UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Anonimización de Imágenes

Informe de Proyecto de Grado presentado por

Aymara Nohely Melo Cuello, Guillermo Maiese Perez,
Martín Corredera Decuadro

En cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Libertad Tansini
Regina Motz

Montevideo, 4 de agosto de 2024

Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a todas las personas que nos han acompañado y apoyado durante la realización de este proyecto de grado y nuestras carreras.

En primer lugar, agradecemos profundamente a nuestras tutoras, Libertad Tansini y Regina Motz, por su guía, paciencia y valiosas contribuciones a lo largo de todo el proceso. Sus conocimientos y dedicación fueron esenciales para el desarrollo exitoso de nuestro trabajo.

Agradecemos también al equipo de EDIGA por proporcionarnos los recursos y el apoyo necesarios para llevar a cabo este proyecto. Su colaboración fue fundamental para que pudiéramos alcanzar nuestros objetivos.

Deseamos extender nuestro agradecimiento a nuestras familias, amigos , parejas y compañeros de trabajo, quienes nos brindaron su constante apoyo, comprensión y aliento. Sin su respaldo incondicional, este logro no habría sido posible.

Gracias a todos.

Resumen

Este proyecto de grado titulado «Anonimización de Imágenes» se centra en el desarrollo de un prototipo que permite proteger la privacidad y seguridad de los individuos mediante técnicas de anonimización de imágenes, específicamente en el contexto de investigaciones sociales. Este trabajo se realiza en el marco del proyecto EDIGA (Entornos Digitales e Identidades de Género en la Adolescencia), que investiga cómo las prácticas en redes sociales influyen en la configuración de identidades de género entre adolescentes en España, Uruguay y México.

Para abordar el problema, el proyecto se centró en técnicas avanzadas de anonimización que aseguran que los rostros y otras características identificables en las imágenes sean irreconocibles, permitiendo así el análisis de datos visuales sin comprometer la identidad de los individuos. La solución desarrollada incluye tanto herramientas de anonimización manual como automática, utilizando tecnologías modernas y robustas.

El desarrollo de la solución se basó en una arquitectura en capas, con una clara separación de responsabilidades entre el frontend, el backend y la base de datos. El frontend fue desarrollado utilizando React, una biblioteca de JavaScript que facilita la creación de interfaces de usuario dinámicas y eficientes. Para la manipulación de imágenes y el marcado de regiones de interés, se utilizó Fabric.js, que permite una interacción intuitiva y precisa con las imágenes. Además, se empleó Material-UI para el diseño de componentes de interfaz de usuario, siguiendo las directrices de Material Design de Google.

El backend de la aplicación se construyó utilizando Flask, un framework minimalista de Python que permite una gestión eficiente de rutas y controladores HTTP. Se implementó una API REST que facilita la comunicación entre el frontend y el backend, y se utilizó SQLAlchemy como ORM (Object-Relational Mapper) para la interacción con la base de datos PostgreSQL. La autenticación y autorización de usuarios se manejaron mediante JWT (JSON Web Tokens), asegurando así la seguridad en el acceso a la aplicación.

Para la detección de rostros en las imágenes, se empleó el algoritmo YOLO (You Only Look Once), conocido por su alta precisión y velocidad en tareas de detección de objetos. YOLO se integró con OpenCV, una biblioteca de código abierto especializada en procesamiento de imágenes, para aplicar los diferentes métodos de anonimización, como difuminado, pixelado y bloqueo de regiones. Además, se utilizó ONNX Runtime para ejecutar el modelo de aprendizaje au-

tomático.

El prototipo también se diseñó para ser desplegada tanto en entornos locales como en la nube, utilizando Docker para la creación de contenedores que facilitan la portabilidad y escalabilidad de la aplicación. Se implementó un reverse proxy con Nginx para manejar de manera segura las solicitudes entrantes y distribuirlas a los diferentes servicios del sistema, asegurando la encriptación de las comunicaciones mediante certificados SSL proporcionados por Let's Encrypt.

En términos de evaluación, la herramienta se sometió a diversas pruebas que demostraron su facilidad de uso, eficiencia y seguridad. La integración de diferentes tecnologías y herramientas permitió desarrollar una solución robusta que cumple con los requisitos de los investigadores del proyecto EDIGA, ofreciendo una forma efectiva y ética de anonimizar imágenes para su uso en investigaciones sociales.

En resumen, el proyecto logró desarrollar un prototipo de herramienta para la anonimización de imágenes que no solo protege la privacidad de los individuos, sino que también facilita la investigación en ciencias sociales, proporcionando una solución escalable, segura y fácil de usar.

Palabras clave: Anonimización de imágenes, Privacidad, Seguridad de datos, Investigaciones sociales

Índice general

Índice general VII

Índice de figuras XI

1. Introducción 1

2. Conceptos de base 5
2.1. Conceptos previos . . . 5
2.2. Métodos de ofuscación o anonimización de imagenes . . . 8
2.3. Modelos de detección y clasificación de objetos . . . 11

3. Relevamiento de requerimientos 17
3.1. Requerimientos funcionales . . . 17
3.1.1. Sesión . . . 17
3.1.2. Configuración de usuario . . . 18
3.1.3. Anonimización manual . . . 18
3.1.4. Anonimización automática . . . 18
3.2. Requerimientos no funcionales . . . 18
3.3. Prueba de concepto . . . 19

4. Análisis de Herramientas 23
4.1. Enfoque de la evaluación . . . 23
4.2. Entorno de la comparación . . . 25
4.3. Pasos realizados para la comparación . . . 26
4.4. Desarrollo de cada herramienta . . . 26
4.5. Fortalezas y debilidades de cada herramienta . . . 29
4.6. Conclusiones del relevamiento de herramientas . . . 32

5. Diseño de la solución 33
5.1. Arquitectura . . . 33
5.2. Modelo relacional . . . 35
5.3. Conclusión . . . 35

6. Tecnologías y herramientas utilizadas **37**
6.1. Investigación de tecnologías . . . 37
6.1.1. Criterios de selección . . . 37
6.2. Frontend . . . 37
6.2.1. React . . . 38
6.2.2. Fabric . . . 38
6.2.3. Material UI . . . 39
6.2.4. JSZip y FileSaver . . . 39
6.3. Backend . . . 40
6.3.1. Flask . . . 40
6.3.2. Autenticación JWT . . . 40
6.3.3. OpenCV . . . 41
6.3.4. YOLO . . . 42
6.3.5. ONXRuntime . . . 42
6.3.6. ORM . . . 42
6.3.7. Envío de correos electrónicos . . . 43
6.4. Base de Datos . . . 43
6.5. Docker . . . 44
6.6. Otras Herramientas . . . 45
6.6.1. Github y Github projects . . . 45
6.6.2. Portainer . . . 45
6.6.3. Sentry . . . 45

7. Implementación **47**
7.1. Sesión de usuario . . . 47
7.1.1. Registro de usuario . . . 48
7.1.2. Inicio de sesión . . . 49
7.1.3. Recuperar contraseña . . . 50
7.1.4. Visualización de rutas protegidas . . . 51
7.1.5. Perfil de usuario . . . 52
7.2. Anonimización manual de imágenes . . . 53
7.2.1. Selección de imagen . . . 53
7.2.2. Regiones de interés . . . 54
7.2.3. Movimiento y Dimensionamiento de Grupos . . . 55
7.2.4. Visualización de resultados . . . 56
7.3. Anonimización automática . . . 58
7.3.1. Nivel de confianza . . . 59
7.3.2. Visualización de resultados . . . 60
7.4. Manejo de errores . . . 62
7.5. Pruebas y validaciones . . . 65
7.5.1. Instancias de retroalimentación . . . 65
7.5.2. Pruebas en Frontend . . . 65
7.5.3. Pruebas en Backend . . . 66

8. Despliegue de la solución **69**
8.1. Despliegue público 69
8.1.1. Docker Engine 70
8.1.2. Portainer 70
8.2. Despliegue local 71
8.2.1. Guía para despliegue local 72

9. Conclusiones **73**

10. Trabajo a futuro **75**

Referencias **79**

A. Anexo Proceso de Desarrollo **81**
A.1. Metodología de desarrollo utilizada 81
A.1.1. Github como plataforma de organización 82

Índice de figuras

2.1. Ejemplo de Red Neuronal Básica 7
2.2. Características de Haar 7
2.3. Desenfoque gaussiano aplicado en toda la imágen 8
2.4. Streamline results F-CNN 12
2.5. Streamline results 13
2.6. Streamline results SSD 14
2.7. Streamline results - YOLOv3 15

4.1. Implementación para la comparación de herramientas. 25
4.2. Demostración de herramienta Celantur 26
4.3. Blur Gaussiano utilizando Fiji ImageJ2 27
4.4. Pipeline de OpenCV para el tratamiento de imágenes 27

5.1. Diagrama de arquitectura del sistema 33
5.2. Modelo Relacional 35

7.1. Visualización de la aplicación con sesión de usuario no iniciada . 48
7.2. Registro de usuario 48
7.3. Inicio de sesión 49
7.4. Diagrama de secuencia de inicio de sesión 50
7.5. Pasos en la recuperación de contraseña 50
7.6. Restablecer contraseña 51
7.7. Visibilidad de rutas protegidas 51
7.8. Perfil de usuario y acciones disponibles 52
7.9. Diagrama de secuencia de cerrar sesión 52
7.10. Herramienta de anonimización manual 53
7.11. Marcado de regiones de interés 54
7.12. Resultados de anonimización manual 57
7.13. Descarga individual de resultados 57
7.14. Opciones de descarga de resultados 58
7.15. Herramienta de anonimización automática 58

7.16. Resultados de la anonimización automática que muestran la capacidad del prototipo para detectar rostros y anonimizarlos con un nivel de confianza del 70 % y la misma imágen con un nivel de confianza del 85 % 59
7.17. Imágenes en procesamiento automático 60
7.18. Resultados generados automaticamente 60
7.19. Descarga individual de resultados 61
7.20. Opciones de descarga de resultados 62
7.21. Alerta de error 63
7.22. Sentry: Visualización de error detectado 63
7.23. Sentry: Demostración de cómo fue generado un error 64
7.24. Sentry: Ejemplo de métrica de performance 64

8.1. Docker Engine en Elastic Cloud 70
8.2. Entorno en Portainer 71
8.3. Servicio local corriendo en Docker Desktop 72

Capítulo 1

Introducción

La anonimización de datos emerge como un tema de suma relevancia en el ámbito de investigaciones sociales y en diversas disciplinas que requieren el manejo de información de identificación personal. La necesidad de divulgar pruebas para respaldar conclusiones, sin comprometer la privacidad y la seguridad de los individuos, ha catalizado el desarrollo de técnicas avanzadas de anonimización.

La anonimización de datos se ha convertido en una herramienta esencial para la investigación, pero su aplicación efectiva y segura no siempre es sencilla. Por tanto, es crucial que los investigadores y otros usuarios no expertos tengan acceso a herramientas que les permitan comparar técnicas de anonimización, evaluar el riesgo de reidentificación y medir la calidad de la anonimización. Esto les brinda la capacidad de utilizar y compartir datos que, de lo contrario, serían inaccesibles debido a la complejidad del proceso y la falta de conocimientos especializados.

Este trabajo se enfocó en brindar una solución a los investigadores del proyecto EDIGA[1] (Entornos Digitales e Identidades de Género en la Adolescencia). EDIGA tiene como objetivo investigar cómo las prácticas en redes sociales influyen en la configuración de subjetividades e identidades de género entre adolescentes. Participan equipos de investigación de España, Uruguay y México, y el grupo de investigación de EDIGA juega un papel fundamental en el desarrollo y validación de las técnicas de anonimización aplicadas.

Por esta razón, este proyecto se centra en la anonimización de imágenes, ya que a partir de la información que se obtiene de las redes sociales, se necesita garantizar la privacidad de los individuos involucrados. La anonimización de imágenes se realiza mediante técnicas avanzadas que aseguran que los rostros y otras características identificables sean irreconocibles, protegiendo así la identidad de las personas mientras se permite el análisis de los datos visuales para la investigación.

En resumen, la implementación de estas técnicas de anonimización no solo proporciona una herramienta valiosa para los investigadores del proyecto EDI-

[1] https://stellae.usc.es/ediga/proyecto

GA, sino que también establece un marco para el uso seguro y ético de datos visuales en investigaciones futuras. Esto contribuye significativamente a la protección de la privacidad y a la integridad de los datos en el ámbito de las ciencias sociales y más allá.

Aspectos generales del problema:

- Privacidad y Seguridad: La información visual, especialmente rostros, contiene datos personales sensibles. La falta de anonimización puede dar lugar a la identificación no deseada, comprometiendo la privacidad y seguridad de los individuos en las imágenes.
- Mejores Técnicas y Análisis de Herramientas: La selección de técnicas eficientes de anonimización es crucial. Se necesita una evaluación de herramientas disponibles, considerando factores como la efectividad de la anonimización, la preservación de características relevantes y la facilidad de uso.
- Reducir la Complejidad: La complejidad asociada con la anonimización de rostros a menudo desalienta su implementación efectiva. Se requiere una simplificación de procesos y la adopción de enfoques que no solo aseguren la protección de la privacidad, sino que también faciliten la aplicación práctica por parte de investigadores.
- Cumplimiento Normativo: Es esencial que las soluciones de anonimización cumplan con los estándares y regulaciones internacionales, como el Reglamento General de Protección de Datos (GDPR) de la Unión Europea, para asegurar la protección legal y ética de los datos personales.

Aspectos particulares del problema:

- Utilización por investigadores: Los investigadores sociales dependen de la recopilación de datos visuales para enriquecer sus estudios y respaldar las conclusiones. Sin embargo, la necesidad de compartir estas imágenes de manera ética y segura requiere la implementación de técnicas efectivas de anonimización.
- Información sensible: Asegurar que la solución de anonimización cumpla con los estándares de protección de datos establecidos por la Unión Europea, garantizando así la conformidad con regulaciones como el Reglamento General de Protección de Datos (GDPR).
- Facilidad de uso: Las herramientas de anonimización deben ser intuitivas y accesibles, permitiendo que los investigadores sin conocimientos técnicos avanzados puedan utilizarlas efectivamente.
- Preservación de Datos: Es fundamental que las técnicas de anonimización mantengan la integridad de los datos esenciales para la investigación, asegurando que la utilidad de los datos no se vea comprometida.

- Acceso global: Las solución debe ser fácilmente accesible para cualquier investigador, independientemente de su ubicación geográfica. Esto se logra mediante la disponibilidad de la herramienta como una aplicación web, permitiendo un acceso sencillo y universal a través de internet.

De estos objetivos generales y particulares se lograron todos los puntos, pudiendo mantener la seguridad de la información, la facilidad de uso y la preservación de los datos.

Capítulo 2

Conceptos de base

Para comprender este trabajo de manera efectiva, es fundamental tener una comprensión exhaustiva de los conceptos presentados a lo largo del documento. En primer lugar, se describen los métodos existentes para ofuscar información, con un enfoque particular en imágenes. A continuación, se analizan diversos algoritmos que permiten detectar regiones de forma automática mediante la detección de objetos. Esta sección concluye con la presentación de la propuesta implementada.

2.1. Conceptos previos

Para facilitar esta comprensión, se introducen algunos conceptos previos que proporcionan un marco teórico sólido. Estos conceptos fundamentales sirven de base para los capítulos siguientes, asegurando que el lector tenga el contexto necesario para entender la evolución del tema y la justificación de la solución propuesta.

Anonimizar

De acuerdo con la definición de la Real Academia Española (ESPAÑOLA, 2024), el término anonimizar se refiere a la acción de eliminar cualquier referencia a la identidad de entidades o personas en un dato específico. En otras palabras, se trata de suprimir la información que podría identificar a un individuo concreto.

Máscara

En el ámbito de la informática y, en particular, en el procesamiento de imágenes, una máscara (R. Fisher, S. Perkins, A. Walker and E. Wolfart, 2004) se refiere a una matriz o filtro que se aplica a una imagen para realizar diversas

operaciones. Estas operaciones pueden incluir el resaltado de características específicas, la eliminación de ruido, la detección de bordes, la segmentación de regiones de interés, entre otras.

Una máscara define qué píxeles de la imagen original serán modificados y de qué manera, permitiendo la manipulación selectiva de partes de la imagen para obtener los resultados deseados. Es una herramienta esencial en muchos algoritmos de procesamiento de imágenes, ya que facilita la aplicación de técnicas como la convolución, la dilatación y la erosión, entre otras.

Segmentación Semántica

La segmentación semántica es una tarea de visión por computadora que asigna una etiqueta de clase a los píxeles utilizando un algoritmo de aprendizaje profundo (Deep Learning). Es una de las tres subcategorías del proceso general de segmentación de imágenes que ayuda a los ordenadores a comprender la información visual. La segmentación semántica identifica colecciones de píxeles y las clasifica según diversas características (IBM, 2024).

Red Neuronal

Una red neuronal (Minsky y Papert, 1969) es un modelo de aprendizaje automático que modela el funcionamiento del cerebro humano, utilizando procesos que imitan cómo las neuronas biológicas trabajan en conjunto para identificar fenómenos, evaluar opciones y llegar a conclusiones.

Una red neuronal consta de varias capas de nodos o neuronas artificiales: una capa de entrada, una o varias capas ocultas y una capa de salida, como se muestra en la Figura 2.1. Cada nodo está conectado a otros nodos y tiene su propia ponderación y umbral asociados. Si la salida de un nodo individual supera el valor umbral especificado, ese nodo se activa y transmite datos a la siguiente capa de la red. De lo contrario, no se pasa ningún dato a la siguiente capa.

Esta estructura permite que las redes neuronales procesen información de manera compleja y sean capaces de aprender y generalizar a partir de los datos, haciéndolas útiles para una amplia variedad de aplicaciones, desde el reconocimiento de imágenes hasta la predicción de tendencias en datos.

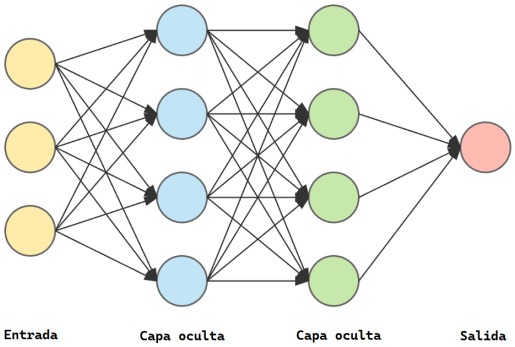


Figura 2.1: Ejemplo de Red Neuronal Básica

Características de Haar

Las características de Haar son un conjunto de cálculos realizados sobre rectángulos adyacentes, como se muestra en la Figura 2.2. Estas características se utilizan para detectar patrones en las imágenes al barrerlas con ventanas de diferentes tamaños y posiciones. A través de estas regiones rectangulares, es posible identificar diferencias de intensidad que son útiles para la detección de objetos. Este método es fundamental en algoritmos de visión por computadora, como los utilizados en la detección de rostros y otros objetos (OpenCV, 2024a).

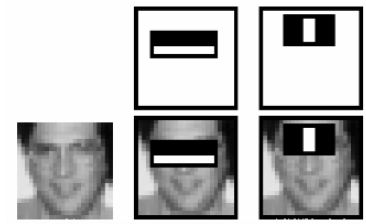


Figura 2.2: Características de Haar

Desenfoque Gaussiano

El desenfoque gaussiano es un método de difuminado que aplica una función gaussiana sobre una región de una imagen. Este proceso genera una nueva imagen, como se muestra en la Figura 2.3 en la que la región seleccionada se encuentra suavizada (OpenCV, 2024b).

