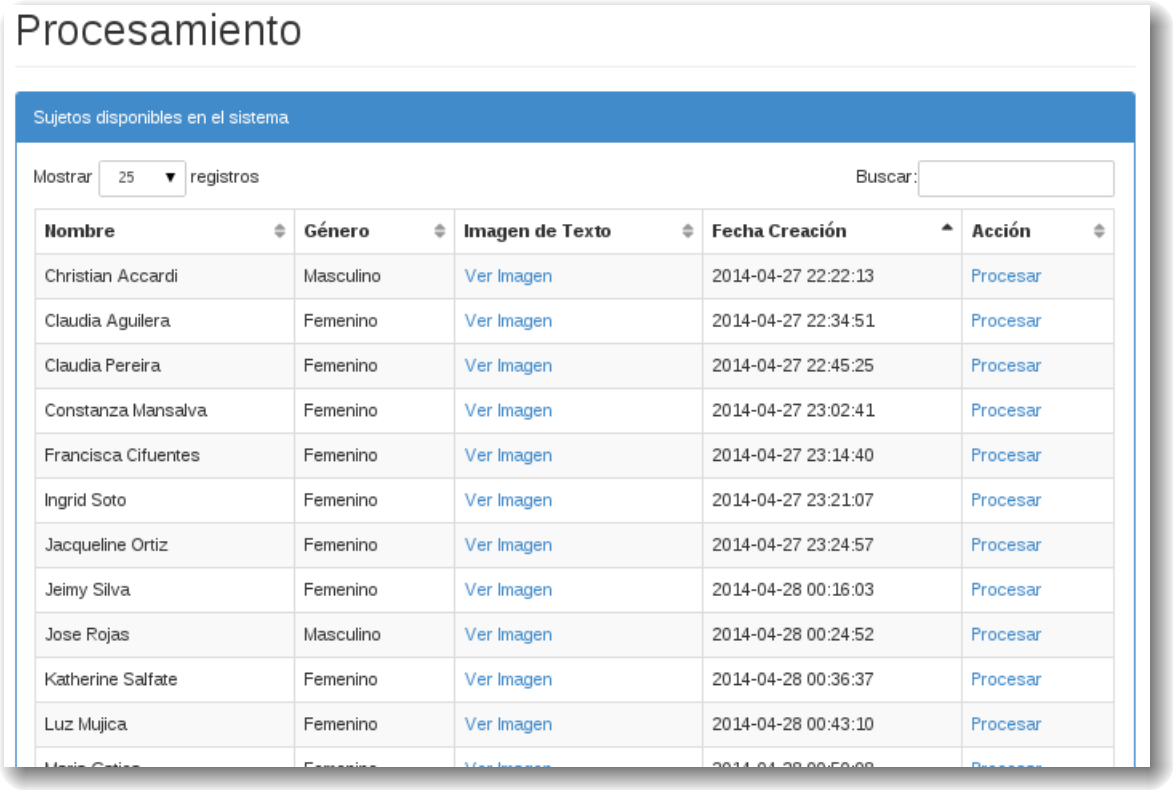


Figura 5.5: Vista de inicio del módulo de procesamiento.

Al acceder al procesamiento de un sujeto, se muestra la vista presentada en la Figura 5.6. En ella se presenta una tabla con la información de la persona, un mensaje con el estado del procesamiento asistido (que puede estar completo o incompleto) y una lista de las etapas inicial e intermedias del procesamiento automático.

Para cada una de las imágenes que componen las etapas de procesamiento, se despliega un contenedor como el mostrado en la Figura 5.7. En él se incluye un enlace para acceder a la edición manual de la identificación automática, otro para acceder a la interfaz de ejecución de procesamiento (a partir de la etapa en cuestión) y un tercero para acceder al reporte con las métricas extraídas.

Al acceder al área de edición manual, se presenta una interfaz interactiva en que se muestra la imagen con su identificación de palabras (en caso de existir) y la clasificación de líneas (determinada por el color que posee cada una de las cajas que envuelven a las palabras). En

Figura 5.6: Vista de procesamiento de un sujeto en particular.

Figura 5.7: Contenedor de las opciones de una etapa intermedia de procesamiento.

esta vista, el usuario es capaz de seleccionar una caja, cambiar sus dimensiones, eliminarla o asignarla a una nueva línea. En la Figura 5.8 se muestra un fragmento de esta vista, en que la primera palabra del texto escrito ha sido seleccionada para su modificación (notar que en el panel superior se muestra la información asociada a dicha palabra).

Por otro lado, al acceder a la interfaz de ejecución de procesamiento (enlace *Procesar*), se

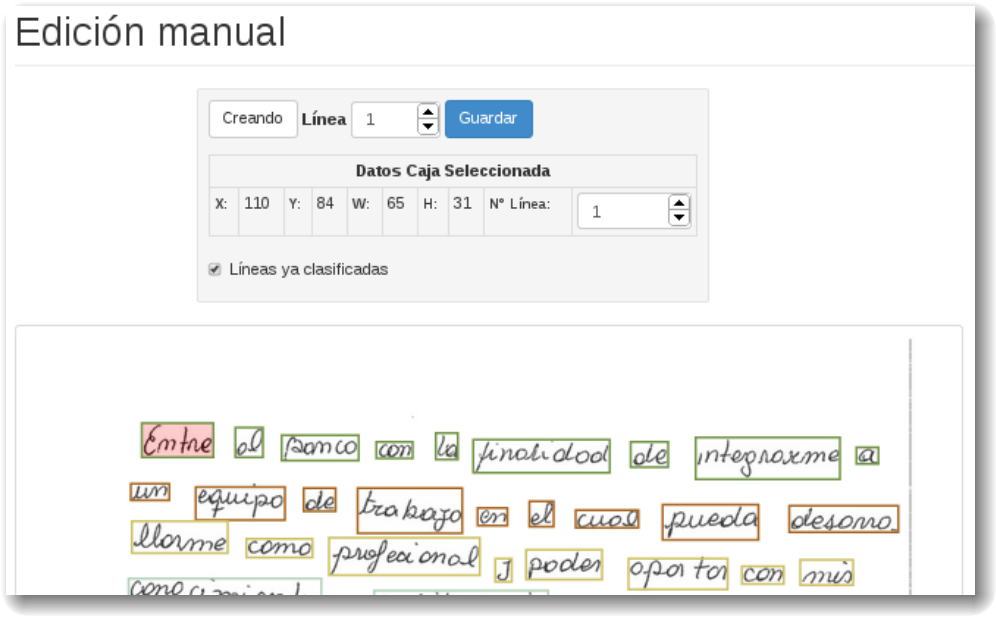


Figura 5.8: Vista de edición manual de la detección realizada automáticamente.

muestra la vista de la Figura 5.9. En ella, además de mostrar la imagen que será utilizada como base para el procesamiento, se proveen tres opciones de ejecución.

La primera opción de ejecución corresponde al procesamiento por defecto completo, que incluye la ejecución de algoritmos de perfeccionamiento de la imagen (remoción de líneas de ruido, binarización, etc), de detección de zonas de interés (palabras, líneas de texto, cuerpos basales, etc) y, finalmente, de cálculo de métricas.

La segunda opción corresponde sólo a la última fase de las anteriormente mencionadas y aplica sólo los algoritmos de cómputo de métricas. Para que esta ejecución sea exitosa, se requiere que la imagen basal que se utilice ya tenga realizada la detección de zonas, o, de lo contrario, no será posible calcular ninguna métrica.

Por último, se provee un pequeño formulario de procesamiento personalizado, que permite ordenar la ejecución de algoritmos en particular. Esta opción está pensada para ser utilizada por un usuario experto para realizar la calibración de los algoritmos o evaluar su efecto paso a paso, pero no se espera que sea usada al momento de querer hacer un análisis estándar. Por este motivo, al cargar la página, dicho formulario se muestra colapsado y no es visible para el usuario.

Finalmente, la otra interfaz que provee el módulo de procesamiento corresponde a la presentada en la Figura 5.10. Ésta tiene el objetivo de permitir al usuario realizar el análisis asistido que complete el proceso. Por lo tanto, se muestra un cuestionario que incluye las características que deben ser identificadas, agrupadas en pestañas según categorías para que puedan ser respondidas de manera eficiente.

Figura 5.9: Vista de ejecución de procesamiento con sus distintas alternativas.

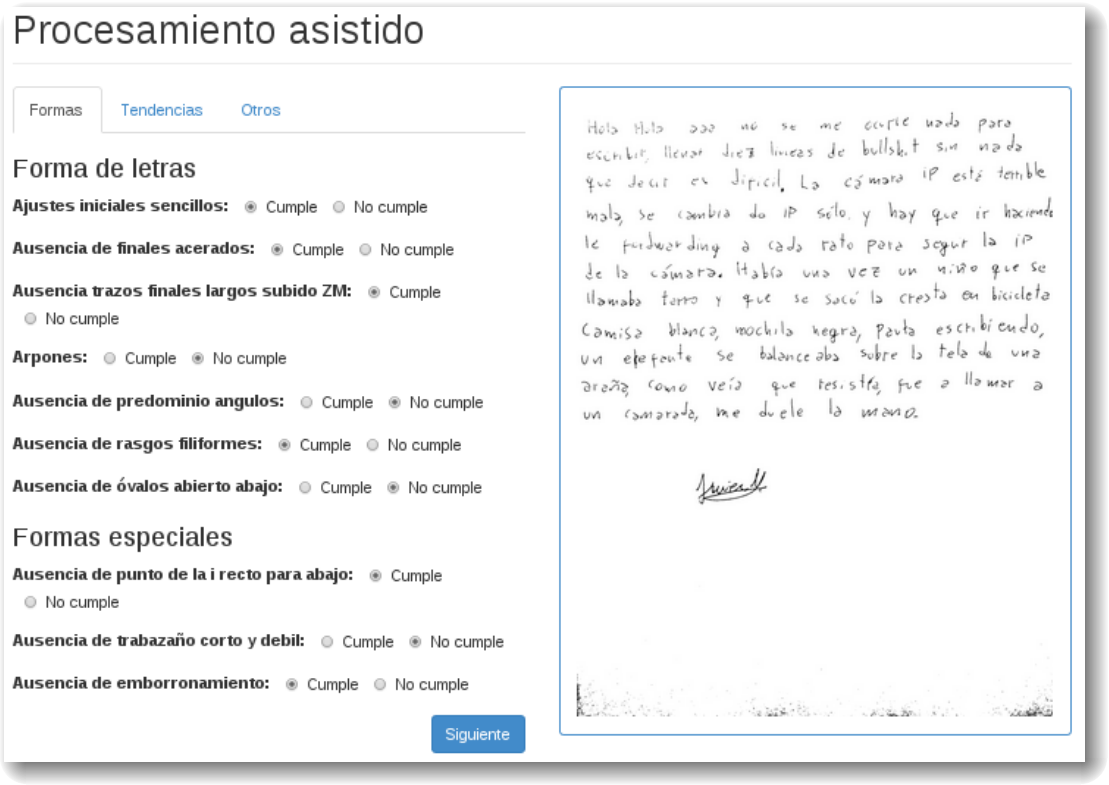
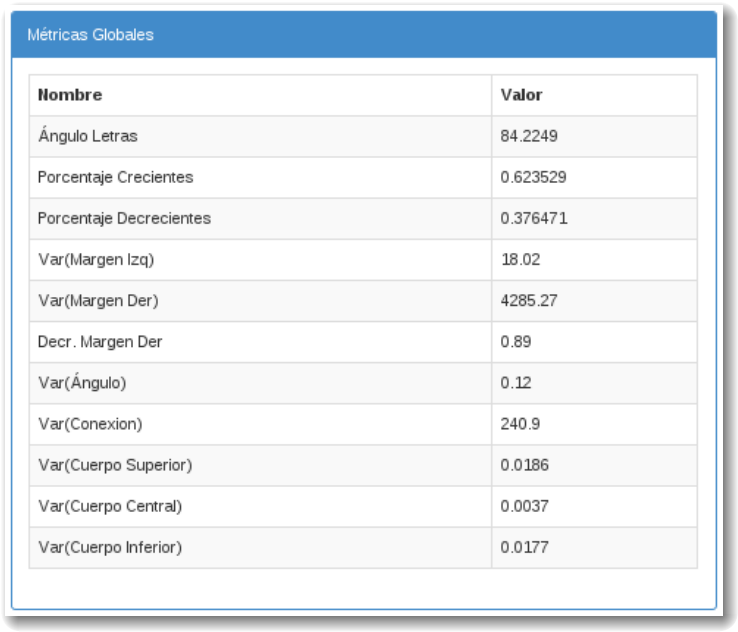


Figura 5.10: Vista de procesamiento asistido.