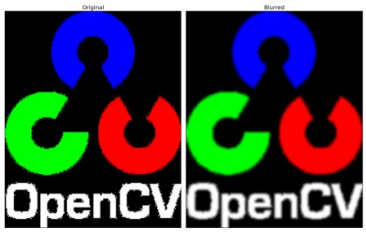


Figura 2.3: Desenfoque gaussiano aplicado en toda la imágen

Precisión Media

La Precisión Media (AP, por sus siglas en inglés) es una métrica utilizada para evaluar la precisión de un modelo de detección de objetos en tareas de visión por computadora. La AP se calcula como el área bajo la curva de precisión-recall (precisión contra recuperación). Formalmente, se define como:

$$\text{AP} = \int_0^1 \text{Precisión}(r)\, dr, \tag{2.1}$$

donde Precisión(r) es la precisión en un nivel de recuperación r. La curva precisión-recall se obtiene evaluando el modelo en un conjunto de datos de validación y midiendo la precisión y la recuperación en diferentes umbrales de confianza. La AP es especialmente útil en conjuntos de datos desbalanceados, donde las clases pueden tener un número desigual de muestras (Everingham, Gool, Williams, Winn, y Zisserman, 2010).

Conjuntos de Datos

En el ámbito de la visión por computadora, un conjunto de datos es una colección de imágenes y, a menudo, sus correspondientes etiquetas o anotaciones, que se utiliza para entrenar, validar y probar modelos de aprendizaje automático y algoritmos de procesamiento de imágenes. Estos conjuntos de datos son fundamentales para el desarrollo y evaluación de técnicas de visión por computadora, ya que proporcionan el material necesario para que los modelos aprendan a reconocer patrones, objetos y características en las imágenes.

El conjunto de datos de rostros, utilizado en este proyecto se obtuvo de Kaggle, una plataforma donde se publican conjuntos de datos de dominio público, como el utilizado para el análisis de las herramientas presentadas en los siguientes capítulos (Gupta, s.f.).

2.2. Métodos de ofuscación o anonimización de imagenes

La ofuscación en imágenes se refiere a técnicas que alteran la representación visual de la información en una imagen con el propósito de hacerla más difícil de

entender o interpretar (ESPAÑOLA, 2024). Estas técnicas son conocidas como métodos de ofuscación o métodos de anonimización y se utilizan a menudo para proteger la privacidad o la información sensible en imágenes. En esta sección se presentaran los métodos más utilizados.

Pixelado

La pixelación (Kerschbaum y Paraboschi, 2018) es una técnica de procesamiento de imágenes que permite disminuir su resolución, proporcionando un medio efectivo para anonimizar o proteger ciertas áreas de la imagen. Este proceso consiste en dividir la imagen en un conjunto de bloques de píxeles adyacentes y luego calcular un color representativo para cada bloque basado en los colores de los píxeles que lo componen. Una vez calculado, este color se utiliza para reemplazar todos los píxeles dentro del bloque, resultando en una imagen con una pérdida de detalles notable en las secciones a las que se les ha aplicado esta técnica.

Matemáticamente, si consideramos un bloque B de tamaño $n \times m$ píxeles, el valor del color representativo C_B se calcula como la media de los valores de los píxeles (x, y) en el bloque:

$$C_B = \frac{1}{n \cdot m} \sum_{(x,y) \in B} I(x, y), \tag{2.2}$$

donde $I(x, y)$ es el valor del píxel en la coordenada (x, y) de la imagen original. Luego, cada píxel (x, y) en el bloque B se reemplaza por C_B:

$$I'(x, y) = C_B \quad \text{para todo } (x, y) \in B. \tag{2.3}$$

Esta técnica reduce efectivamente la resolución local de la imagen y oculta los detalles finos en las áreas seleccionadas, proporcionando una forma sencilla y eficaz de proteger la información visual sin alterar significativamente el contexto global de la imagen.

Difuminado

El difuminado (Blur) (Gonzalez y Woods, 2002) es una técnica de procesamiento de imágenes que permite obtener una imagen con una sección suavizada, en la cual se pierden los detalles sutiles y más representativos de la región a la que se aplica. Este proceso consiste en generar una región de adyacencia alrededor de un píxel y calcular el color promedio de los píxeles que la componen. Este color calculado se convierte en el nuevo color del píxel original, suavizando notoriamente los bordes entre los píxeles.

Matemáticamente, si $I(x, y)$ representa el valor del píxel en la coordenada (x, y) de la imagen original, y $N(x, y)$ es el conjunto de píxeles vecinos alrededor de (x, y), el valor del píxel difuminado $I'(x, y)$ se puede calcular como:

$$I'(x,y) = \frac{1}{|N(x,y)|} \sum_{(i,j)\in N(x,y)} I(i,j), \tag{2.4}$$

donde $|N(x,y)|$ es el número de píxeles en el vecindario $N(x,y)$. La suavidad o dureza del difuminado puede ajustarse mediante el tamaño del vecindario y el peso asignado a cada píxel durante el cálculo del promedio. En versiones más avanzadas, se pueden utilizar diferentes funciones de ponderación para los píxeles, tales como la función gaussiana, que asigna más peso a los píxeles cercanos al píxel central y menos a los más alejados, resultando en un desenfoque gaussiano.

Esta técnica es útil para reducir el ruido y los detalles finos en la imagen, proporcionando una apariencia más suave y menos detallada en las áreas seleccionadas.

Bloqueo

La funcionalidad de bloqueo (Kerschbaum y Paraboschi, 2018) se refiere a una técnica en la cual una región detectada de la imagen es completamente reemplazada por un bloque de valores cero. Este procedimiento tiene como efecto recortar la región específica, eliminando cualquier información presente en ella. Como resultado, se obtiene una imagen en la que la región seleccionada está cubierta por un bloque de color sólido (negro en este caso), ocultando así cualquier elemento que estuviera presente en dicha área.

Esta técnica es útil en diversos contextos donde se requiere anonimizar o eliminar información sensible de una imagen antes de su procesamiento o análisis posterior. Al aplicar el bloqueo, se asegura que ningún detalle visual de la región cubierta sea accesible, manteniendo la privacidad y confidencialidad de los datos.

Matemáticamente, si $I(x,y)$ representa el valor del píxel en la coordenada (x,y) de la imagen original, y R es la región detectada que se desea bloquear, la operación de bloqueo se puede expresar como:

$$I'(x,y) = \begin{cases} 0 & \text{si } (x,y) \in R, \\ I(x,y) & \text{en otro caso,} \end{cases} \tag{2.5}$$

donde $I'(x,y)$ es la imagen resultante después de aplicar el bloqueo.

Esta metodología no solo permite mantener la integridad de la imagen fuera de la región bloqueada, sino que también facilita la anonimización efectiva y segura de la información visual.

Sal y Pimienta

El ruido sal y pimienta (Gonzalez y Woods, 2002) es un tipo de ruido impulsivo que se manifiesta como puntos blancos y negros dispersos aleatoriamente en una imagen. Este ruido se caracteriza por la aparición de píxeles que se establecen en los valores extremos de la escala de intensidad, es decir, 0 (negro) y 255 (blanco) en una imagen de 8 bits.

Matemáticamente, el ruido sal y pimienta se puede modelar mediante una función de probabilidad que describe la probabilidad de que un píxel tome el valor mínimo o máximo:

$$P(x) = \begin{cases} p & \text{si } x = 0, \\ q & \text{si } x = 255, \\ 0 & \text{en otro caso,} \end{cases} \tag{2.6}$$

donde $P(x)$ es la probabilidad de que un píxel tenga el valor x, p es la probabilidad de que un píxel sea negro (0), y q es la probabilidad de que un píxel sea blanco (255). Generalmente, p y q se toman iguales y su suma representa la densidad del ruido en la imagen.

El ruido sal y pimienta implica que los algoritmos de procesamiento de imágenes deben ser robustos para detectar y eliminar este tipo de ruido sin afectar significativamente los detalles importantes de la imagen. Técnicas comunes para mitigar este ruido incluyen el uso de filtros de mediana y algoritmos de detección de bordes que son menos sensibles a los valores extremos.

$$P(x) = \begin{cases} p & \text{si } x = 0, \\ q & \text{si } x = 255, \\ 0 & \text{en otro caso,} \end{cases} \tag{2.7}$$

Este tipo de ruido es particularmente problemático en aplicaciones de visión por computadora y análisis de imágenes, donde la precisión y la claridad de los datos visuales son críticas para el rendimiento de los algoritmos de procesamiento subsecuentes.

2.3. Modelos de detección y clasificación de objetos

Un aspecto clave para lograr la anonimización automática de una imagen es la detección del objeto o la sección relevante para este proceso. Para comprender cómo funcionan las herramientas actuales con esta funcionalidad, en esta sección se realiza un estudio de publicaciones académicas, seleccionando y destacando los modelos de detección de objetos más utilizados. Esto permite compararlos basándose en publicaciones científicas y en el soporte de la comunidad. Algunas comparaciones se llevan a cabo utilizando tablas comparativas presentes en cada publicación científica, ya que, en general, se emplean los mismos conjuntos de datos para las evaluaciones.

Boosted Cascade of Simple Features

Este enfoque (Viola y Jones, 2001) trabaja con un conjunto de características de Haar, para obtener esas características se utiliza la representación integral de imágenes que se obtiene realizando operaciones por pixel en la imagen original.

Posteriormente se utiliza un clasificador creado mediante AdaBoost seleccionando determinadas características importantes.

La clave en este método es que se combinan clasificadores sucesivamente más complejos formando una estructura de cascada, que es lo que permite mejorar la velocidad del detector ya que se centra en regiones prometedoras de la imagen.

Fast R-CNN

Fast R-CNN (Ren, He, Girshick, y Sun, 2015) es una arquitectura de modelos de aprendizaje automático que combina métodos de propuesta de regiones y redes neuronales convolucionales. Mediante una red de propuesta de regiones que recibe una imagen, se obtiene un conjunto de propuestas de objetos rectangulares, los cuales son clasificados en categorías utilizando otra red convolucional y también se refinan las coordenadas que se predijeron inicialmente.

El enfoque busca aumentar la velocidad de detección de estos objetos, comparando con arquitecturas similares, sin comprometer la precisión con la que los objetos son detectados.

En la tabla 2.4 puede verse el desempeño de Fast-CNN para detectar objetos utilizando el conjunto PASCAL VOC 2007[1] el cual es utilizado en las comparaciones clásicas de evaluaciones de desempeño de modelos en la academia.

train-time region proposals		**test-time region proposals**		
method	# boxes	method	# proposals	mAP (%)
SS	2k	SS	2k	58.7
EB	2k	EB	2k	58.6
RPN+ZF, shared	2k	RPN+ZF, shared	300	**59.9**
ablation experiments follow below				
RPN+ZF, unshared	2k	RPN+ZF, unshared	300	58.7
SS	2k	RPN+ZF	100	55.1
SS	2k	RPN+ZF	300	56.8
SS	2k	RPN+ZF	1k	56.3
SS	2k	RPN+ZF (no NMS)	6k	55.2
SS	2k	RPN+ZF (no *cls*)	100	44.6
SS	2k	RPN+ZF (no *cls*)	300	51.4
SS	2k	RPN+ZF (no *cls*)	1k	55.8
SS	2k	RPN+ZF (no *reg*)	300	52.1
SS	2k	RPN+ZF (no *reg*)	1k	51.3
SS	2k	RPN+VGG	300	59.2

Figura 2.4: Desempeño F-CNN en la detección de objetos de PASCAL VOC 2007, utilizando diferentes métodos de propuesta para entrenamiento y realización de pruebas. (Huaizu y Erik, 2016)

A su vez, en el trabajo de investigación Face Detection with the Faster R-CNN (Huaizu y Erik, 2016) se puede observar su desempeño dentro del marco de

[1]PASCAL VOC es un conjunto estandarizado de datos para el reconocimiento y clasificación de imágenes. https://paperswithcode.com/dataset/pascal-voc-2007

reconocimiento facial. El mismo fué realizado utilizando los conjuntos de datos Face Detection Dataset and Benchmark (FDDB)[2] y IJB-A benchmark[3].

En el sitio web asociado a FDDBpuede verse Fast R-CNN posicionado como el segundo modelo con mejor desempeño, logrando un Average Precision (AP)[4] de 0.990.

Mask R-CNN

La arquitectura Mask R-CNN (Kaiming, Georgia, Piotr, y Ross, 2018) es una extensión de Fast R-CNN, agrega una rama en la cual se intenta predecir una máscara para el objeto que se está detectando al mismo tiempo que se reconoce el rectángulo delimitador. De esta forma, utiliza en una etapa inicial la misma CNN de propuesta de regiones que Fast R-CNN y en la segunda etapa además de la clasificación y refinamiento de coordenadas, genera paralelamente la máscara.
En este enfoque se combina la detección de objetos y la segmentación semántica, la cuál busca clasificar cada píxel según un conjunto de categorías.
En la tabla 2.5 puede verse el desempeño de Mask R-CNN al utilizar distintos umbrales, y distintos tamaños del dataset, frente a otros métodos o modelos de segmentación, se utilizó para la comparación el dataset COCO[5], el cual es un conjunto de datos utilizado para medir desempeño.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Figura 2.5: Desempeño de Mask R-CNN frente a Multi-task Network Cascade (MNC), ganador del challenge de segmentación COCO 2015 y Fully Convolutional Instance-aware Semantic Segmentation (FCIS), ganador del challenge de segmentación COCO 2016. (Kaiming y cols., 2018)

Single Shot MultiBox Detector (SSD)

SSD (Wei y cols., 2016) es un método de detección de objetos en imágenes basado en la utilización de una única red neuronal profunda. A diferencia de los

[2]Conjunto de imágenes etiquetadas de rostros, con distintas resoluciones, ángulos y desenfoques. Incluye imágenes en escala de grises y a color. https://paperswithcode.com/dataset/fddb

[3]Conjunto de imágenes de rostros con variaciones de posición, iluminación, distintas expresiones, resolución y oclusión. https://paperswithcode.com/dataset/ijb-a

[4]AP es una métrica utilizada para evaluar los modelos de detección de objetos.

[5]COCO es un conjunto de datos de detección, segmentación y subtítulos de objetos a gran escala. https://cocodataset.org

métodos presentados anteriormente, en este caso se elimina la etapa de generación de propuestas y refinamiento de características o píxeles, efectuando todo el cálculo en la única red.

Su funcionamiento se da utilizando la red neuronal para generar una cantidad determinada de rectángulos ponderados que delimitan las instancias de clases de objetos encontradas, para luego obtener las detecciones finales.
Se puede observar en la tabla 2.6 como se desempeña SSD sobre el dataset COCO en comparación con otros métodos.

Method	data	Avg. Precision, IoU:			Avg. Precision, Area:			Avg. Recall, #Dets:			Avg. Recall, Area:		
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast [6]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast [24]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster [2]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [24]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster [25]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**	**24.8**	**37.5**	**39.8**	**14.0**	**43.5**	**59.0**

Figura 2.6: Desempeño SSD comparado con Faster R-CNN en el conjunto COCO del año 2015. SSD512 supera a Faster R-CNN en AP al 0.75 y en la detección de objetos grandes, pero tiene una mejora menor en AP al 0.5 y en la detección de objetos pequeños. (Wei y cols., 2016)

You Only Look Once (YOLO)

Entre estos algoritmos también se encuentra You Only Look Once (YOLO) (Joseph y Ali, s.f.). El mismo aplica una única red neuronal sobre la imagen, con esto se logra dividir la imagen en secciones (se les llama cajas de detección y se predicen cuatro vértices para su construcción), clasificar las mismas en clases a las cuáles se las pondera en base a cierta probabilidad y lograr detectar un objeto. Para el entrenamiento de sus redes neuronales utilizan el framework Darknet.

En la presente tabla 2.7 se visualiza la comparativa de YOLOv3 y otros enfoques de reconocimiento de objetos según la métrica Average Precision (AP) utilizando el conjunto de datos COCO.

Método elegido

En conclusión, a partir de los métodos de detección presentados, se procedió con la elección del último presentado YOLO, existiendo versiones más modernas como la versión 8[6].

[6]Yolo V8 github release https://github.com/ultralytics/ultralytics/tree/v8.2.50

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Figura 2.7: Desempeño YOLOv3 comparado con otros metodos de detección en el dataset COCO (Joseph y Ali, s.f.)

Capítulo 3

Relevamiento de requerimientos

En este capítulo se describe el relevamiento de requerimientos para la herramienta de anonimización de imagenes que se busca implementar. En el proceso de determinación de los requisitos para el presente proyecto, se llevaron a cabo una serie de reuniones con el equipo de investigación. A lo largo de estos encuentros, los investigadores nos transmitieron sus necesidades.

A su vez, dado que llevamos adelante una metodología de trabajo de tipo ágil, resulta crucial contar con una etapa de validación de requerimientos en la cual los usuarios finales puedan brindar un feedback temprano que afecte positivamente al desarrollo de las funcionalidades, la misma también será abordada dentro de este capítulo.

3.1. Requerimientos funcionales

Las historias de usuario se definen en base a las funcionalidades esenciales que la herramienta debe incorporar y se elaboran en reuniones con el equipo de investigadores. A continuación, se presentan las siguientes historias de usuarios relevadas:

3.1.1. Sesión

- Como investigador registrado en la plataforma, quiero poder iniciar sesión con mi usuario y contraseña, para acceder a las funcionalidades de la plataforma.

- Como investigador autenticado en la plataforma, quiero poder cerrar sesión en cualquier momento, para asegurar la privacidad de mi cuenta.

3.1.2. Configuración de usuario

- Como investigador autenticado en la plataforma, quiero poder modificar mis datos de usuario, para mantener actualizada mi cuenta.
- Como investigador autenticado en la plataforma, quiero poder cambiar mi contraseña en cualquier momento, para mejorar la seguridad de mi cuenta.
- Como investigador registrado en la plataforma, quiero poder restablecer mi contraseña si la olvido o si necesito cambiarla por motivos de seguridad, para recuperar el acceso a mi cuenta y mantenerla segura.

3.1.3. Anonimización manual

- Como investigador autenticado en la plataforma, quiero poder cargar un conjunto de imágenes desde el sistema de archivos, para aplicar filtros de ofuscación seleccionables y enviar las imágenes procesadas al servidor.
- Como investigador autenticado en la plataforma, quiero poder tener opciones para descargar las imágenes anonimizadas y guardarlas en mi sistema de archivos.

3.1.4. Anonimización automática

- Como investigador autenticado en la plataforma, quiero poder cargar un conjunto de imágenes desde el sistema de archivos, para aplicar métodos de ofuscación automáticamente en zonas que un modelo entrenado considere que son regiones de interés.
- Como investigador autenticado en la plataforma, quiero poder tener opciones para descargar las imágenes anonimizadas y guardarlas en mi sistema de archivos.

3.2. Requerimientos no funcionales

Junto con los requerimientos funcionales, surgen algunos requerimientos no funcionales propios de desarrollar un prototipo dentro del marco de este proyecto. Estos son:

- **Software libre:** La herramienta debe priorizar el uso de software libre. En particular se buscan herramientas y tecnologías que respeten los principios de código abierto y la libertad de uso, modificación y distribución.
- **Infraestructura portable:** La herramienta de anonimización debe ser versátil, permitiendo su despliegue tanto en entornos locales como en servicios en la nube, proporcionando escalabilidad y accesibilidad remota para usuarios distribuidos geográficamente.

- **Seguridad y privacidad:** Dado que la herramienta opera con datos sensibles, es de suma importancia que no almacene ni los datos de entrada ni los resultados de la anonimización, garantizando así la máxima confidencialidad y cumplimiento de normativas de protección de datos.
- **Rendimiento:** La herramienta debe demostrar una capacidad robusta para procesar y anonimizar imágenes de manera eficiente, asegurando tiempos de respuesta rápidos y consistentes incluso bajo cargas de trabajo elevadas, lo cual es crucial para mantener la eficiencia y la efectividad en el tratamiento de datos visuales sensibles.
- **Usabilidad:** La herramienta debe ser diseñada con una interfaz intuitiva y fácil de usar, proporcionando una experiencia amigable que permita a los usuarios interactuar sin complicaciones, asegurando así una adopción rápida y eficiente del proceso de anonimización de imágenes.

Se lograron cumplir casi todos los requerimientos. La evaluación de si un tiempo de respuesta se considera rápido depende del contexto específico del procesamiento de imágenes. Como se detalla en secciones posteriores, los tiempos de respuesta observados con la infraestructura disponible se consideraron exitosos.

3.3. Prueba de concepto

En esta sección se presentan los resultados de una prueba de concepto realizada al inicio del desarrollo, cuyo objetivo es evaluar la funcionalidad central del proyecto: la anonimización manual de imágenes.

Este componente es fundamental no solo por su capacidad para proteger la privacidad de datos visuales sensibles, sino también por su interfaz diseñada para facilitar la interacción del usuario. El objetivo principal de esta fase es validar los requisitos básicos de usabilidad y funcionalidad, permitiendo que los usuarios interactúen directamente con la herramienta. Esta interacción proporciona retroalimentación esencial para mejorar la experiencia de usuario y sirve como un indicador clave para evaluar la alineación del desarrollo con los objetivos y expectativas del proyecto.

Además, se busca garantizar que los usuarios puedan interactuar de manera intuitiva con la herramienta, validando su capacidad para anonimizar imágenes de forma manual y evaluando su usabilidad. Los objetivos específicos incluyen:

- Validar la capacidad de la herramienta para cargar y anonimizar imágenes manualmente.
- Evaluar la interfaz de usuario en términos de facilidad de uso y accesibilidad.
- Recibir retroalimentación de los usuarios para alinear el desarrollo de la herramienta con los objetivos y expectativas del proyecto.

Para realizar la prueba de concepto, se desarrollaron e implementaron una interfaz de usuario inicial (ver Figura 3.1) y las siguientes funcionalidades:

- Carga individual de imágenes para anonimizar manualmente.
- Selección de método e intensidad de ofuscación.
- Delimitación de zonas a anonimizar.
- Visualización de imágenes anonimizadas.
- Descarga individual de imágenes anonimizadas.

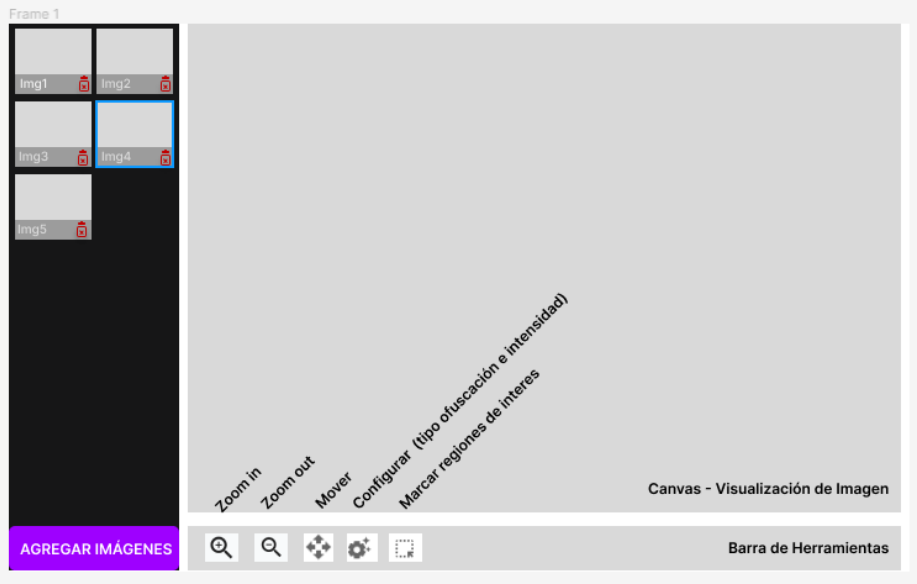


Figura 3.1: Interfaz de Usuario

Los resultados de la prueba de concepto proporcionan valiosa retroalimentación de los investigadores de EDIGA, destacando varios aspectos clave para mejorar la herramienta:

- **Carga de imágenes**: Los investigadores de EDIGA expresan interés en la capacidad de cargar múltiples imágenes simultáneamente, lo que facilita el proceso de anonimización en lotes.
- **Configuración de ofuscación**: Se sugiere separar claramente las opciones para configurar el tipo y el grado de ofuscación en la barra de herramientas, mejorando la usabilidad y reduciendo errores.

- **Descarga de imágenes**: Los investigadores de EDIGA valoran la opción de descargar imágenes individualmente, pero también manifiestan la necesidad de descargar todas las imágenes anonimizadas en un solo archivo.
- **Claridad de textos**: Se observa que algunos textos (botones o tooltips) no son suficientemente claros, lo que afecta la experiencia del usuario.

A partir de los resultados obtenidos, se decide agregar los siguientes requerimientos funcionales:

- Permitir a los investigadores autenticados seleccionar múltiples imágenes para cargar en la herramienta de anonimización manual.
- Permitir a los investigadores autenticados seleccionar múltiples imágenes para cargar en la herramienta de anonimización automática.
- Permitir la descarga de todas las imágenes anonimizadas manualmente en un archivo .zip.
- Permitir la descarga de todas las imágenes anonimizadas automáticamente en un archivo .zip.
- Permitir seleccionar un conjunto de imágenes anonimizadas manualmente y descargarlas en un archivo .zip.
- Permitir seleccionar un conjunto de imágenes anonimizadas automáticamente y descargarlas en un archivo .zip.

Respecto a los requerimientos no funcionales, en base a la retroalimentación se pudieron tomar decisiones respecto a la parte de usabilidad:

- Rediseñar la barra de herramientas para lograr independencia entre las funcionalidades que ofrece y facilitar un uso más sencillo, en particular la selección de tipo y grado de ofuscación.
- Ajustar algunos textos para hacerlos más mnemotécnicos, lo cual contribuye significativamente a mejorar la usabilidad.

Si bien se ha tomado en cuenta que los investigadores de EDIGA podrían desear marcar áreas de interés dibujando libremente en lugar de utilizar los polígonos que brinda la herramienta, se considera que esto queda fuera del alcance del proyecto en la fase actual.

Capítulo 4

Análisis de Herramientas

En este capítulo se presentan las herramientas analizadas disponibles hasta la fecha. Se evalúan cuatro herramientas en el mercado que ofrecen diferentes métodos para anonimizar conjuntos de imágenes, ya sea de forma automática o manual. Sobre el conjunto de datos total que existía en Kaggle (aproximadamente 8500 imágenes), se decidió trabajar con un subconjunto de 50 imágenes (Gupta, s.f.) Las herramientas sobre las cuales se trabajó fueron: Celantur (*Celantur*, s.f.), Api4AI (*Api4AI*, s.f.), Fiji ImageJ2 (*Fiji Image2J*, s.f.), OpenCV (*OpenCV*, s.f.). Estas herramientas, brindan posibilidad de anonimizar imágenes de forma manual o automática, brindando alguna de las principales características de casos de uso que se relevaron.

Para evaluar las herramientas se acotaron los problemas, tratando de enfocarse en una solución del tipo "Divide y Vencerás" y de esta forma asignar puntos a cada herramienta y valorizándolas en cuanto a las soluciones, la modularidad que brindan, y asignar evaluarlas de la forma mas equitativa posible.

Estándares comparativos

Para establecer los estándares de comparación de las herramientas, partimos de los requerimientos principales anteriormente descriptos, dónde se enfatizó en:

- Seguridad de los datos, no deben de ser publicados a terceros
- Posibilidad de anonimización manual
- Posibilidad de realizar anonimización automática
- Debe de ser realizable con software libre

4.1. Enfoque de la evaluación

Una vez establecidos los estándares de comparación, se analizaron las fortalezas y debilidades de algunas herramientas existentes, a partir de esto, se

obtuvo como resultado el seleccionar 4 herramientas para ser analizadas en más profundidad para utilizar en el trabajo.

Estas herramientas son:

- OpenCV: Es una biblioteca de código abierto[1] especializada en procesamiento de imágenes. Ofrece una amplia gama de funciones y algoritmos que permiten a los desarrolladores trabajar con imágenes y vídeos de manera eficiente. Desde la detección de rostros hasta la manipulación de imágenes, OpenCV es una herramienta versátil utilizada en aplicaciones de aprendizaje automático.

- Fiji ImageJ2: Fiji, basado en ImageJ2, es un software de código abierto[2] utilizado en el procesamiento de imágenes que se centra en la visualización y análisis de datos de imágenes científicas. Diseñado para ser extensible y modular, Fiji simplifica tareas complejas en la investigación biológica y médica.

- Celantur: Es un software especializado en la anonimización de datos, centrándose en la protección de la privacidad y confidencialidad de la información. Es utilizado para ocultar o enmascarar datos sensibles, especialmente en conjuntos de datos utilizados en investigación y análisis. Sobre esta herramienta no se encontró documentación de su licencia, pero si se pudo utilizar, entendiendo que es de código cerrado.

- Api4AI: Es una API diseñada para facilitar la integración de funciones de inteligencia artificial en aplicaciones y sistemas. Ofrece acceso a una variedad de algoritmos de aprendizaje automático y procesamiento de lenguaje natural, permitiendo a los desarrolladores aprovechar las capacidades avanzadas de la inteligencia artificial en sus propias aplicaciones sin la necesidad de desarrollar algoritmos desde cero. Sobre esta herramienta tampoco se encontró documentación de su licencia, entendiendo que es de código cerrado.

[1] Licencia Apache 2.0
[2] Licencia GPL 3.0

4.2. Entorno de la comparación

Para construir la comparación de herramientas, de forma tal de evaluar las imágenes en forma simultánea, se enfocó en minimizar la redundancia de los componentes, optimizar el uso de recursos durante la ejecución y reducir la complejidad de los mismos. En esta sección se nombran brevemente las tecnologías y protocolos utilizados en el desarrollo de la comparación, así como un diagrama de componentes utilizados.

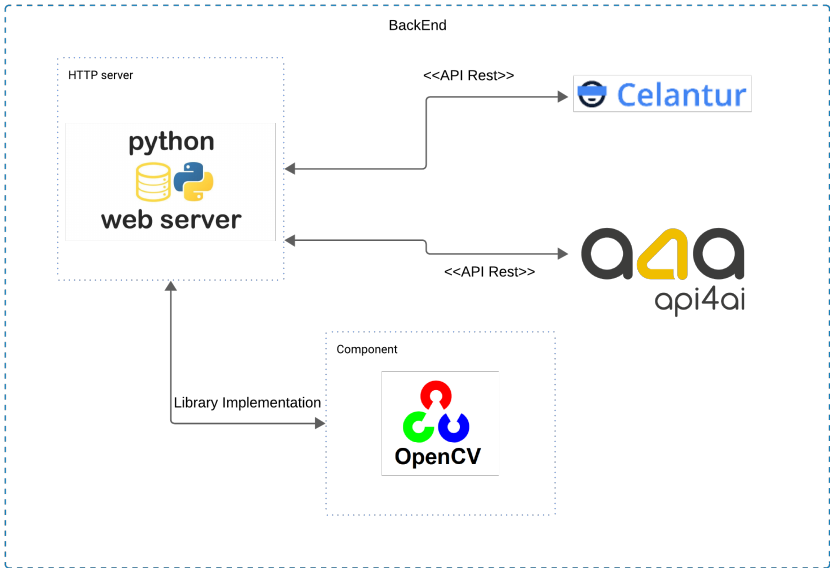


Figura 4.1: Implementación para la comparación de herramientas.

Como se observa en la Figura 4.1 para aquellos casos donde es necesario realizar un desarrollo adicional para poder utilizar o consumir los servicios de las herramientas herramientas, se empleó en Python el desarrollo de los módulos necesarios. Cuando las herramientas proporcionaban un servicio de API REST, los módulos implementan la función de generar la solicitud donde se incluía la imágen, y manejar la respuesta recibida. Para el caso en el que las herramientas formaban parte de una librería, los módulos desarrollados cumplen la función de orquestar todo el proceso haciendo uso de las funciones que aporta la librería para el manejo de imágenes.

4.3. Pasos realizados para la comparación

Construcción del Set de Datos: Se comienza con lo mencionado anteriormente en la Sección 2.1, que es la elección de un subconjunto de 50 rostros humanos de dominio público (Gupta, s.f.). Con este conjunto de datos, se obtienen métricas sobre la calidad de anonimización de las distintas herramientas.

4.4. Desarrollo de cada herramienta

A continuación se muestra la implementación realizada para cada herramienta, utilizando el Set de datos descrito anteriormente

Celantur: Esta solución de software es especializada en la anonimización de datos. Para la implementación se registró en el sitio y obtuvieron las credenciales necesarias para acceder a la API, de esta manera autenticarse utilizando email, contraseña e incluir la clave única para el consumo de los servicios. A continuación, se configuró el entorno de desarrollo, en este caso nuestro servidor en Python. Esta herramienta cuenta con una extensa API por lo que resultó indispensable consultar la documentación oficial de Celantur (*Celantur*, s.f.) para entender los métodos, los endpoints y los distintos parámetros disponibles para la anonimización de imágenes.

Con el objetivo de evaluar la eficacia de la herramienta, se realizó un conjunto de pruebas y una demostración como se muestra en la Figura 4.2, utilizando diferentes rangos de parámetros. Este enfoque de modificación de parámetros fue de utilidad para poder obtener y comparar los resultados obtenidos.

(a) Imágen original (b) Imagen resultante

Figura 4.2: Demostración de herramienta Celantur

Fiji ImageJ2: Este software (*Fiji Image2J*, s.f.) instalable en una PC, que permite aplicar distintos filtros a una imagen, como por ejemplo Blur Gaussiano, Bloqueo o Noise, entre otros. Para realizar esto, se selecciona una imagen desde el sistema de archivos y se aplica un bounding box definidos por el usuario, seguido del filtro deseado, resultando en la imagen anonimizada.

En el ejemplo de la Figura 4.3 se utilizó un desenfoque gaussiano. Esta opción permite definir un parámetro 'Sigma', el cual define la profundidad del desenfoque. A mayor Sigma, mayor será el desenfoque aplicada.

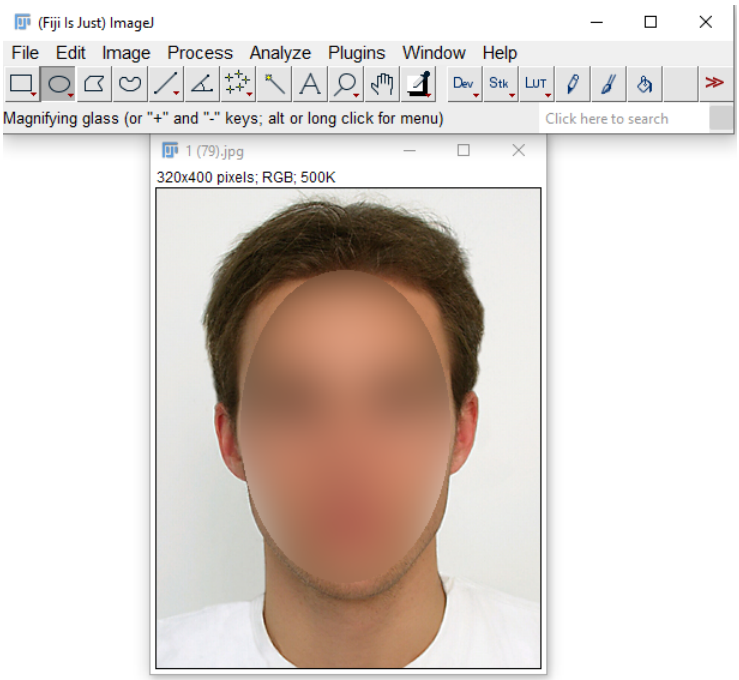


Figura 4.3: Blur Gaussiano utilizando Fiji ImageJ2

OpenCV: Para aplicar una posible solución con OpenCV (*OpenCV*, s.f.), se implementaron distintos módulos para la anonimización de las imágenes. Estos módulos se utilizaron de forma secuencial o iterativa dependiendo de los parámetros ingresados, construyendo así un pipeline de procesamiento de imágenes.

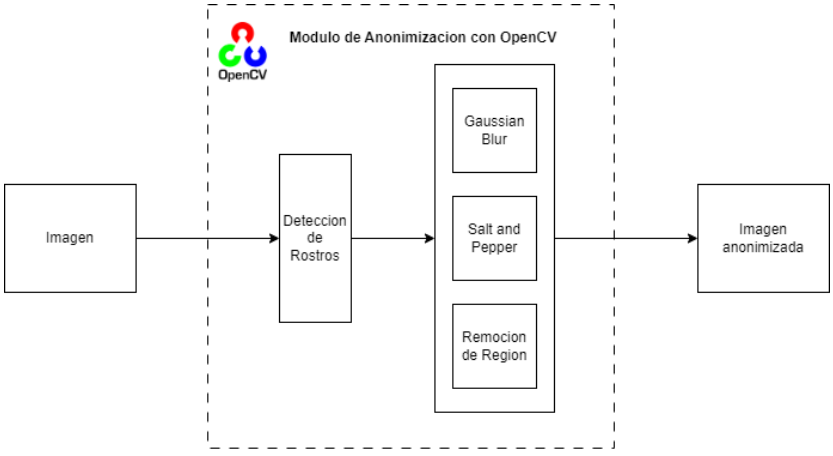


Figura 4.4: Pipeline de OpenCV para el tratamiento de imágenes

Como se muestra en la Figura 4.4, el módulo compuesto por los distintos métodos para agregar ruido o eliminar una porción de las imágenes permite personalizar cómo se desea realizar este proceso. Esta flexibilidad facilita la completa modificación del módulo para su reutilización en el desarrollo del prototipo.

Aunque OpenCV permite una manipulación de imágenes fácil y completa, se requiere un desarrollo adicional para convertir esta solución en algo completamente funcional y accesible para cualquier usuario sin conocimientos técnicos.

Api4AI: La plataforma en la nube de Api4AI ofrece una amplia gama de herramientas, entre las cuales se encuentra la funcionalidad de anonimización de imágenes, como se describe en su sitio web (*Api4AI*, s.f.) como una solución para la identificación de áreas sensibles en imágenes y la aplicación automática de difuminado. Esta herramienta tiene la capacidad de identificar rostros en imágenes y aplica una máscara de difuminado de forma automática, con el fin de volver esté contenido sensible en las imágenes irreconocibles. Para lograr este cometido la herramienta utiliza algoritmos de aprendizaje automáticos y de esta forma detectar las áreas donde se encuentran las caras dentro de una imagen, estas áreas se denominan bounding boxes y son utilizadas más tarde para determinar donde aplicar la mascara de difuminado.

Esta herramienta cuenta con una API que ofrece un servicio en la cual se envía un imagen y se obtiene como resultado la imagen luego de haber sido procesada (aplicar difuminado en caso de detectar una cara) y metadata donde se agrega información sobre el procesamiento de dicha imagen.

4.5. Fortalezas y debilidades de cada herramienta

En esta Sección se presentan las fortalezas y debilidades propias de cada herramienta, donde se identifican algunos requerimientos que no se encontraban directamente relevados, y al realizar las pruebas sobre estas herramientas, obtuvimos algunos como lo son:

- Facilidad de uso
- Facilidad de integración
- Cantidad de opciones de anonimización
- Documentación fácilmente accesible
- Comunidad activa sobre la herramienta

A continuación, se muestran las tablas 4.1, 4.2, 4.3 y 4.4 que detallan las fortalezas y debilidades de las herramientas Fiji ImageJ2, OpenCV, Api4AI, Cenaltur en ese orden.