5.4. Implementación módulo Reportes de Métricas

El módulo de reportes tiene dos funcionalidades principales. Por una parte, obtiene los datos obtenidos como resultado del procesamiento y calcula promedios de acuerdo a las zonas del texto; y por otra, realiza la representación gráfica de los valores obtenidos para las métricas de interés.

En la Figura 5.11 se muestra un extracto de la vista en que se despliega una tabla con las métricas globales de la imagen, tales como ángulo promedio de inclinación de las letras, varianza en los márgenes, varianza en los tamaños de los cuerpos de las palabras y porcentaje de líneas con tendencia creciente y decreciente.


Métricas Globales

Nombre	Valor
Ángulo Letras	84.2249
Porcentaje Crecientes	0.623529
Porcentaje Decrecientes	0.376471
Var(Margen Izq)	18.02
Var(Margen Der)	4285.27
Decr. Margen Der	0.89
Var(Ángulo)	0.12
Var(Conexion)	240.9
Var(Cuerpo Superior)	0.0186
Var(Cuerpo Central)	0.0037
Var(Cuerpo Inferior)	0.0177

Figura 5.11: Tabla de métricas globales en vista de reporte.

Por su parte, en la Figura 5.12 se muestra la representación gráfica del cómputo de métricas por zonas de texto, además del cálculo de los promedios globales para los valores que tiene sentido calcularlos. Ejemplos de estas métricas son razón de espaciamiento inter-línea (EIL), razón de espaciamiento intra-palabras (EIP), promedio de margen izquierdo y promedio de inclinación de líneas.

Adicionalmente, en la Figura 5.13 se muestra el valor de algunas métricas representadas gráficamente de acuerdo a un desglose por líneas. En la imagen, se puede observar un gráfico de la cantidad de palabras por línea de texto y otro que representa el tamaño de los cuerpos (inferior, central y superior) promediado por línea.

Por último, en esta vista general de reporte se presenta un panel (ver Figura 5.14) que resume el estado del procesamiento automático y del procesamiento asistido, de tal manera que si ambos están completos, es posible generar el informe de protocolos asociado al sujeto.

Figura 5.12: Visualización de métricas computadas por zonas del texto.

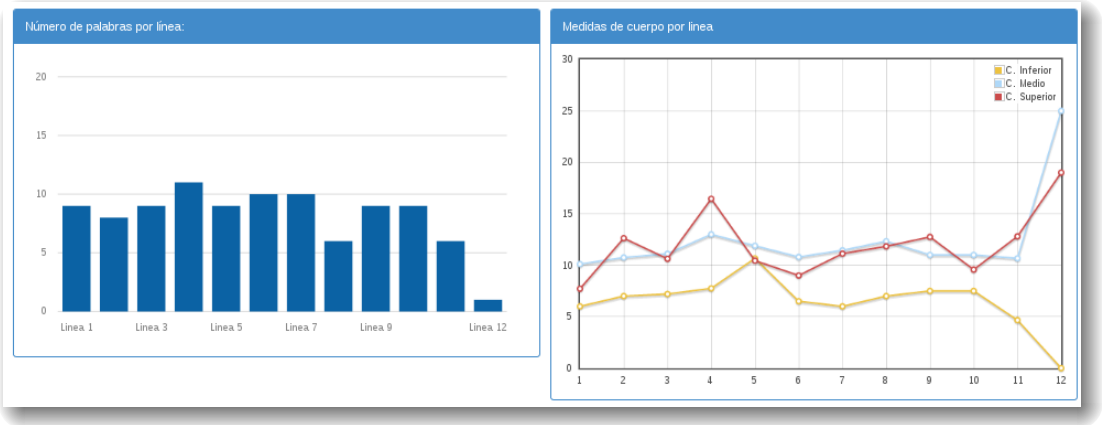


Figura 5.13: Gráfico de métricas desglosadas por línea de texto.

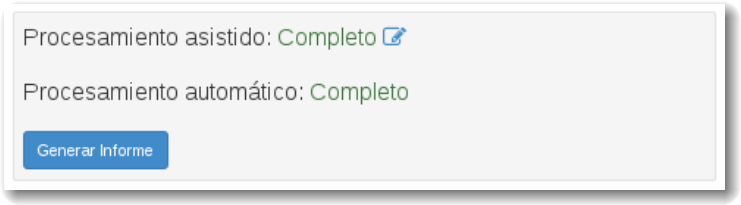


Figura 5.14: Panel con estado del procesamiento.

5.5. Implementación módulo Informes

El modulo de informes tiene por objetivo generar el documento final que resume los niveles de logro del sujeto de acuerdo al uso de los protocolos definidos. Sin embargo, para poder generar dicho documento resumen, es necesario realizar un procedimiento previo.

La aplicación de los protocolos de evaluación se basa en características que se construyen a partir del valor de las métricas. Sin embargo, esta transformación no es directa.

De acuerdo a la información que utilizan los expertos neuroescriturales, es posible determinar cuáles métricas influyen en cada una de las características de los sujetos, pero los umbrales requeridos para determinar el cumplimiento (o no) de cada una de ellas, es distinto al usado manualmente.

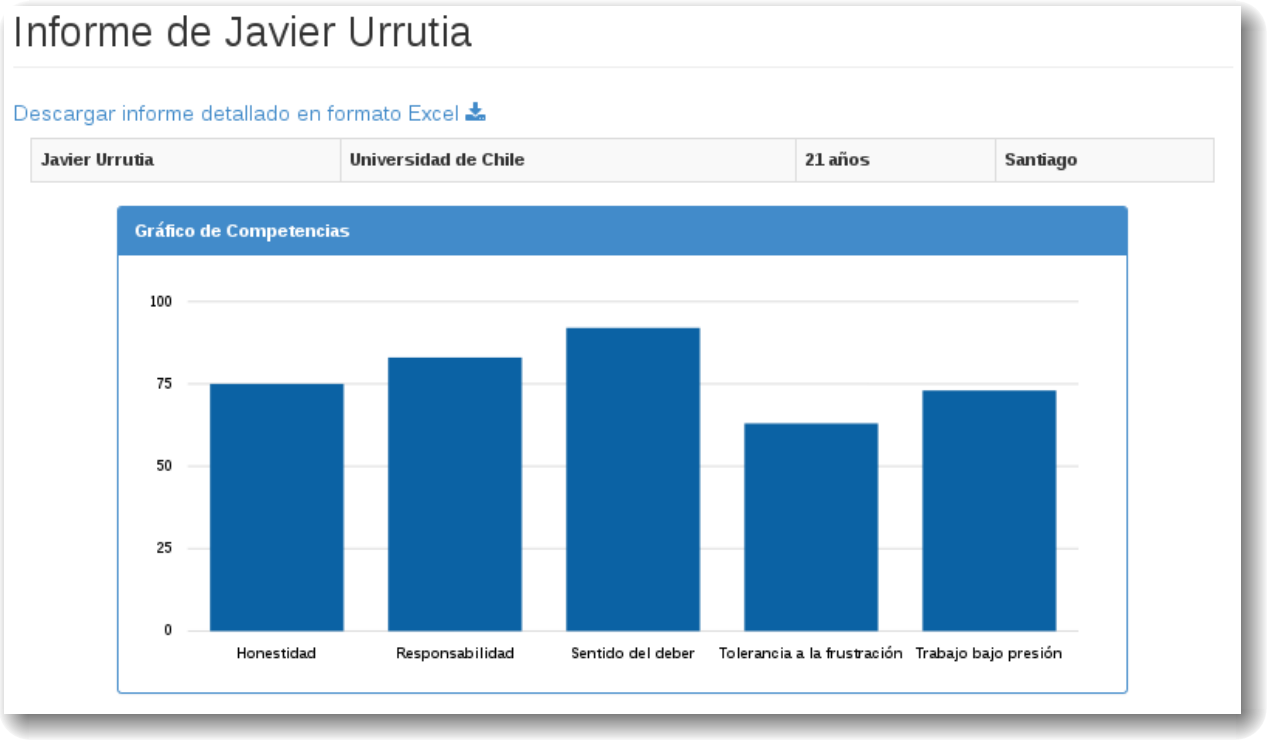


Figura 5.15: Gráfico de niveles de logro en informe de sujeto.

A modo de ejemplo, para el protocolo *trabajo bajo presión*, se debe considerar la característica *margen izquierdo regular*. A partir del conocimiento de los expertos, es posible determinar que esta característica se evalúa como *cumple* o *no cumple* a partir de la varianza en el margen izquierdo. Sin embargo, la interrogante a responder es cuál es el valor de dicha varianza que determina el umbral entre la regularidad y no regularidad.

Esta problemática surge en cada una de las características a detectar. Luego, para definir umbrales adecuados y consistentes con el procesamiento manual, la metodología que se utiliza consiste en calcular la métrica relacionada para cada una de las imágenes de muestra que se posee. Posteriormente, a partir de la evaluación entregada por el perito en cada caso, se

ajusta el umbral de la métrica a un valor tal que se maximice la cantidad de casos clasificados correctamente.

Cabe destacar que realizar este procedimiento de ajuste basado en las métricas que los peritos utilizan para las características, garantiza que se use una relación concreta que emule el análisis manual y evita el fenómeno de *overfitting* o sobreajuste para los casos en evaluación.

Las partes del documento final generado se presenta en las Figuras 5.15 y 5.16. En ellas se muestra un gráfico que resume los niveles de logro de manera porcentual para cada uno de los protocolos. Posteriormente, se presenta una clasificación de cada nivel de logro, junto a una descripción estándar de lo que significa alcanzar dicho nivel.