Tabla 4.1: Fiji ImageJ2

Fortalezas	**Debilidades**
▪ Fácil utilización ▪ Interfaz de usuario amigable ▪ Permite definir bounding boxes personalizados ▪ Instalación rápida y sencilla ▪ Gran abanico de opciones de modificación de imágenes	▪ Difícil de integrar con otras herramientas ▪ No es posible utilizar de forma automatizada ▪ Poco eficiente a la hora de procesar un gran volumen de datos ▪ No ofrece métricas que midan el nivel de anonimización, ni el de re-identificación

Tabla 4.2: OpenCV

Fortalezas	**Debilidades**
▪ Ofrece varios modos de anonimización ▪ Permite una completa manipulación de imágenes ▪ Fácil de integrar con otras herramientas ▪ Extensa documentación que permite utilizar de forma precisa la herramienta ▪ Permite procesar un gran volumen de datos	▪ Requiere de desarrollo en python ▪ Requiere conocimiento de la librería ▪ Procesos complejos requieren soluciones a través de pasos intermedios ▪ La detección de rostros no es su principal funcionalidad

Tabla 4.3: Api4AI

Fortalezas	**Debilidades**
▪ Utiliza una Api web ▪ Fácil de integrar con otras herramientas debido a la documentación ▪ Permite procesar un gran volumen de datos ▪ Devuelve información relevante, como por ejemplo el porcentaje de detección de rostros	▪ Un solo método de anonimización ▪ Herramienta paga ▪ Requiere de desarrollo de código que consuma la Api de procesamiento de imágenes

Tabla 4.4: Celantur

Fortalezas	**Debilidades**
▪ Ofrece varios métodos para la anonimización ▪ Devuelve información relevante, como por ejemplo porcentaje de detección de rostros, nivel de re-identificación, y el bounding box aplicado ▪ Mantiene altos niveles de confidencialidad a la hora de anonimizar ▪ Fácil de integrar ▪ Permite procesar un gran volumen de datos ▪ Buena documentación	▪ Tiene una prueba gratis por tiempo limitado ▪ Herramienta paga

4.6. Conclusiones del relevamiento de herramientas

Al lograr el objetivo principal del análisis, que consiste en investigar el abanico de tecnologías actuales y proporcionar comparaciones objetivas de las herramientas disponibles, se concluye que se debe utilizar OpenCV para el desarrollo principal del prototipo. Esto se debe a la facilidad de implementación y reutilización de lo ya existente, así como a la seguridad de que el manejo de los datos queda limitado a lo implementado y no se comparte con terceros.

Capítulo 5

Diseño de la solución

En este capítulo se describe el diseño de la solución, en particular la arquitectura implementada, el tipo y modelo de base de datos utilizados, y se explica el motivo de estas decisiones. Un aspecto a tener en cuenta es que el diseño se ve fuertemente afectado por la necesidad de manejar la menor cantidad de datos de usuario posible y por la solicitud de los investigadores de no persistir datos o metadatos asociados a las imágenes dentro de la aplicación.

5.1. Arquitectura

La arquitectura del proyecto se organiza en tres módulos principales: la aplicación web, el servidor y la base de datos. Estos se muestran en la Figura 5.1.

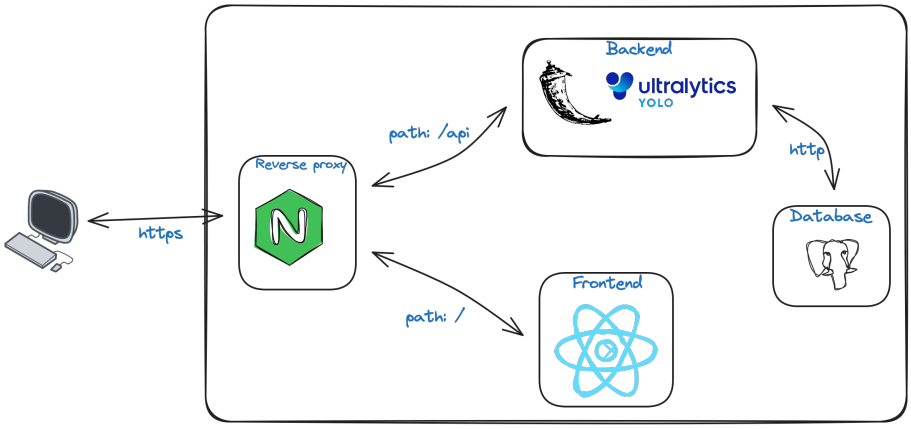


Figura 5.1: Diagrama de arquitectura del sistema

El patrón de diseño que utiliza esta arquitectura es conocido como arquitectura en capas. En este enfoque, la aplicación se organiza en diferentes capas,

cada una con una responsabilidad específica y comunicándose únicamente con las capas adyacentes. Esta estructura promueve la separación de responsabilidades, lo que facilita el desarrollo, el mantenimiento y la escalabilidad del sistema.

La capa de presentación o usuario (frontend), está desarrollada en React. React es una biblioteca de JavaScript que permite crear aplicaciones de una sola página (SPA) de manera eficiente. Esta tecnología se basa en componentes reutilizables y auto-contenidos, lo que facilita la modularidad y el mantenimiento del código. Además, React utiliza un Virtual DOM para optimizar el rendimiento, realizando actualizaciones mínimas en el DOM real y mejorando la velocidad de la aplicación. React promueve un flujo de datos unidireccional, lo que hace que el estado de la aplicación sea más predecible y sencillo de gestionar. Con la introducción de hooks, React permite el uso de características avanzadas sin necesidad de clases, facilitando así el manejo del estado y otros efectos secundarios en los componentes funcionales.

La capa de lógica de negocio (backend) está desarrollada utilizando Flask. Flask es un framework minimalista de Python que facilita el desarrollo de aplicaciones web y servicios API de manera rápida y sencilla. Permite una gestión eficiente de rutas y controladores para manejar las solicitudes HTTP, y es conocido por su flexibilidad y capacidad para integrarse con diversas extensiones que amplían su funcionalidad. Su estructura minimalista permite elegir las herramientas y bibliotecas que mejor se adapten a las necesidades del sistema, evitando la sobrecarga de funcionalidades innecesarias. Además, Flask es ideal para la autenticación y autorización de usuarios, el procesamiento de datos y la validación de entradas, lo que asegura que la aplicación funcione de manera correcta y segura. El acceso a los datos se realiza a través de extensión SQLAlchemy, el ORM (Object-Relational Mapper) utilizado por Flask. SQLAlchemy facilita la interacción con la base de datos, permitiendo trabajar con bases de datos mediante objetos de Python. Cada tabla se mapea a una clase y cada columna a un campo de dicha clase. Además, SQLAlchemy permite mapear las relaciones entre tablas como relaciones entre objetos.

La capa de datos (database) está implementada utilizando PostgreSQL, un sistema de gestión de bases de datos relacional de código abierto, conocido por su robustez y escalabilidad. Esta capa se encarga de almacenar y gestionar los datos persistentes de la aplicación, proporcionando un entorno seguro y eficiente para la manipulación de datos. En esta base de datos, se alojarán exclusivamente los datos relacionados con los usuarios y la gestión de tokens de acceso.

En nuestra arquitectura del sistema, utilizamos un reverse proxy configurado sobre Nginx para servir nuestra aplicación web y exponer el backend de forma segura. Nginx es un servidor web de código abierto conocido por su alta velocidad, eficiencia y capacidad para actuar como reverse proxy, balanceador de carga y caché HTTP. Es ampliamente utilizado para servir aplicaciones web, APIs y gestionar el tráfico web de manera eficiente y segura.

Su función de reverse proxy, nos permite gestionar todas las solicitudes entrantes y distribuyéndolas a los diferentes servicios del sistema. Este proceso lo realiza de forma segura utilizando certificados SSL que son proporcionados por Let's Encrypt, una autoridad de certificación gratuita y automatizada que nos

garantiza que todas las comunicaciones sean encriptadas y seguras, protegiendo los datos sensibles durante su transmisión.

5.2. Modelo relacional

Como se menciona anteriormente, en la base de datos relacional solo se almacenan datos para la gestión de usuarios y tokens de acceso. Como se puede observar en la Figura 5.2, que muestra el diagrama MER (Modelo Entidad-Relación), el modelo relacional es bastante sencillo y cuenta con las siguientes entidades:

- User: Esta entidad modela a los usuarios del sistema e incluye los siguientes datos: nombre, apellido, email y contraseña.
- Token_blocklist: Esta entidad modela los tokens de acceso generados por los usuarios que ya no son válidos y no pueden ser usados para autenticarse en el sistema. Incluye el identificador del token y una referencia al usuario al cual le pertenece.

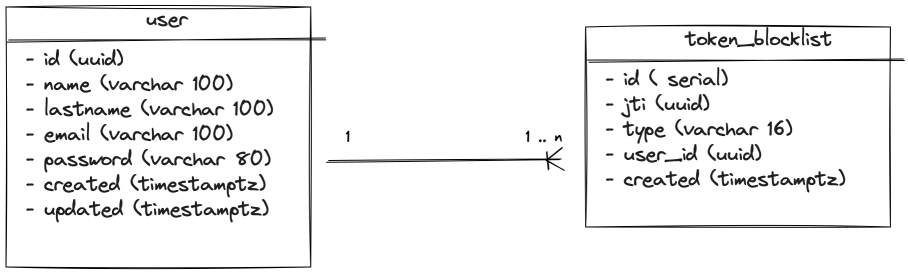


Figura 5.2: Modelo Relacional

5.3. Conclusión

La arquitectura en capas adoptada en este proyecto ofrece una estructura sólida que facilita el desarrollo, mantenimiento y escalabilidad del sistema. La elección de tecnologías como React para el frontend, Flask para el backend y PostgreSQL para la capa de datos asegura una implementación eficiente y flexible, manteniendo un rendimiento óptimo y una gestión segura de la información.

El diseño minimalista de la base de datos, limitado a los datos esenciales de usuarios y tokens de acceso, responde a la preferencia de los usuarios finales de almacenar la menor cantidad de datos posible. Este enfoque no solo simplifica el sistema, sino que también refuerza la privacidad y seguridad, evitando la persistencia de datos innecesarios como fotos u otros detalles personales. En conjunto,

esta arquitectura equilibra la funcionalidad y la seguridad, adaptándose a las necesidades del usuario y garantizando la integridad y eficiencia del sistema.

Capítulo 6

Tecnologías y herramientas utilizadas

En este capítulo se describen todas las tecnologías y herramientas utilizadas a lo largo del proyecto.

6.1. Investigación de tecnologías

A continuación, se detalla la investigación realizada para definir las tecnologías utilizadas en el desarrollo del proyecto.

6.1.1. Criterios de selección

Para la elección de las tecnologías a utilizar, se definió un conjunto de criterios a tener en cuenta durante su evaluación. A continuación, se describe el conjunto de criterios resultante:

- Experiencia previa: Considerar el conocimiento y la familiaridad del equipo con las tecnologías, lo que puede facilitar el desarrollo y reducir el tiempo de aprendizaje.
- Software libre: Priorizar el uso de software libre para garantizar la accesibilidad, la posibilidad de personalizaciónes y la reducción de costos de licencias.
- Comunidad activa: Elegir tecnologías respaldadas por una comunidad activa, lo que asegura soporte continuo, actualizaciones frecuentes

6.2. Frontend

Para el frontend de nuestra aplicación web, hemos elegido utilizar React, una biblioteca de JavaScript ampliamente reconocida por su eficiencia en la creación

de interfaces de usuario dinámicas y escalables. A continuación se presentan las tecnologías y herramientas utilizadas en el frontend.

6.2.1. React

React [1] es una biblioteca de JavaScript de código abierto desarrollada por Facebook, conocida por su eficiencia y rendimiento en la construcción de interfaces de usuario interactivas y dinámicas, especialmente en aplicaciones de una sola página (SPA). Al ser un proyecto de código abierto[2], React es desarrollado y mantenido por Facebook en colaboración con una comunidad global de desarrolladores. Esto permite que su código fuente sea accesible públicamente para que cualquiera pueda verlo, modificarlo y contribuir a su desarrollo. Los desarrolladores no solo utilizan la biblioteca, sino que también tienen la capacidad de mejorarla, corregir errores, proponer nuevas funcionalidades y adaptarla a diferentes necesidades y plataformas.

React utiliza un enfoque de desarrollo basado en componentes, donde la interfaz de usuario se descompone en pequeños elementos reutilizables. Estos componentes pueden renderizarse de manera independiente según el estado de la aplicación, lo que simplifica la gestión y actualización eficiente de la interfaz. React utiliza un DOM virtual (Virtual DOM) para optimizar las actualizaciones del DOM real. Este DOM virtual actúa como una capa intermedia entre el estado de la aplicación y el DOM real del navegador. Cuando ocurre un cambio en el estado de la aplicación, React actualiza primero el DOM virtual en lugar del DOM real. Luego, compara estos dos DOM para determinar las diferencias y actualiza solo los elementos necesarios en el DOM real. Esta estrategia minimiza las operaciones costosas de manipulación del DOM real, mejorando así significativamente el rendimiento de las aplicaciones desarrolladas con React. El uso del DOM virtual no solo optimiza el rendimiento, sino que también simplifica el desarrollo al proporcionar una abstracción eficiente y predecible del proceso de renderizado del DOM, permitiendo a los desarrolladores enfocarse en la lógica de la aplicación sin preocuparse por detalles innecesarios de renderizado.

6.2.2. Fabric

Fabric.js[3] es una biblioteca JavaScript robusta que facilita el manejo interactivo de imágenes y el marcado de regiones de interés en aplicaciones web. Con Fabric.js, los usuarios pueden cargar, visualizar y manipular imágenes de manera eficiente dentro de un lienzo HTML5. Sus ventajas principales respecto al proyecto a desarrollar incluyen:

- **Manipulación Interactiva:** Permite el movimiento, redimensionamiento y rotación fluidos de imágenes y objetos dentro del lienzo, lo que facilita la precisión en el marcado de regiones.

[1] https://react.dev
[2] https://github.com/facebook/react
[3] https://fabricjs.com/

- **Marcado de Regiones:** Proporciona herramientas para dibujar formas personalizadas, como rectángulos y polígonos, sobre las imágenes. Esto es fundamental para identificar y delimitar áreas específicas de interés.

- **Personalización Avanzada:** Ofrece opciones detalladas para modificar propiedades visuales como color, transparencia y estilos de borde, lo que permite adaptar el marcado según las necesidades específicas del proyecto.

- **Compatibilidad y Facilidad de Integración:** Es compatible con múltiples navegadores y se integra fácilmente con otras bibliotecas y frameworks de JavaScript, lo que simplifica el desarrollo y la implementación de aplicaciones web interactivas.

- **Documentación Completa y Activa Comunidad:** Cuenta con una documentación extensa y una comunidad activa de desarrolladores, lo que facilita la resolución de problemas y la adopción de nuevas funcionalidades.

6.2.3. Material UI

Material-UI[4] es una biblioteca de componentes de interfaz de usuario diseñada para React y otros frameworks de JavaScript. Proporciona componentes preconstruidos y estilizados que siguen las directrices de diseño de Material Design de Google. Estos componentes incluyen botones, sliders, barras de progreso, entre otros, que mejoran la experiencia del usuario al ofrecer una interfaz moderna y consistente. Material-UI facilita el desarrollo de aplicaciones web al permitir una fácil personalización a través de temas y estilos, asegurando coherencia visual en toda la aplicación. Es ampliamente utilizada debido a su documentación completa, comunidad activa y soporte robusto, lo que facilita la adopción y el mantenimiento de aplicaciones web de alta calidad estéticamente.

6.2.4. JSZip y FileSaver

JSZip y FileSaver.js son bibliotecas JavaScript utilizadas para la manipulación y descarga de archivos en formato .zip directamente desde el navegador. JSZip permite la creación dinámica de archivos .zip a partir de múltiples archivos o datos, optimizando la transferencia y el almacenamiento de información. Por otro lado, FileSaver.js proporciona métodos simples para guardar archivos generados en el navegador en el sistema de archivos local del usuario. Estas bibliotecas son esenciales para aplicaciones web como la que se desea implementar ya que permiten descargar conjuntos de datos, como imágenes, de manera eficiente y sin depender de servidores adicionales, mejorando así la experiencia del usuario y la eficiencia en la gestión de archivos.

[4] https://mui.com/material-ui/

6.3. Backend

Para el backend del prototipo, se optó por utilizar Flask, un framework minimalista de Python ampliamente reconocido por su simplicidad y flexibilidad en la creación de APIs RESTful. En las siguientes subsecciones se pueden visualizar las tecnologías y herramientas utilizadas en el backend.

6.3.1. Flask

Flask [5] es un framework web minimalista de Python que pertenece al proyecto Pallets y se utiliza para desarrollar aplicaciones web y APIs. Sigue una filosofía minimalista, ofreciendo solo los componentes esenciales, pero es altamente extensible, permitiendo añadir funcionalidades adicionales mediante extensiones. Además, Flask se basa en Werkzeug, una biblioteca de utilidades WSGI que proporciona herramientas avanzadas para la creación de aplicaciones web robustas, incluyendo enrutamiento y gestión de solicitudes.

Para dar soporte a nuestro frontend, hemos creado una API REST. Una API REST (Representational State Transfer) es una arquitectura para diseñar servicios web que interactúan utilizando operaciones básicas del protocolo HTTP (GET, POST, PUT, DELETE). REST es conocido por su simplicidad y capacidad para crear servicios web escalables y fáciles de mantener. Algunas características destacables de una API REST son:

- **Stateless (Sin Estado):** Cada solicitud del cliente al servidor debe contener toda la información necesaria para entender y procesar la solicitud. El servidor no debe almacenar el estado del cliente entre las solicitudes.
- **Recursos:** Los recursos se identifican mediante URLs.
- **Operaciones HTTP:** Utiliza las operaciones estándar de HTTP.
- **Representaciones:** Los recursos pueden ser representados en múltiples formatos, comúnmente JSON o XML.

6.3.2. Autenticación JWT

Para el sistema de autenticación y autorización se utilizó JWT (JSON Web Token) es un estándar abierto (RFC7519)[6] que define una forma compacta y autónoma de transmitir información de manera segura entre las partes como un objeto JSON. La información contenida en el token puede ser verificada y confiada porque está firmada digitalmente. JWT es ampliamente utilizado para la autenticación y autorización en aplicaciones web. Los JWT son compactos y eficientes, ideales para entornos con baja latencia y transmisiones rápidas de datos a través de URL, parámetros POST o encabezados HTTP. Contiene toda la información necesaria sobre el usuario en sí mismo, eliminando la necesidad de

[5] https://github.com/pallets/flask
[6] https://datatracker.ietf.org/doc/html/rfc7519

almacenar sesiones en el servidor. JWT puede ser firmado digitalmente utilizando un secreto o claves públicas/privadas, asegurando la integridad de los datos y permitiendo el cifrado de información sensible. Además, sigue un estándar bien definido que garantiza su interoperabilidad entre sistemas y plataformas diferentes. La autenticación es el proceso de verificar la identidad del usuario con, el flujo típico a través de JWT cuenta con los siguientes pasos:

1. Primera autenticación:
   - El usuario envía sus credenciales (email y contraseña) al servidor de autenticación.
   - El servidor verifica las credenciales. Si son correctas, se genera un JWT.
2. Generación del JWT:
   - El servidor crea un JWT que incluye información sobre el usuario (claims), como su identificador, roles y una fecha de expiración.
   - El JWT se compone de tres partes: La primera es el header el cual contiene el tipo de token y algoritmo de firma, la segunda es el payload el cual contiene la información asociados al usuario y por último tenemos la signature que se crea al firmar el header y payload.
3. Envio de JWT al usuario:
   - El servidor envía el JWT al usuario, el mismo se puede almacenar en una cookie o en el almacenamiento del navegador.
4. Uso del JWT:
   - En solicitudes posteriores, el usuario incluye el JWT en el encabezado de autorización de la solicitud HTTP.

Una vez que el usuario realice su primera autenticación recibirá dos JWT, uno llamado access token y otro llamado refresh token. El access token permite al usuario autenticarse a recursos protegidos, mientras que el refresh token permite obtener nuevos access token sin tener que volverse a autenticar. Nuestro sistema configura el tiempo de vida del access token en 8hs, luego de ese tiempo el mismo dejara de ser valido, por otro lado el refresh token tiene un tiempo de vida de 30 días.