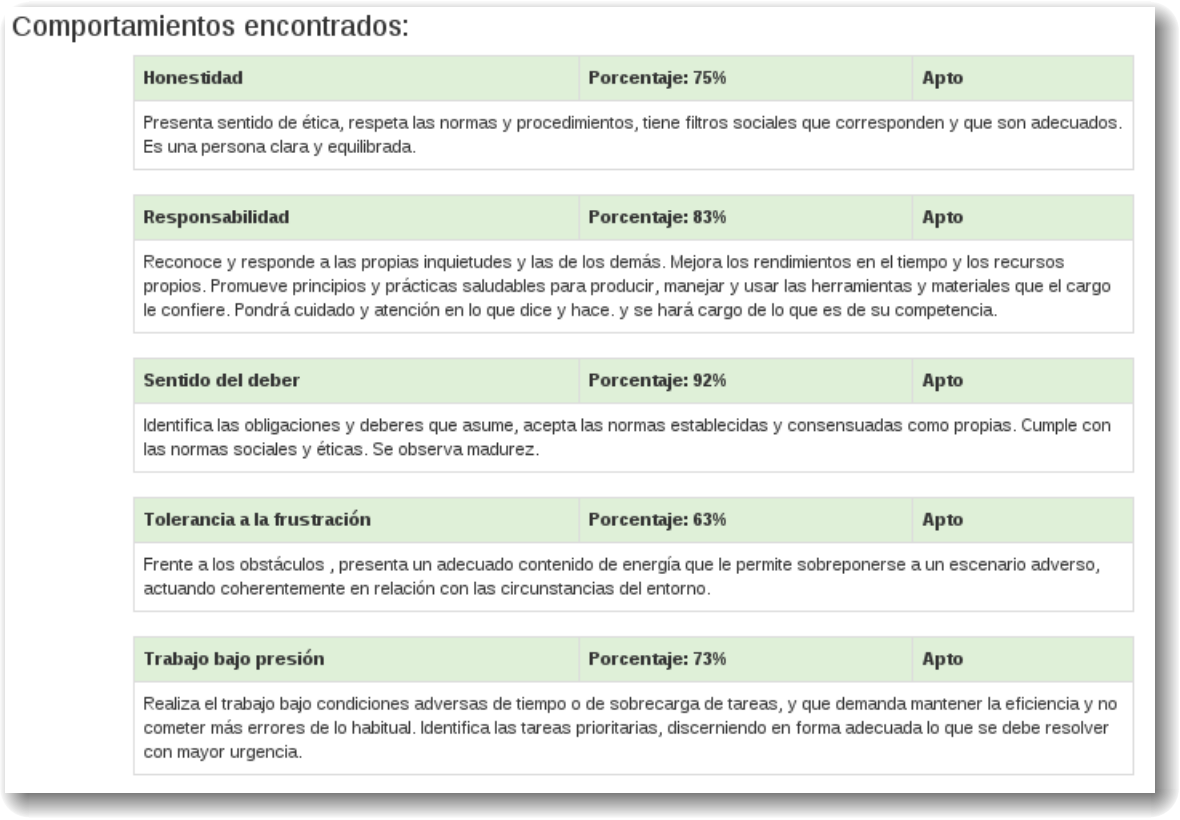
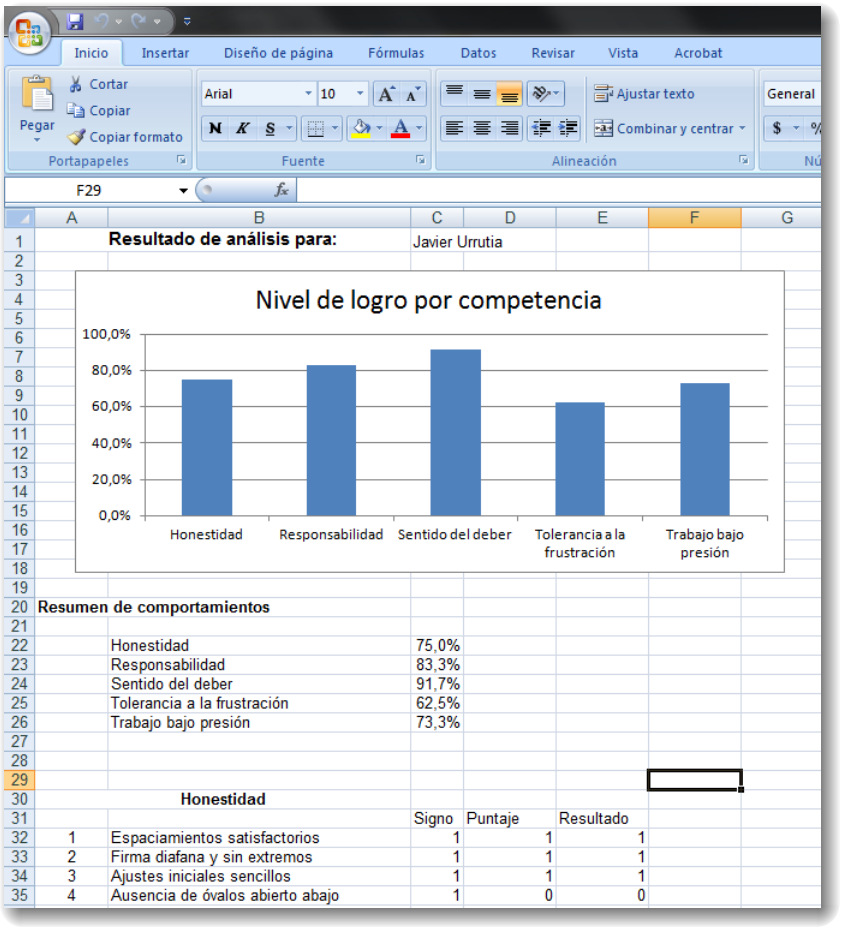


Figura 5.16: Tabla resumen con descripción de los niveles de logro.

Tal como se puede apreciar en la Figura 5.15, existe un enlace para generar un informe detallado en formato *xlsx* (compatible con Microsoft Excel). Al accederlo, se genera un archivo en el mencionado formato que tiene el detalle del resultado de la evaluación de cada métrica asociada a cada protocolo, además de un gráfico que resume los niveles de logro del individuo. Una vista parcial de este archivo se presenta en la Figura 5.17. La importancia de contar con este archivo, radica en que posee exactamente la misma estructura que los archivos que usan los evaluadores actualmente, por lo que facilita la comparación y adpatación al uso de este *software*.

Por último, este módulo también provee una interfaz que despliega una lista con todos los sujetos que tienen un informe disponible y provee un enlace de acceso directo a éste.



1		**Resultado de análisis para:**	Javier Urrutia			
20	**Resumen de comportamientos**					
22		Honestidad	75,0%			
23		Responsabilidad	83,3%			
24		Sentido del deber	91,7%			
25		Tolerancia a la frustración	62,5%			
26		Trabajo bajo presión	73,3%			
30		**Honestidad**				
31			Signo	Puntaje	Resultado	
32	1	Espaciamientos satisfactorios	1	1	1	
33	2	Firma diafana y sin extremos	1	1	1	
34	3	Ajustes iniciales sencillos	1	1	1	
35	4	Ausencia de óvalos abierto abajo	1	0	0	

Figura 5.17: Vista parcial del informe detallado en formato *xlsx*.

5.6. Implementación módulo Administración

El módulo de administración implementa una lógica típica de sistema de usuarios dentro de una arquitectura web y, además, se hace cargo de la mantención de las relaciones entre métricas y protocolos.

El sistema de usuarios posibilita crear cuentas de acceso a la plataforma de *frontoffice*, lo cual es vital para poder ofrecer el análisis de sujetos como un servicio. De esta forma, cada sujeto que sea cargado al sistema por un usuario creado por este medio queda automáticamente vinculado a la empresa correspondiente, permitiendo luego realizar un filtrado por este atributo. Una vista de la pantalla inicial del sistema de administración de usuarios puede verse en la Figura 5.18. En ella se listan los perfiles creados, teniendo la posibilidad de editar su información, eliminarlos o agregar nuevos.

Por su parte, en la Figura 5.19 se muestra el formulario de creación de un nuevo usuario. Si bien en esta primera iteración se mantenienе de manera sencilla, se ha considerado que en el futuro puede ser necesario incluir información adicional de los usuarios.

Por otro lado, la administración de protocolos consiste esencialmente en crear asociaciones

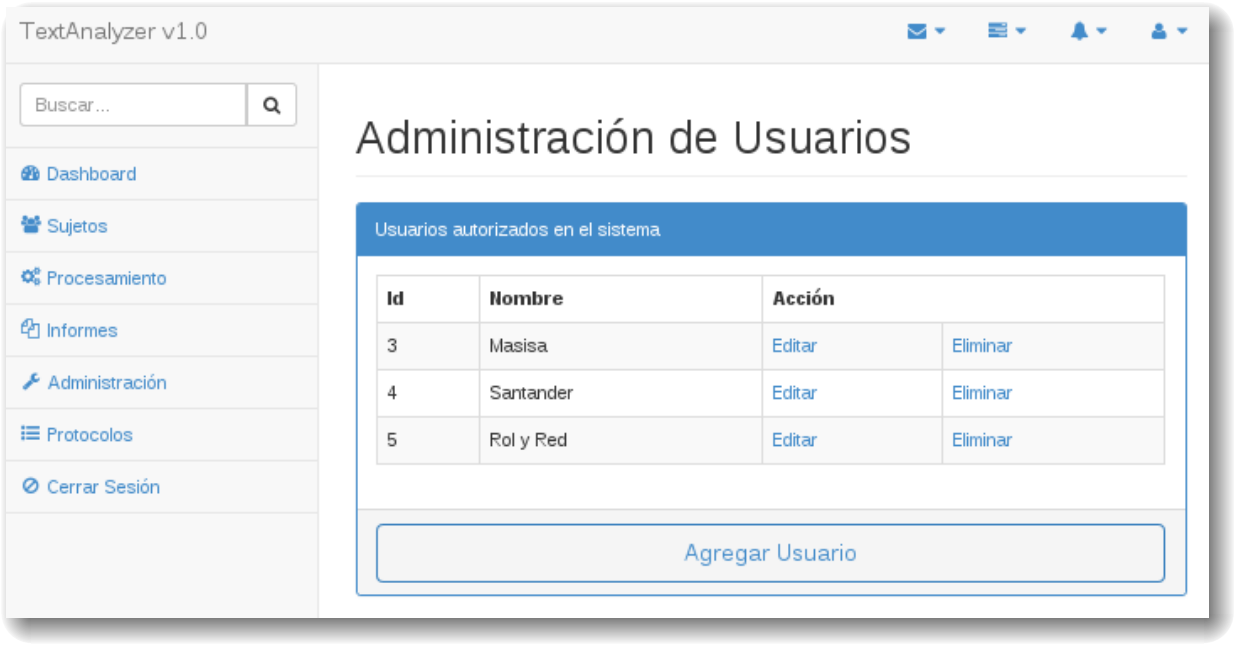


Figura 5.18: Vista principal de administración de usuarios.

Figura 5.19: Vista de creación de nuevo usuario.

entre métricas soportadas por el sistema y criterios de evaluación. Una vista de la pantalla principal de esta subsistema de presenta en la Figura 5.20. En ella se puede observar una lista de los protocolos actualmente definidos, que corresponden justamente a los usados actualmente por los peritos escriturales.

Al momento de agregar o editar un protocolo de evaluación (ver Figura 5.21), se muestra una lista de cada una de las métricas que soporta el sistema actualmente, detallando de qué tipo es (manual o automática), el tipo de influencia que genera en el protocolo (directa o inversa) y si está incluida o no en el protocolo. Cabe destacar, que la correcta definición de estos protocolos, incide directamente en la generación de los informes de evaluación. Por lo tanto, contar con esta interfaz de administración es de vital importancia para el usuario del

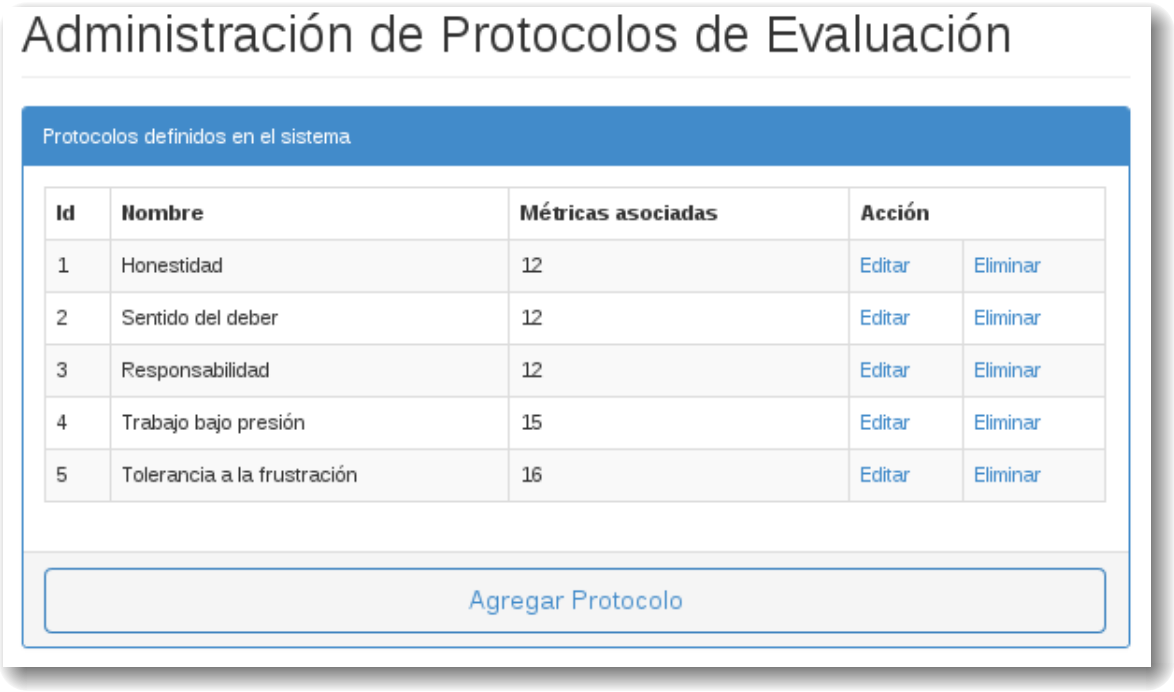


Figura 5.20: Vista principal de administración de protocolos.

sistema, puesto que le permite introducir nuevos protocolos en caso de integrar nuevas formas de evaluar y mejorar los ya existentes, ante cambios en la metodología de análisis.

5.7. Implementación de Frontoffice

La plataforma de *frontoffice* es, a grandes rasgos, una versión acotada del sistema de *backoffice*. De hecho, la implementación de los módulos propuestos en el diseño es compartida entre ambos sistemas.

Además de la vista de inicio de sesión, que es igual a la del otro sistema, éste provee una interfaz que muestra una lista de los sujetos disponibles para esa empresa (Ver Figura 5.22), ya sea que hayan sido agregados por el mismo usuario o por el perito desde la interfaz de *backoffice*. Sin embargo, esta vista es levemente distinta a la ya presentada, puesto que, además de algunos datos generales del sujeto, muestra el estado de análisis que posee (que puede ser *Completo* o *Pendiente*). En el caso de los análisis ya completos, se provee un enlace para consultar su informe, que es exactamente igual al presentado en las secciones precedentes.

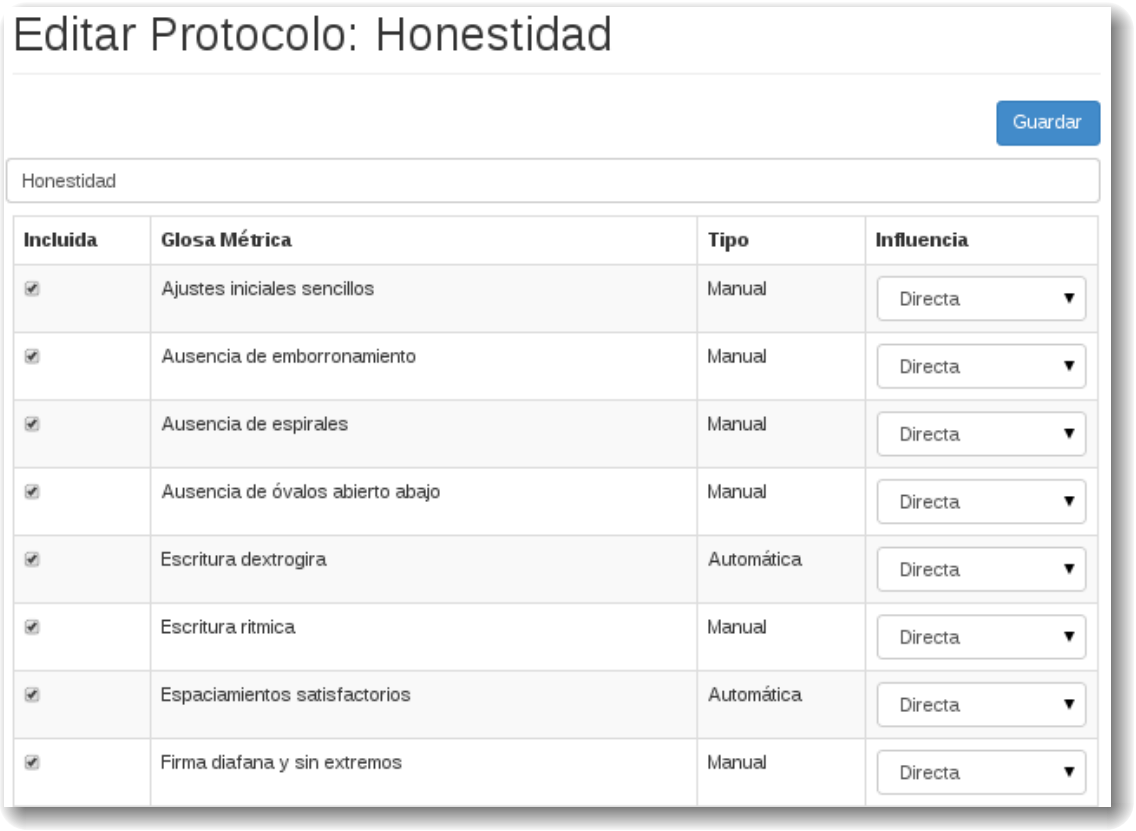


Figura 5.21: Vista de edición de un protocolo de evaluación.

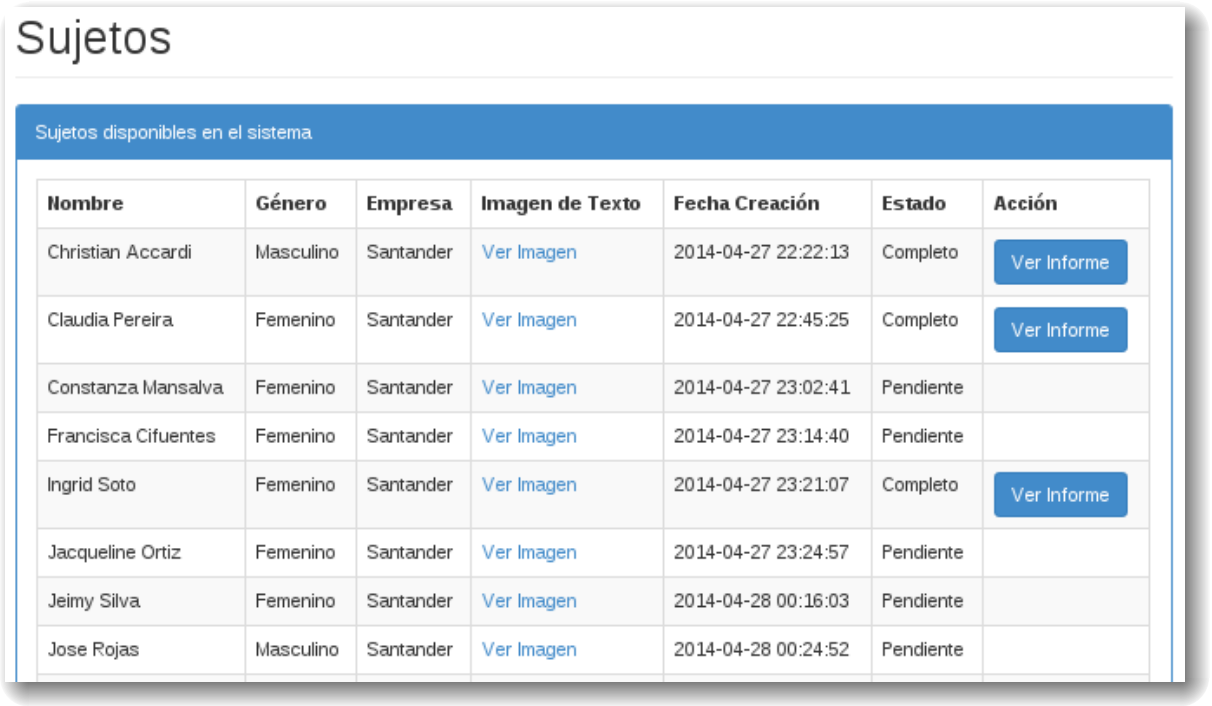


Figura 5.22: Vista de inicio del sistema del cliente.

Capítulo 6

Evaluación

El presente proyecto se sustenta en el objetivo de automatizar una tarea que actualmente se realiza de manera manual. Sin embargo, para que ésta sea realmente efectiva, es necesario evaluar su nivel de precisión respecto a la alternativa tradicional.

Para llevar a cabo esta tarea, se cuenta con un conjunto de 15 muestras, que ya han sido analizadas por un perito neuroescritural, junto a sus respectivos informes. Por lo tanto, lo que se busca es contrastar el resultado obtenido mediante la utilización del sistema aquí propuesto frente al análisis manual, en distintos niveles.

El primer nivel corresponde a una evaluación global, el segundo corresponde a una evaluación general por sujeto y, por último, se realiza una evaluación particular por métrica automatizada.

6.1. Evaluación global

En una primera instancia se busca realizar una evaluación general a partir de los rangos de clasificación de cada una de las competencias, para cada uno de los sujetos en análisis, calculando para cada una de ellas, métricas de efectividad de los métodos.

Para ello se debe recordar que mediante la aplicación de protocolos neuroescriturales, se realiza un cálculo de nivel de logro que puede tomar un valor de 0 % a 100 % para cada una de 5 competencias (honestidad, responsabilidad, sentido del deber, tolerancia a la frustración y trabajo bajo presión). Luego, según el porcentaje obtenido, cada nivel de logro es clasificado como *apto* (A), *apto con reparos* (R) o *no apto* (N).

Considerando estos antecedentes, en la sección Anexos B.1 se presenta la clasificación realizada tanto por el perito, como por el sistema, para cada una de las 5 competencias en evaluación. A modo de síntesis, en la Tabla 6.1 se presenta la cantidad de casos presentes en cada competencia (identificadas como $C1$, $C2$, $C3$, $C4$ y $C5$).

	C1		C2		C3		C4		C5	
	Real	Pred	Real	Pred	Real	Pred	Real	Pred	Real	Pred
A	15	15	15	15	15	14	13	13	13	13
R	0	0	0	0	0	1	1	2	2	2
N	0	0	0	0	0	0	1	0	0	0

Tabla 6.1: Clasificaciones reales y predichas en cada competencia

Como se puede observar, la muestra se encuentra sesgada hacia los casos *apto*, por lo que en caso de obtener una evaluación satisfactoria del sistema, solo permitiría comprobar la eficacia del mismo en el caso de detectar la categoría *apto*.