6.3.3. OpenCV

OpenCV es una biblioteca de visión por computadora de código abierto que proporciona una amplia gama de funciones para el procesamiento de imágenes y vídeo. Con más de 2500 algoritmos optimizados, OpenCV es ampliamente utilizado en aplicaciones que van desde la detección de objetos hasta la segmentación de imágenes y el reconocimiento facial.

En esta implementación, OpenCV se utiliza para preprocesar las imágenes antes de enviarlas al modelo YOLO Face. Esto incluye tareas como la lectura de imágenes, el cambio de tamaño, la normalización y la conversión de formatos de color. Además, OpenCV se emplea para modificar las regiones detectadas, aplicándole alguno de los métodos mencionado en la Sección 2.2 como son el difuminado o bloqueo.

6.3.4. YOLO

En este prototipo, se utiliza una versión especializada de YOLO, conocida como YOLO Face [7], optimizada para la detección de rostros. El modelo está entrenado en un conjunto de datos específico de rostros humanos, permitiéndole identificar y localizar rostros en diversas condiciones y entornos. La precisión y velocidad de YOLO Face son cruciales para cumplir con los requisitos de seguridad, debido a que nos permite dentro del mismo servidor mantener toda la solución. Además, este modelo se encuentra bajo licencia de código abierto[8].

6.3.5. ONXRuntime

ONNX Runtime[9] es un motor de inferencia de alto rendimiento para modelos entrenados en diversos marcos de aprendizaje profundo como PyTorch, TensorFlow y otros. ONNX (Open Neural Network Exchange) es un formato de intercambio abierto[10], implementado por Microsoft, que permite a los desarrolladores usar modelos entrenados en diferentes marcos y optimizarlos para diversas plataformas.

Para esta solución, se convierte el modelo YOLO Face a formato ONNX para aprovechar las capacidades de ONNX Runtime. Esto permite ejecutar el modelo de manera eficiente en diferentes entornos, incluyendo CPU y GPU. ONNX Runtime proporciona una plataforma flexible y escalable para la inferencia, mejorando la velocidad y eficiencia del proceso de predicción de rostros.

6.3.6. ORM

Un ORM (Object-Relational Mapper) es una técnica de programación que permite interactuar con bases de datos relacionales utilizando el paradigma de la programación orientada a objetos. En lugar de escribir consultas SQL manualmente, un ORM traduce las operaciones en objetos a consultas SQL equivalentes, proporcionando una forma más intuitiva y productiva de manejar datos almacenados en bases de datos. Además, ofrece una capa de abstracción que permite cambiar la base de datos subyacente sin modificar el código de la aplicación. Algunas de las ventajas de utilizar un ORM son:

[7] https://github.com/akanametov/yolo-face
[8] Licencia GPL-3.0
[9] https://github.com/microsoft/onnxruntime
[10] Licencia MIT

- **Mapeo de clases a tablas:** Las clases en el lenguaje de programación se mapean directamente a tablas en la base de datos.
- **Mapeo de atributos a columnas:** Los atributos de las clases se mapean a columnas en las tablas de la base de datos.
- **Manejo de relaciones:** Facilita la definición y manejo de relaciones entre tablas utilizando objetos.

En el proyecto se utiliza SQLAlchemy [11], un ORM muy popular para Python. Específicamente, se hace uso de la extensión Flask-SQLAlchemy, que simplifica la integración de SQLAlchemy con aplicaciones Flask.

6.3.7. Envío de correos electrónicos

Dado que existe la necesidad de enviar correos electrónicos a los usuarios para confirmar cuentas o restablecer contraseñas, se utiliza el servicio de Gmail SMTP (Simple Mail Transfer Protocol). SMTP[12] es un protocolo estándar para el envío de correos electrónicos a través de redes, facilitando la transmisión de mensajes desde un cliente de correo a un servidor de correo electrónico, y entre servidores de correo para la entrega final al destinatario. Gmail SMTP[13] es el servicio de protocolo de transferencia de correo electrónico proporcionado por Google a través de su plataforma Gmail, permitiendo enviar correos electrónicos programáticamente desde una aplicación o cliente de correo utilizando la infraestructura de Gmail. Una limitación de este servicio es que permite enviar únicamente 100 correos diarios para cuentas que no pertenecen al plan de pago de Google Workspace.

En el proyecto se utiliza la extensión Flask-Mail, la cual facilita el envío de correos electrónicos. Flask-Mail permite configurar servidores de correo, gestionar plantillas de correo electrónico y enviar mensajes de forma sincrónica o asincrónica.

6.4. Base de Datos

El proyecto utiliza PostgreSQL[14] que es un sistema de gestión de bases de datos relacional de código abierto conocido por su robustez y extensibilidad. Es compatible con el estándar SQL, permite definir tipos de datos personalizados, operadores y funciones, y soporta tipos de datos avanzados como JSON y arrays. Sus características incluyen mecanismos de integridad de datos, control de concurrencia multiversión, índices avanzados, particionamiento de tablas y paralelización de consultas, además de replicación física y lógica. PostgreSQL garantiza propiedades ACID y ofrece altos niveles de seguridad con autenticación, control de acceso y encriptación de datos. Soporta múltiples lenguajes de

[11] https://www.sqlalchemy.org
[12] https://www.ietf.org/rfc/rfc2821.txt
[13] https://support.google.com/a/answer/176600?hl=es
[14] https://www.postgresql.org

programación para procedimientos almacenados. Es altamente fiable, flexible, escalable, y cuenta con una activa comunidad de desarrollo que proporciona soporte y herramientas adicionales.

6.5. Docker

Docker[15] es una plataforma de código abierto diseñada para facilitar la creación, implementación y ejecución de aplicaciones utilizando contenedores. Un contenedor Docker encapsula una aplicación junto con todas sus dependencias (bibliotecas, frameworks, configuraciones, etc.) en un entorno autónomo y liviano que puede ejecutarse de manera consistente en cualquier máquina que ejecute Docker. Esto garantiza que la aplicación funcione de la misma manera independientemente del entorno de desarrollo, pruebas o producción.
Se ha seleccionado tanto para el Frontend como para el Backend por varias razones:

- Consistencia y Portabilidad: Los contenedores Docker proporcionan un entorno consistente y reproducible para el desarrollo y la ejecución de aplicaciones en diferentes etapas del ciclo de vida del software. Esto asegura que la aplicación se comporte de la misma manera en todos los entornos, desde el desarrollo local hasta la implementación en producción.

- Aislamiento y Seguridad: Cada contenedor Docker es independiente y ejecuta la aplicación con sus propios recursos aislados del sistema operativo subyacente. Esto mejora la seguridad al prevenir conflictos entre diferentes aplicaciones o componentes de software que se ejecutan en el mismo servidor.

- Eficiencia y Escalabilidad: Docker permite empaquetar y distribuir aplicaciones junto con sus dependencias en contenedores ligeros y rápidos. Esto facilita la escalabilidad horizontal, es decir, la capacidad de aumentar o reducir la capacidad de la aplicación añadiendo o eliminando contenedores según la demanda.

Teniendo en cuenta que la aplicación a desarrollar debe ser apta para funcionar como aplicación de escritorio, Docker ofrece la ventaja de simplificar la distribución y la instalación de la aplicación al empaquetarla en un contenedor que puede ser distribuido y ejecutado en múltiples plataformas y sistemas operativos sin necesidad de configuraciones adicionales, es independiente del sistema operativo subyacente.
La configuración y el comportamiento de la aplicación son consistentes en todos los entornos, independientemente de si se ejecuta en un contenedor Docker en la nube o como una aplicación de escritorio instalada localmente.

[15] https://www.docker.com/

6.6. Otras Herramientas

En las siguientes subsecciones se presentan algunas herramientas utilizadas para la gestión del proyecto, disponibilización en la nube, y gestión de errores. Estas herramientas fueron de gran utilidad facilitando la gestión del cambio, integración continua en la nube y manejos de errores en momento de ejecución.

6.6.1. Github y Github projects

GitHub es una plataforma de desarrollo colaborativo y de control de versiones basada en Git. Permite a los desarrolladores gestionar proyectos de software, colaborar en el código, realizar revisiones de código, y automatizar flujos de trabajo de desarrollo. GitHub proporciona una interfaz gráfica que facilita la visualización y gestión de repositorios, ramas, y commits. También ofrece características avanzadas como GitHub Actions para CI/CD (Integración Continua y Entrega Continua), GitHub Packages para gestionar dependencias, y GitHub Pages para desplegar sitios web estáticos.

GitHub Projects es una herramienta de gestión de proyectos integrada en GitHub que utiliza un enfoque basado en tableros Kanban. Permite a los equipos planificar, organizar y realizar el seguimiento del trabajo a través de tarjetas (cards) que representan tareas, issues, y pull requests.

6.6.2. Portainer

Portainer[16] es una herramienta esencial para el desarrollo del proyecto ya que es idóneo para trabajar con Docker y desplegar aplicaciones web en la nube, facilita la gestión de contenedores de forma centralizada.
Se considera que la principal ventaja de Portainer es su capacidad para simplificar el proceso de despliegue y escalamiento de aplicaciones ya que desde su interfaz se puede iniciar, detener, reiniciar y eliminar contenedores. También proporciona herramientas básicas de monitoreo y estadísticas para los contenedores Docker. Esto incluye métricas de uso de CPU, memoria y red, que te permiten optimizar el rendimiento y detectar problemas.

6.6.3. Sentry

Sentry[17] es una herramienta de monitoreo y gestión de errores que se utiliza para rastrear y resolver errores en aplicaciones de software. Está diseñada para ayudar a los desarrolladores a identificar, diagnosticar y corregir problemas en sus aplicaciones, proporcionando informes detallados sobre los errores y el rendimiento de la aplicación. Se integra en la aplicación mediante SDKs específicos para diferentes lenguajes y plataformas, como JavaScript, Python, Java, entre otros. Cuando se detecta un error, genera un informe detallado que

[16] https://www.portainer.io/
[17] https://sentry.io/welcome/

incluye información contextual como la pila de llamadas, las variables locales, la configuración del sistema y del entorno. Además de la gestión de errores, Sentry también proporciona herramientas para monitorear el rendimiento de la aplicación, permitiendo a los desarrolladores rastrear métricas como el tiempo de respuesta de las solicitudes, la latencia y otros indicadores de rendimiento.
Se considera una herramienta muy valiosa para la aplicación ya que los errores en el procesamiento de imágenes, como fallos en algoritmos de anonimización o problemas de compatibilidad con diferentes formatos de imagen, pueden ser difíciles de detectar, Sentry captura estos errores y proporciona detalles completos para facilitar su resolución.

Capítulo 7

Implementación

En este capítulo se detalla la implementación del proyecto. Como se menciona en la Sección 5.1 la estructura del backend está construida utilizando Flask, un microframework de Python que facilita la creación de servidores web y APIs. Para el frontend, se emplea React.js, una biblioteca de JavaScript ampliamente utilizada para construir interfaces de usuario interactivas y dinámicas.

En cuanto al procesamiento de imágenes, se opta por descargar el modelo YOLO Face, conocido por su eficiencia en la detección de rostros, disponible públicamente para fines de investigación. Este modelo se convierte al formato ONNX, permitiendo su integración y utilización mediante la librería ONNX Runtime, que optimiza su desempeño en entornos de producción.

El manejo y manipulación de imágenes en la interfaz de usuario se realiza mediante las librerías Fabric.js y Canvas, las cuales proporcionan herramientas potentes y flexibles para la edición y visualización de imágenes.

A lo largo del capítulo, se abordarán en detalle estos componentes y sus respectivas implementaciones, ofreciendo una visión completa del proceso de desarrollo y las decisiones técnicas adoptadas.

7.1. Sesión de usuario

Cuando el usuario ingresa por primera vez a la plataforma web, se le presenta una vista genérica que incluye una página de inicio que describe la aplicación, una página para iniciar sesión y una página para registrarse, como se muestra en la Figura 7.1.

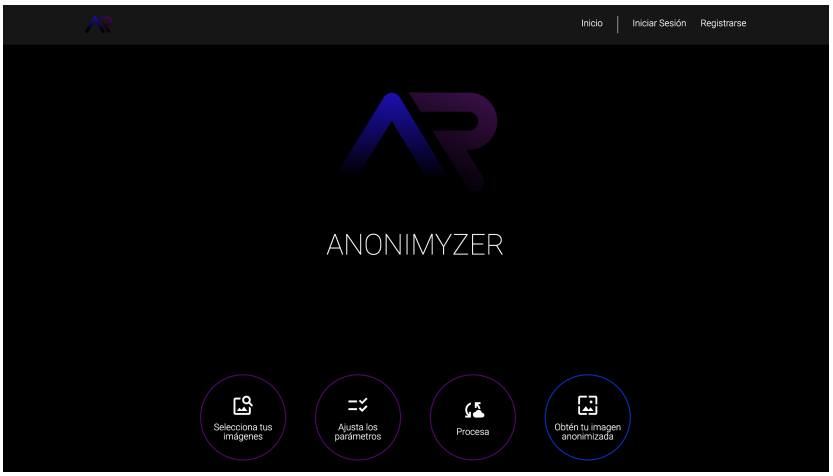


Figura 7.1: Visualización de la aplicación con sesión de usuario no iniciada

7.1.1. Registro de usuario

En la página accesible a través del menú para el registro de usuario, se encuentra un formulario que los usuarios deben completar para unirse a la comunidad que usa la web. Los datos solicitados incluyen nombre y apellido, correo electrónico y contraseña, como se muestra en la Figura 7.2. La contraseña cuenta con un campo adicional para ser ingresada una segunda vez como confirmación, una práctica común en los portales web.

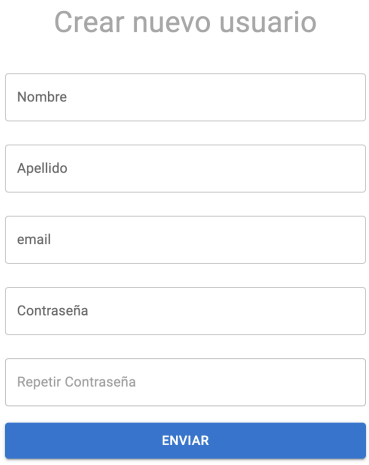


Figura 7.2: Registro de usuario

7.1.2. Inicio de sesión

Cuando un usuario registrado ingresa a la aplicación debe iniciar sesión a través del formulario que se muestra en la Figura 7.3. Los datos solicitados para el mismo son su email y contraseña.
Una vez enviado el formulario se recibe un JWT (Json Web Token) que el usuario mantendrá durante toda su sesión, es utilizado como medida de seguridad para acceder a los servicios que ofrece la API, se envía en el cabezal de cada llamada posterior que se realice a backend como consecuencia de una interacción del usuario con la interfaz.
A modo de garantizar una mayor medida de seguridad para el flujo de datos dentro de la aplicación, dado su carácter sensible, se decide utilizar la biblioteca js-cookie. La misma permite guardar en una cookie el token y establecer configuraciones para asegurar que solo sea enviada a través de conexiones HTTPS. A su vez se aplica una configuración que permite mitigar ataques de tipo CSRF (Cross-Site Request Forgery), la cookie solo se envía si la solicitud se origina en el mismo sitio que fue creada. En la Figura 7.4 pueden verse las interacciones a través de un diagrama de secuencia.

Figura 7.3: Inicio de sesión

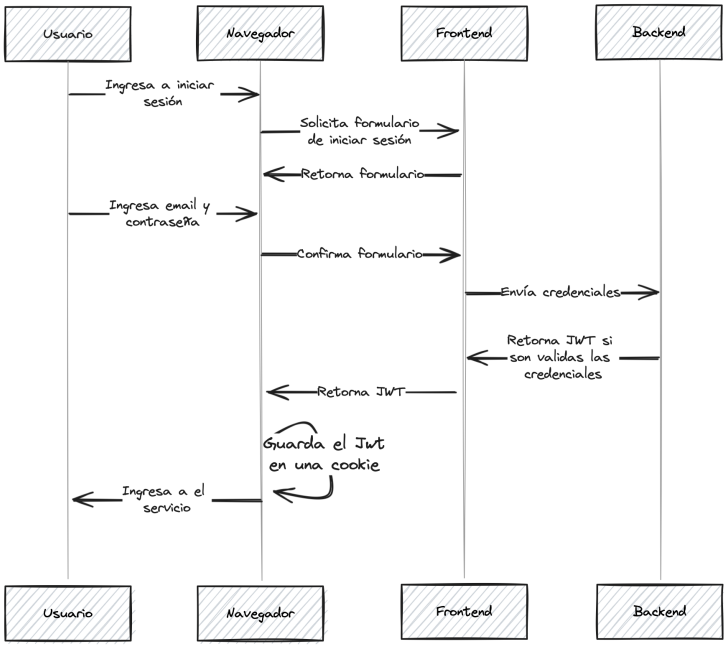
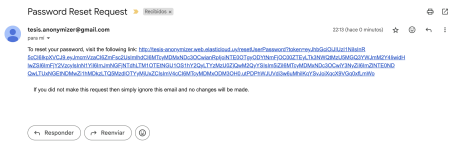


Figura 7.4: Diagrama de secuencia de inicio de sesión

7.1.3. Recuperar contraseña

Este es un flujo que suele implementarse en aplicaciones web, dado que el usuario puede llegar a olvidar o confundir su contraseña. Es accesible a través del link "Olvide mi contraseña" presente en la pagina de inicio de sesión como se puede observar en la Figura 7.3 y dirige al usuario hacia un nuevo formulario en el cual se le solicita ingresar su email (ver Figura 7.5a) para posteriormente enviar un correo de recuperación (ver Figura 7.5b).

(a) Recuperación de contraseña



(b) Email de recuperación de contraseña

Figura 7.5: Pasos en la recuperación de contraseña

Cuando el destinatario navega a través del link es dirigido hacia una pagina como se muestra en la Figura 7.6, en la cual puede ingresar un nuevo valor para su contraseña y será aplicada a los inicios de sesión posteriores.
La seguridad de este flujo esta dada por un JWT (Json Web Token) que es enviado al email del usuario, el mismo es captado en la pagina de recuperación y se utiliza en la llamada a la API para restablecer la contraseña.

Figura 7.6: Restablecer contraseña

7.1.4. Visualización de rutas protegidas

Una vez la sesión esta iniciada los usuarios podrán acceder a un conjunto de rutas protegidas como se muestra en la Figura 7.7, las cuales solo son visibles para los usuarios registrados en la aplicación que han iniciado sesión. Mediante ellas se puede hacer uso de las herramientas de anonimización que se ofrecen en la web y al perfil de usuario.

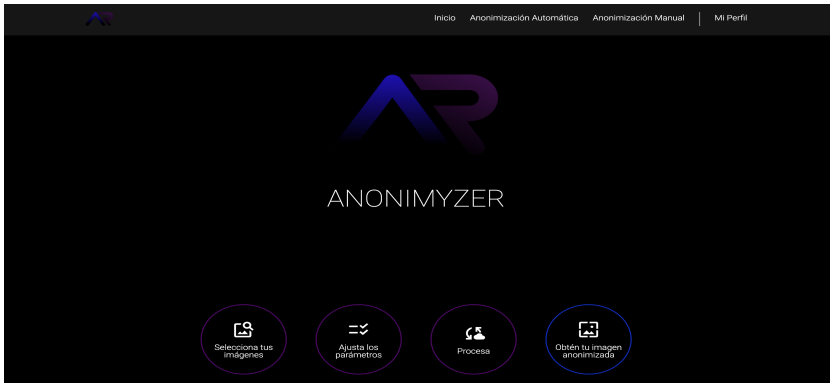
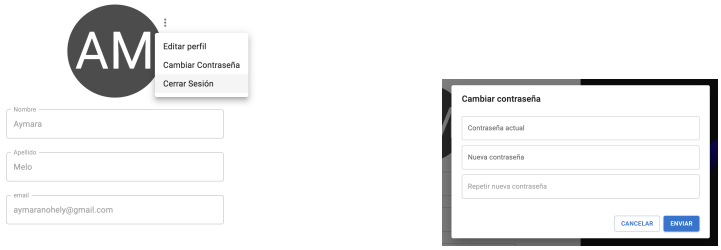


Figura 7.7: Visibilidad de rutas protegidas

7.1.5. Perfil de usuario

Cuando el usuario ingresa a su perfil accede a la visualización de sus datos como se muestra en la Figura 7.8a, que son los mismos que fueron solicitados al registrarse. También obtiene acceso a las funcionalidades para editar esos datos, cambiar su contraseña y cerrar sesión.
Se decide que en el flujo de edición de datos sea posible editar nombre y apellido pero el email con el que se crea la cuenta permanece no editable. Si el usuario desea utilizar un email diferente, deberá crear una cuenta asociada al mismo.
Para el cambio de contraseña se le muestra un modal en el cual se solicita al usuario la contraseña actual como muestra la Figura 7.8b.