Debido a esta tendencia, se decide construir una matriz de confusión por competencia, considerando únicamente si la clasificación fue *apto* o no. Estas matrices se presentan en las tablas 6.2, 6.3, 6.4, 6.5 y 6.6.

		Predicho	
		Apto	Otro
Real	Apto	15	0
	Otro	0	0

Tabla 6.2: Matriz de confusión para C1 (Honestidad)

		Predicho	
		Apto	Otro
Real	Apto	15	0
	Otro	0	0

Tabla 6.3: Matriz de confusión para C2 (Responsabilidad)

		Predicho	
		Apto	Otro
Real	Apto	14	1
	Otro	0	0

Tabla 6.4: Matriz de confusión para C3 (Sentido del deber)

		Predicho	
		Apto	Otro
Real	Apto	13	0
	Otro	0	2

Tabla 6.5: Matriz de confusión para C4 (Tolerancia a la Frustración)

Para evaluar el desempeño de esta clasificación, se construye las métricas de evaluación *accuracy*, *recall* (o *true positive rate*) y *false positive rate*, cuyas definiciones se presentan en las Ecuaciones 6.1, 6.2 y 6.3, respectivamente, donde TP corresponde a la cantidad de muestras correctamente clasificadas en la categoría *apto*, TN a aquellas correctamente clasificadas

		Predicho	
		Apto	Otro
Real	Apto	12	1
	Otro	1	1

Tabla 6.6: Matriz de confusión para C5 (Trabajo bajo presión)

como *otro*, P a la totalidad de casos reales en la categoría *apto* y N a la totalidad de casos reales en la categoría *otro*.

$$Accuracy = \frac{TP + TN}{P + N} \tag{6.1}$$

$$Recall = \frac{TP}{P} \tag{6.2}$$

$$Accuracy = \frac{FP}{N} \tag{6.3}$$

El resultado de construir dichas métricas se presenta en la Tabla 6.7. En ella se puede observar que para las primeras dos y para la cuarta competencia, se logra una clasificación perfecta. Sin embargo, para la tercera y la quinta, se cometen errores de clasificación, siendo de mayor envergadura los de esta última.

	C1	C2	C3	C4	C5
Accuracy	100.00 %	100.00 %	93.33 %	100.00 %	86.67 %
Recall	100.00 %	100.00 %	93.33 %	100.00 %	92.31 %
FP Rate	0.00 %	0.00 %	0.00 %	0.00 %	50.00 %

Tabla 6.7: Métricas de efectividad de clasificación por competencia.

Si bien estos resultados pueden parecer alentadores, es importante recalcar que esta evaluación solo permite verificar la eficacia del sistema en una dirección del problema. Por este motivo, se propone complementarlo con otras evaluaciones más particulares.

6.2. Evaluación general de características por sujetos

Como una manera de evaluar de manera más precisa el desempeño del sistema, se decide desagregar las competencias y medir la calidad de clasificación en cada una de las 15 características automatizadas por sujeto.

La metodología utilizada para llevar a cabo esta medición de eficacia consiste en identificar, para cada sujeto, la cantidad de características que son clasificadas correctamente y las que no, de manera de calcular la *accuracy* por sujeto. Al conocer este valor, es posible evaluar la calidad de los algoritmos de detección en distintos escenarios y descartar posibles sesgos para ciertos tipos de escritura.

El resultado de llevar a cabo esta evaluación se presenta en la Tabla 6.8 y, a partir de ella, se puede concluir que existe una alta efectividad para todos los sujetos. En particular, un

ID	Correctas	Incorrectas	Accuracy
S1	14	1	93.33 %
S2	14	1	93.33 %
S3	15	0	100.00 %
S4	15	0	100.00 %
S5	15	0	100.00 %
S6	14	1	93.33 %
S7	13	2	86.67 %
S8	14	1	93.33 %
S9	14	1	93.33 %
S10	15	0	100.00 %
S11	13	2	86.67 %
S12	14	1	93.33 %
S13	15	0	100.00 %
S14	14	1	93.33 %
S15	14	1	93.33 %

Tabla 6.8: Evaluación de clasificación de características por sujeto.

33 % de los sujetos poseen todas sus características determinadas de forma perfecta; un 54 % solo contienen un error de clasificación; y el restante 23 % falla en dos.

Esto indica que el conjunto de algoritmos desarrollados para detectar las características de escritura a partir del cálculo de métricas, logran una buena efectividad al analizar el resultado por sujeto. Sin embargo, aún resta evaluar cuán buena es la calidad de cada uno de los algoritmos por métrica.

6.3. Evaluación particular por característica

Hasta este punto, las evaluaciones realizadas a nivel agregado y general han sido satisfactorias, pero aún resta medir la efectividad de los algoritmos en un nivel de granularidad más fino, lo cual se logra mediante una evaluación particular a nivel de características.

En primer lugar, se debe tener en consideración que una característica puede tomar un valor de *sí* o *no* y que se determina a partir del valor de una o más métricas (para una explicación más detallada, consultar Sección 5.3).

En concreto, esta última medición consiste en contar la cantidad de aciertos de clasificación por cada característica, de manera de poder calcular las mismas métricas de la Sección 6.2. Nuevamente, para ver el contexto de la evaluación y determinar si existe sesgo en el conjunto de muestras basales, se realiza un breve análisis exploratorio de los datos. De esta forma, en la Tabla 6.9 se muestra la cantidad de casos positivos y negativos por cada característica.

Al estudiar la distribución de casos, se puede observar que del total de características, ocho de ellas presentan clases muy desbalanceadas en la muestra, es decir, que una clase posee más

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
sí	9	13	15	12	14	11	12	10	5	13	12	10	8	12	9
no	6	2	0	3	1	4	3	5	10	2	3	5	7	3	6

Tabla 6.9: Distribución de casos por característica.

del 80 % de los casos. Esta situación provoca que para este conjunto de características una alta *accuracy* pueda deberse a un sesgo y no a una verdadera buena capacidad de predicción.

Teniendo este antecedente en consideración, se procede a construir matrices de confusión para cada una de las características (disponibles en Anexos C) y, a partir de ellas, se calculan las métricas de evaluación *accuracy*, *recall* y *false positive rate*, ya que son estas dos últimas las que pueden evaluar con mayor certeza la calidad de la clasificación en situaciones de clases desbalanceadas. Estos resultados se presentan en la Tabla 6.10.

	C1	C2	C3	C4	C5
Accuracy	86.67 %	93.33 %	100.00 %	93.33 %	100.00 %
Recall	88.89 %	100.00 %	100.00 %	100.00 %	100.00 %
FPR	16.67 %	50.00 %	0.00 %	33.33 %	0.00 %

	C6	C7	C8	C9	C10
Accuracy	100.00 %	93.33 %	93.33 %	86.67 %	100.00 %
Recall	100.00 %	91.67 %	100.00 %	80.00 %	100.00 %
FPR	0.00 %	0.00 %	25.00 %	10.00 %	0.00 %

	C11	C12	C13	C14	C15
Accuracy	93.33 %	93.33 %	93.33 %	93.33 %	100.00 %
Recall	100.00 %	100.00 %	87.50 %	100.00 %	100.00 %
FPR	33.33 %	33.33 %	0.00 %	33.33 %	0.00 %

Tabla 6.10: Evaluación del desempeño de la clasificación por características.

A partir de estos resultados se puede extraer una serie de conclusiones.

En primer lugar, para las características que poseen clases balanceadas (números 1, 13 y 15) se logran niveles de *accuracy* y de *recall* que oscilan entre el 86 % y el 100 %, mientras que el nivel de *false positive rate* es inferior al 17 %. Por lo tanto, se puede concluir que de acuerdo a los rangos de calidad definidos en la Tabla 1.1 los algoritmos involucrados presentan un alto grado de efectividad y pueden ser catalogados como buenos y muy buenos, según sea el caso de la característica a evaluar.

En segundo lugar, para las clases que presentan un balanceo moderado (números 6, 8, 9 y 12) los niveles de *accuracy* y de *recall* obtenidos se mueven en el mismo rango, sin embargo, los niveles de *false positive rate* suben hasta un 33 %. La explicación de este comportamiento es que, si bien a modo general el rendimiento del algoritmo es bueno, la tasa de falla en la clase que tiene menos datos es superior. En este escenario, solo dos algoritmos (asociados a las características 6 y 9) pueden ser catalogados como muy buenos, mientras que los restantes caen en la categoría regular.

Por otro lado, para las características con clases desbalanceadas (números 4, 7, 11 y 14) los niveles de las tres métricas comparten el mismo rango que las del grupo anterior. Sin embargo, en este punto, los resultados deben ser interpretados con mayor precaución. Al existir un desbalanceo y, por tanto, un sesgo hacia una de las clases, no es posible concluir que los algoritmos posean una efectividad razonable en un escenario de clases balanceadas.

Este mismo argumento aplica también para las características extremadamente desbalanceadas (números 3, 5 y 10), con la diferencia de que para éstas se tienen métricas de desempeño perfectas. Por lo tanto, debido a la falta de datos, no se puede concluir respecto a la efectividad en la clasificación de estos casos.

Capítulo 7

Conclusión

El análisis neuroescritural es una técnica cada vez más usada en procesos de selección y evaluación de personal para extraer información respecto del nivel de desarrollo de ciertas competencias deseadas.

Debido a la naturaleza manual de este tipo de análisis, el objetivo principal de esta memoria corresponde a automatizar parte de este proceso. Esta meta se logró gracias a la implementación de una serie de algoritmos que, mediante procesamiento de imágenes, son capaces de identificar de manera autónoma un conjunto de 15 características del texto, que luego son utilizadas para calcular el nivel de desarrollo de 5 competencias laborales.

A partir de este trabajo se puede concluir que dentro de un proceso de análisis neuroescritural tradicional existe una cantidad importante de patrones relacionados con las formas y los trazos del texto, cuya detección adecuada es compleja de lograr de manera computacional, por este motivo se propuso un sistema semiautomático de análisis.

Por otro lado, se concluye que la automatización del proceso es posible en la medida que sean desarrollados algoritmos particulares para la detección de cada uno de los patrones, los cuales deben ser calibrados en un volumen considerable de muestras. Al respecto se estima que son necesarias al menos 50 muestras para lograr un nivel de confianza adecuado.

Aún así, con la cantidad acotada de imágenes analizadas que se posee, es posible comprobar altos grados de precisión en la detección de 7 de un total de 15 métricas. En el resto, si bien se obtienen buenas métricas de efectividad de clasificación, no es posible sacar conclusiones debido a la naturaleza de los datos existentes.

Adicionalmente, el profundo análisis llevado a cabo sobre los requerimientos de la solución, permite proponer un diseño extensible para la incorporación de la detección de nuevas características. De esta forma, se realiza un aporte desde el punto de vista de la ingeniería de software detrás del desarrollo de *frameworks* para el procesamiento de imágenes.

En otro ámbito, el sistema propuesto en este trabajo, prueba ser una herramienta que genera valor al usuario debido a la considerable disminución en el tiempo requerido para generar un análisis de un sujeto. En promedio, un perito demora aproximadamente una hora

en realizar un análisis completo de manera manual, mientras que, según pruebas preliminares, este tiempo se reduce a 8 minutos con el uso del sistema. De esta forma, permite aumentar su capacidad de generación de análisis en un factor de 7. Además, debido a los altos niveles de efectividad que provee el sistema, éste puede ser clasificado como una herramienta de apoyo a la toma de decisiones.

7.1. Trabajo Futuro

En el desarrollo de este trabajo se pudo comprobar la factibilidad de automatizar la detección de características neuroescriturales en una imagen de texto. Sin embargo, nuevas líneas de desarrollo pueden ser abordadas con el fin de perfeccionar las técnicas aquí expuestas.

En primer lugar, se propone realizar una evaluación más exhaustiva del rendimiento de los algoritmos desarrollados, utilizando un mayor volumen de muestras de contraste, poniendo énfasis en contar con cantidades balanceadas de muestras de los distintos casos en cada característica. En este sentido, es probable que algunos de los algoritmos necesiten ser recalibrados para su correcto funcionamiento.

Posteriormente, pensando en ampliar el conjunto de características que pueden ser detectadas de manera automática, se propone construir un *dataset* a partir de las evaluaciones manuales que deben ser ingresadas al sistema, para ser usado como base en la formulación de modelos de *data mining* que puedan encontrar relaciones entre métricas ya existentes (o nuevas) con características de detección manual.

Desde un punto de vista de más alto nivel, se propone realizar pruebas de usabilidad formales con los usuarios de las distintas plataformas de *frontend*. De esta forma, se sugiere realizar cambios a la interfaz de acuerdo a las necesidades e inquietudes detectadas en los usuarios al momento de utilizar los sistemas.

Por último, y pensando en llevar este sistema a una implantación real de negocio, se propone extender el sistema de *frontoffice* para incluir un módulo que permita realizar los pagos al momento de realizar los análisis, de manera de sistematizar y agilizar el proceso de comunicación entre el cliente y los expertos neuroescriturales.

Bibliografía

[1] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. *CRISP-DM 1.0 Steb-by-step data mining guide*. SPSS Publications, 2000.

[2] M. Muñoz Colado. *Estudio de la ansiedad en pacientes con cáncer de laringe mediante test grafológico*. PhD thesis, Universidad Complutense de Madrid, 1994.

[3] The Unicode Consortium. *The Unicode Standard*. The Unicode Consortium, Mountain View, CA, 2013.

[4] Florian Coulmas. *The Writing Systems of the World*. Basil Blackwell, 1989.

[5] E. Frias-Martinez, A. Sanchez, and J. Velez. Support vector machines versus multilayer perceptrons for efficient off-line signature recognition. *Engineering Applications of Artificial Intelligence*, 19(6):693 – 704, 2006. ISSN 0952-1976. doi: http://dx.doi.org/10.1016/j.engappai.2005.12.006. URL `http://www.sciencedirect.com/science/article/pii/S0952197606000352`. Special Section on Innovative Production Machines and Systems (I*PROMS).

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994. ISBN 9780321700698.

[7] B. Gatos, N. Stamatopoulos, and G. Louloudis. Icdar 2009 handwriting segmentation contest. In *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pages 1393–1397, July 2009. doi: 10.1109/ICDAR.2009.245.

[8] R. González and R. Woods. *Digital Image Processing*. Prentice Hall, Inc., 2001.

[9] Honroth and Ribera. *Grafología: teoría y práctica*. Ediciones Troquel, 1961.

[10] J. Jensen. *Introductory digital image processing: a remote sensing perspective*. Prentice-Hall Inc., 1996.

[11] S. Kothari, Q. Chaudry, and M.D. Wang. Automated cell counting and cluster segmentation using concavity detection and ellipse fitting techniques. In *Biomedical Imaging: From Nano to Macro, 2009. ISBI '09. IEEE International Symposium on*, pages 795–798, June 2009. doi: 10.1109/ISBI.2009.5193169.

[12] Rebecca A. Langmaid, Nicole Papadopoulos, Beth P. Johnson, James G. Phillips, and

Nicole J. Rinehart. Handwriting in children with adhd. *Journal of Attention Disorders*, 2012. doi: 10.1177/1087054711434154. URL http://jad.sagepub.com/content/early/2012/05/11/1087054711434154.abstract.

[13] U.-V. Marti and H. Bunke. Hidden markov models. chapter Using a Statistical Language Model to Improve the Performance of an HMM-based Cursive Handwriting Recognition Systems, pages 65–90. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2002. ISBN 981-02-4564-5. URL http://dl.acm.org/citation.cfm?id=505741.505745.

[14] R. Lévano Pari. *Variables Grafológicas, Rasgos de Personalidad e Inteligencia en procesos de Evaluación y Selección de Personal*. PhD thesis, Universidad Nacional Mayor de San Marcos, 2010.

[15] R. Plamondon and S.N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):63–84, Jan 2000. ISSN 0162-8828. doi: 10.1109/34.824821.

[16] Claudio Sacchi, Gianluca Gera, Lucio Marcenaro, and Carlo S Regazzoni. Advanced image-processing tools for counting people in tourist site-monitoring applications. *Signal Processing*, 81(5):1017 – 1040, 2001. ISSN 0165-1684. doi: http://dx.doi.org/10.1016/S0165-1684(00)00280-2. URL http://www.sciencedirect.com/science/article/pii/S0165168400002802.

[17] O. Santana, C.M. Travieso, J.B. Alonso, and M.A. Ferrer. Writer identification based on graphology techniques. *Aerospace and Electronic Systems Magazine, IEEE*, 25(6):35–42, June 2010. ISSN 0885-8985. doi: 10.1109/MAES.2010.5525319.

[18] I. Sommerville. *Software Engineering*. Addison-Wesley, 9th edition edition, 2010.

[19] Milan Sonka, Vaclav Hlavac, Roger Boyle, et al. *Image processing, analysis, and machine vision*, volume 3. Thomson, 2007.

[20] S.N. Srihari and Z. Shi. Forensic handwritten document retrieval system. In *Document Image Analysis for Libraries, 2004. Proceedings. First International Workshop on*, pages 188–194, 2004. doi: 10.1109/DIAL.2004.1263248.

[21] T. Steinherz, D. Doermann, E. Rivlin, and N. Intrator. Offline loop investigation for handwriting analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):193–209, Feb 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.68.

[22] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32 – 46, 1985. ISSN 0734-189X. doi: http://dx.doi.org/10.1016/0734-189X(85)90016-7. URL http://www.sciencedirect.com/science/article/pii/0734189X85900167.

[23] Paul Viola and MichaelJ. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000013087.49260.fb. URL http://dx.doi.org/10.1023/B%3AVISI.0000013087.49260.fb.

Anexo A

Diagrama de clases

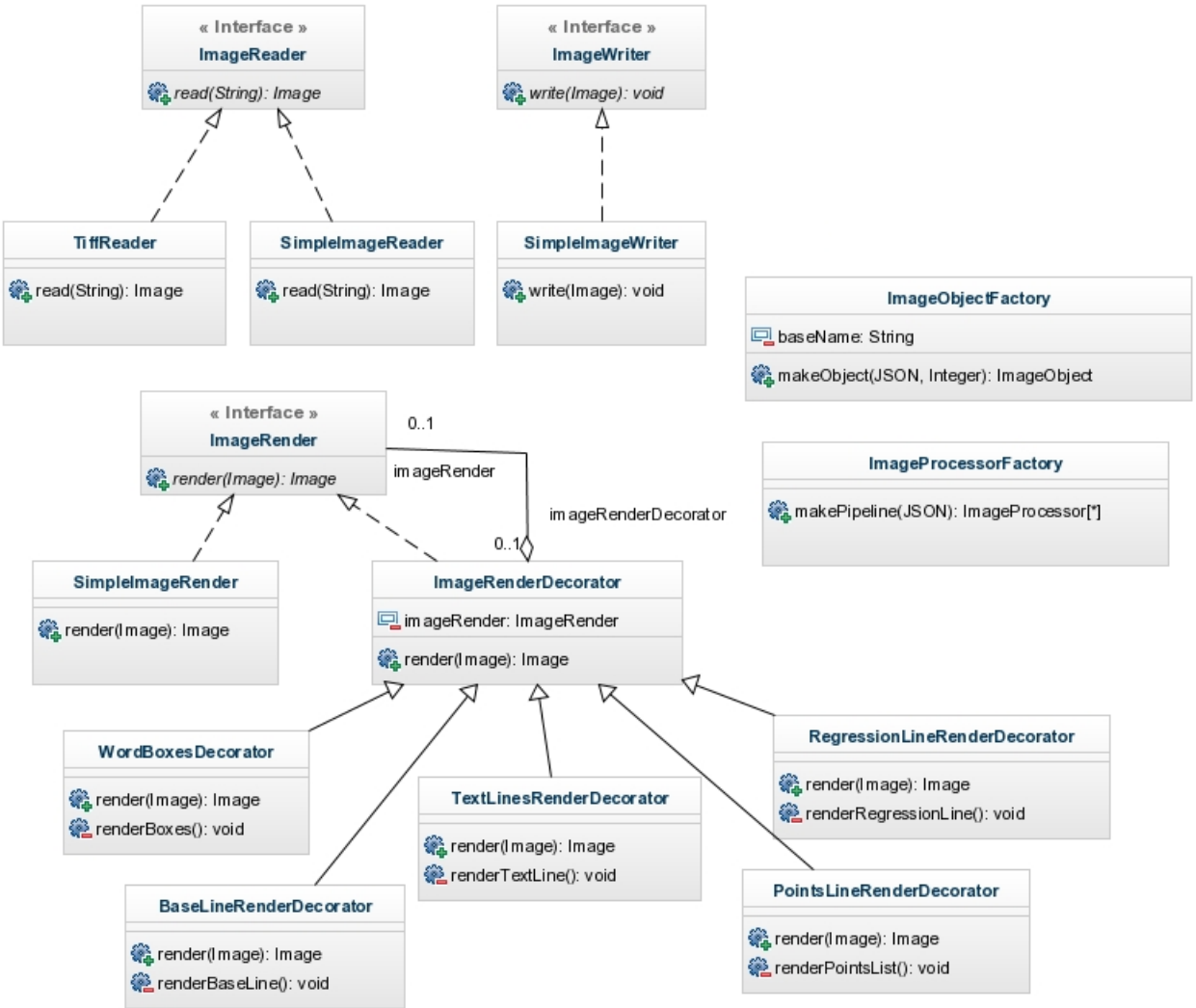


Figura A.1: Sección del diagrama con clases de funcionalidades anexas.