Cuando el usuario decide cerrar la sesión, se realiza una llamada a la API para invalidar el token actual, agregándolo a la lista de tokens bloqueados. Simultáneamente, la web elimina los datos almacenados en la cookie que ya no son válidos. En la Figura 7.9 pueden verse las interacciones a través de un diagrama de secuencia.



(a) Perfil de usuario (b) Cambio de contraseña

Figura 7.8: Perfil de usuario y acciones disponibles

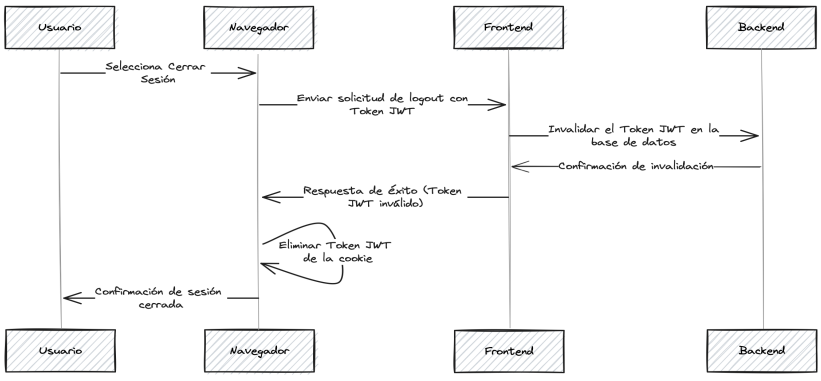


Figura 7.9: Diagrama de secuencia de cerrar sesión

7.2. Anonimización manual de imágenes

En la Figura 7.10 puede verse el diseño implementado para la anonimización de imágenes manual. El mismo cuenta con un panel de imágenes situado a la izquierda en el cual se cargan las imágenes con las que el usuario va a trabajar. En la parte inferior puede verse la barra de herramientas que ofrece las funcionalidades de zoom (acercar y alejar), mover y dimensionar la imagen, seleccionar un tipo de ofuscación y la intensidad con la que será aplicada y marcado de regiones de interés.

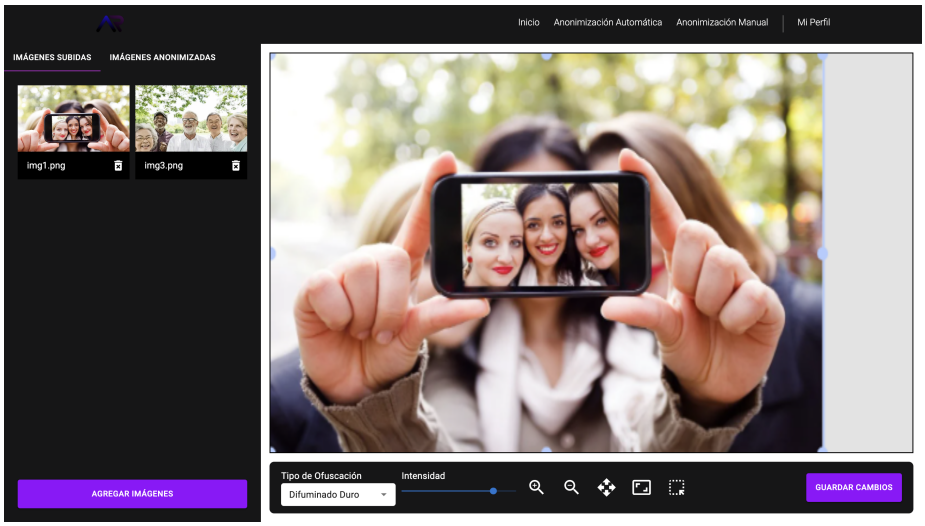


Figura 7.10: Herramienta de anonimización manual

Al iniciar la herramienta, se configura un lienzo (canvas) utilizando Fabric.js. Se configuran también propiedades globales para los objetos dentro del lienzo. El mismo se mantiene sin permitir la selección múltiple de objetos para evitar interacciones no deseadas.

7.2.1. Selección de imagen

Cuando se cargan nuevas imágenes, estas se añaden al estado de la aplicación con configuraciones predeterminadas para el grupo de objetos, el tipo de anonimización y la intensidad de la misma, todas vacías hasta que el usuario fije las deseadas.
Al seleccionar una imagen, se carga en el lienzo, escalándola para que se ajuste adecuadamente al tamaño del mismo. Este factor se determina comparando las dimensiones de la imagen con las dimensiones del lienzo (ver ecuación 7.1). Se utiliza el menor valor entre el factor de escalado horizontal y el vertical para

asegurar que la imagen se ajuste dentro del lienzo sin distorsionarse.

$$factor\_escala = \text{mín}\left(\frac{ancho\_lienzo}{ancho\_imagen}, \frac{alto\_lienzo}{alto\_imagen}\right) \tag{7.1}$$

Esta imagen se agrupa en un conjunto de objetos (grupo) para facilitar su manipulación conjunta. El grupo contendrá a futuro todas las regiones de interés que el usuario haya marcado en esa imagen. Si la imagen seleccionada ya tenía un grupo asociado, se reutiliza dicho grupo, de lo contrario, se crea uno nuevo.

7.2.2. Regiones de interés

El manejo de las regiones de interés se realiza mediante rectángulos dibujados sobre la imagen en el lienzo como se puede observar en la Figura 7.11. Estos rectángulos permiten al usuario marcar las áreas específicas que desea anonimizar.

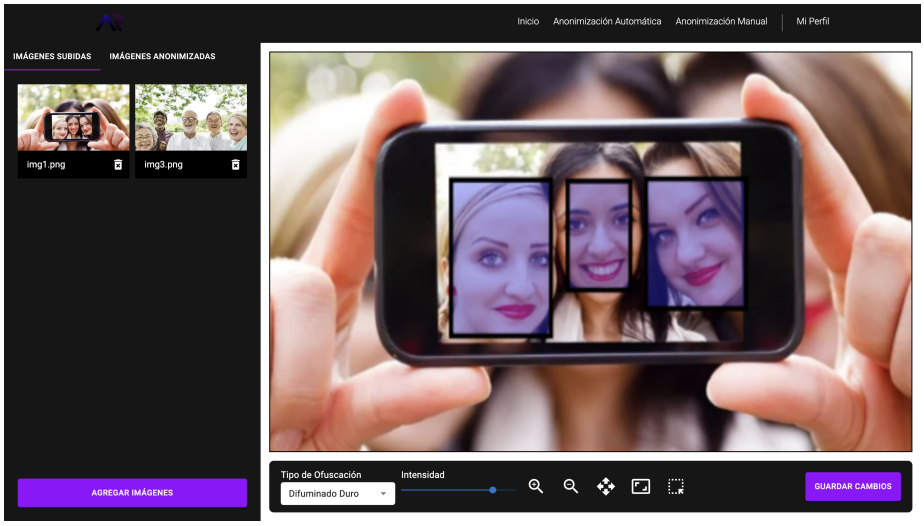


Figura 7.11: Marcado de regiones de interés

Proceso de marcado de región

Cuando el usuario hace clic en el lienzo, se captura la posición inicial del puntero. En este punto, se crea un pequeño rectángulo en esa posición.
A medida que el usuario arrastra el puntero, el tamaño del rectángulo se ajusta dinámicamente. Esto se logra capturando la posición actual del puntero y ajustando las dimensiones del rectángulo en consecuencia. Se aseguran también las condiciones de límite para que el rectángulo no salga de los bordes de la imagen.

Al soltar el botón del mouse, el tamaño final del rectángulo se establece, y se añaden al grupo de objetos en el lienzo.

$$(x_{\text{start}}, y_{\text{start}}) \tag{7.2}$$

$$x'_{\text{current}} = \text{mín}(\text{máx}(x_{\text{current}}, x_{\text{min}}), x_{\text{max}}) \tag{7.3}$$
$$y'_{\text{current}} = \text{mín}(\text{máx}(y_{\text{current}}, y_{\text{min}}), y_{\text{max}}) \tag{7.4}$$

$$\delta x = |x'_{\text{current}} - x_{\text{start}}| \tag{7.5}$$
$$\delta y = |y'_{\text{current}} - y_{\text{start}}| \tag{7.6}$$

$$\text{rect} = \text{Rect}(x_{\text{start}}, y_{\text{start}}, \delta x, \delta y) \tag{7.7}$$

donde:

- $(x_{\text{start}}, y_{\text{start}})$ son las coordenadas iniciales del puntero cuando se hace clic en el lienzo.
- $x_{\text{current}}, y_{\text{current}}$ son las coordenadas actuales del puntero.
- $x'_{\text{current}}, y'_{\text{current}}$ son las coordenadas actuales ajustadas para asegurar que el rectángulo no salga de los límites del lienzo definidos por $x_{\text{min}}, x_{\text{max}}$ y $y_{\text{min}}, y_{\text{max}}$, respectivamente.
- δx y δy representan las diferencias en las coordenadas x y y, respectivamente, entre la posición inicial del puntero y su posición actual ajustada.
- $\text{Rect}(x_{\text{start}}, y_{\text{start}}, \delta x, \delta y)$ representa la creación de un rectángulo en el lienzo con las dimensiones calculadas.

7.2.3. Movimiento y Dimensionamiento de Grupos

El grupo se configura para que sea seleccionable, lo que permite que sea manipulado por el usuario a través de eventos como clics y arrastres. Además, se desbloquean las propiedades de movimiento horizontal y vertical del grupo para permitir una manipulación libre.
Para lograr el redimensionamiento de la imagen y sus regiones de interés (todos estos elementos componen el grupo), los controles específicos necesarios para el redimensionamiento (es decir, las esquinas y bordes del grupo) se activan para que sean visibles y accesibles al usuario. Durante este modo, el usuario puede hacer clic y arrastrar los controles visibles del grupo para cambiar las dimensiones, asegurando así que el redimensionamiento sea intuitivo y controlado.
Para proporcionar una mejor experiencia de usuario, se implementan las funciones zoomIn y zoomOut. Las mismas permiten modificar el nivel de zoom del lienzo de Fabric.js, lo que afecta directamente la escala y la visualización de los

objetos presentes en el lienzo.
Acercamiento (zoomIn):

$$Z_{\text{nuevo}} = Z_{\text{actual}} \times 1{,}1$$

El acercamiento se realiza multiplicando el zoom actual Z_{actual} por un factor mayor a 1 (1.1 en este caso). Esto aumenta el tamaño visual de los objetos en el lienzo, proporcionando un detalle aumentado para áreas específicas. **Alejamiento (zoomOut):**

$$Z_{\text{nuevo}} = Z_{\text{actual}} \times 0{,}9$$

El alejamiento disminuye el nivel de zoom actual multiplicándolo por un factor menor a 1 (0.9 en este caso). Esto permite visualizar los objetos en una escala más reducida, útil para obtener una vista general del lienzo o trabajar en áreas extensas con mayor comodidad.

Mantenimiento de Proporciones y Coherencia Visual

Durante tanto el movimiento y redimensionamiento del grupo como el uso de zoom, Fabric.js maneja internamente la lógica para mantener la proporción adecuada entre los objetos y la imagen de fondo en el lienzo. Esto se logra ajustando dinámicamente las escalas y dimensiones de los objetos conforme se manipulan o cambia el nivel de zoom. Así, se garantiza que los elementos conserven su relación visual correcta, proporcionando una experiencia coherente y precisa para el usuario final.

7.2.4. Visualización de resultados

Una vez que el usuario termina de manipular la imagen con la que se encuentra trabajando, selecciona el botón guardar cambios que se encuentra en la barra de herramientas y envía la imagen a backend para ser procesada.
Los resultados obtenidos pueden ser apreciados en la tab “imágenes anonimizadas” como se muestra en la Figura 7.12.

Figura 7.12: Resultados de anonimización manual

Descarga de resultados

Los usuarios pueden descargar los resultados obtenidos de tres maneras distintas. Se decidió implementar varias formas dada la naturaleza de trabajo de los usuarios finales de la aplicación y el interés que mostraron en contar con estas opciones.
Se puede realizar una descarga individual de cada imagen anonimizada como se observa en la Figura 7.13, una selección de ellas en un archivo .zip o todas las imágenes obtenidas como resultado en un archivo .zip como se puede ver en la Figura 7.14. Para la implementación de estas funcionalidades se hace uso de las librerias JSZip y FileSaver.

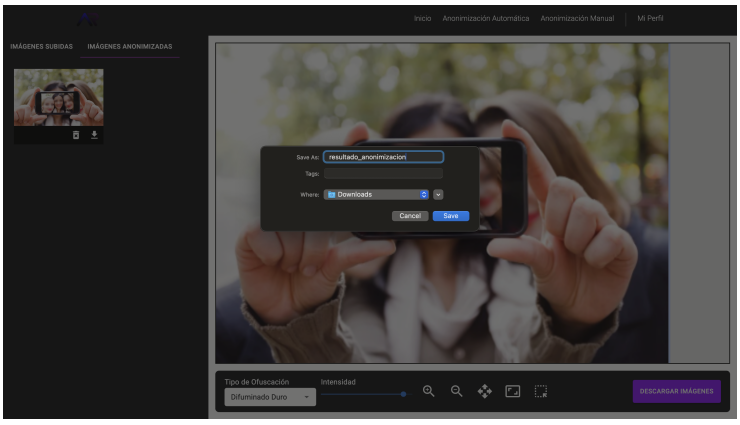


Figura 7.13: Descarga individual de resultados

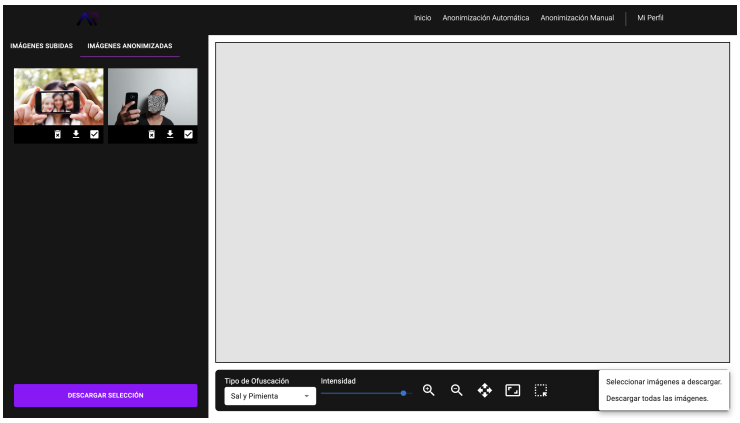


Figura 7.14: Opciones de descarga de resultados

7.3. Anonimización automática

En la Figura 7.15 puede verse el diseño implementado para la anonimización de imágenes de forma automática. El mismo cuenta con un panel de imágenes en el cual el usuario puede subir todas las imágenes que desee anonimizar, utilizando el botón de agregar imágenes. En la parte inferior puede verse la barra de herramientas que ofrece las funcionalidades de seleccionar un tipo de ofuscación, la intensidad con la que será aplicada y el nivel de confianza del algorítmo.

Si bien esta habilitada la posibilidad de subir una única imagen, el enfoque de la solución apunta a la posibilidad de subir múltiples imágenes para que se haga la detección de rostros y anonimización de los mismos automáticamente.

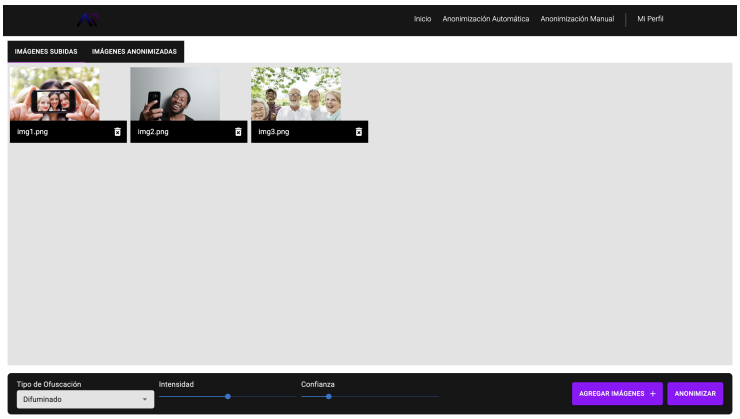


Figura 7.15: Herramienta de anonimización automática

7.3.1. Nivel de confianza

La posibilidad de establecer un nivel de confianza a través de la barra de herramientas tiene como objetivo brindar al usuario una forma de establecer la efectividad con la que los rostros son detectados.

El modelo utilizado para la detección de objeto trabaja con probabilidades. Cuando la probabilidad es baja, implica que el algoritmo no es capaz de decidir si el objeto identificado es un rostro, mientras que una probabilidad alta significa que el algoritmo detectó un rostro con mayor certeza.
A partir de estas probabilidades obtenidas, y el nivel de confianza configurado, si la probabilidad que indica el modelo es mayor al número obtenido por

$$\frac{nivel\_de\_confianza}{100}$$

entonces esa región obtenida se considera como aceptable, y se procede a anonimizar dicha región. Dependiendo de las necesidades del usuario puede configurarse el nivel de confianza con un valor bajo, para que tome en cuenta probabilidades menos significativas. Esto puede generar falsos positivos pero da mayor garantía que casi todos de los rostros serán anonimizados. Por otro lado podría ajustarse el nivel de confianza en un valor alto, el cual hará que el algoritmo tome en cuenta probabilidades más significativas, con el riesgo de que un rostro difícil de identificar pueda no quedar anonimizado.

(a) Nivel de confianza de 85 % (b) Nivel de confianza de 70 %

Figura 7.16: Resultados de la anonimización automática que muestran la capacidad del prototipo para detectar rostros y anonimizarlos con un nivel de confianza del 70 % y la misma imágen con un nivel de confianza del 85 %

En la Figura 7.16a puede verse el resultado del algoritmo al establecer un nivel de confianza de 85 %, puede apreciarse que claramente faltan rostros por anonimizar. La misma imagen con un nivel de confianza del 70 % (ver Figura 7.16b) brinda resultados de mejor calidad anonimizando todos los rostros presentes.

7.3.2. Visualización de resultados

Una vez que el usuario carga las imágenes que desea procesar y selecciona el botón para anonimizar se envían al backend todos los archivos y parámetros de anonimización establecidos a través de la barra de herramientas. Durante el proceso de anonimizacion puede verse un loader para indicar que los resultados están siendo generados como puede observarse en la Figura 7.17 y una vez se termina el procesamiento los mismos quedan disponibles para ser descargados (ver Figura 7.18).

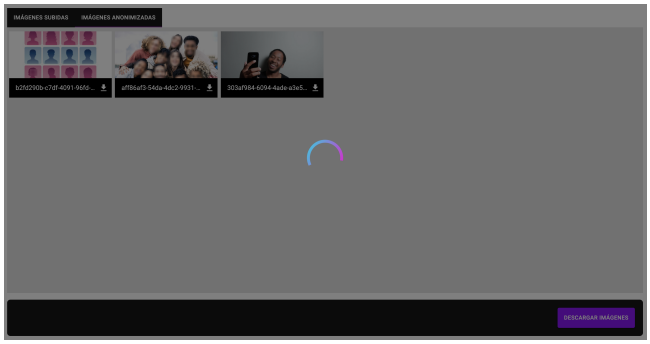


Figura 7.17: Imágenes en procesamiento automático

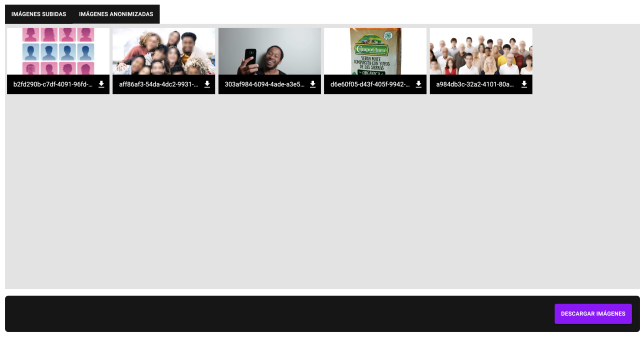


Figura 7.18: Resultados generados automaticamente

Desde el lado del servidor, este proceso implica: la carga del modelo YOLO desde un archivo ONNX, realizar la inferencia de objetos en una imagen y procesar las salidas de la red neuronal para obtener las ubicaciones de los objetos detectados.
En el siguiente bloque de código se muestra de forma sencilla cómo se realiza el proceso de predicción:

```
1 # Cargar el modelo YOLO
```

```
model_session = load_yolo_model('app/model/yolov8l-face.onnx')
# Preprocesar la imagen de entrada
image = preprocess_input(image, input_size)
# Realizar la inferencia utilizando el modelo cargado
outputs = run_inference(model_session, image)
# Postprocesar las salidas para obtener las coordenadas de los objetos detectados
boxes = postprocess_output(outputs, original_image, input_shape, threshold)
```

Descarga de resultados

Los usuarios pueden descargar los resultados obtenidos de tres maneras distintas. Se implementa para esta funcionalidad las mismas formas utilizadas en anonimización manual.
Se puede realizar una descarga individual (ver Figura 7.19) de cada imagen anonimizada, una selección de ellas en un archivo .zip o todas las imágenes obtenidas como resultado en un archivo .zip (ver Figura 7.20). Para la implementación de estas funcionalidades se hace nuevamente uso de las librerías JSZip y FileSaver.

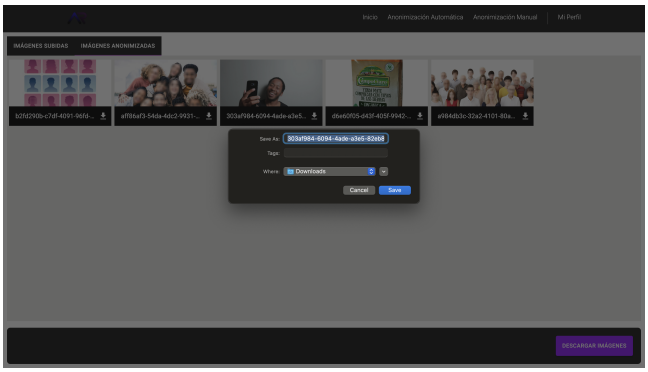


Figura 7.19: Descarga individual de resultados

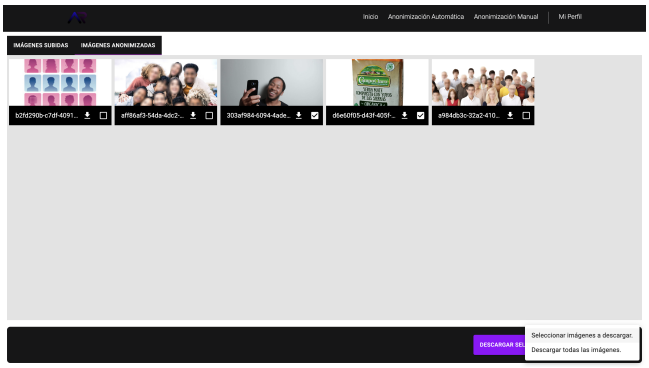


Figura 7.20: Opciones de descarga de resultados

7.4. Manejo de errores

Para asegurar una respuesta uniforme ante errores y mejorar la comunicación con los usuarios, se implementó un interceptor de respuesta en Axios. Este interceptor se encarga de capturar cualquier error que surja durante las solicitudes HTTP y procesarlo para que sea visualizable en la aplicación web.
Utilizando "*api.interceptors.response.use*" , se definió una función de manejo de errores que se ejecuta cada vez que una solicitud devuelve una respuesta de error desde el servidor. Dentro del interceptor, se verifica la presencia de un objeto de respuesta (response) y se extrae el mensaje de error específico proporcionado por backend, el cual contiene mensajes adaptados a cada funcionalidad. Este mensaje se muestra al usuario mediante una función dedicada para mostrar alertas en la interfaz de la aplicación como puede visualizarse en la siguiente Figura 7.21.

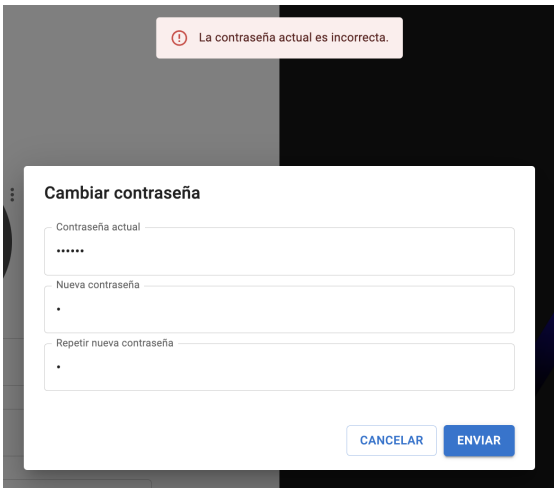


Figura 7.21: Alerta de error

A su vez se ha integrado la herramienta Sentry con el objetivo de llevar un registro detallado de los errores no manejados y monitorear el rendimiento de la aplicación. Sentry captura y registra automáticamente los errores que ocurren en la aplicación, incluso aquellos que no han sido manejados explícitamente. Esto incluye excepciones no capturadas, errores de red, problemas de configuración y otros, luego envía alertas en tiempo real cuando se detectan estos nuevos errores o problemas de rendimiento.
En la Figura 7.22 puede verse un ejemplo de un error detectado en la funcionalidad de anonimización manual. Se brinda información de la url en la cual el usuario se encontraba cuando ocurrió, si el error estaba siendo o no manejado y datos de el navegador que se estaba utilizando.

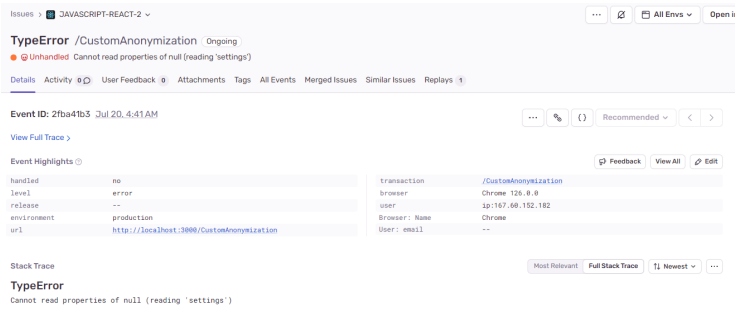


Figura 7.22: Sentry: Visualización de error detectado

Un recurso de gran valor que brinda esta herramienta es la posibilidad de visualizar una demostración de como fue que llegó a producirse el error (ver Figura 7.23). La misma es un vídeo que reproduce el comportamiento del usuario,

mostrando los elementos con los que interactuó hasta generar la falla. De este modo se permite al desarrollador seguir exactamente los pasos para reproducir el error y corregir el mismo lo más rápido posible.

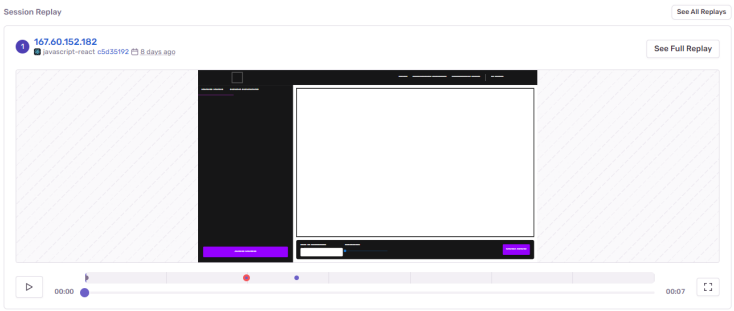


Figura 7.23: Sentry: Demostración de cómo fue generado un error

Además de lo expuesto anteriormente, la herramienta brinda distintas métricas de performance que permiten llevar un registro de como se comporta la aplicación. También puede ser conectada con Github para manejar mediante tickets la resolución de determinados errores y poder tener un historial de los mismos.
Un ejemplo de métrica de performance utilizada puede observarse en la Figura 7.24, dónde se evalúa la capacidad de respuesta de la página observando la latencia de las interacciones de usuario como por ejemplo clicks o uso del teclado. El valor final de INP es la interacción más larga observada.

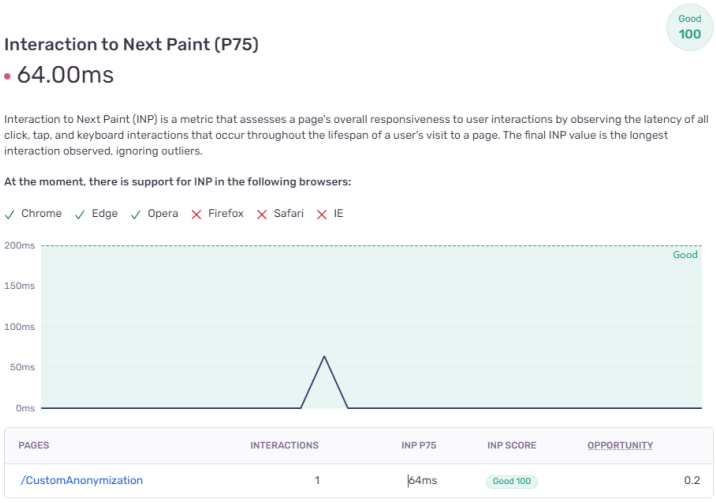


Figura 7.24: Sentry: Ejemplo de métrica de performance

7.5. Pruebas y validaciones

En la siguiente sección se detallan las pruebas y validaciones realizadas a lo largo del proyecto, para así poder garantizar la completitud de los requerimientos funcionales y no funcionales descriptos en el capítulo 3. Estas pruebas incluyen la retroalimentación continua con el equipo de investigadores de EDIGA, pruebas unitarias y de integración entre el frontend y el backend, así como pruebas de rendimiento y carga para asegurar la eficiencia del sistema bajo condiciones de uso real.

7.5.1. Instancias de retroalimentación

A lo largo de la realización del proyecto, se llevan a cabo varias instancias de validación con el equipo de investigadores de EDIGA. En estas sesiones, los investigadores proporcionan sugerencias y cambios sobre la solución, los cuales se toman en cuenta en iteraciones posteriores. Estas reuniones de validación son esenciales para asegurar que la herramienta desarrollada cumpla con los requisitos y expectativas del grupo de investigación. Cada sesión de retroalimentación incluye una demostración del estado actual del proyecto, seguida de una discusión abierta sobre las mejoras y ajustes necesarios.

Durante estas sesiones, se discuten diversos aspectos de la interfaz de usuario, la funcionalidad del sistema y la efectividad de las técnicas de anonimización implementadas. Las sugerencias abarcan desde cambios menores en la interfaz de usuario hasta ajustes significativos en los algoritmos de procesamiento de imágenes. Este enfoque iterativo permite un desarrollo ágil y adaptativo, asegurando que cada nueva versión de la herramienta incorpore mejoras basadas en el uso práctico y la experiencia de los investigadores.

Además, estas instancias de validación no solo se centran en la funcionalidad técnica, sino también en la usabilidad y accesibilidad de la herramienta. Se realizan pruebas de usabilidad para identificar posibles obstáculos en la experiencia del usuario y se ajustan los flujos de trabajo en consecuencia. La retroalimentación continua garantiza que la herramienta no solo sea técnicamente robusta, sino también intuitiva y fácil de usar para los investigadores.

En resumen, las instancias de validación con el equipo de investigadores de EDIGA son fundamentales para el éxito del proyecto. La colaboración estrecha y la retroalimentación constante permiten desarrollar una herramienta que satisface las necesidades específicas del grupo de investigación, asegurando así una implementación efectiva y eficiente en su contexto de trabajo.

7.5.2. Pruebas en Frontend

Sobre el frontend se realizan pruebas de integración para asegurar que las funcionalidades del sistema funcionen correctamente en conjunto. Al momento de agregar una nueva funcionalidad en el proceso de desarrollo, se integra y se realizan las pruebas desde el frontend para verificar su correcto funcionamiento

de manera sencilla y rápida. Esto permite identificar y solucionar problemas de compatibilidad entre el frontend y el backend de manera temprana.

Durante estas pruebas de integración, se verifica que todas las partes del sistema interactúen de manera efectiva y sin conflictos. Se realizan pruebas exhaustivas que incluyen la interacción del usuario con la interfaz, la correcta comunicación con la API del backend y la correcta visualización y manipulación de los datos. Estas pruebas ayudan a garantizar que cada componente del sistema funcione como se espera cuando se combina con otros componentes.

En resumen, las pruebas de integración en el frontend son esenciales para asegurar que las nuevas funcionalidades se incorporen sin problemas y que todo el sistema funcione de manera coherente y eficiente. Este enfoque proactivo en las pruebas permite desarrollar un sistema robusto y fiable, minimizando el riesgo de problemas de compatibilidad y asegurando una experiencia de usuario óptima.

7.5.3. Pruebas en Backend

Dado que el backend es una API Rest, se realizan pruebas unitarias para cada endpoint, considerando las respuestas del sistema tanto en casos de éxito como de falla. Antes de ejecutar cada prueba, se generan los diferentes conjuntos de datos necesarios. Para dichas pruebas, se utiliza la herramienta Postman[1], la cual permite enviar solicitudes a las APIs y recibir las respuestas correspondientes a través de una interfaz gráfica intuitiva y fácil de usar.

Para las pruebas de autenticación, se invocan los endpoints y se verifica que la respuesta tenga un código de estado 401 si no se proporcionan credenciales válidas para el recurso solicitado, y 200 si se cuenta con ellas. Luego, para los endpoints accesibles mediante el método GET, se verifica que la respuesta sea idéntica a los datos creados previamente en el conjunto de datos y que el código de estado sea 200. Por último, para los endpoints accesibles mediante métodos POST o PUT, se realiza el llamado y se comprueba que el código de estado sea 200 o 201, según corresponda.

Como se estableció en los requerimientos no funcionales, el servicio debe tener un tiempo de respuesta bajo para los endpoints que procesan imágenes, ya que el manejo y procesamiento de imágenes genera una gran carga para el backend. Para comprobar el rendimiento y la capacidad de carga, se realizaron pruebas de estrés utilizando la herramienta Apache JMeter[2].

Apache JMeter es una herramienta de software libre diseñada para realizar pruebas de carga y medir el rendimiento de aplicaciones. Permite simular múltiples usuarios concurrentes enviando solicitudes a un servidor web, API, base de datos, entre otros, con el fin de evaluar cómo se comporta el sistema bajo diferentes condiciones de carga. Con JMeter, se pueden crear y ejecutar planes de prueba detallados, configurar diversos parámetros de prueba y analizar los resultados a través de informes gráficos y tabulares. Estas pruebas permiten evaluar

[1] https://www.postman.com
[2] https://jmeter.apache.org

cómo el sistema maneja el procesamiento intensivo de imágenes bajo condiciones de alta demanda, asegurando que el servicio mantenga su rendimiento óptimo incluso en escenarios de carga máxima.

Para la prueba de carga realizada en JMeter, se configuró un grupo de pruebas con 20 hilos de ejecución concurrentes, en un intervalo (ramp-up) de 1 segundo. Esto simula a 20 usuarios concurrentes accediendo al servicio en un corto período de tiempo. En este grupo de pruebas se incluyeron dos de los endpoints más utilizados y críticos: el de anonimización manual y el de anonimización automática. Cada solicitud se realizó a través de HTTPS, proporcionando la información necesaria para aplicar la anonimización sobre las imágenes adjuntadas. Durante la prueba, no se detectaron errores en las solicitudes, lo cual indica un rendimiento estable del servicio bajo las condiciones de carga especificadas. Los resultados obtenidos fueron los siguientes, la anonimización manual tuvo un tiempo promedio de 8.7 segundos por solicitud, mientras que la anonimización automática presentó un tiempo promedio de 35.0 segundos por solicitud. Estos resultados se pueden consultar en la Tabla 7.1. La diferencia en los tiempos de procesamiento refleja las distintas complejidades y demandas de los dos métodos de anonimización.

URL	**Average (ms)**	**Min (ms)**	**Max (ms)**	**Error %**
/automatic_blur	35068	16513	55980	0.0
/blur_image	8666	6537	10887	0.0

Tabla 7.1: Resultados de las pruebas de carga con JMeter

Capítulo 8

Despliegue de la solución

En este capítulo se describe el proceso mediante el cual se liberó públicamente la solución a través de la web. Este procedimiento se realizó con el objetivo de que el grupo de investigación de EDIGA pudiera llevar a cabo tanto anonimizaciones manuales como automáticas. La disponibilidad de la solución en línea no solo facilita el acceso y uso por parte de los investigadores de EDIGA, sino que también promueve la colaboración y la validación por parte de la comunidad científica en general. Este enfoque asegura que las técnicas y herramientas desarrolladas sean utilizadas de manera eficiente y efectiva, potenciando así los esfuerzos de investigación en el ámbito de la anonimización de datos.

8.1. Despliegue público

El servicio utilizado para el despliegue es la plataforma Elastic Cloud de Minube Antel.[1] Elastic Cloud es una plataforma como servicio (PaaS) ofrecida por Antel, diseñada para acelerar el desarrollo de aplicaciones, reducir costos y complejidad en la gestión de infraestructura, y mejorar la disponibilidad y seguridad. Permite desplegar aplicaciones en diversos lenguajes (Java, PHP, Node.js, Ruby, Python, Go) sin necesidad de modificar el código, utilizando tecnologías como Docker, GIT, SVN, entre otras. Ofrece escalabilidad automática vertical y horizontal durante picos de carga, distribución equitativa de tráfico entre instancias mediante balanceo de carga, facilitando la implementación de alta disponibilidad y alto rendimiento con clusterización automática y balanceo inteligente de carga. Elastic Cloud emplea un modelo de pago por uso basado en el consumo de recursos (cloudlets). Para este proyecto utilizamos un código de promoción que nos permite utilizar la plataforma sin costo por 12 meses.

[1] https://minubeantel.uy/index.php?NAME_PATH=Elastic_Cloud

8.1.1. Docker Engine

Dentro de la plataforma Elastic Cloud, se utiliza Docker Engine CE (ver Figura 8.1) para desplegar el sistema. Docker Engine CE es una plataforma de contenedorización de código abierto que permite a los desarrolladores empaquetar aplicaciones junto con todas sus dependencias en contenedores portátiles y autosuficientes, asegurando una ejecución consistente en cualquier entorno. Con un núcleo basado en el daemon de Docker, esta plataforma gestiona contenedores, imágenes, redes y volúmenes mediante una API y una interfaz de línea de comandos, proporcionando una solución robusta y eficiente para el despliegue de aplicaciones. Además, se agrega la herramienta de gestión de contenedores Portainer dentro de Docker Engine CE para simplificar la tarea.