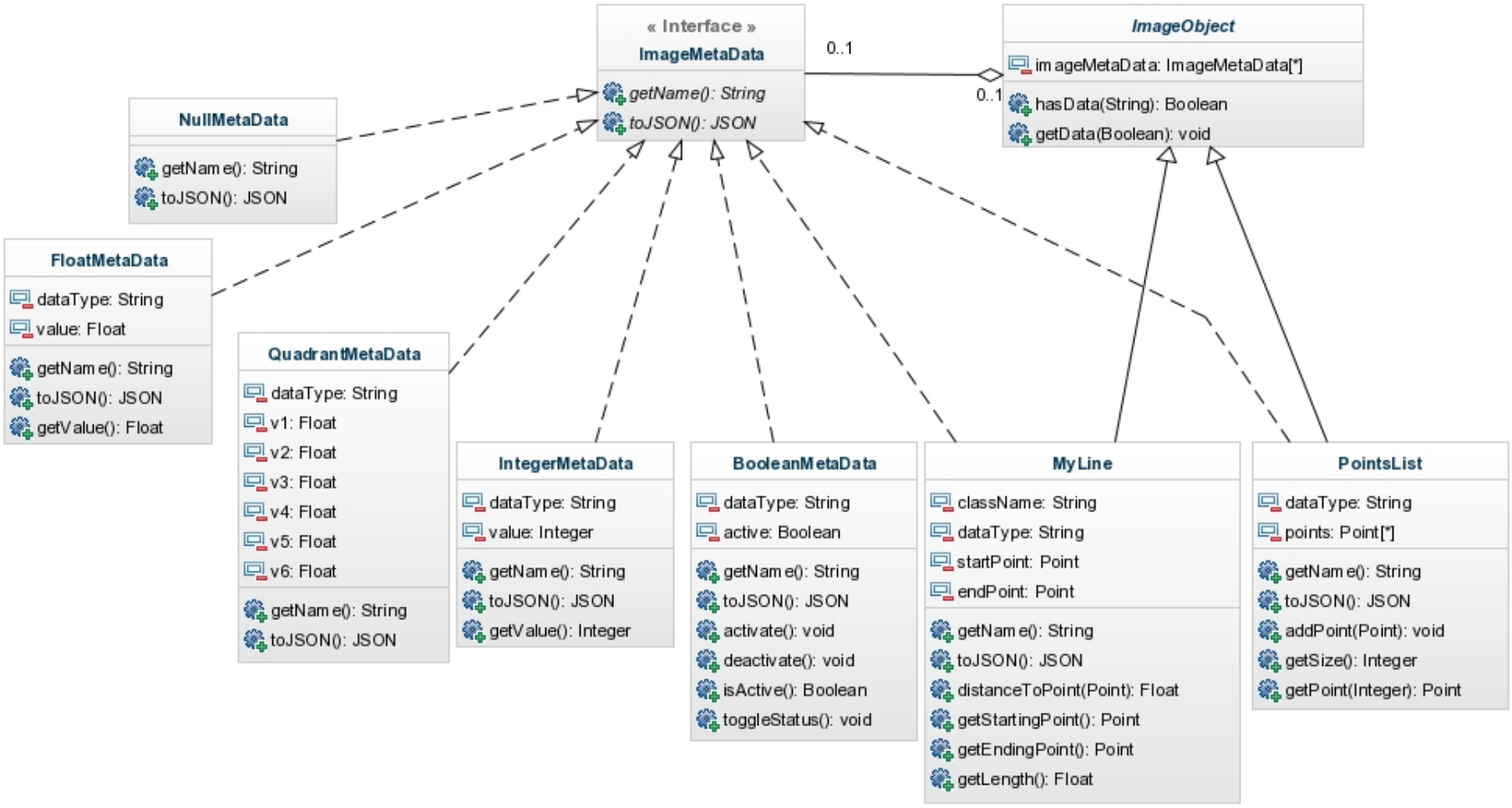


Figura A.2: Sección del diagrama con clases de representación de imágenes (Parte 1 de 2).

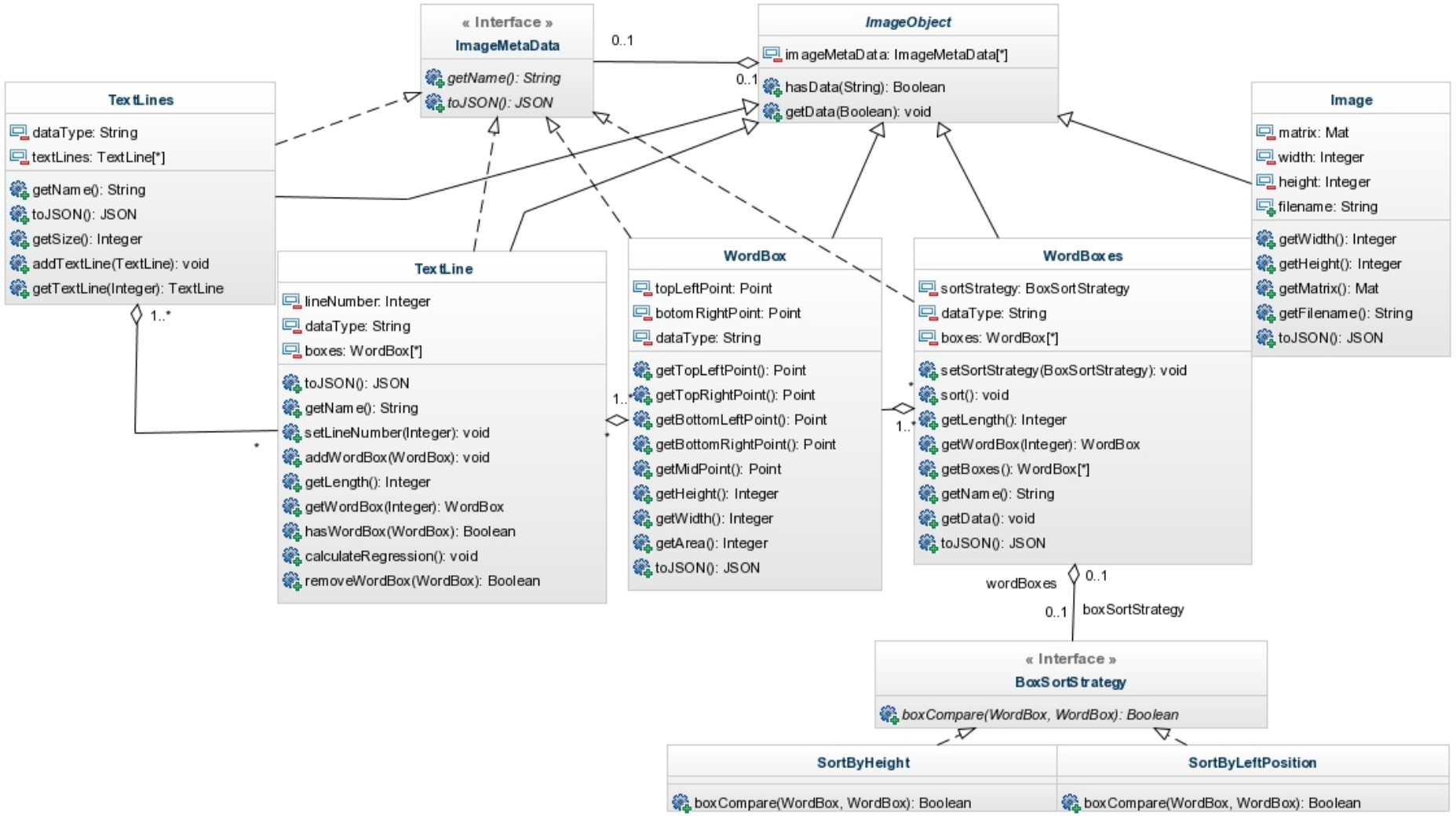


Figura A.3: Sección del diagrama con clases de representación de imágenes (Parte 2 se 2).

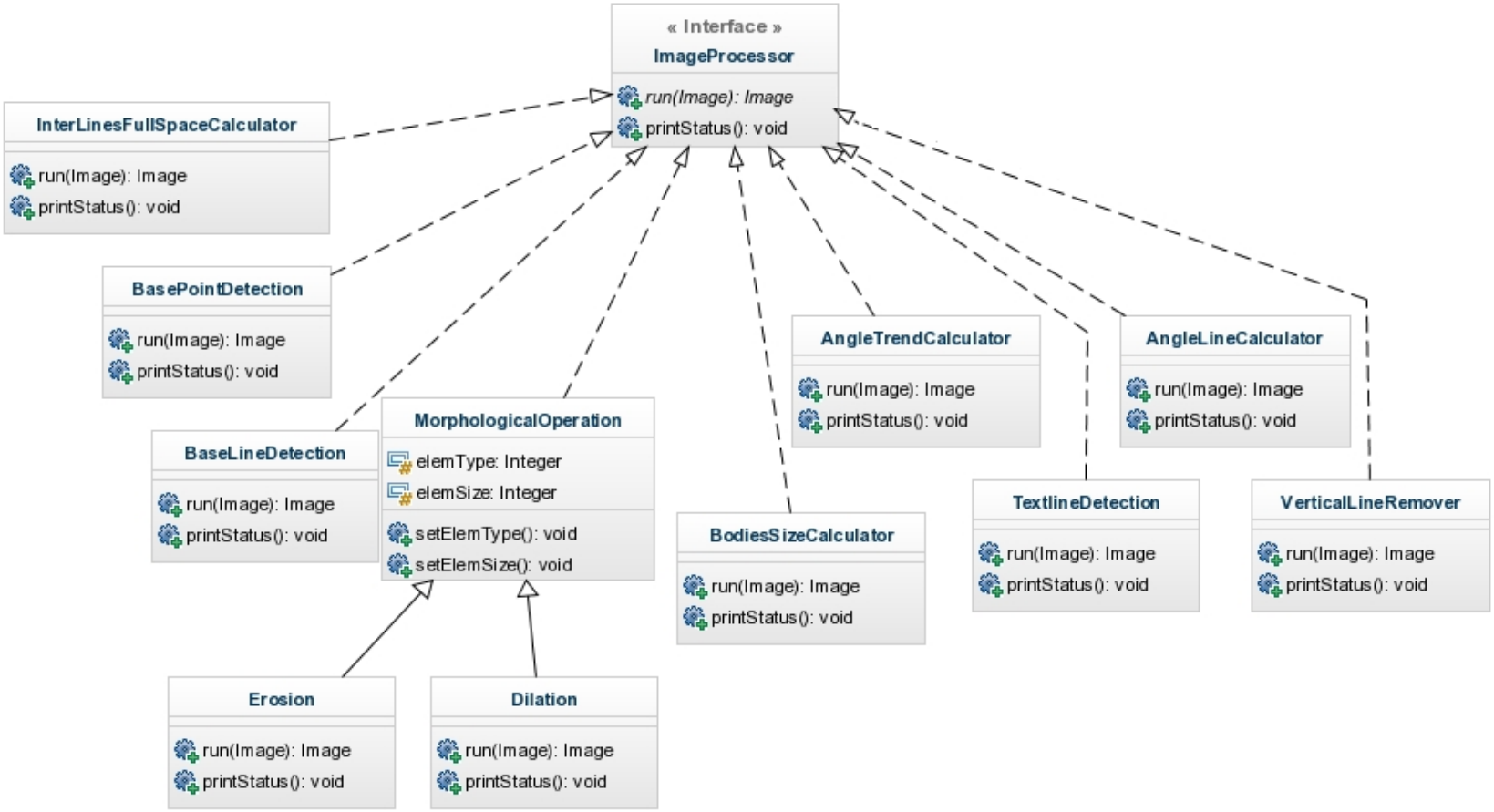


Figura A.4: Sección del diagrama con clases de procesamiento de imágenes (Parte 1 de 3).

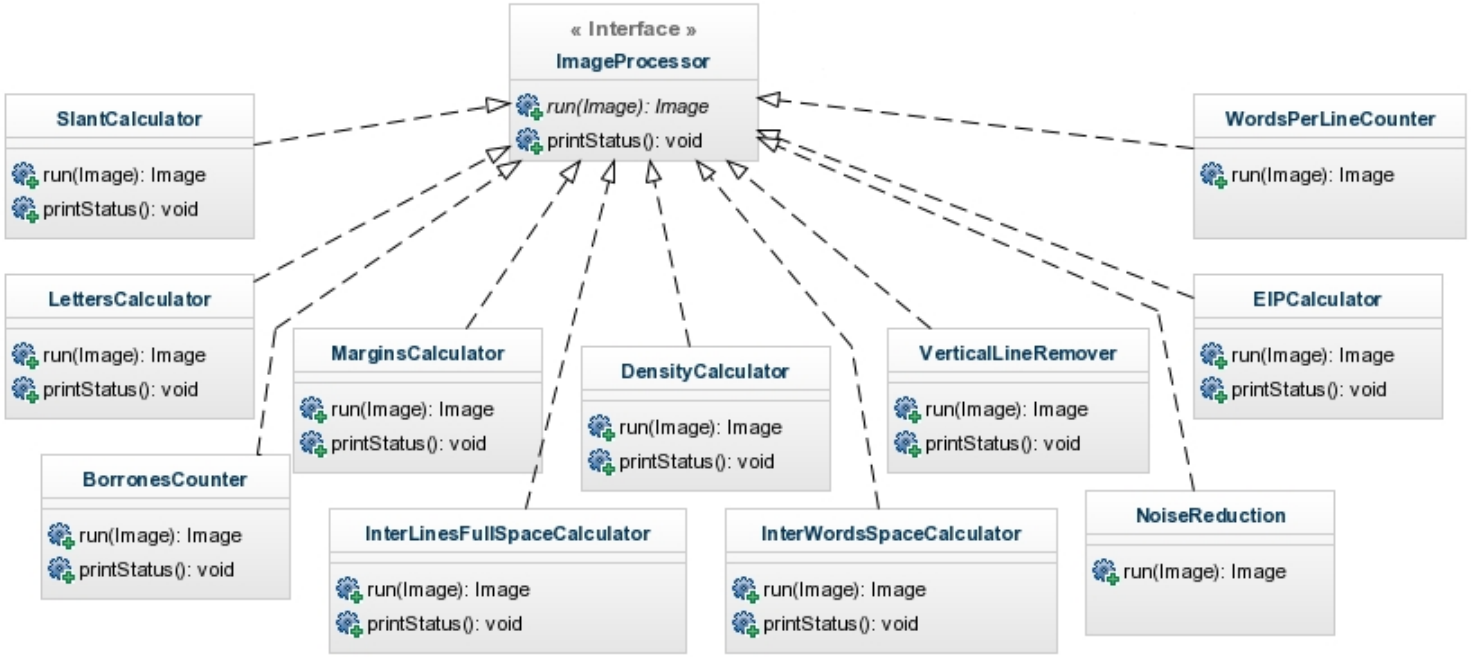


Figura A.5: Sección del diagrama con clases de procesamiento de imágenes (Parte 2 de 3).

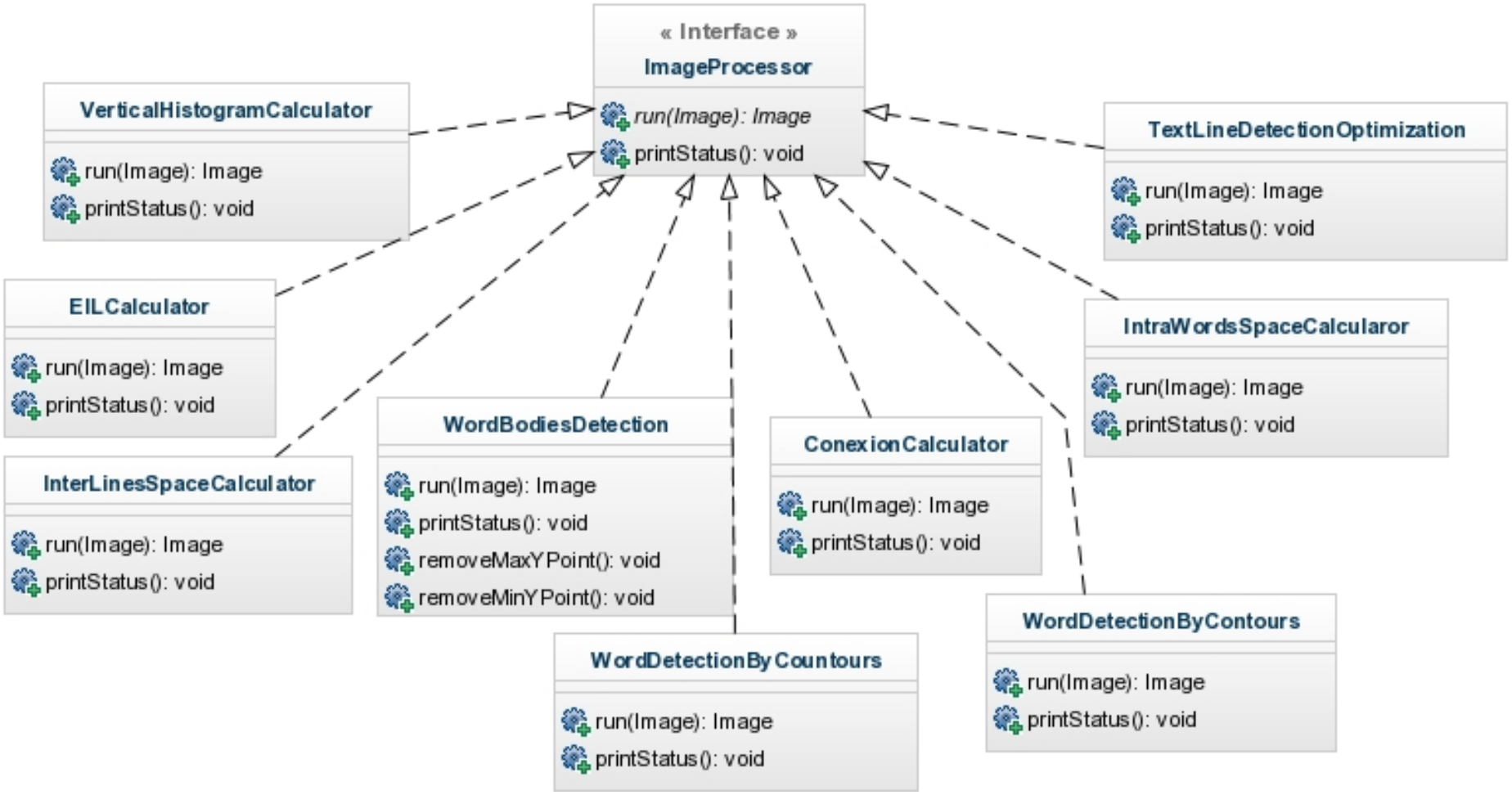


Figura A.6: Sección del diagrama con clases de procesamiento de imágenes (Parte 3 de 3).

Anexo B

Clasificación de sujetos según niveles en competencias

	C1		C2		C3		C4		C5	
id	Real	Pred	Real	Pred	Real	Pred	Real	Pred	Real	Pred
1	A	A	A	A	A	A	A	A	A	A
2	A	A	A	A	A	A	A	A	A	A
3	A	A	A	A	A	A	A	A	R	R
4	A	A	A	A	A	A	A	A	A	A
5	A	A	A	A	A	A	A	A	A	A
6	A	A	A	A	A	A	A	A	A	A
7	A	A	A	A	A	A	N	R	A	A
8	A	A	A	A	A	A	A	A	A	A
9	A	A	A	A	A	A	A	A	A	A
10	A	A	A	A	A	A	A	A	A	A
11	A	A	A	A	A	A	A	A	A	A
12	A	A	A	A	A	R	A	A	R	A
13	A	A	A	A	A	A	A	A	A	A
14	A	A	A	A	A	A	R	R	A	R
15	A	A	A	A	A	A	A	A	A	A
Total A	15	15	15	15	15	14	13	13	13	13
Total R	0	0	0	0	0	1	1	1	2	2
Total N	0	0	0	0	0	0	1	1	0	0

Tabla B.1: Comparación de clasificación de sujetos según niveles en competencias

Anexo C

Matrices de confusión por característica

A continuación se presentan las matrices de confusión utilizadas para la evaluación particular por característica

C1		Predicho	
		No	Sí
Real	No	5	1
	Sí	1	8

Accuracy	86.67 %
Recall	88.89 %
FPR	16.67 %

Tabla C.1: Matriz de confusión y métricas de evaluación para Competencia 1.

C2		Predicho	
		No	Sí
Real	No	1	1
	Sí	0	13

Accuracy	93.33 %
Recall	100.00 %
FPR	50.00 %

Tabla C.2: Matriz de confusión y métricas de evaluación para Competencia 2.

C3		Predicho	
		No	Sí
Real	No	0	0
	Sí	0	15

Accuracy	100.00 %
Recall	100.00 %
FPR	0.00 %

Tabla C.3: Matriz de confusión y métricas de evaluación para Competencia 3.

C4		Predicho	
		No	Sí
Real	No	2	1
	Sí	0	12

Accuracy	93.33 %
Recall	100.00 %
FPR	33.33 %

Tabla C.4: Matriz de confusión y métricas de evaluación para Competencia 4.

C5		Predicho	
		No	Sí
Real	No	1	0
	Sí	0	14

Accuracy	100.00 %
Recall	100.00 %
FPR	0.00 %

Tabla C.5: Matriz de confusión y métricas de evaluación para Competencia 5.

C6		Predicho	
		No	Sí
Real	No	4	0
	Sí	0	11

Accuracy	100.00 %
Recall	100.00 %
FPR	0.00 %

Tabla C.6: Matriz de confusión y métricas de evaluación para Competencia 6.

C7		Predicho	
		No	Sí
Real	No	3	0
	Sí	1	11

Accuracy	93.33 %
Recall	91.67 %
FPR	0.00 %

Tabla C.7: Matriz de confusión y métricas de evaluación para Competencia 7.

C8		Predicho	
		No	Sí
Real	No	3	1
	Sí	0	11

Accuracy	93.33 %
Recall	100.00 %
FPR	25.00 %

Tabla C.8: Matriz de confusión y métricas de evaluación para Competencia 8.

C9		Predicho	
		No	Sí
Real	No	9	1
	Sí	1	4

Accuracy	86.67 %
Recall	80.00 %
FPR	10.00 %

Tabla C.9: Matriz de confusión y métricas de evaluación para Competencia 9.

C10		Predicho	
		No	Sí
Real	No	2	0
	Sí	0	13

Accuracy	100.00 %
Recall	100.00 %
FPR	0.00 %

Tabla C.10: Matriz de confusión y métricas de evaluación para Competencia 10.

C11		Predicho	
		No	Sí
Real	No	2	1
	Sí	0	12

Accuracy	93.33 %
Recall	100.00 %
FPR	33.33 %

Tabla C.11: Matriz de confusión y métricas de evaluación para Competencia 11.

C12		Predicho	
		No	Sí
Real	No	4	1
	Sí	0	10

Accuracy	93.33 %
Recall	100.00 %
FPR	20.00 %

Tabla C.12: Matriz de confusión y métricas de evaluación para Competencia 12.

C13		Predicho	
		No	Sí
Real	No	7	0
	Sí	1	7

Accuracy	93.33 %
Recall	87.50 %
FPR	0.00 %

Tabla C.13: Matriz de confusión y métricas de evaluación para Competencia 13.

C14		Predicho	
		No	Sí
Real	No	2	1
	Sí	0	12

Accuracy	93.33 %
Recall	100.00 %
FPR	33.33 %

Tabla C.14: Matriz de confusión y métricas de evaluación para Competencia 14.

C15		Predicho	
		No	Sí
Real	No	6	0
	Sí	0	9

Accuracy	100.00 %
Recall	100.00 %
FPR	0.00 %

Tabla C.15: Matriz de confusión y métricas de evaluación para Competencia 15.