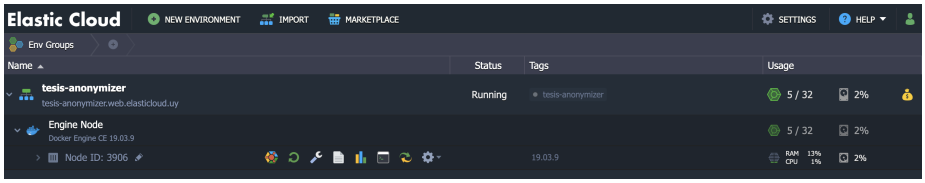


Figura 8.1: Docker Engine en Elastic Cloud

8.1.2. Portainer

Portainer es una herramienta de gestión de contenedores de código abierto que proporciona una interfaz gráfica de usuario fácil de usar para administrar entornos Docker y Kubernetes. Diseñada para simplificar la administración de contenedores, imágenes, redes y volúmenes, Portainer ofrece funcionalidades clave como la creación y gestión de contenedores, la descarga y administración de imágenes Docker, la configuración de volúmenes y redes, y la orquestación de servicios en Docker Swarm. Además, incluye controles de acceso basados en roles para gestionar permisos de usuarios y equipos, mejorando la seguridad y el control. Portainer es altamente valorada por su capacidad de mejorar la productividad, reducir la complejidad y hacer accesible la administración de Docker y Kubernetes.

Dentro de Portainer, se configura un ambiente de tipo Docker Standalone (ver Figura 8.2) a través de un Docker Compose para gestionar el despliegue de la plataforma. Docker Compose es una herramienta que permite definir y gestionar aplicaciones Docker multi-contenedor de manera declarativa a través de un archivo YAML. Permite especificar los servicios, redes y volúmenes que componen una aplicación, facilitando su configuración y despliegue de manera reproducible y consistente. Con Docker Compose, se pueden definir la estructura de su aplicación, incluyendo la configuración de cada servicio, la interconexión entre ellos y la gestión de variables de entorno.

En el Docker Compose se definieron los siguientes contenedores:

- **Backend:** Contenedor con la aplicación en Flask.

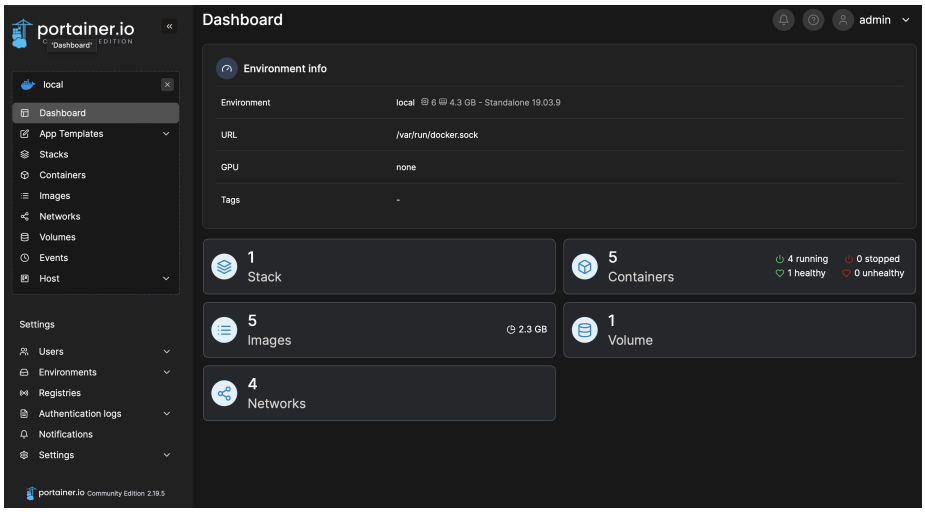


Figura 8.2: Entorno en Portainer

- **Frontend:** Contenedor con la aplicación en React sobre un Nginx.
- **Db:** Contenedor con la base de datos PostgreSQL.
- **Adminer:** Contenedor con un gestor de base de datos.

Además, se configuró una red que permite la comunicación entre todos los contenedores a través de Internet. Se estableció también un volumen dedicado para almacenar los datos de la base de datos, asegurando que la información persista de manera separada de los contenedores y sea accesible de manera eficiente durante el funcionamiento de la aplicación.

Portainer facilita la configuración adjuntando un archivo de configuración tipo .env al crear el entorno con Docker Compose. Esto permite utilizar variables de entorno en la definición de los contenedores para configurar usuarios, contraseñas, puertos, rutas, URIs y otras configuraciones necesarias de manera organizada y segura.

8.2. Despliegue local

Parte de los requerimientos no funcionales que nos plantea el equipo de EDIGA es la posibilidad de desplegar la solución de manera local, para no tener que compartir ningún tipo de información fuera de su equipo personal.

Dado que elegimos un desarrollo basado en contenedores para nuestra solución, aprovechamos esa característica para el despliegue de forma local a través del software Docker Desktop[2].

[2]https://www.docker.com/products/docker-desktop/

8.2.1. Guía para despliegue local

1. Descargar e instalar Docker Desktop.
2. Descargar archivos de configuración para entorno local.
3. Ejecutar el archivo *install.bat* desde el CMD en Windows o *install.sh* desde la terminal en Linux o MacOS, para instalar el servicio de forma local. Es necesario tener Docker Desktop abierto.
4. Desde Docker Desktop es posible iniciar y detener el servicio, que se encuentra en sección de actions, dentro de la instancia del contenedor llamado standalone (ver Figura 8.3).

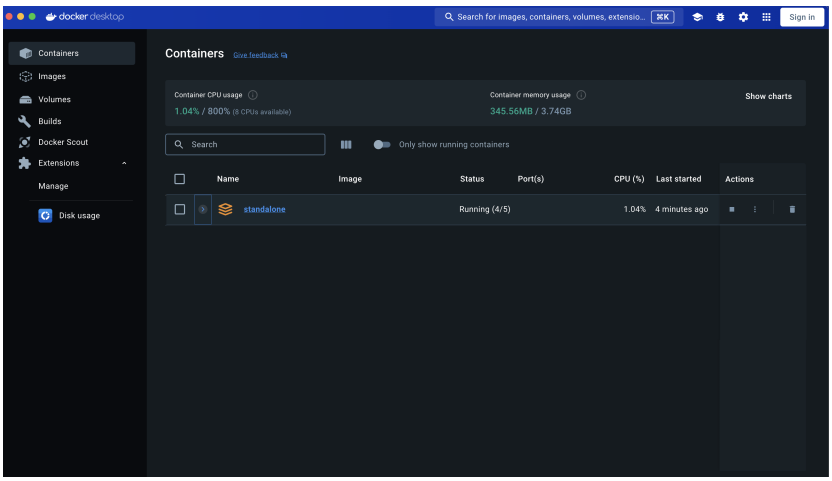


Figura 8.3: Servicio local corriendo en Docker Desktop

Capítulo 9

Conclusiones

En este proyecto se abordó el desafío de la anonimización de imágenes en el contexto de investigaciones sociales, destacando la importancia de proteger la privacidad y seguridad de los individuos en un mundo cada vez más digitalizado. El trabajo se enmarcó dentro del proyecto EDIGA (Entornos Digitales e Identidades de Género en la Adolescencia), cuyo objetivo es investigar cómo las prácticas en redes sociales influyen en la configuración de subjetividades e identidades de género entre adolescentes. Participan equipos de investigación de España, Uruguay y México, y el grupo de investigación de EDIGA juega un papel fundamental en el desarrollo y validación de las técnicas de anonimización aplicadas.

Se desarrolló un prototipo efectivo que permite a los investigadores del proyecto EDIGA anonimizar imágenes mediante técnicas avanzadas, asegurando que los rostros y otras características identificables sean irreconocibles. Esto se logró mediante la implementación de herramientas que facilitan tanto la anonimización manual como automática de imágenes, utilizando algoritmos como YOLO para la detección de rostros y técnicas de difuminado, pixelado y bloqueo para la protección de datos visuales.

Las pruebas realizadas demostraron que la solución desarrollada cumple con los requisitos de facilidad de uso, eficiencia y seguridad, proporcionando a los investigadores una herramienta robusta y accesible. La elección de tecnologías como OpenCV, React y Flask, junto con una arquitectura en capas bien definida, garantizó un desempeño óptimo y una gestión segura de la información.

Las fortalezas de la solución incluyen su capacidad para manejar grandes volúmenes de datos, la flexibilidad para integrarse con otras herramientas y la posibilidad de desplegarse tanto en entornos locales como en la nube, ofreciendo escalabilidad y accesibilidad remota. La herramienta desarrollada se destacó por su capacidad para anonimizar imágenes de manera efectiva, preservando la privacidad de los individuos mientras se mantiene la utilidad de los datos para la investigación.

A pesar de estos logros, se identificaron áreas para el trabajo futuro, como la mejora de la interfaz de usuario para permitir el marcado libre de áreas de

interés y la incorporación de métricas más precisas para evaluar el nivel de anonimización y el riesgo de reidentificación. También se sugiere explorar el uso de técnicas de aprendizaje profundo para mejorar aún más la precisión y eficiencia de los algoritmos de anonimización.

En resumen, la implementación de estas técnicas de anonimización no solo proporciona una herramienta valiosa para los investigadores del proyecto EDIGA, sino que también establece un marco para el uso seguro y ético de datos visuales en investigaciones futuras. Esto contribuye significativamente a la protección de la privacidad y a la integridad de los datos en el ámbito de las ciencias sociales y más allá.

Capítulo 10

Trabajo a futuro

Visualización del trabajo actual

En el contexto de un proyecto en desarrollo, se considera que el producto actual representa una base sólida que satisface las necesidades actuales de los usuarios finales. Este punto de partida se visualiza como el inicio de un proyecto más amplio y ambicioso, donde se planea integrar nuevas funcionalidades para mejorar la experiencia del usuario.
Las limitaciones financieras son un factor determinante en las decisiones de diseño e implementación del proyecto. Dado que no se cuenta con el financiamiento necesario para establecer servidores propios, se ha optado por soluciones que aprovechan infraestructuras y servicios disponibles en la nube. Esto ha permitido mantener los costos operativos dentro de límites razonables mientras se garantiza la escalabilidad y disponibilidad del sistema.

Propuestas de nuevas funcionalidades

Ofuscación de imágenes mediante dibujo libre

Esta función permitirá a los usuarios proteger la privacidad de imágenes mediante la superposición de trazos libres sobre áreas específicas de las imágenes. En lugar de la utilización de polígonos para delimitar las regiones de interés, seria posible realizar cualquier trazado sobre la imagen y anonimizarla con el tipo e intensidad deseado.
Se reconoce que es una de las funcionalidades a realizar a futuro que resultaría de gran valor para los usuarios finales, en este caso por tema de tiempos quedo fuera de alcance del presente proyecto.

Detección y anonimización de texto de forma automática

Esta función permitirá a los usuarios proteger la privacidad de imágenes mediante la detección de cualquier texto presente en ellas, ya sea de calles, lugares, o cualquier otra entidad identificable que pueda asociarse a la persona.
Para ello se requeriría la utilización de otro modelo especificado en texto ya que al evaluar distintas alternativas para el presente proyecto se concluyo que es más efectivo un modelo y algoritmo dedicado a un objetivo particular (rostros, texto, etc) que un modelo capaz de detectar cualquier objeto según parámetros, ya que la efectividad disminuye.
Esta nueva funcionalidad podría ser agregada dentro de la existente para anonimización automática y permitir al usuario seleccionar si quiere enfocarse en texto o rostros.

Historial de ediciones y registro de cambios

Para llevar a cabo esta funcionalidad, se considera necesario establecer un servidor localizado dentro del país de origen. Este servidor cumpliría con las normativas de protección de datos de la Unión Europea, como el Reglamento General de Protección de Datos (GDPR), que requiere que los datos personales sean almacenados y procesados dentro de jurisdicciones específicas para garantizar la seguridad y privacidad de los datos de los usuarios.

Se implementaría un sistema de registro detallado que mantendría un historial de todas las modificaciones realizadas en las imágenes. Cada cambio contaría con una etiqueta de fecha en la cual fue realizado y asociado al usuario que lo realizó, permitiendo así una trazabilidad completa de la evolución de cada imagen. Esta funcionalidad no solo serviría para auditorías internas, sino que también proporcionaría transparencia y control sobre los procesos de edición.

Edición compartida entre usuarios

Se desarrollaría la capacidad de editar imágenes de forma colaborativa entre múltiples usuarios autorizados. Esto incluiría la posibilidad de asignar permisos de edición a diferentes usuarios según roles definidos, como editores, revisores y administradores. Los cambios realizados por un usuario serían visibles en tiempo para otros colaboradores, facilitando el trabajo dentro de distintos proyectos o de revisión del contenido visual sobre el que han estado trabajando otros usuarios.

Implementación de roles de usuario

Esta funcionalidad esta fuertemente motivada por las dos funcionalidades anteriores. Se cree fuertemente que al implementarla mejoraría notoriamente la efectividad y el trabajo colaborativo entre los usuarios finales de la aplicación.

Roles definidos

Se establecerán roles específicos para usuarios dentro de la aplicación, como editores, revisores y administradores. Cada rol tendría permisos y privilegios distintos para acceder a la visualización y modificación de datos dentro de la aplicación.

Control de acceso y permisos

Los roles permitirán controlar quién puede acceder a qué funcionalidades y datos dentro de la plataforma. Por ejemplo, los editores podrían tener acceso completo para realizar ediciones en las imágenes propias y de otros, mientras que los revisores podrían tener permisos limitados para revisar y aprobar cambios pero sin la posibilidad de realizar modificaciones.

Desarrollo de una nueva vista de administración

Se desarrollará una interfaz de administración para la gestión centralizada de roles de usuario. Esto permitirá a los administradores asignar, modificar y revocar roles según las necesidades del equipo y del proyecto.

Referencias

Api4ai. (s.f.). https://api4.ai/apis/image-anonymization. (Accessed: 2023-15-11)

Celantur. (s.f.). https://www.celantur.com/. (Accessed: 2023-06-11)

ESPAÑOLA, R. A. (2024). *Diccionario de la lengua española, 23.ª ed.* [versión 23.7 en línea]. https://dle.rae.es. (Accessed: 2024-05-11)

Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., y Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, *88*(2), 303–338. Descargado de http://proxy.timbo.org.uy/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=48746637&lang=es&site=eds-live

Fiji image2j. (s.f.). https://imagej.net/learn/. (Accessed: 2023-10-11)

Gonzalez, R. C., y Woods, R. E. (2002). *Digital image processing* (2nd ed.). Upper Saddle River, NJ, USA: Prentice Hall.

Gupta, A. (s.f.). *Kaggle - dataset human-faces*. https://www.kaggle.com/datasets/ashwingupta3012/human-faces/. (Accessed: 2023-20-11)

Huaizu, J., y Erik, L. M. (2016). *Face detection with the faster r-cnn*. https://arxiv.org/pdf/1606.03473.pdf. (Accessed: 2023-29-10)

IBM. (2024). *¿qué es la segmentación semántica?* https://www.ibm.com/es-es/topics/semantic-segmentation. (Accessed: 2024-05-11)

Joseph, R., y Ali, F. (s.f.). *Yolov3: An incremental improvement*. https://arxiv.org/pdf/1804.02767.pdf. (Accessed: 2023-28-10)

Kaiming, H., Georgia, G., Piotr, D., y Ross, G. (2018). *Mask r-cnn*. https://arxiv.org/pdf/1703.06870.pdf. (Accessed: 2023-28-10)

Kerschbaum, F., y Paraboschi, S. (2018). *Data and Applications Security and Privacy XXXII* (Vol. LNCS-10980). Springer International Publishing. Descargado de https://inria.hal.science/hal-01954400 doi: 10.1007/978-3-319-95729-6

Minsky, M., y Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. Cambridge, MA, USA: MIT Press.

Opencv. (s.f.). https://opencv.org/. (Accessed: 2023-12-11)

OpenCV. (2024a). *Cascade classifier*. https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html. (Accessed: 2024-06-06)

OpenCV. (2024b). *Smoothing images*. https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html. (Accessed: 2024-06-06)

R. Fisher, S. Perkins, A. Walker and E. Wolfart. (2004). *Masking*. https://homepages.inf.ed.ac.uk/rbf/HIPR2/mask.htm.

Ren, S., He, K., Girshick, R., y Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. En C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, y R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 28). Curran Associates, Inc. Descargado de https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

Viola, P., y Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. En *Proceedings of the 2001 ieee computer society conference on computer vision and pattern recognition. cvpr 2001* (Vol. 1, p. I-I). doi: 10.1109/CVPR.2001.990517

Wei, L., Dragomir, A., Dumitru, E., Christian, S., Scott, R., Cheng-Yang, F., y Alexander C., B. (2016). *Ssd: Single shot multibox detector*. https://arxiv.org/pdf/1512.02325.pdf. (Accessed: 2023-29-10)

Anexo A

Anexo Proceso de Desarrollo

Este anexo tiene como objetivo proporcionar una visión del proceso de desarrollo utilizado en este proyecto específico. Se abordarán aspectos como: la planificación inicial, la metodología adoptada, las herramientas y tecnologías empleadas para garantizar la eficiencia y calidad del producto final.

A.1. Metodología de desarrollo utilizada

En este proyecto, se adoptó la metodología ágil para asegurar una gestión flexible del desarrollo, generando adaptabilidad frente a cambios que surgen en los requerimientos, con la posibilidad de realizar entregas de forma iterativa. Este ultimo aspecto permite obtener una retroalimentación temprana que se considera muy valiosa dentro del proceso de desarrollo del proyecto actual. A lo largo del tiempo el proceso mantuvo la comunicación activa entre los miembros del equipo, los tutores a cargo y los usuarios finales del producto.
A continuación se pueden visualizar características particulares del proceso

- Cada semana, el equipo se reúne para planificar y decidir las tareas y puntos a seguir. Estas reuniones, similares a las reuniones diarias en Scrum (daily stand-ups), permiten revisar el progreso, identificar obstáculos y ajustar el plan de trabajo. La colaboración constante y la comunicación abierta son esenciales para mantener a todo el equipo alineado y enfocado en los objetivos del proyecto.

- Se llevan a cabo reuniones con los usuarios finales en varias etapas del desarrollo. Estas sesiones permiten realizar un relevamiento detallado de los requerimientos y validar que las soluciones propuestas cumplen con las expectativas y necesidades del cliente. La retroalimentación continua de los usuarios finales ayuda a ajustar y mejorar el producto de manera iterativa.

- Se utiliza un board de Github con tickets para dar seguimiento a las tareas. Este tablero, similar a un tablero Kanban, proporciona una visualización clara del flujo de trabajo, permitiendo al equipo gestionar las tareas de manera eficiente y priorizar las actividades según su importancia y prioridad.
- Los requerimientos se modifican según las necesidades surgidas de las reuniones con los usuarios y respecto a la adopción de nuevas tecnologías. Esta flexibilidad permite al equipo adaptarse a los cambios y asegurar que el producto final tenga valor.

A.1.1. Github como plataforma de organización

En el proyecto se utilizó GitHub[1] para organizar y gestionar el desarrollo de software.
Se creó una organización en GitHub denominada "tesis-anonymizer". Esta organización actúa como un contenedor centralizado para todos los repositorios relacionados con el proyecto. La creación de una organización permite gestionar fácilmente los permisos de acceso y la colaboración entre los miembros del equipo.
Dentro de la organización, se crearon tres repositorios (backend, frontend y managment). Cada repositorio se destina a diferentes componentes del proyecto, permitiendo una separación clara de los módulos y funcionalidades. Esta estructura facilita la gestión, el seguimiento de cambios y la colaboración entre los desarrolladores.

GitHub Projects

Se creó un proyecto dentro de la organización "tesis-anonymizerütilizando la funcionalidad de GitHub Projects[2]. Este proyecto incluye un board de tickets que se utiliza para gestionar y priorizar las funcionalidades a desarrollar. Los tickets en el board representan tareas específicas y se pueden mover a través de diferentes columnas para reflejar el estado del trabajo.

[1] https://github.com/tesis-anonymizer
[2] https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects