UNMDP - Facultad de Ingeniería

Sistema de procesamiento de patrones de Speckle láser dinámico

Alumnos

- Angélico, Matías
- Bruses, Lautaro
- Suarez Quilis, Ignacio

Directores

- Guzmán, Marcelo
- Cujano, Estefany

Referente Funcional

- Meschino, Gustavo

Proyecto final para optar por el grado de Ingeniero en Informática

Departamento de Informática

Mar del Plata, 30 de abril de 2025

Autorización Repositorio Institucional -RINFI

Se presenta conjuntamente con la versión final del Trabajo Final

Repositorio Institucional RINFI, Facultad de Ingeniería, UNMDP

En calidad de TITULARES de los derechos de autor de la obra que se detalla a continuación, y sin infringir según mi conocimiento derechos de terceros, por la presente informo a la Facultad de Ingeniería de la UNMDP mi decisión de concederle en forma gratuita, no exclusiva y por tiempo ilimitado la autorización para:

1) Publicar el texto del trabajo más abajo indicado, exclusivamente en medio digital, en el sitio web de la Facultad y/o Universidad, por Internet, a título de divulgación gratuita de la producción científica generada por la Facultad, a partir de la fecha especificada.

2) Permitir a la Biblioteca que, sin producir cambios en el contenido, establezca los formatos de publicación en la web para su más adecuada visualización y la realización de copias digitales y migraciones de formato necesarias para la seguridad, resguardo y preservación a largo plazo de la presente obra:

Autor 1: ANGÉLICO MATÍAS NICOLÁS Documento: 33646199 Teléfono: 223.5556184 E-mail: MAngelico@gmail.com	Firma 1
Autor 2: LAUTARO NAHUEL BRUSES Documento: 42828582 Teléfono: 2236825109 E-mail: lautarobruses@gmail.com	Firma 2
Autor 3: IGNACIO GUSTAVO Suarez Quilis Documento: 43456655 Teléfono: 2235731797 E-mail: Ignacio.gustavosuarez1234@gmail.com	Firma 3
Director/a: Guzmán, Marcelo Nicolás Documento: 27130302 Leg. 12522	Firma Director/a
Codirector/a: Estefany Cujano Ayala Documento: 95601852 Leg. 18962-00	Firma Codirector/a

2. Título obtenido: INGENIERO EN INFORMÁTICA.

3. Identificación/Título de la Obra: SISTEMA DE PROCESAMIENTO DE PATRONES DE SPECKLE LÁSER DINÁMICO.

4. AUTORIZO la publicación bajo con la licencia Creative Commons BY-NC-ND Atribución-NoComercial-Sin Obra Derivada.

5. **Nota de Embargo:** Para aquellas obras que NO pueden ser de acceso a texto completo por razones de acuerdos previos con empresas o instituciones; por razones de índole comercial u otras razones; se procederá según lo establecido en Art. 6 de la Ley 26899 de Repositorios digitales institucionales de acceso abierto:

> **ARTICULO 6º** — En caso que las producciones científico-tecnológicas y los datos primarios estuvieran protegidos por derechos de propiedad industrial y/o acuerdos previos con terceros, los autores deberán proporcionar y autorizar el acceso público a los metadatos de dichas obras intelectuales y/o datos primarios, comprometiéndose a proporcionar acceso a los documentos y datos primarios completos a partir del vencimiento del plazo de protección de los derechos de propiedad industrial o de la extinción de los acuerdos previos antes referidos.
>
> Asimismo, podrá excluirse la difusión de aquellos datos primarios o resultados preliminares y/o definitivos de una investigación no publicada ni patentada que deban mantenerse en confidencialidad, requiriéndose a tal fin la debida justificación institucional de los motivos que impidan su difusión. Será potestad de la institución responsable en acuerdo con el investigador o equipo de investigación, establecer la pertinencia del momento en que dicha información deberá darse a conocer. A los efectos de la presente ley se entenderá como "metadato" a toda aquella información descriptiva sobre el contexto, calidad, condición o características de un recurso, dato u objeto, que tiene la finalidad de facilitar su búsqueda, recuperación, autentificación, evaluación, preservación y/o interoperabilidad.

En razón de lo expuesto, si el Trabajo se encuentra comprendido en el caso de que su producción esté protegida por derechos de Propiedad Industrial y/o acuerdos previos con terceros que implique la confidencialidad de los mismos, el/la directora/a debe indicar a continuación motivos y fecha de finalización del embargo:

NO SE AUTORIZA la publicación antes de la fecha __/__/____ por lo siguientes motivos:

.....

.....

.....

.....

.....

Cumplido el plazo del embargo, estará accesible a texto completo según contempla la normativa vigente.

Director/a del TF

UNMDP - Facultad de Ingeniería

Sistema de procesamiento de patrones de Speckle láser dinámico

Alumnos

- Angélico, Matías
- Bruses, Lautaro
- Suarez Quilis, Ignacio

Directores

- Guzmán, Marcelo
- Cujano, Estefany

Referente Funcional

- Meschino, Gustavo

Proyecto final para optar por el grado de Ingeniero en Informática

Departamento de Informática

Mar del Plata, 30 de abril de 2025

Agradecimientos

A nuestra familia y amigos, gracias por el apoyo, la confianza y el cariño de cada día. Este proyecto es fruto de un gran camino colectivo; que sirva de homenaje a todo el amor que nos brindaron durante el proceso.

Al Laboratorio de Bioingeniería de la Facultad de Ingeniería, especialmente al Dr. Ing. Marcelo Guzmán y a la Ing. Estefany Cujano Ayala, por la paciencia, las correcciones y las facturas de los miércoles. También al Dr. Ing. Gustavo Meschino por la oportunidad y la claridad de sus ideas.

A la Facultad de Ingeniería por el espacio y las herramientas brindadas. Y a todos los docentes de la carrera por su conocimiento, su influencia y la dedicación con que nos formaron en lo académico, lo profesional y lo personal.

1. Resumen

Este proyecto responde a una necesidad específica del Laboratorio de Bioingeniería de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata: contar con un software dedicado que, a partir de videos obtenidos mediante la técnica de Speckle Láser Dinámico (DLS), permita identificar de manera integral e intuitiva la actividad subyacente en las muestras analizadas.

El objetivo principal de este proyecto es el desarrollo de una aplicación web que abarque las etapas fundamentales del ciclo de vida del software, desde la planificación y el diseño hasta la implementación y puesta en funcionamiento. El sistema permitirá procesar videos experimentales y, a través de su integración con modelos de inteligencia computacional, generar una imagen segmentada resultante.

La aplicación está diseñada para ser accesible, sin requerir conocimientos en programación ni el uso de software técnico especializado. Además, los resultados son exportables, lo que facilita la reproducibilidad y el análisis comparativo de los experimentos de laboratorio.

La implementación de un producto funcional que satisface las necesidades del Laboratorio de Bioingeniería valida el éxito en el cumplimiento de los objetivos iniciales. La aplicación se encuentra funcionando en el Laboratorio de Bioingeniería de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata.

2. Introducción

Cuando un haz de luz coherente incide sobre una superficie o un medio con imperfecciones del orden de la longitud de onda de la fuente de luz, se genera un patrón de interferencia compuesto por zonas de mayor y menor intensidad, denominadas *speckles*. Si la muestra presenta alguna actividad que altere su índice de refracción en el tiempo, este patrón evoluciona dinámicamente, dando lugar al fenómeno conocido como *Speckle dinámico* o por sus siglas en inglés DLS (*Dynamic Speckle Láser)*. En el caso de muestras biológicas, este efecto se denomina *Biospeckle*.

A nivel microscópico, la luz incide sobre todas las moléculas de la muestra, las cuales difractan la luz en todas direcciones. La interferencia entre las ondas difractadas puede ser constructiva (regiones claras) o destructiva (regiones oscuras), formando patrones de moteado que cambian en el tiempo. Estos patrones pueden clasificarse como patrones en ebullición, donde los granos de speckle se mueven, deforman, desaparecen y reaparecen sin un desplazamiento significativo de su posición media. Para analizar estos cambios se compara la intensidad de la luz en cada punto a lo largo del tiempo, permitiendo extraer información sobre la dinámica del sistema.

El patrón de speckle dinámico varía de forma aparentemente aleatoria; sin embargo, está relacionado con fenómenos internos que ocurren en la muestra, por lo que el DLS proporciona una herramienta interesante, rápida, no destructiva y no invasiva para el estudio de muestras biológicas. La DLS se ha utilizado, entre otras cosas, para detectar la quimiotaxis bacteriana y para diferenciar el crecimiento de hongos y bacterias en medios semisólidos. Recientemente se ha aplicado en la monitorización en tiempo real de la cinética bacteriana, para estimar la eficacia en la desinfección del agua y para estimar la concentración bacteriana en muestras acuosas.

Cuando los dispersores son móviles , como por ejemplo las bacterias, que realizan movimientos deliberados de natación, se producen fluctuaciones en el patrón de speckle. Si la acción de nadar es irregular y aleatoria, entonces el espectro de

frecuencia de las fluctuaciones del speckle será cualitativamente como el del movimiento browniano, pero con una escala de tiempo apropiada para la velocidad de nado. La luz dispersada, al ser coherente, interfiere consigo misma de forma constructiva y/o destructiva. Dentro de esta fluctuación de intensidad se encuentra información sobre la escala temporal del movimiento de los dispersores. El espectro resultante será continuo y contendrá todas las frecuencias hasta un límite superior, determinado por el tamaño de la mota y la velocidad de nado. Las señales generadas por los cambios de intensidad en cada píxel a través de la secuencia de imágenes Speckle se procesan para identificar la actividad subyacente de la muestra.

Para la adquisición de este tipo de señales se emplean dos técnicas: backscattering y forwardscattering.

La técnica de backscattering, o retrodispersión, consiste en detectar la luz que ha sido reflejada o dispersada en dirección contraria a su incidencia sobre la muestra. En este caso, tanto la fuente de iluminación como el detector se ubican del mismo lado, lo que permite analizar la superficie o estructuras internas cercanas sin necesidad de atravesar el material.

Por otro lado, la técnica de forwardscattering implica la detección de la luz que atraviesa la muestra y se dispersa hacia adelante, es decir, en la misma dirección de la fuente emisora. En este caso, el detector se coloca en el lado opuesto a la fuente de iluminación, lo que permite obtener información relevante sobre estructuras internas más profundas o sobre materiales semitransparentes. Esta configuración resulta especialmente útil cuando se busca analizar la transmisión de luz a través de una muestra.

Aunque los métodos de adquisición difieren en su configuración, los patrones generados por ambas técnicas presentan características similares, lo que permite su análisis mediante enfoques comparables.

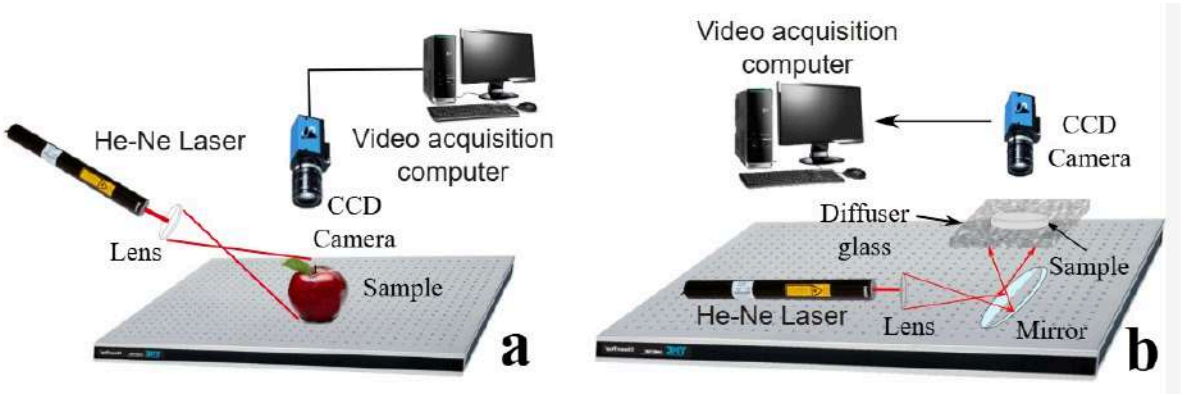

Figura 1. Arreglos experimentales. Imagen A: backscattering. Imagen B: forwardscattering.

Para el análisis de las experiencias adquiridas se calculan distintas funciones llamadas descriptores, las cuales permiten obtener una medida cuantitativa del nivel de actividad presente en una serie temporal. Estos descriptores capturan diferentes propiedades dinámicas de la señal, facilitando la representación compacta de la información contenida.

Una vez calculados, los descriptores se utilizan como entrada para la aplicación de técnicas de clustering, un enfoque de aprendizaje no supervisado que permite agrupar patrones similares sin necesidad de contar con etiquetas previas. Este proceso identifica regiones o comportamientos comunes, realizando una clasificación automática de distintos niveles de actividad.

Posteriormente, con los datos de descriptores y agrupamientos obtenidos, se entrena un modelo computacional basado en redes neuronales artificiales, el cual se inspira en el funcionamiento del cerebro humano, simulando la manera en que las neuronas biológicas procesan la información. Este modelo permite aprender relaciones complejas dentro de los datos y generalizar sobre nuevos casos.

Finalmente, estos modelos son capaces de clasificar los niveles de actividad en nuevas experiencias, a partir de los patrones previamente aprendidos.

3. Planificación del proyecto

3.1. Objetivos del proyecto

3.1.1. Objetivo general

Desarrollar un software integral que cubra las etapas del ciclo de vida del sistema: desde el análisis y diseño, pasando por la implementación y pruebas, hasta el despliegue del mismo para facilitar el procesamiento de patrones de DLS adquiridos en el Laboratorio de Bioingeniería, con el fin de simplificar la evaluación del nivel de actividad en muestras biológicas e industriales con una interfaz intuitiva y consistente, que permita al usuario interactuar de manera sencilla con el sistema.

3.1.2. Objetivos específicos

- Gestionar un proyecto con impacto real con usuarios que utilizan el sistema en sus tareas diarias.
- Interactuar con demandantes que requieren soluciones funcionales y respuestas concretas.
- Aplicar herramientas aprendidas durante la formación para recorrer todas las etapas del ciclo de vida del software.
- Adaptarse a un entorno profesional cambiante, distinto al entorno académico controlado.
- Entregar valor real a los usuarios mediante soluciones útiles, estables y utilizables.
- Aprender nuevas tecnologías demandadas actualmente en el mercado laboral.

3.2. Problema a resolver

En la actualidad, el cálculo de descriptores para el análisis de datos de experiencias de Speckle Dinámico dentro del LABI se realiza mediante software propietario como Matlab®. Este enfoque presenta múltiples limitaciones: no solo implica costos de

licenciamiento, sino que también requiere conocimientos técnicos avanzados, restringiendo y limitando su acceso a usuarios no especializados.

Asimismo, el uso de herramientas que no fueron diseñadas específicamente para el procesamiento de descriptores ralentiza y complejiza los flujos de trabajo, lo que afecta tanto la calidad de los resultados como la eficiencia en los tiempos de análisis.

Otro problema significativo es la falta de integración de los procesos en una única herramienta, lo que aumenta la posibilidad de error humano y compromete la objetividad en el análisis.

En resumen, el proceso actual de análisis en experiencias de Speckle Dinámico es costoso en términos de licenciamiento, técnico, fragmentado y poco accesible. La falta de herramientas especializadas, la necesidad de conocimientos avanzados y la ausencia de automatización en tareas clave dificultan la eficiencia, precisión y reproducibilidad de los datos.

3.3. Alcance del proyecto

En una etapa temprana del proyecto, el alcance se limitaría al cálculo de descriptores integrado con modelos predictivos como Mapas Autoorganizados (*SOM, Self Organizing Maps*) y algoritmos de *clustering*. Sin embargo, por solicitud del Referente funcional, la técnica de Mapas Autoorganizados no fue implementada y, en su lugar, se optó por reemplazarla con el uso de redes neuronales (*feed-forward*) supervisadas. Este ajuste fue parte de la evolución de los requerimientos y objetivos, lo que dio lugar al alcance que se describe a continuación.

El sistema se estructura en dos procesos principales: entrenamiento y consulta, cada uno con un flujo de pasos bien definidos y un sistema de validación activa de la información. Para asociar cada proceso a su respectivo usuario, se implementa una gestión de usuarios que permite persistir la información generada durante la interacción con el sistema.

Finalmente, el sistema ofrece la opción de descargar los resultados obtenidos en varios formatos. Garantiza su exportabilidad y facilita su uso en distintos entornos. Los resultados pueden ser representados como matrices o visualizados en mapas de calor.

3.4. Características de innovación del proyecto

El proyecto propone una solución innovadora que permitirá al usuario interactuar de manera dinámica con los descriptores y la configuración de sus hiperparámetros, ofreciendo una experiencia de usuario personalizada para la evaluación de secuencias de imágenes obtenidas mediante la técnica de Speckle Láser Dinámico. Esta herramienta no solo optimiza el proceso de análisis de datos, sino que facilita el ajuste de parámetros en tiempo real, adaptándose a las necesidades específicas de cada experiencia.

Esta aplicación proporciona una solución integral, que abarca desde el cálculo de descriptores hasta la clasificación mediante redes neuronales, todo ello a través de una interfaz gráfica que abstrae los detalles técnicos y complejos del usuario final. Su enfoque innovador es relevante para el LABI, ya que ofrece una plataforma accesible para investigadores sin experiencia previa en herramientas complejas ni conocimientos técnicos avanzados, a la vez que conserva la flexibilidad necesaria permitiéndole a investigadores principiantes obtener resultados comparables con los de investigadores más experimentados de forma rápida y sencilla.

3.5. Análisis FODA

El análisis FODA es una herramienta de planificación estratégica que permite evaluar la situación de un proyecto de manera integral a partir de 4 dimensiones claves: fortalezas, oportunidades, debilidades y amenazas.

Las fortalezas y debilidades corresponden a aspectos internos del equipo o del producto, mientras que las oportunidades y amenazas provienen del entorno externo. Esta metodología no solo facilita la comprensión del contexto actual del proyecto, sino que también orienta la toma de decisiones estratégicas: se busca

potenciar las fortalezas, aprovechar las oportunidades, mitigar las debilidades y anticipar o reducir el impacto de las amenazas.

Inicialmente se plantearon los factores que se detallan a continuación:

3.5.1. Fortalezas

- **Equipo de trabajo**: El grupo ha colaborado en diversos proyectos académicos a lo largo de la carrera, lo que ha permitido identificar las fortalezas individuales de cada integrante.
- **Experiencia de los desarrolladores**: Cada integrante del equipo cuenta con experiencia previa en el desarrollo de aplicaciones web y APIs, aunque en distintos niveles y enfoques. Este bagaje técnico permite una distribución eficiente de las tareas, asignando responsabilidades de acuerdo con las competencias particulares de cada miembro.
- **Especialización del software**: Al estar dedicado exclusivamente al cálculo de descriptores en experiencias de Speckle Dinámico, el software se adapta a las necesidades específicas del área, simplificando el proceso para los investigadores.
- **Facilidad de uso**: Eliminar la necesidad de que los usuarios posean conocimientos avanzados en programación, incrementa notablemente la accesibilidad del sistema y se reduce la barrera técnica de entrada.

3.5.2. Oportunidades

- **Ampliación a otros laboratorios:** El software posee el potencial de ser escalado o adaptado para su implementación en otros laboratorios o centros de investigación que empleen la técnica de Speckle Dinámico. Entre los posibles beneficiarios se encuentran grupos como el GIB (Grupo de Ingeniería Bioquímica) o el GIPCAL (Grupo de Investigación en Preservación y Calidad de Alimentos), cuyos trabajos podrían verse favorecidos por una herramienta especializada y de fácil utilización.
- **Aplicación en otras áreas:** El proyecto posee un alto grado de versatilidad, lo que permitiría su adaptación a otros campos que requieran análisis de

imágenes y extracción de descriptores. Ámbitos como la medicina o la ingeniería de materiales podrían beneficiarse de una herramienta con características similares, adaptada a sus necesidades específicas.

- **Innovación en el campo de la bioingeniería:** La incorporación de inteligencia computacional en una herramienta especializada para el análisis de Speckle Dinámico representa un aporte innovador dentro del ámbito de la bioingeniería. Esta integración no solo optimiza los procesos de análisis, sino que también tiene el potencial de abrir nuevas líneas de investigación y generar oportunidades para futuras publicaciones científicas.

3.5.3. Debilidades

- **Dominio del problema:** Dado que el área de aplicación del software pertenece a un campo específico de la bioingeniería, el equipo de desarrollo no posee un conocimiento profundo del dominio del problema.
- **Escala del proyecto:** Ninguno de los integrantes del equipo había participado previamente en un proyecto de esta magnitud y complejidad, que implica abarcar de forma integral todas las etapas del desarrollo de software.
- **Incertidumbre de la investigación:** El software no garantiza por sí mismo el éxito ni la validez de los resultados obtenidos. El desempeño de cualquier investigación depende de múltiples factores adicionales, como la calidad de los datos recolectados, el diseño experimental implementado y la correcta interpretación de los resultados por parte del equipo investigador.
- **Falta de conocimientos técnicos específicos:** La ausencia de experiencia en la implementación del cálculo de descriptores y el manejo de ciertas tecnologías clave representan una barrera inicial, que implica una inversión adicional de tiempo en procesos de aprendizaje y capacitación.

3.5.4. Amenazas

- **Evolución tecnológica:** La constante y acelerada evolución de las tecnologías asociadas a la inteligencia computacional y al procesamiento de imágenes representa un desafío para la vigencia del software desarrollado.

- **Disponibilidad de tiempo:** La disponibilidad horaria de los integrantes del equipo puede variar de forma impredecible debido a compromisos académicos, laborales o personales.
- **Lentitud en el procesamiento:** La implementación de modelos de inteligencia computacional en servidores remotos puede implicar tiempos de respuesta elevados, especialmente trabajando con procesos muy demandantes de recursos.

3.6. Metodología de trabajo

El equipo de trabajo está compuesto por seis integrantes: tres estudiantes de la carrera de Ingeniería Informática, encargados del desarrollo y la gestión del proyecto; dos directores de proyecto y un referente funcional, quienes participaron activamente como *stakeholders* en la definición del producto y actuarán como usuarios finales.

La metodología planteada para la gestión del proyecto es de tipo tradicional, con un enfoque secuencial basado en las etapas del ciclo de vida del desarrollo de software. Como parte de esta planificación, se establecieron reuniones quincenales del equipo de trabajo para informar avances, resolver dudas y tomar decisiones relevantes.

Siguiendo la arquitectura definida, la cual se detalla más adelante, el trabajo se distribuye en tres módulos principales: frontend, backend y API de cálculo. La asignación de tareas se realiza según las especializaciones e intereses de cada miembro, lo que permite una organización eficiente y explota las fortalezas de cada miembro del equipo, como la especialización técnica en distintas áreas, lo que facilita el desarrollo y optimiza el aprovechamiento de las capacidades individuales.

3.7. Planificación inicial

A continuación, se presenta el diagrama de Gantt inicial del proyecto, que muestra el inicio y final de cada tarea, distribuida en semanas.

- **Fase de Análisis (Semanas 1-5):** Se define el problema, se identifican los objetivos y se establecen los requisitos iniciales del proyecto.
- **Fase de Diseño (Semanas 6-8):** Se piensa cómo se va a construir el sistema: arquitectura, herramientas a usar y estructura general del software.
- **Fase de Implementación (Semanas 9-18):** Se empieza a programar. Se desarrollan todas las funciones que se definieron dentro de la etapa de diseño.
- **Fase de Testing (Semanas 19-21):** Se prueban las funcionalidades desarrolladas para encontrar errores y corregirlos antes de poner el sistema en marcha.
- **Fase de Despliegue (Semanas 22-23):** El sistema se pone en funcionamiento en un entorno real o de producción.
- **Fase de Finalización (Semanas 24-27):** Se documenta todo, se realiza la redacción del informe, se entregan los resultados finales ante los *stakeholders* y se da por cerrado el proyecto

El plan inicial contempla una dedicación aproximada de 10 horas semanales por integrante, con flexibilidad en la distribución. Esto permite que, en determinadas semanas, se destinen más o menos horas, compensando el tiempo según la disponibilidad de cada miembro.

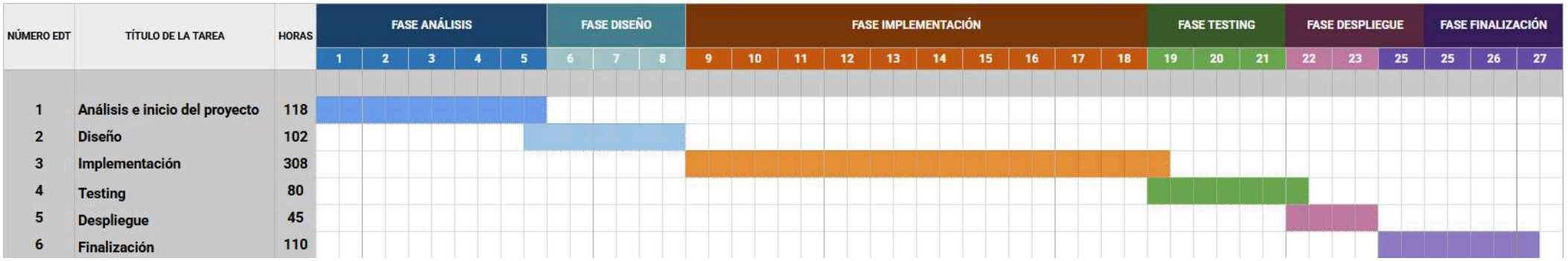

Figura 2. Diagrama de Gantt del proyecto mostrando la planificación detallada de las fases y tareas involucradas. Se puede observar la distribución inicial de las horas asignadas a cada fase del proyecto, que abarca desde el análisis y diseño hasta la implementación, testing, despliegue y finalización.

4. Desarrollo del sistema

4.1. Análisis del problema

La fase de análisis tiene como objetivo identificar y definir las necesidades que el sistema debe satisfacer. En este contexto, los requerimientos representan las condiciones, funcionalidades y expectativas que la aplicación debe cumplir de acuerdo al dominio del problema.

En este proyecto, el análisis del problema se realizó de forma iterativa y continua. A lo largo del desarrollo, varios requerimientos fueron añadidos, modificados o descartados, en función de los avances técnicos y la retroalimentación recibida, con el objetivo de asegurar que el producto final se ajustara de manera precisa a las necesidades del dominio del problema.

4.1.1. Requerimientos

El sistema debe calcular múltiples descriptores a partir de un video, ofreciendo la posibilidad de ajustar los valores de cálculo. A partir de las matrices generadas por estos descriptores, el sistema aplicará algoritmos de *clustering*, permitiendo al usuario modificar los parámetros de estos métodos para obtener resultados acordes a la experiencia realizada.

El sistema debe incluir la arquitectura de una red neuronal (*feed-forward*), permitiendo configurar características como el número de capas, las neuronas en cada capa, la cantidad de épocas de entrenamiento y la inclusión de técnicas como *Batch Normalization* y *Dropout*. Para entrenar la red neuronal se utilizara un vector de características que se genera utilizando el resultado de aplicar los descriptores y las técnicas de clustering. Este modelo podrá almacenarse lo que permitirá al usuario realizar consultas sobre nuevos vídeos. Esta funcionalidad brindará información clave que contribuirá a la optimización de procesos, análisis y toma de decisiones, por parte de los investigadores del Laboratorio de BioIngeniería de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata (LABI).

Los requerimientos se dividen en requerimientos funcionales, que establecen qué debe hacer el sistema, y no funcionales, que indican cómo debe comportarse bajo determinadas condiciones.

4.1.1.1. Requerimientos funcionales (RF)

- **RF01:** El sistema debe permitir la carga de un video en formato AVI para el entrenamiento y la consulta.
- **RF02:** El sistema debe ser capaz de calcular 17 descriptores específicos del video cargado.
- **RF03:** El sistema debe permitir la selección y configuración de los descriptores a calcular.
- **RF04:** El sistema debe ser capaz de calcular 5 métodos de *clustering* específicos tomando como entrada las matrices resultantes del cálculo de descriptores.
- **RF05**: El sistema debe permitir la selección y configuración de los algoritmos de *clustering* a utilizar.
- **RF06**: El sistema debe ser capaz de generar la matriz de características a partir de los descriptores y método de *clustering* seleccionados.
- **RF07:** El sistema debe permitir entrenar un modelo de red neuronal (*feedforward)* a partir de una matriz de características.
- **RF08:** El sistema debe permitir la configuración de parámetros para el proceso de entrenamiento, tales como *batch normalization*, número de épocas y *dropout*.
- **RF09:** El sistema debe mostrar la matriz de confusión como resultado del entrenamiento.
- **RF10:** El sistema debe permitir persistir el modelo entrenado junto con sus configuraciones e información asociada.
- **RF11:** El sistema debe permitir a los usuarios buscar y filtrar los modelos de entrenamiento guardados por nombre y fecha de creación.
- **RF12:** El sistema debe permitir reutilizar un modelo previamente entrenado para evaluar un nuevo video y obtener una clasificación basada en dicho modelo.

- **RF13:** El sistema debe calcular los descriptores de un nuevo video utilizando la configuración de un modelo entrenado.
- **RF14:** El sistema debe generar una representación visual de las matrices de los descriptores, de las matrices de *clustering* y el resultado del proceso de consulta.
- **RF15:** El sistema debe permitir la descarga de las imágenes generadas en diferentes formatos (PNG, JPG, SVG, PDF).
- **RF16:** El sistema debe permitir la descarga de las matrices obtenidas en diferentes formatos (TXT, CSV, JSON).
- **RF17:** El sistema debe permitir que los usuarios se registren en el sistema utilizando un nombre de usuario y contraseña.
- **RF18:** El sistema debe permitir que los usuarios inicien sesión en el sistema utilizando su nombre de usuario y contraseña.
- **RF19:** El sistema debe permitir que los usuarios se registren en el sistema utilizando su cuenta de Google.
- **RF20:** El sistema debe permitir que los usuarios inicien sesión en el sistema utilizando su cuenta de Google.

4.1.1.2. Requerimientos no funcionales (RNF)

- **RNF01:** La aplicación debe ser compatible con las dos versiones estables más recientes de Chrome, Firefox, Edge y Safari vigentes en mayo de 2025.
- **RNF02**: En caso de error crítico, el sistema debe mostrar una página de error personalizada.
- **RNF03**: El sistema deberá ser compatible con una resolución mínima de 1024x600 píxeles en dispositivos de escritorio.
- **RNF04**: El sistema debe permitir agregar nuevos módulos sin necesidad de modificar los existentes.
- **RNF05:** El sistema debe ser accesible y fácil de usar para investigadores sin experiencia técnica avanzada.

4.2. Diseño del sistema

Durante esta etapa se definió el flujo de información dentro del sistema, así como las interacciones entre sus componentes y con el usuario final. Además, se modeló

la arquitectura general de la solución y se seleccionaron las tecnologías más adecuadas para su implementación teniendo en cuenta los requerimientos planteados.

4.2.1. Arquitectura del sistema

El sistema se organiza en tres dominios lógicos: frontend, backend y una API de cálculo.

El frontend (capa de presentación), desarrollado como una interfaz web, se encarga de la interacción con el usuario final. Este componente se comunica con Auth0 para gestionar la autenticación de usuarios, y luego realiza peticiones al backend a través de una API REST.

El backend (capa de negocio), valida los tokens JWT emitidos por Auth0, aplica la lógica de negocio y orquesta la interacción entre los distintos componentes del sistema. Funciona como nexo entre el frontend, la base de datos y la API de cálculo (capa de procesamiento), con la cual se comunica a través de una API REST para delegar las operaciones numéricas complejas.

La persistencia en la base de datos se delega en MongoDB Atlas, un servicio administrado que simplifica el escalado y el backup.

Todas las comunicaciones —internas y externas— se realizan a través de HTTPS (TLS 1.3), garantizando confidencialidad e integridad de los datos.

Esta separación facilita el mantenimiento, la evolución y brinda autonomía a cada componente ante cambios futuros en los requerimientos.

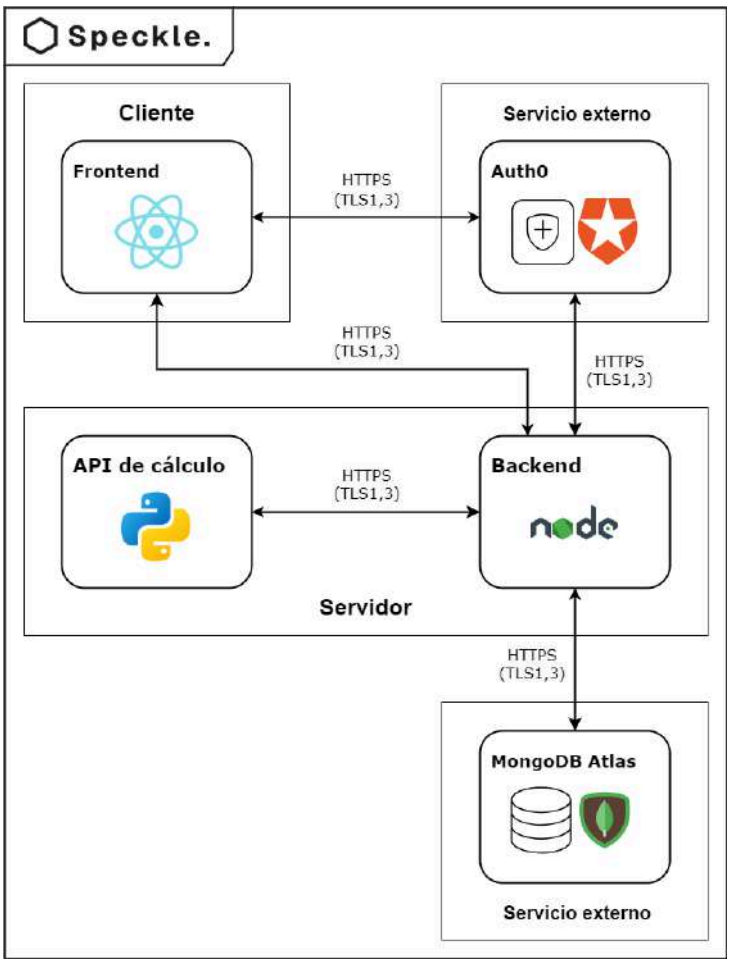

Figura 3. Arquitectura lógica del sistema.

4.2.2. Diagramas de secuencia de procesos principales

A continuación se presentan los diagramas de secuencia que ilustran el flujo de interacción entre los componentes del sistema para los dos procesos principales: Entrenamiento y Consulta. Estos diagramas permiten visualizar de manera detallada los mensajes intercambiados entre los distintos módulos involucrados.

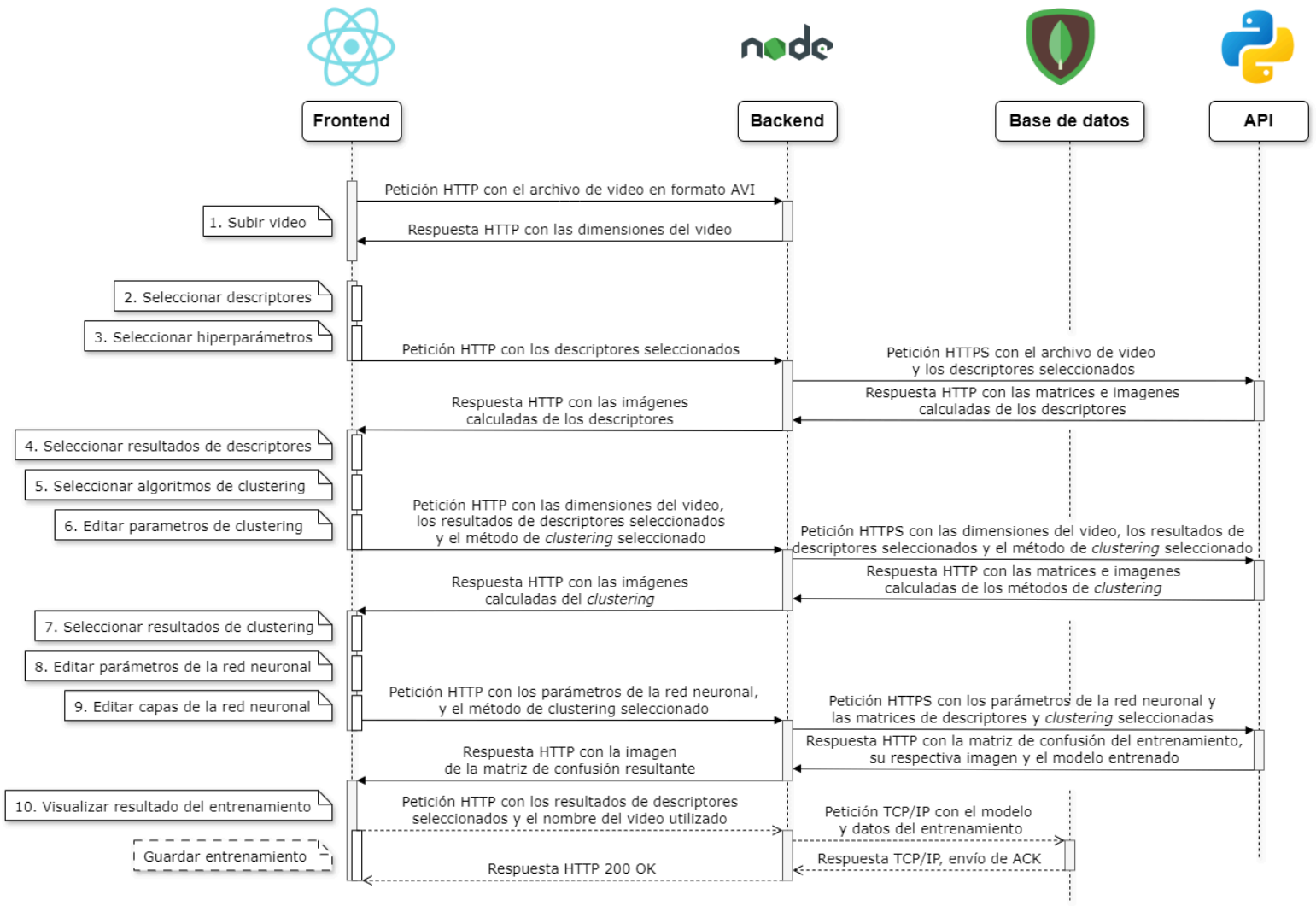
Frontend
Backend
Base de datos
API
Petición HTTP con el archivo de video en formato AVI
1. Subir video
Respuesta HTTP con las dimensiones del video
2. Seleccionar descriptores
3. Seleccionar hiperparámetros
Petición HTTP con los descriptores seleccionados
Petición HTTPS con el archivo de video y los descriptores seleccionados
Respuesta HTTP con las matrices e imagenes calculadas de los descriptores
Respuesta HTTP con las imágenes calculadas de los descriptores
4. Seleccionar resultados de descriptores
5. Seleccionar algoritmos de clustering
6. Editar parametros de clustering
Petición HTTP con las dimensiones del video, los resultados de descriptores seleccionados y el método de clustering seleccionado
Petición HTTPS con las dimensiones del video, los resultados de descriptores seleccionados y el método de clustering seleccionado
Respuesta HTTP con las matrices e imagenes calculadas de los métodos de clustering
Respuesta HTTP con las imágenes calculadas del clustering
7. Seleccionar resultados de clustering
8. Editar parámetros de la red neuronal
9. Editar capas de la red neuronal
Petición HTTP con los parámetros de la red neuronal, y el método de clustering seleccionado
Petición HTTPS con los parámetros de la red neuronal y las matrices de descriptores y clustering seleccionadas
Respuesta HTTP con la matriz de confusión del entrenamiento, su respectiva imagen y el modelo entrenado
Respuesta HTTP con la imagen de la matriz de confusión resultante
10. Visualizar resultado del entrenamiento
Guardar entrenamiento
Petición HTTP con los resultados de descriptores seleccionados y el nombre del video utilizado
Petición TCP/IP con el modelo y datos del entrenamiento
Respuesta TCP/IP, envío de ACK
Respuesta HTTP 200 OK

Figura 4. Representación detallada del flujo de interacción entre frontend, backend, base de datos y API durante el proceso de entrenamiento, abarcando la configuración de descriptores, algoritmos de clustering, parámetros de red neuronal y el almacenamiento final del entrenamiento generado (opcional).

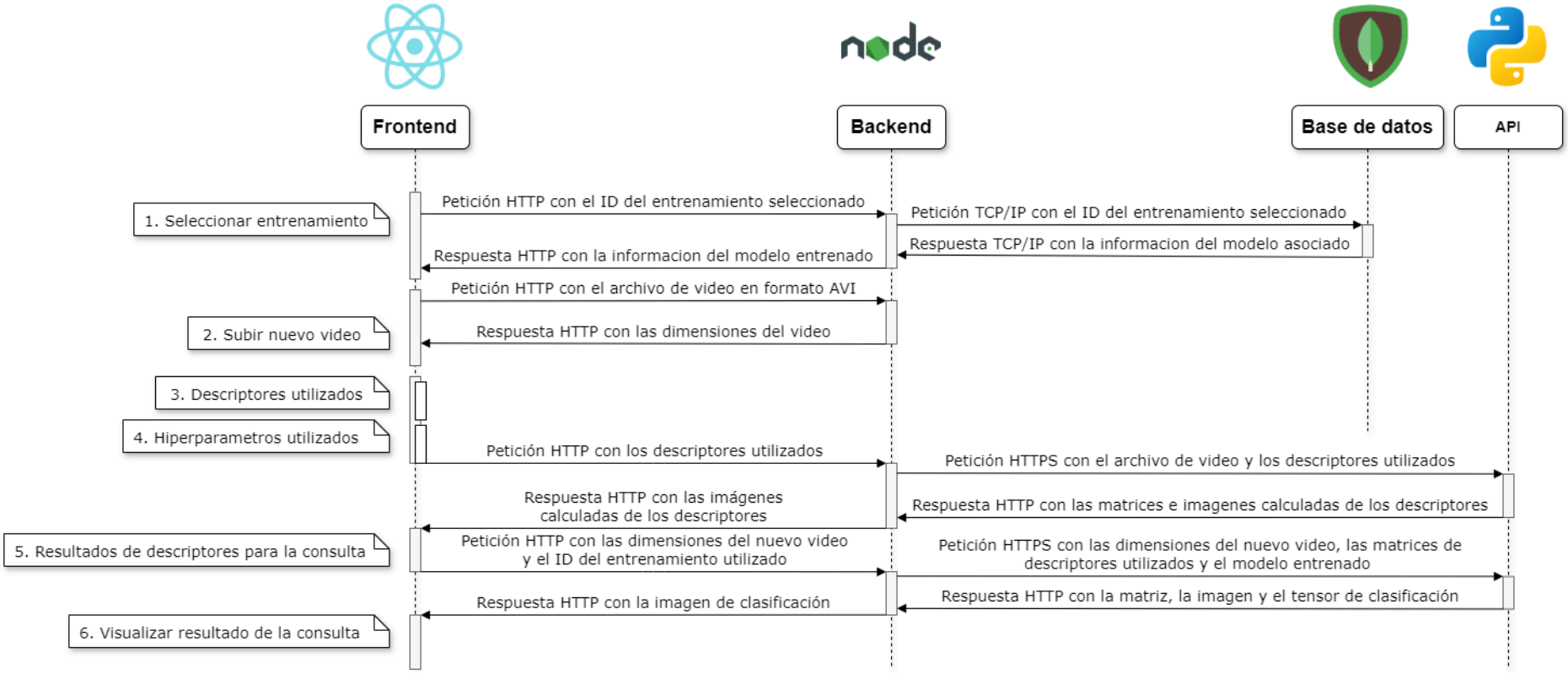

Figura 5. Representación detallada del flujo de interacción entre frontend, backend, base de datos y API durante el proceso de consulta, desde la selección del entrenamiento y el envío del video hasta la visualización de los resultados de predicción.

4.3. Frontend

Es la interfaz visual que permite a los usuarios interactuar con el sistema, ocultando las complejidades de la lógica y funcionalidades que ocurren en las capas inferiores.

4.3.1. Tecnologías utilizadas

Para el desarrollo del frontend se emplearon tecnologías modernas que permiten una experiencia de usuario fluida, una arquitectura escalable y un entorno de desarrollo eficiente.

El núcleo del frontend fue desarrollado utilizando React, una biblioteca de JavaScript ampliamente adoptada para la construcción de interfaces de usuario basadas en componentes. Su enfoque declarativo y la posibilidad de reutilizar componentes facilitaron el mantenimiento y la escalabilidad del proyecto.

La elección de React como biblioteca principal para el desarrollo del frontend se fundamentó, en primer lugar, en la experiencia previa del desarrollador con esta tecnología, lo que permitió optimizar los tiempos de desarrollo y reducir significativamente la curva de aprendizaje. Además, React ofrece una arquitectura basada en componentes que facilita la modularidad del código, lo que mejora tanto la escalabilidad como la mantenibilidad del proyecto. Su eficiencia en la gestión del estado y el uso de un DOM virtual permite actualizaciones rápidas de la interfaz sin sobrecargar el rendimiento.

La familiaridad con React también permitió una implementación ágil, con un menor margen de error y una integración más sencilla con otras herramientas y bibliotecas. Esta combinación de factores hizo que React fuera la opción ideal para el desarrollo del frontend.

Para el entorno de desarrollo y empaquetado se optó por Vite, una herramienta de construcción moderna que destaca por su velocidad de arranque y recarga en caliente.

4.3.2. Principales librerías de terceros utilizadas

4.3.2.1. Manejo del estado y flujo de datos

El estado global de la aplicación se gestionó mediante Redux. Asimismo, se utilizó redux-thunk como middleware para manejar operaciones asíncronas, como llamadas a servicios externos o APIs.

4.3.2.2. Enrutamiento

Para la navegación entre los procesos de entrenamiento y consulta, se utilizó React Router DOM, que permite la definición de rutas anidadas, rutas dinámicas y una navegación más estructurada.

4.3.2.3. Comunicación con APIs

Se utilizó Axios como cliente HTTP para la comunicación con la API del backend. Esta herramienta facilita el manejo de solicitudes asíncronas, la gestión de cabeceras personalizadas y el tratamiento de errores, lo cual resulta esencial para una aplicación robusta.

4.3.2.4. Autenticación

La autenticación de usuarios se realizó mediante @auth0/auth0-react, un SDK específico para integrar los servicios de Auth0 en aplicaciones React. Esta herramienta facilitó la implementación del inicio de sesión, cierre de sesión y gestión de tokens de manera segura.

4.3.2.5. Estilos y diseño visual

El diseño visual se implementó mediante styled-components, una librería que permite escribir estilos CSS directamente en los componentes de React, facilitando la encapsulación y modularización del estilo.

4.3.2.6. Máquinas de estado

Se incorporó XState para modelar máquinas de estado finito en componentes complejos, con el objetivo de mejorar la previsibilidad del comportamiento y la organización del flujo lógico de los procesos principales de entrenamiento y consulta.

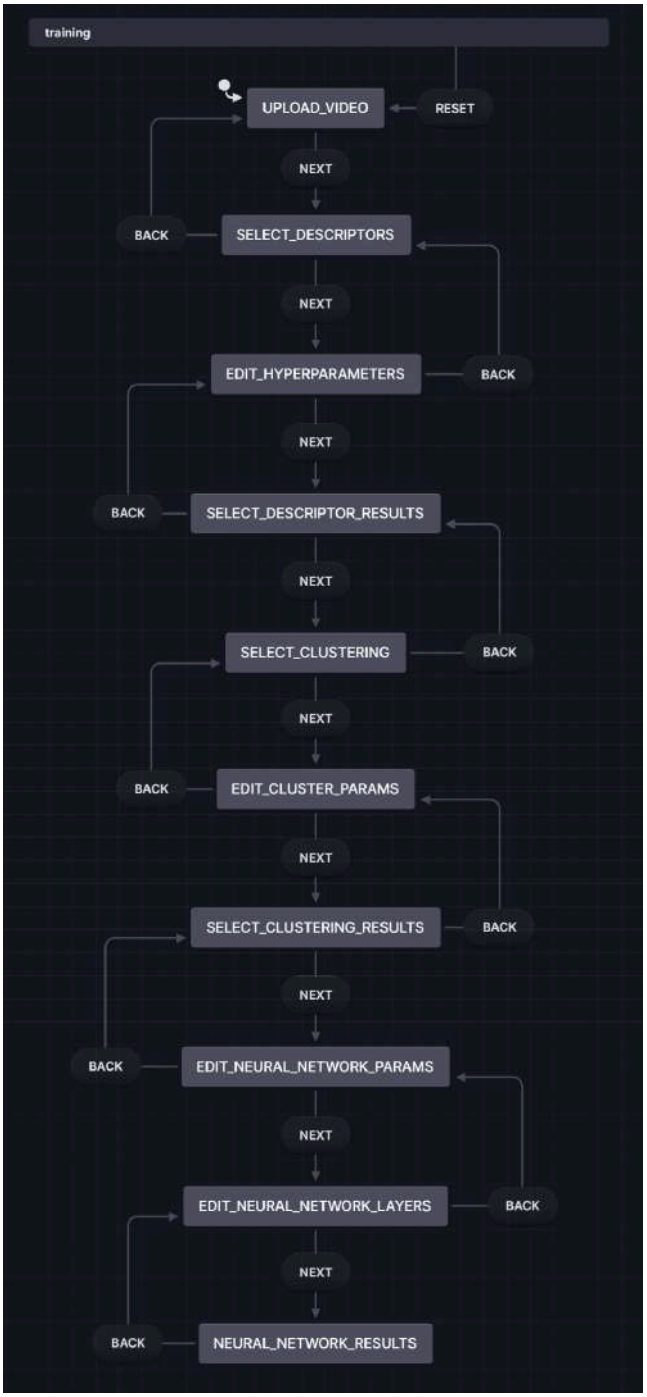

Figura 6. Diagrama de flujo de la máquina de estado del proceso de entrenamiento (*training*).

4.3.3. Arquitectura del Frontend

La interfaz se diseñó conforme al paradigma de arquitectura basada en componentes característico de React, que descompone la UI en unidades autocontenidas, reutilizables y testeables. Este enfoque se fundamenta en tres principios:

1) **Modularidad** – Cada componente implementa una única responsabilidad (*Single-Responsibility Principle*), lo que facilita el paralelismo en el desarrollo y reduce el acoplamiento.
2) **Reutilización** – Los componentes pueden combinarse y anidarse, lo que disminuye la duplicación de código y acelera la construcción de nuevas vistas.
3) **Escalabilidad y mantenimiento** – Las actualizaciones se aplican de forma localizada; la propagación de cambios se circunscribe al árbol de componentes afectado, preservando la estabilidad del resto del sistema.

Para organizar el código de forma coherente se definieron siete módulos lógicos:

- ***Components*:** Elementos visuales de propósito específico.
- ***Shared*:** Componentes atómicos y utilidades transversales, como botones, inputs, notificaciones o alertas.
- ***Hooks*:** Hooks personalizados que encapsulan lógica reutilizable.
- ***Machines*:** Máquinas de estado finito modeladas con XState.
- ***Reducers*:** Gestión del estado global con Redux.
- ***Services*:** Acceso a recursos externos y orquestación de peticiones HTTP mediante Axios.
- ***Utils*:** Funciones auxiliares puras, como por ejemplo funciones de descargas de archivos e imágenes.

Esta distribución evita dependencias circulares y garantiza la autonomía de cada módulo, de modo que un cambio en, por ejemplo, *Services* no impacta directamente en *Components* ni *Machines*.

De esta forma, el frontend está preparado para evolucionar sin comprometer la robustez ni la legibilidad del código fuente.

4.3.4. Diseño de la interfaz

El objetivo principal de la interfaz es facilitar al usuario la comprensión de la lógica y del flujo de procesos que ofrece el sistema. Para ello, se ha optado por un diseño minimalista e intuitivo, centrado exclusivamente en guiar la interacción hacia los procesos principales de entrenamiento y consulta, eliminando cualquier elemento que pueda generar distracción.

Cada etapa incluye instrucciones claras y precisas sobre las acciones necesarias para alcanzar el propósito del sistema, promoviendo así una experiencia de uso eficiente y orientada a resultados.

La interfaz fue diseñada pensando en usuarios con conocimientos especializados en la temática tratada. Aunque se ha procurado utilizar un lenguaje claro y accesible, se incluyen ciertos términos técnicos propios del dominio que resultan necesarios para mantener la precisión conceptual.

Durante su desarrollo, se aplicaron principios fundamentales de accesibilidad. En particular:

- **Perceptibilidad**: El contenido es comprensible mediante alternativas textuales a elementos visuales, ajustes visuales y un diseño que garantiza un adecuado contraste.
- **Comprensibilidad**: La información y la navegación han sido diseñadas para ser intuitivas y predecibles, utilizando un lenguaje claro.
- **Robustez**: La estructura del contenido garantiza la compatibilidad con diversas tecnologías de asistencia, como lectores de pantalla, permitiendo su correcta interpretación por parte de la mayoría de agentes de usuario.

Cabe señalar que el sistema fue concebido inicialmente para su uso en computadoras de escritorio o portátiles, por lo que no incluye actualmente soporte responsive para dispositivos móviles.

En cuanto al diseño visual, se emplea una paleta cromática en escalas de blanco y negro, cuyo propósito es resaltar los resultados de los procesos, los cuales se presentan en colores vivos y variados para captar la atención del usuario y facilitar su interpretación.

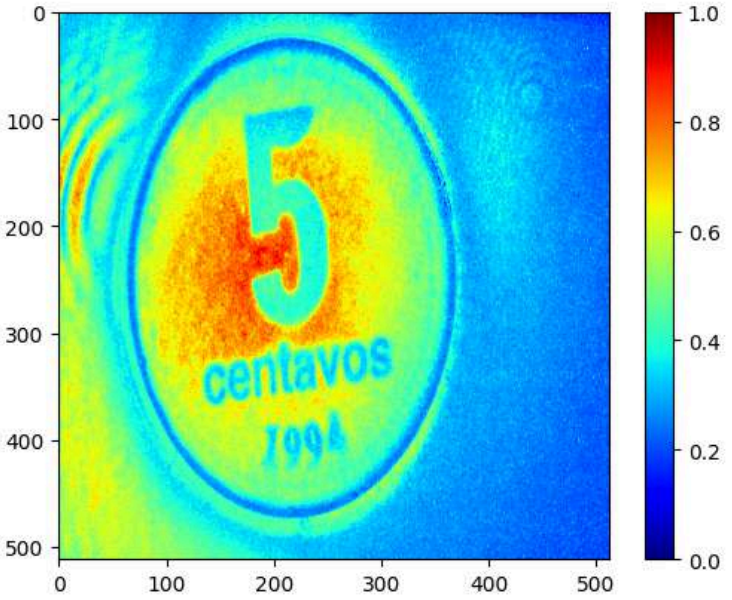

Figura 7. Ejemplo de resultado obtenido mediante el cálculo de descriptores

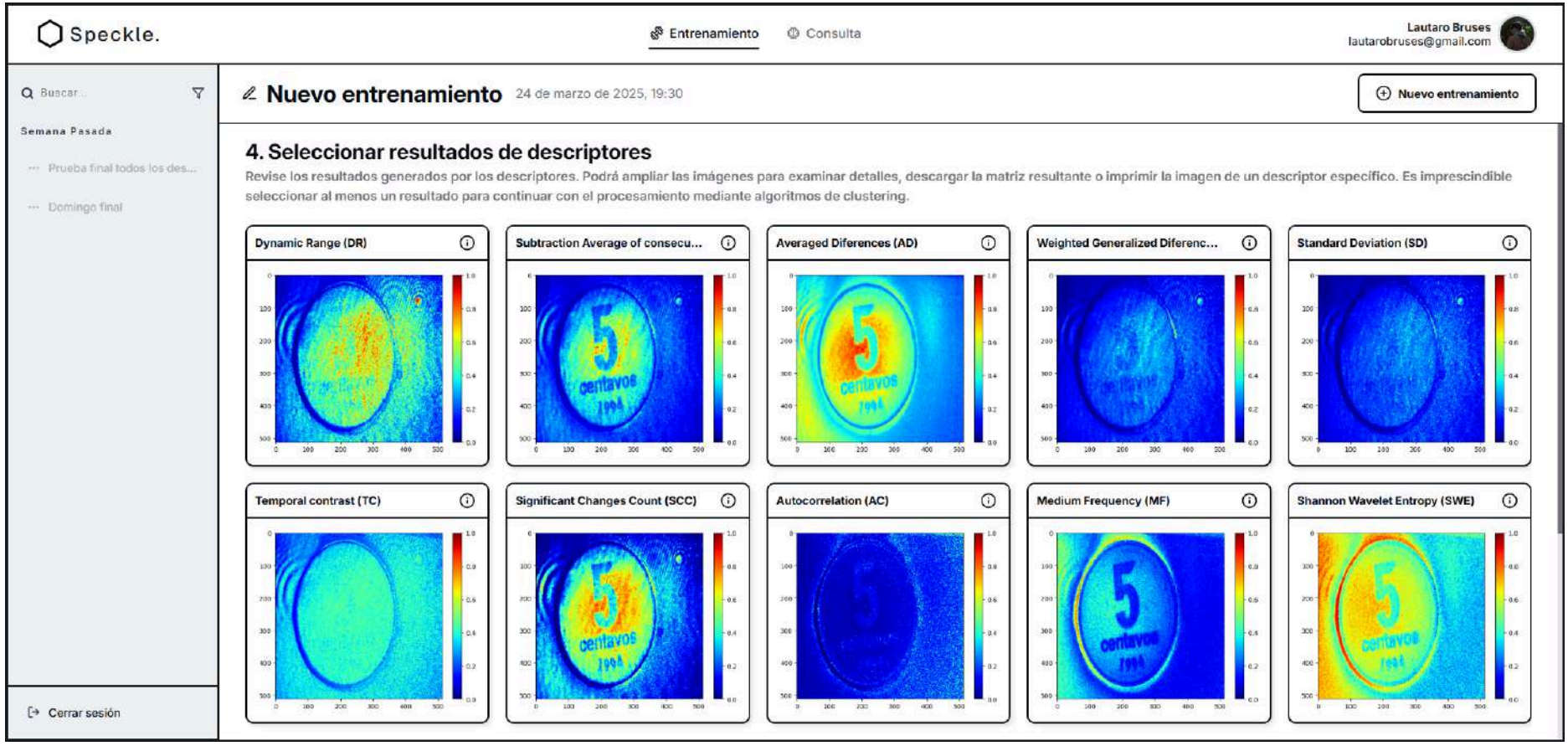

Figura 8. Captura de pantalla de la interfaz en la selección de resultados de descriptores

4.3.5 Prototipos

Los prototipos de la interfaz se diseñaron inicialmente en Figma. Una vez definidos los contenedores principales, su disposición, el contenido y los componentes clave, se procedió a implementarlos en código.

4.3.5.1 Iteraciones

El primer prototipo incluía un historial de experiencias —inicialmente no estaba claro el concepto de modelos de entrenamiento— que permitía al usuario navegar por las experiencias guardadas, una etapa de configuración inicial —en la que podía adjuntar un vídeo y seleccionar los descriptores necesarios para el cálculo— y una sección de resultados donde se mostraban las imágenes generadas.

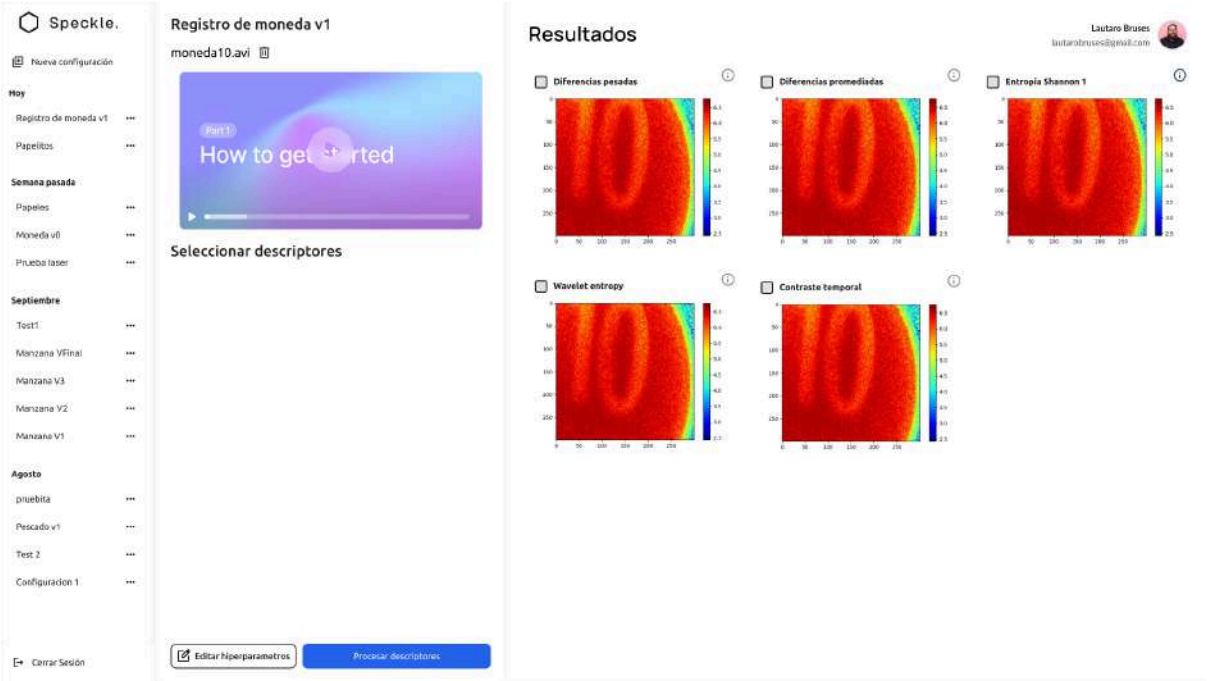

Figura 9. Primer prototipo diseñado en Figma

Tras revisiones y la redefinición de los requerimientos del sistema, se añadió un encabezado con las secciones Experiencias, Favoritas y Estadísticas. Estas secciones se descartaron posteriormente a partir de la retroalimentación recabada durante las pruebas con usuarios. A su vez, se diseñó el que sería finalmente el contenedor principal —resultado de fusionar los contenedores de configuración y de

resultados en una secuencia de pasos—, dedicado exclusivamente al flujo de los procesos de entrenamiento y consulta. Se evidencia que, desde un principio, la idea de un flujo secuencial de pasos estaba clara. Además, en esta etapa se definió la identidad visual del sitio, optando por una interfaz monocromática basada en escala de grises (blanco y negro).

Figura 10. Segundo prototipo y boceto final diseñado en Figma

4.3.6. Manejo de errores

Para gestionar correctamente los errores, se coordinaron situaciones de fallos con el backend. Se contemplaron situaciones posibles de errores que podrían surgir durante los procesos de entrenamiento y consulta. Para ello, se implementaron notificaciones que informan al usuario cuando algo sale mal y evitan que la ejecución normal del frontend se interrumpa.

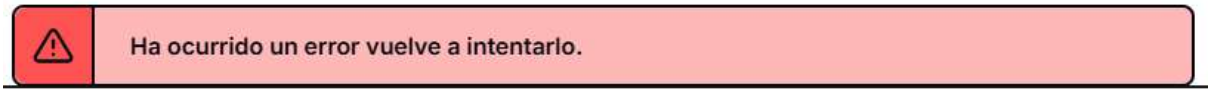

Figura 11. Prototipo final de notificación de error

4.3.6.1. Límites de errores

También se implementó un límite de errores (*Error Boundary*), que permite gestionar los errores y mostrar una interfaz de usuario alternativa con mensaje personalizado, en lugar de que la aplicación se detenga o muestre una pantalla en blanco.

Figura 12. Mensaje principal del límite de errores implementado

4.4. Backend

El backend cumple un rol fundamental como intermediario entre el frontend, la base de datos y la API de cálculo. Encapsula la lógica de negocio del sistema y se encarga de gestionar de forma segura y eficiente el flujo de información: responde a las solicitudes del frontend, persiste datos relevantes en la base de datos y consulta a la API de cálculo.

4.4.1. Tecnologías utilizadas

Se desarrolló utilizando Node.js junto con el framework Express, lo que permitió construir una API REST modular y escalable.

El modelo de ejecución asincrónico y orientado a eventos de Node.js le permitió ser capaz de manejar múltiples solicitudes concurrentes de manera eficiente, aprovechando el paralelismo y optimizando el rendimiento general del sistema.

Se utilizó Multer para la carga de videos, los cuales son enviados al módulo de Python para su procesamiento.

4.4.2. Estructura del backend

Para mantener una estructura clara y escalable del backend, se optó por organizar el código en distintas carpetas cada una con responsabilidades bien definidas:

- La carpeta *routes* contiene todas las rutas de la API, permitiendo una separación lógica y ordenada de los distintos endpoints.
- La lógica de negocio asociada a cada ruta fue encapsulada dentro de la carpeta *controllers*.
- La autenticación del módulo con el backend se implementó a través del uso de *middlewares* que fueron ubicados dentro de la carpeta con el mismo nombre.
- Finalmente, los esquemas de la base de datos fueron definidos en la carpeta *models*, integrando Mongoose para estructurar y manipular los datos almacenados en MongoDB.

4.4.3. Gestión de carpetas temporales

Una de las decisiones de diseño clave en el desarrollo del backend fue la utilización de carpetas temporales para almacenar resultados intermedios del procesamiento.

Esta alternativa resultó más eficiente que persistir datos en la base de datos, ya que:

- Evita el almacenamiento de archivos voluminosos que podrían impactar negativamente en el rendimiento.
- Permite un acceso rápido y directo a los datos generados por el procesamiento.

- Aprovecha la autenticación JWT para asignar un espacio exclusivo a cada usuario, evitando accesos no autorizados.

Esta estrategia no solo se ajusta a las limitaciones de infraestructura, sino que también permite una gestión más eficiente de los recursos disponibles.

4.4.4. Conexión con la base de datos

La conexión con la base de datos se estableció mediante Mongoose, una biblioteca que permite definir modelos basados en esquemas, facilitando una integración eficiente con MongoDB.

Una de las ventajas principales de Mongoose es su capacidad para definir esquemas que representan la estructura esperada de cada documento, especificando tipos de datos, campos requeridos, valores por defecto, y otras restricciones. Estas validaciones a nivel de esquema aseguran que la información almacenada en la base de datos mantenga coherencia y calidad, y reducen la probabilidad de errores provocados por entradas inválidas o malformadas.

4.4.5. Autenticación y seguridad

Se implementaron mecanismos de autenticación y seguridad con el objetivo de proteger los recursos del sistema y garantizar una comunicación controlada y segura.

4.4.5.1. Autenticación con el Frontend (JWT)

Para la comunicación con el frontend, se utiliza la validación a través de JSON Web Tokens (JWT).

- El usuario inicia sesión proporcionando sus credenciales y recibe un token JWT firmado.
- El token se incluye en el encabezado de autorización de cada solicitud y es verificado mediante un middleware.
- Si el token es válido, la solicitud continúa su curso. Si no lo es, el servidor responde con un error de autenticación.

El uso de JWT en el sistema permite gestionar la autenticación de manera segura, eficiente y escalable, asegurando que solo los usuarios autenticados puedan acceder a recursos protegidos, mientras se mantiene un control estricto sobre la integridad de la información del usuario.

4.4.5.3. Autenticación mediante API Key

Para controlar el acceso a la API de cálculo y prevenir el uso no autorizado, cada petición debe enviar una *API Key*. Este mecanismo de autenticación permite validar a los clientes que interactúan con el sistema, asegurando que solo usuarios autorizados puedan acceder a los servicios. Si la clave proporcionada es inválida o está ausente, la solicitud es rechazada con una respuesta de error.

4.4.5.4. Cifrado de la comunicación con HTTPS

Para proteger la integridad y confidencialidad de los datos transmitidos, la API opera exclusivamente sobre el protocolo HTTPS, que emplea TLS para cifrar las comunicaciones, evitando que terceros puedan interceptar la información en tránsito. De esta manera, se previenen ataques de intercepción de datos y se garantiza que la información se mantenga segura.

4.5. Base de datos

El sistema requiere una gestión eficiente de la información debido a la naturaleza de los datos con los que trabaja, incluyendo matrices de gran tamaño y modelos de inteligencia artificial.

4.5.1. Elección de una base de datos NoSQL

La elección de MongoDB por sobre una base de datos relacional se basó en la necesidad de adaptabilidad y eficiencia. Desde el inicio, el proyecto presentaba incertidumbre en sus requerimientos, lo que hacía inviable definir un esquema rígido debido a que la estructura exacta de los datos y los requisitos específicos evolucionarían a lo largo del desarrollo limitando la flexibilidad del sistema y generando una carga adicional en futuras modificaciones. MongoDB permite realizar

cambios en la estructura de datos sin necesidad de migraciones complejas, asegurando flexibilidad durante el desarrollo.

Además, dado que el sistema maneja grandes volúmenes de información, la capacidad de MongoDB para almacenar estructuras complejas en un solo documento evita consultas y combinaciones de datos innecesarias, mejorando el rendimiento. Esta elección permitió gestionar eficientemente los datos sin comprometer la escalabilidad del sistema.

4.5.2. Almacenamiento temporal

Para seguir optimizando el rendimiento del sistema, se tomó la decisión de no almacenar directamente información temporal dentro de la base de datos, sino utilizar un esquema basado en carpetas temporales organizadas por usuario dentro del backend. Esta estrategia permitió manejar eficientemente la información durante el proceso de análisis, sin sobrecargar la base de datos con datos transitorios que no requerían persistencia. Al combinar la flexibilidad de MongoDB con un sistema de almacenamiento temporal, se garantizó un equilibrio entre escalabilidad y eficiencia, asegurando que los recursos del sistema fueran utilizados de manera óptima.

4.5.3. MongoDB Atlas

Para la implementación de la base de datos, se optó por utilizar MongoDB Atlas, la solución en la nube de MongoDB. Esta elección se fundamentó en la necesidad de contar con un servicio administrado que garantizara disponibilidad, escalabilidad y seguridad sin la sobrecarga operativa de gestionar servidores propios.

4.5.4. Facilidad de administración y despliegue

MongoDB Atlas permite gestionar la base de datos de manera centralizada, con herramientas avanzadas para la monitorización, copias de seguridad automatizadas y escalabilidad flexible. Gracias a esta infraestructura, se evitó la necesidad de configurar y mantener servidores manualmente, lo que permitió al equipo centrarse

en el desarrollo del sistema sin preocuparse por la administración de la base de datos.

4.5.5. Integración inmediata con el desarrollo

Desde el inicio del proyecto, MongoDB Atlas permitió trabajar con una base de datos funcional sin necesidad de configuraciones complejas. Esto facilitó la implementación temprana del esquema de datos y permitió iteraciones ágiles en el desarrollo, asegurando que la base de datos se adaptara a las necesidades cambiantes del sistema sin ralentizar el flujo de trabajo.

4.5.6. Limitaciones de la versión gratuita

Una desventaja de MongoDB Atlas es que la versión gratuita ofrece sólo 512 MB de almacenamiento. Si bien esta cantidad es suficiente para el desarrollo inicial y pruebas, no fue adecuada para manejar los grandes volúmenes de datos que el sistema necesita procesar, especialmente al tratar con matrices de gran tamaño y modelos de IC entrenados. Sin embargo, esta limitación se complementa perfectamente con el uso de carpetas temporales dentro del sistema. Las carpetas temporales permiten almacenar archivos transitorios durante el proceso de análisis sin sobrecargar la base de datos, lo que evitó el riesgo de alcanzar el límite de almacenamiento de la versión gratuita de Atlas.

4.5.7. Esquemas definidos

Para la gestión de la base de datos se definieron esquemas que representan entidades clave del proyecto.

1) **Parámetros:** Definen las propiedades asociadas a descriptores, *clustering* y redes neuronales, como nombre, valor y opciones disponibles.
2) **Descriptores:** Cada descriptor agrupa un conjunto de parámetros y se utiliza para configurar el análisis de datos o imágenes.
3) ***Clustering*:** Similar a los descriptores, pero enfocado en técnicas de agrupamiento de datos, también con parámetros configurables.

4) **Redes Neuronales:** Definen los parámetros y capas de las redes neuronales, permitiendo configuraciones flexibles de las redes.
5) **Experiencias de Usuario:** Almacenan las interacciones de los usuarios con el sistema, incluyendo los videos utilizados, descriptores seleccionados y modelos entrenados.

4.6. API de cálculo

4.6.1. Tecnologías utilizadas

Se optó por utilizar Python, lenguaje de programación de código abierto compatible con GLP, en su versión estable 3.12.10 para el desarrollo de la API de Cálculo. Esta elección se tomó principalmente por la experiencia previa en su uso, tanto en la implementación de métodos de *clustering* como en el desarrollo de redes neuronales, conocimientos adquiridos durante el curso de la materia Inteligencia Artificial.

Además, al explorar otras opciones, identificamos que Python es el lenguaje de referencia en el desarrollo de modelos de inteligencia computacional y *machine learning*, debido a su simplicidad y facilidad de uso, características que se deben a su sintaxis clara y concisa. Asimismo, su capacidad de integración con tecnologías avanzadas, el respaldo de una comunidad global activa y su amplio ecosistema de bibliotecas especializadas para el tratamiento de datos lo convierten en una opción ideal para este tipo de desarrollos.

4.6.2. Principales librerías de terceros utilizadas

- **NumPy**: proporciona un paquete fundamental para la computación científica en Python. Esta librería ofrece un objeto de matriz multidimensional, varios objetos derivados, y un conjunto de rutinas que permiten realizar operaciones rápidas con matrices, tales como operaciones matemáticas, lógicas, manipulación de formas, ordenación, selección, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas, entre otras.

- **Matplotlib**: permiten la visualización de datos, esencial para el análisis exploratorio. Biblioteca para crear visualizaciones estáticas, animadas e interactivas.
- **Scikit-Learn**: proporciona herramientas sencillas, eficaces, accesibles y reutilizables en diversos contextos, para el análisis predictivo de datos, tanto de *clustering*, regresión, clasificación, reducción de dimensionalidad, preprocesamiento y selección de modelo. Métodos basados en NumPy, SciPy y matplotlib. Biblioteca de código abierto, utilizable comercialmente bajo licencia BSD.
- **TensorFlow y Keras**: ofrecen herramientas avanzadas para el desarrollo de redes neuronales y modelos de aprendizaje profundo. Biblioteca de código abierto para aprendizaje automático a través de un rango de tareas, y desarrollado por Google para satisfacer necesidades de sistemas capaces de construir y entrenar redes neuronales, para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos.
- **SciPy**: proporciona herramientas adicionales para el procesamiento numérico avanzado. Biblioteca libre y de código abierto para la realización de operaciones matemáticas de ciencias e ingeniería. SciPy contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, FFT, procesamiento de señales y de imagen, resolución de ODEs entre otros.
- **OpenCV**: Esencial en aplicaciones de visión por computadora. Biblioteca multiplataforma de código abierto, publicada bajo licencia BSD para la detección de movimiento, reconocimiento de objetos, reconstrucción 3D a partir de imágenes, entre otros.
- **PyWt**: Biblioteca gratuita de código abierto para transformadas wavelet en Python. Combina una sencilla interfaz de alto nivel con prestaciones de bajo nivel en C y Cython.

Asimismo, se utilizó el framework FastAPI debido a sus características técnicas que lo hacen adecuado para el desarrollo de esta API. Entre sus principales ventajas se encuentra las siguientes:

- Se basa en estándares abiertos, lo que facilita la interoperabilidad con otras tecnologías y sistemas.
- El uso de OpenAPI para la creación y documentación automática permite la definición de operaciones de ruta, parámetros, cuerpos de solicitud, seguridad, entre otros.
- La integración con JSON Schema para la documentación automática del modelo de datos, lo cual es compatible con la especificación OpenAPI.
- La capacidad de generar código cliente automáticamente en varios lenguajes, lo que simplifica la interacción desde diferentes entornos.
- La inclusión de funcionalidades esenciales para la construcción, como la validación de datos y la serialización, sin necesidad de implementación adicional por parte del desarrollador. Además, la documentación generada es accesible de manera automática sin causar sobrecarga en la aplicación durante su ejecución, ya que se genera al inicio.

FastAPI permitió reducir el tiempo de desarrollo y minimizar los errores, al tiempo que optimizó la cantidad de líneas de código requeridas. Gracias a estas características, se pudo obtener un rendimiento comparable al de implementaciones manuales, garantizando una solución robusta y eficiente para el desarrollo de la API.

4.6.3. Procesamiento de video de entrada

A partir de un video de entrada en formato AVI, se obtiene un tensor Numpy de números enteros en formato uint8, es decir valores en el rango de [0:255]. Este arreglo se logra utilizando el módulo *cv2* (uno de los más importantes de OpenCV). El tensor resultante es fundamental para los cálculos posteriores que se realizan en el proceso de análisis.

4.6.4. Cálculo de descriptores

Los descriptores en el análisis de videos de speckle láser dinámico son funciones matemáticas o métricas que transforman la información contenida en un video en representaciones compactas y relevantes (matrices). Cada descriptor extrae características específicas de la variabilidad temporal de cada pixel del video de

patrón speckle. La matriz resultante de cada cálculo genera una imagen que resume cierto aspecto del comportamiento dinámico de la muestra analizada.

El uso de descriptores optimiza la extracción de información clave, facilita la automatización del análisis y permite una mejor clasificación de patrones mediante técnicas de machine learning. Esto los vuelve una alternativa robusta y eficiente en términos de sensibilidad y capacidad de segmentación de actividad.

4.6.4.1. Desarrollo

A partir de un conjunto de rutinas previamente desarrolladas en Matlab, se implementaron las funciones necesarias para el cálculo de las matrices de descriptores, así como para su visualización en mapas de calor. Para la programación de estas funciones, se utilizó la capacidad vectorial de la librería NumPy, lo cual permitió simplificar y optimizar las funciones, evitar ciclos explícitos y hacer el código más compacto y fácil de leer.

Para el cálculo de las matrices de descriptores se utilizaron los siguientes métodos específicos:

- **scipy.*signal.welch*** – para la estimación de la densidad espectral de potencia, la cual divide los datos en segmentos superpuestos, calculando un periodograma modificado para cada segmento y promediando los periodogramas mediante el método de Welch.
- ***scipy.signal.ellip*** – para el diseño de filtros elípticos de orden N (Cauer) digitales y analógicos (devuelve los coeficientes del filtro)
- ***scipy.signal.sosfilt*** – para filtrar una secuencia de datos a lo largo de una dimensión utilizando un filtro IIR digital definido por secciones de segundo orden en cascada.
- ***scipy.signal.ellipord*** – para seleccionar y devolver el orden del filtro elíptico (Cauer) digital o analogico de orden más bajo que no pierda más de gpass dB en la banda de paso y tenga al menos gstop dB de atenuación en la banda de rechazo.

- ***pywt.wavedec*** – para realizar transformada wavelet discreta multinivel y obtener lista ordenada de matrices de coeficientes en la que *n* indica el nivel de descomposición.

Para la verificación del correcto cálculo, se dispuso de matrices calculadas mediante otro software desarrollado en Matlab®. De este modo, a medida que se implementaban las distintas funciones, se verificaba que los resultados obtenidos coincidieran con los esperados.

No obstante, en el caso del cálculo de matrices de descriptores que emplean la función de Welch, no se logró reproducir exactamente el comportamiento obtenido en Matlab. A pesar de realizar múltiples ajustes en los parámetros de esta función y de revisar en detalle la documentación de ambas bibliotecas (*Matlab* y *SciPy*), los resultados no fueron completamente idénticos.

Sin embargo, al analizar las matrices obtenidas mediante mapas de calor, se verificó que las funciones implementadas representaban correctamente la información esperada. En consecuencia, tras una evaluación en conjunto con el director del proyecto, se decidió considerar estos valores como válidos y continuar con el desarrollo.

4.6.4.2. Descriptores y definición

- **Dynamic Range (DR)**: mide el rango de intensidad (diferencia entre el valor máximo y mínimo) a lo largo del tiempo en cada píxel.
- **Weighted Generalized Differences (WGD)**: mide el cambio abrupto en el tiempo, dando más peso al primer cuadro respecto a los siguientes.
- **Subtraction Average of consecutive pixel intensities (SA)**: mide el cambio promedio de intensidad entre cuadros consecutivos.
- **Averaged Differences (AD)**: mide la relación de cambios entre cuadros consecutivos, resaltando pequeñas variaciones relativas en el tiempo.
- **Standard Deviation (SD)**: mide la variabilidad de la intensidad en el tiempo en cada píxel.
- **Temporal Contrast (TC)**: mide el contraste dinámico, comparando la variación (desvío estándar) respecto al promedio temporal.

- **Autocorrelation (AC)**: mide cuánto tiempo tarda la señal de cada píxel en perder la similitud consigo misma.
- **Fuzzy Granularity (FG)**: mide cuántas veces cambia el estado de intensidad de un píxel entre rangos de valores a lo largo del tiempo.
- **Medium Frequency (MF)**: estima la frecuencia promedio de las fluctuaciones de intensidad a lo largo del tiempo.
- **Shannon Wavelet Entropy (SWE)**: mide la complejidad o desorden de la distribución de frecuencias de cada píxel.
- **Cut off Frequency (CF)**: mide la frecuencia en la que la energía espectral cae a la mitad.
- **Wavelet Entropy (WE)**: mide la complejidad temporal usando descomposición en distintas escalas (ondas de diferentes frecuencias).
- **High to Low Ratio (HLR)**: mide la proporción de energía entre frecuencias altas y bajas en cada píxel.
- **Low Frequency Energy Band (LFEB)**: mide la energía filtrada de las señales en un rango de bajas frecuencias.
- **Medium Frequency Energy Band (MFEB)**: mide la energía filtrada de las señales en un rango de frecuencias medias.
- **High Frequency Energy Band (HFEB)**: mide la energía filtrada de las señales en un rango de alta frecuencia.
- **Significant Changes Count (SCC)**: mide la frecuencia de cambios abruptos en la intensidad.

4.6.4.3. Normalización y tipo de datos

Como se comentó anteriormente, el video de entrada es transformado en un tensor NumPy con valores enteros en el rango de 0 a 255. Sin embargo, muchas de las funciones utilizadas para calcular las matrices de los descriptores requieren datos en formato float64 (doble precisión) para obtener resultados precisos. Por esta razón, los cálculos internos se realizan utilizando este tipo de dato.

Sin embargo, antes de generar una respuesta, las matrices se convierten a float32 (precisión simple). Esta decisión se tomó debido a que la precisión de float32 es suficiente para los fines del proyecto, por otro lado, de esta forma se reduce en

promedio un 30% el tamaño de los archivos de respuesta, optimizando el almacenamiento y la transmisión de datos.

4.6.4.4. Normalización de datos

Para garantizar la coherencia en el procesamiento de los datos, las matrices se normalizan en el rango [0,1]. Esta normalización se utiliza por dos razones principales:

1) **Unificación del formato de datos**: Permite que todas las matrices de descriptores se mantengan dentro de un mismo rango.
2) **Optimización del entrenamiento de la red neuronal**: Dado que los valores normalizados son utilizados como entrada en la red neuronal, esta transformación contribuye a mejorar la eficiencia de la función de activación ReLU, favoreciendo un entrenamiento más estable y eficiente.

4.6.4.5. Disponibilidad de datos crudos

En una etapa avanzada del proyecto, se incorporó un requerimiento adicional por parte de los directores, el cual consistía en proporcionar acceso tanto a las matrices normalizadas como a las matrices crudas (sin normalización). En respuesta a esta solicitud, se hicieron los cambios necesarios para que el sistema permita descargar ambas versiones de las matrices una vez que han sido calculadas, ofreciendo mayor flexibilidad en el análisis de los datos.

4.6.5. Cálculo de métodos de agrupamiento o *clustering*

El *clustering* es la clasificación no supervisada de patrones (observaciones, elementos de datos o vectores de características) en grupos (*clusters*). El problema de la agrupación ha sido abordado en muchos contextos y por investigadores de diversas disciplinas, lo que refleja su gran atractivo y utilidad como uno de los pasos clave del análisis exploratorio de datos.

En este caso, los algoritmos de agrupamiento segmentan zonas tomando como entrada matrices de descriptores. Para la selección de los métodos, los directores

del proyecto solicitaron la exploración de diversas técnicas, con la condición de que el *clustering* sustractivo debía ser incluido en la evaluación.

4.6.5.1. Clustering Sustractivo

El *clustering* sustractivo asume que cada punto de datos es un posible centro de conglomerado. El algoritmo calcula la probabilidad de que cada punto de datos se convierta en un centro de *cluster* en función de la densidad de los puntos circundantes. Luego, selecciona el punto con mayor potencial como primer centro y elimina aquellos dentro de un radio determinado. Posteriormente, el proceso se repite con el siguiente punto de mayor potencial hasta que todos los datos estén dentro del rango de influencia de un centro.

Para su implementación, se evaluaron diversas soluciones, pero muchas de ellas no cumplían con los requisitos del proyecto, ya que al procesar grandes volúmenes de datos presentaban problemas de rendimiento o fallaban. Se incorporó una implementación bien documentada que permitió realizar el *clustering* de manera eficaz. Sin embargo, se requirió optimizar el código mediante paralelización para mejorar su eficiencia. Para ello, se utilizó la biblioteca *Numba*, un compilador *Just-In-Time* (JIT) que traduce un subconjunto de Python y NumPy en código de máquina utilizando LLVM (*Low-Level Virtual Machine*).

4.6.5.2. Métodos Basados en Scikit-Learn

Además del *clustering* sustractivo, se exploraron otros métodos de agrupamiento utilizando la biblioteca *scikit-learn*. Se evaluaron algoritmos que, a priori, presentaban una escalabilidad adecuada para este proyecto, seleccionando aquellos que lograban realizar los cálculos de manera eficiente.

Cabe destacar que varios de los métodos disponibles no pudieron utilizarse debido al elevado volumen de datos a procesar, ya que el poder computacional requerido supera ampliamente los recursos disponibles. Un caso particular fue el *clustering* espectral, propuesto por el referente funcional como una alternativa acorde al trabajo requerido. Sin embargo, su implementación requiere la construcción de una

matriz de afinidad que supera, en este caso, los 60 GiB de memoria, lo que imposibilitó su uso en este contexto y obligó a descartarla.

Finalmente, los métodos seleccionados incluyen *K-Means* y dos variantes de este (*Bisecting K-Means* y *Mini Batch K-Means*), además de un modelo de mezcla gaussiana. De esta manera, el sistema desarrollado cuenta con un total de cinco métodos de *clustering*.

4.6.5.3. K-Means y sus Variantes

El método *K-Means* agrupa los datos dividiéndolos en *n* conglomerados de igual varianza, minimizando un criterio conocido como suma de inercia o suma de cuadrados dentro de un *cluster*. Este algoritmo requiere que se especifique previamente la cantidad de grupos a formar. Su elección se debió a su capacidad para manejar grandes volúmenes de datos y a su amplio uso en diversas áreas de aplicación.

El algoritmo asigna cada muestra a un conglomerado disjunto, definido por la media de los puntos que lo componen. Estas medidas, denominadas *centroides*, representan el centro de cada *cluster*, aunque en general no corresponden a puntos específicos del conjunto de datos, sino que constituyen una referencia dentro del espacio de características. El objetivo de *K-Means* es seleccionar los *centroides* de manera que se minimice la inercia, es decir, el grado de dispersión interna de los conglomerados, optimizando así la cohesión dentro de cada grupo.

4.6.5.4. Bisecting K-Means

Bisecting K-Means es una variante iterativa de *K-Means* que utiliza un enfoque de agrupamiento jerárquico divisivo. En lugar de calcular todos los *centroides* simultáneamente, los conglomerados se generan progresivamente a partir de una segmentación inicial. El algoritmo selecciona un *cluster* y lo divide en dos nuevos, repitiendo este proceso hasta alcanzar el número objetivo de conglomerados.

Este método resulta más eficiente en términos computacionales que *K-Means* cuando se requiere un gran número de conglomerados, ya que opera sobre

subconjuntos de datos en cada bisección. Esto permite obtener agrupaciones más estructuradas y facilita la interpretación jerárquica de los resultados.

4.6.5.5. Mini Batch K-Means

Mini Batch K-Means es otra variante de *K-Means* que emplea *mini-lotes* (*mini-batches*) para reducir el tiempo de cálculo, manteniendo el mismo criterio de optimización. Los *mini-batches* son subconjuntos aleatorios del conjunto de datos de entrada, extraídos en cada iteración del proceso de entrenamiento. Esta técnica disminuye significativamente el tiempo necesario para alcanzar la convergencia hacia una solución local.

El algoritmo opera en dos pasos principales. Primero, se selecciona aleatoriamente un *mini-lote* de datos y se asigna cada muestra al *centroide* más cercano. Luego, en el segundo paso, se actualizan los *centroides* de manera incremental. A diferencia de *K-Means*, la actualización no se realiza sobre la totalidad de los datos en cada iteración, sino de forma progresiva: cada *centroide* se ajusta tomando el promedio entre la muestra actual y todas las asignadas previamente. Este mecanismo reduce la variabilidad de los *centroides* a lo largo del tiempo, estabilizando la solución final.

Si bien *Mini Batch K-Means* converge más rápidamente que *K-Means*, la precisión de los resultados puede ser ligeramente inferior debido a la estimación basada en subconjuntos de datos.

4.6.5.6. Modelo de Mezcla Gaussiana

El modelo de mezcla gaussiana es un enfoque probabilístico que asume que todos los puntos de datos provienen de una combinación de distribuciones gaussianas con parámetros desconocidos. Este método puede verse como una generalización de *K-Means* que no solo considera la posición de los *centroides*, sino también la estructura de covarianza de los datos, permitiendo una mayor flexibilidad en la definición de los conglomerados.

Este modelo utiliza un algoritmo de maximización de expectativas (*Expectation-Maximization*, EM) para ajustar las distribuciones gaussianas y

determinar el número óptimo de *clusters* mediante el cálculo del Criterio de Información Bayesiano (*Bayesian Information Criterion*, BIC). Finalmente, cada muestra es asignada a la distribución gaussiana a la que tiene mayor probabilidad de pertenecer.

4.6.6. Red neuronal

Para la sección de inteligencia computacional, se utilizó una red *feedforward* profunda, también conocida como red neuronal *feedforward* o perceptrón multicapa (*Multilayer Perceptron*, MLP), debido a su relevancia como modelo fundamental en el aprendizaje profundo (*deep learning*).

4.6.6.1. Definición y funcionamiento

Las redes *feedforward* tienen como objetivo aproximar una función desconocida. Se denominan de esta manera porque la información fluye en una única dirección: desde la entrada x, a través de una serie de transformaciones intermedias definidas por la estructura de la red, hasta producir la salida y. A diferencia de otros modelos, estas redes no presentan conexiones de retroalimentación en las que las salidas influyen sobre las entradas en iteraciones posteriores.

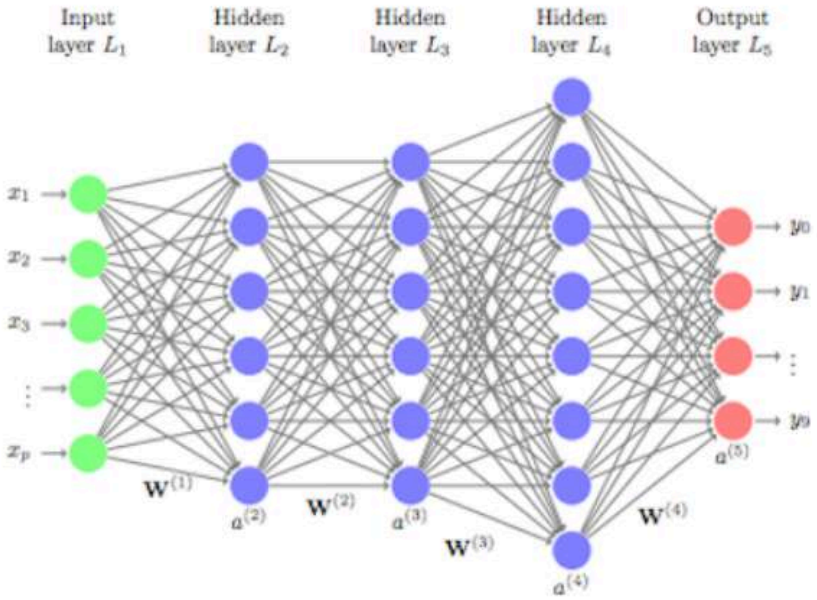

Figura 13. Red neuronal feed-forward con tres capas ocultas.

Las redes neuronales *feedforward* son ampliamente utilizadas en el aprendizaje automático debido a su capacidad para modelar relaciones complejas entre los datos. Su estructura puede representarse mediante un grafo acíclico dirigido, en el que cada nodo corresponde a una operación matemática y las conexiones indican el flujo de información. La profundidad de la red está determinada por la cantidad de capas intermedias o *capas ocultas* presentes en la arquitectura. El término *aprendizaje profundo* proviene precisamente de la posibilidad de diseñar modelos con múltiples capas, lo que permite representar funciones de alta complejidad.

4.6.6.2. Entrenamiento y capas ocultas

El proceso de entrenamiento de una red neuronal *feedforward* consiste en ajustar los parámetros de la función $f(x)$ para que se aproxime a la función objetivo $f^*(x)$. Para ello, se dispone de un conjunto de datos de entrenamiento, en el que cada entrada x está asociada a una etiqueta $y \approx f^*(x)$.

La capa de salida de la red es la única cuyo comportamiento está explícitamente determinado por los datos de entrenamiento, ya que la función objetivo dicta los valores esperados. En contraste, las capas ocultas no tienen salidas directamente supervisadas por los datos, sino que su función es determinada por el algoritmo de aprendizaje para optimizar la representación de los patrones en los datos. Estas capas permiten transformar la información de manera progresiva, extrayendo características de mayor nivel a medida que la señal avanza a través de la red.

4.6.6.3. Relación con la neurociencia

El término *red neuronal* proviene de una inspiración conceptual en la neurociencia. Cada capa oculta de la red se representa como un vector de valores, cuya dimensionalidad determina la *anchura* del modelo. Cada elemento de este vector puede interpretarse como una unidad de procesamiento análoga a una neurona biológica, ya que recibe información de múltiples unidades anteriores y calcula su propia activación.

Si bien la estructura de las redes neuronales artificiales se basa en principios inspirados en el cerebro humano, su desarrollo se encuentra más influenciado por

técnicas matemáticas y de optimización que por modelos neurocientíficos estrictos. En este sentido, las redes neuronales deben considerarse principalmente como modelos de aproximación de funciones diseñados para lograr generalización estadística, más que como representaciones detalladas del funcionamiento cerebral.

4.6.6.4. La arquitectura de la red neuronal

La arquitectura de la red neuronal es configurable en tiempo de ejecución, permitiendo al usuario definir su estructura según los requerimientos específicos del problema. Se puede seleccionar entre una y ocho capas ocultas, con un máximo de 128 neuronas por capa.

4.6.6.5. Capas densas (*Dense Layers*)

Cada capa oculta de la red corresponde a una de conexión densa (*dense layer*), la cual implementa la siguiente operación matemática:

$$\text{output} = \text{activation}(\text{dot}(\text{input}, \text{kernel}) + \text{bias})$$

donde:

- ***activation*** es la función de activación aplicada elemento a elemento.
- ***kernel*** es la matriz de pesos entrenables de la capa.
- ***bias*** es un vector de sesgo también ajustado durante el entrenamiento.

La capa realiza el producto punto entre las entradas y el *kernel* a lo largo del último eje de las entradas y el eje 0 del *kernel*, lo que permite la transformación de la información en cada capa de la red.

4.6.6.6. Función de activación ReLu

La función de activación seleccionada para las capas ocultas de la red neuronal es la *Rectified Linear Unit (ReLU)*, definida como $f(x)=max(0,x)$. Esta función es ampliamente utilizada en redes neuronales profundas debido a su simplicidad computacional y su capacidad para mitigar el problema del desvanecimiento del gradiente (*vanishing gradient problem*), que afecta a funciones de activación

sigmoideas o tangenciales en redes profundas. A diferencia de funciones como *sigmoid* o *tanh*, ReLU no sufre de saturación en valores positivos, lo que permite un entrenamiento más eficiente al proporcionar gradientes más estables y reducir la complejidad computacional. Además, introduce cierta dispersión en la activación de las neuronas, ya que las unidades con entradas negativas se desactivan, promoviendo una mayor eficiencia en el cálculo y mejorando la capacidad de generalización del modelo. Debido a estas propiedades, ReLU es una opción adecuada para mejorar el rendimiento y la velocidad de convergencia de la red neuronal implementada en este proyecto.

4.6.6.7. Normalización por Lotes (*Batch Normalization*)

El usuario puede configurar la aplicación de normalización por lotes (*batch normalization*), una técnica que estandariza las activaciones de la capa para mejorar la estabilidad y aceleración del entrenamiento. Esta transformación mantiene la media de salida cercana a 0 y la desviación estándar cercana a 1.

Durante el entrenamiento, la normalización se aplica utilizando la media y la desviación estándar del lote actual de entradas mediante la siguiente ecuación:

$$\text{gamma} * (\text{lote} - \text{media(lote)}) / \text{sqrt}(\text{var(lote)} + \text{epsilon}) + \text{beta}$$

donde:

- ***epsilon*** es un valor pequeño para evitar divisiones por cero.
- ***gamma*** es un factor de escala aprendido (inicializado en 1).
- ***beta*** es un factor de desplazamiento aprendido (inicializado en 0).

La normalización por lotes ayuda a reducir la dependencia del modelo en la inicialización de los pesos y contribuye a la regularización, mitigando posibles problemas de sobreajuste.

4.6.6.8. Regulación mediante *Dropout*

Otra opción configurable en la arquitectura de la red es la aplicación de *dropout*, un método de regularización que reduce el sobreajuste desactivando aleatoriamente ciertas neuronas durante el entrenamiento.

Esta técnica consiste en establecer aleatoriamente ciertas unidades de entrada en 0 con una frecuencia definida por el usuario. Las unidades restantes se escalan por un factor de 1/(1−tasa),asegurando que la suma total de las activaciones se mantenga constante.

La inclusión de *dropout* introduce una forma de ensamble dentro de la red neuronal, ya que diferentes subconjuntos de neuronas aprenden representaciones distintas, mejorando la capacidad de generalización del modelo.

4.6.6.9. Capa de entrada

La primera capa de la red está compuesta por una capa de entrada (*Input*) con una dimensión igual al número de características de los datos. En este caso ese número coincide con la cantidad de descriptores seleccionados para realizar el entrenamiento. Esta capa tiene como finalidad la recepción de los datos, definiendo explícitamente la dimensión de entrada del modelo y asegurando que la red pueda procesar correctamente las matrices generadas por los descriptores.

4.6.6.10. Última capa: Salida y asignación de probabilidades

La última capa de la red es una capa densa con un número de neuronas igual a la cantidad de clusters determinados por el método de agrupamiento seleccionado. Esta capa utiliza la función de activación *Softmax*, un modelo de regresión que convierte las representaciones aprendidas en una distribución de probabilidad sobre las posibles clases. De este modo, la salida del modelo indica la probabilidad de pertenencia de cada entrada a un cluster específico.

Dada una instancia x, el modelo de regresión Softmax calcula primero una puntuación sk(x) para cada clase k y, a continuación, estima la probabilidad de cada

clase aplicando la función Softmax, también conocida como exponencial normalizada. La ecuación se define de la siguiente manera:

$$\exp(x) = \exp(x - \max(x))$$

$$f(x) = \exp(x) / \Sigma\exp(x)$$

Esta función garantiza que la suma de las probabilidades asignadas a cada *clúster* sea igual a 1. El cálculo de la exponencial de cada puntuación y su posterior normalización facilitan la interpretación de los resultados, permitiendo que la red neuronal aprenda de manera efectiva a diferenciar entre los distintos grupos.

4.6.6.11. Compilación

Para la compilación del modelo se utilizó el optimizador Adam, un método estocástico de descenso de gradiente basado en la estimación adaptativa de momentos de primer y segundo orden. Este algoritmo es computacionalmente eficiente, ya que requiere poca memoria, es invariante al reescalado diagonal de los gradientes y se adapta bien a problemas de gran escala, tanto en términos de datos como de parámetros.

Por otro lado, como función de costo se empleó *Sparse Categorical Cross-Entropy*, una variante de la entropía cruzada. Esta función tiene su origen en la teoría de la información de Claude Shannon y mide la cantidad promedio de información transmitida por operación. Se eligió esta función debido a que el problema involucra múltiples clases representadas por etiquetas enteras, lo que hace que *Sparse Categorical Cross-Entropy* sea una opción adecuada para manejar este tipo de datos de manera eficiente.

4.6.6.12. Entrenamiento

Para el entrenamiento, las matrices de los descriptores y la matriz de clustering seleccionada y calculadas en los pasos anteriores, son transformadas mediante la operación de *flattening*, es decir, reestructuradas desde su forma bidimensional a una unidimensional. El resultado es una matriz de características, en la que cada fila

representa una matriz original, la cual será utilizada como conjunto de datos para entrenar el modelo.

4.6.6.13. Separación de los datos

Entrenar y evaluar un modelo sobre los mismos datos puede llevar a un error metodológico conocido como *sobreajuste* (*overfitting*). En este caso, el modelo memorizaría las etiquetas de las muestras vistas durante el entrenamiento y obtendría una puntuación perfecta en esos datos, pero no lograría generalizar ni predecir correctamente sobre datos nuevos.

Para evitar el sobreajuste, una práctica común en aprendizaje automático es reservar una parte de los datos disponibles como conjunto de prueba. *Scikit-learn* permite realizar una división aleatoria rápida en conjuntos de entrenamiento y prueba, facilitando la evaluación del modelo. De esta forma, el modelo se entrena con el conjunto de entrenamiento, se evalúa con un conjunto de validación y, una vez que el experimento parece exitoso, se realiza la evaluación final con el conjunto de prueba.

4.6.6.14. Parámetros configurables

El usuario puede ajustar la cantidad de *épocas* de entrenamiento, es decir, el número de ciclos de corrección de propagación necesarios para reducir la pérdida. Asimismo, puede definir el *batch size*, que corresponde al número de muestras introducidas en cada iteración del proceso de entrenamiento.

Por último, el usuario tiene la opción de activar el mecanismo de *Early Stopping*. Esta técnica permite finalizar el entrenamiento de manera anticipada en caso de que la métrica de pérdida deje de mejorar, evaluando su evolución al final de cada época. De ser así, restablece los valores de los parámetros correspondientes a la época en la que se obtuvo la mínima pérdida.

4.6.6.15. Matriz de confusión

Finalmente, se evalúa el modelo utilizando los datos del conjunto de prueba y se genera una *matriz de confusión*. En esta matriz, cada columna representa el número

de predicciones realizadas para cada clase, mientras que cada fila representa las instancias de la clase real. Los elementos diagonales indican la cantidad de muestras correctamente clasificadas, mientras que los elementos fuera de la diagonal corresponden a errores de clasificación. Esta matriz proporciona una visión detallada del rendimiento del modelo y permite identificar los errores cometidos en la asignación incorrecta de cada clase durante la clasificación.

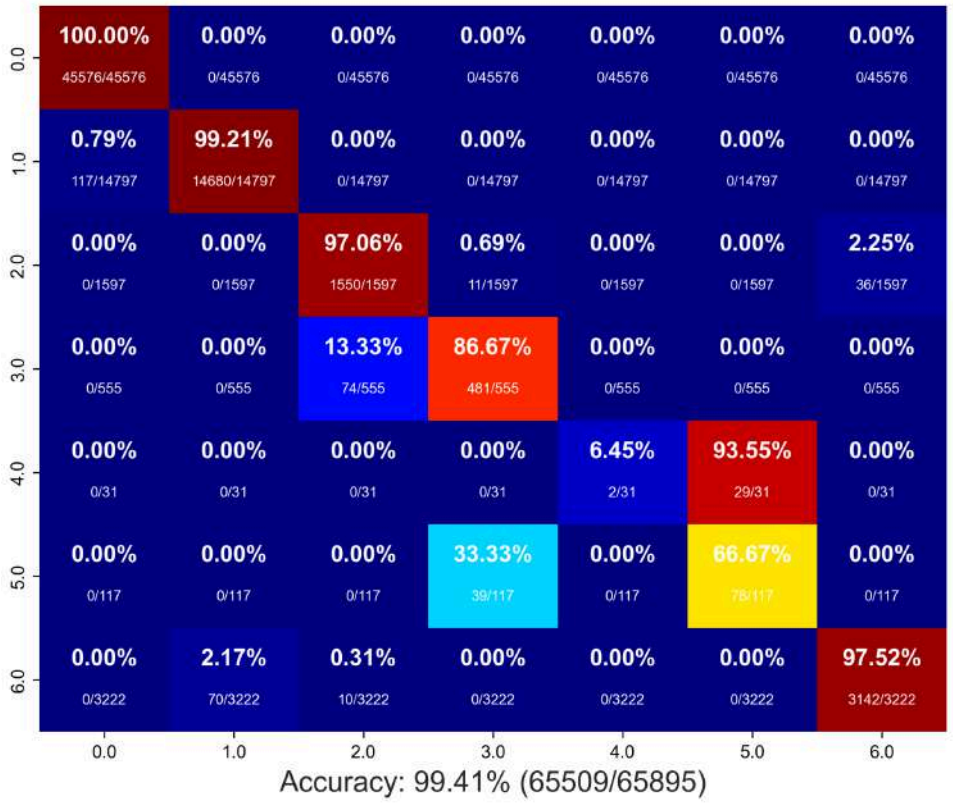

Figura 14. Matriz de confusión del modelo evaluado, mostrando una precisión general del 99.41%.

4.6.7. Comunicación con el backend

La comunicación con el backend se realiza a partir de métodos POST, para ello la API cuenta con cuatro endpoints, los cuales realizan el cálculo de descriptores, el cálculo de *clustering*, el entrenamiento de la red neuronal y la consulta a un modelo previamente entrenado.

4.6.8. Respuesta a peticiones

Las respuestas a las peticiones realizadas desde el backend en Node.js se generan, en su mayoría, en formato JSON, un estándar ligero de intercambio de datos. JSON está basado en un subconjunto del lenguaje de programación JavaScript y fue seleccionado debido a sus ventajas en términos de legibilidad y facilidad de uso tanto para humanos como para máquinas.

Este formato es ampliamente utilizado porque permite estructurar datos de manera sencilla y comprensible, es independiente del lenguaje de programación, lo que facilita la interoperabilidad entre sistemas y además porque su sintaxis se basa en convenciones bien conocidas, lo que lo convierte en un estándar ideal para el intercambio de información.

JSON está compuesto por dos estructuras principales:

1) **Colección de pares clave/valor:** Se representa mediante llaves ({}), donde cada elemento está compuesto por una clave y un valor, separados por dos puntos (:). Los distintos pares están separados por comas.
2) **Lista ordenada de valores:** Se representa mediante corchetes ([]), con los elementos separados por comas. Esta estructura mantiene el orden de los elementos.

Dentro de la API, este formato es utilizado como respuesta en tres de los cuatro endpoints, incluyendo matrices en formato de lista e imágenes codificadas en Base64.

El único endpoint que no devuelve una respuesta en formato JSON es aquel que envía el modelo entrenado de la red neuronal. Esto se debe a que un modelo de este tipo no puede ser transmitido directamente dentro de un archivo JSON.

Por esta razón, se optó por generar una respuesta en un archivo ZIP (formato de compresión sin pérdida), que contiene el modelo junto con la imagen de la matriz de confusión, la cual se incluye en formato Base64 para su fácil manejo y transferencia.

4.6.9. Codificación Base64

Base64 es un esquema de codificación que convierte datos binarios en una representación de 64 caracteres alfanuméricos. Esta codificación es ampliamente utilizada para transmitir información a través de protocolos que manejan únicamente texto, asegurando que los datos no se corrompan durante la transferencia.

El uso de Base64 en la API permite codificar imágenes y otros datos binarios en formato de texto, lo que facilita su intercambio entre módulos y sistemas sin riesgo de pérdida o alteración de la información.

4.7. Convenciones adoptadas

A nivel de desarrollo, se definieron ciertas convenciones con el objetivo de mantener un código claro, coherente y fácilmente mantenible. Entre ellas, se estableció el uso de la notación *Camelcase* para nombrar variables, funciones y componentes.

5. Producto

El producto resultante es una aplicación web interactiva orientada al análisis de experiencias basadas en la técnica de Speckle Dinámico. A través de una interfaz intuitiva, el sistema permite a los usuarios realizar entrenamientos que incluyen la carga de videos, el cálculo de descriptores de actividad, el uso de métodos de *clustering*, la configuración de una red neuronal (*feed-forward)* y la obtención del modelo entrenado.

El sistema incorpora una gestión de sesiones que permite registrarse e iniciar sesión. Cada usuario puede consultar sus entrenamientos guardados, utilizar modelos previamente generados para evaluar nuevas secuencias de video de laboratorio y obtener clasificaciones basadas en sus experiencias anteriores.

5.1. Módulos del sistema

La aplicación se encuentra estructurada en módulos que organizan sus funcionalidades en torno a un flujo claro de interacción con el usuario. Cada módulo responde a una necesidad específica y contribuye al proceso general de análisis basado en la técnica de Speckle Láser Dinámico.

El sistema dispone de tres módulos principales, los cuales constituyen el núcleo de su funcionalidad:

- **Módulo de autenticación:** Permite al usuario registrarse e iniciar sesión.
- **Módulo de Entrenamiento:** Permite configurar y ejecutar procesos de entrenamiento con redes neuronales, a partir de un video procesado mediante el cálculo de descriptores y la aplicación de algoritmos de *clustering*.
- **Módulo de Consulta:** Facilita la consulta utilizando nuevos videos a los modelos de redes neuronales previamente entrenados.

Estos módulos trabajan de manera integrada para ofrecer una herramienta eficiente y adaptable a las necesidades del usuario en el análisis de datos experimentales.

5.1.1. Módulo de autenticación

Este módulo controla el acceso al sistema, requiriendo que los usuarios se autentiquen antes de utilizar las funcionalidades disponibles. La autenticación puede realizarse mediante el método tradicional, ingresando un nombre de usuario y contraseña, o a través de autenticación utilizando una cuenta de Google, lo que simplifica el acceso y mejora la experiencia del usuario.

Figura 15. Interfaz de inicio de sesión del sistema.

Esta funcionalidad permite gestionar sesiones personalizadas con el objetivo de asociar modelos de entrenamiento a usuarios específicos.

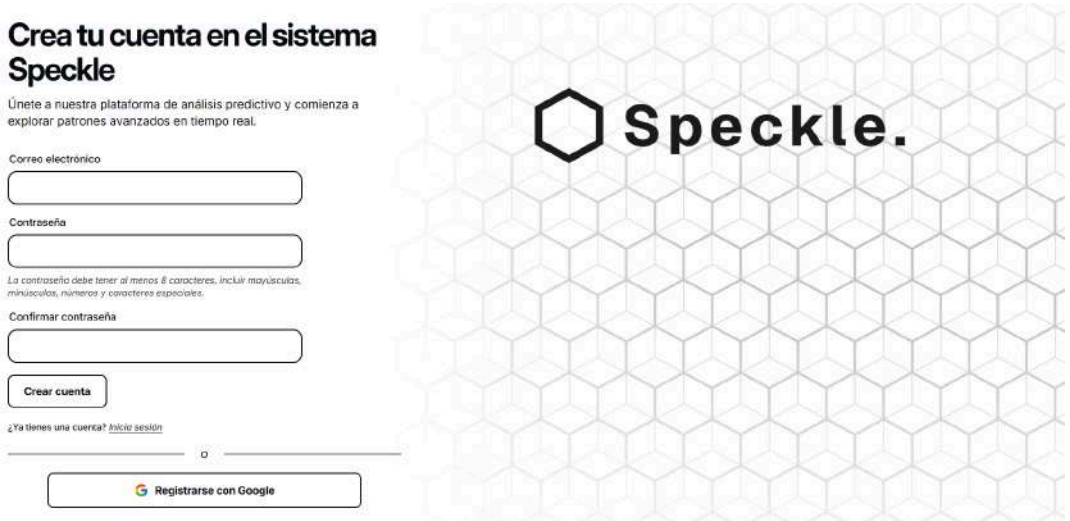

Figura 16. Interfaz de registro de usuarios.

5.1.2. Módulo de entrenamiento

El módulo de entrenamiento constituye el núcleo funcional de la aplicación. A través de una navegación guiada, basada en un flujo secuencial de pasos, el usuario puede configurar las entradas —video a procesar, descriptores, algoritmos de clustering y parámetros de la red neuronal— según sus necesidades y aplicar conocimiento específico para generar modelos personalizados de entrenamiento.

Una vez finalizado y almacenado el entrenamiento, el modelo queda disponible en el historial, desde donde pueden ser reutilizados en el módulo de consulta.

El flujo de trabajo del módulo consta de 10 pasos secuenciales, los cuales permiten al usuario tener un control detallado sobre cada etapa del proceso:

1) **Subir video** – El usuario carga una secuencia de video en formato .avi. El sistema muestra su tamaño, dimensiones y frames.
2) **Seleccionar descriptores** – Se ofrece la posibilidad de seleccionar entre 17 descriptores desarrollados específicamente para el análisis de experiencias basadas en Speckle Láser Dinámico (*Ver sección 4.6.4).*
3) **Seleccionar hiperparámetros** – En caso de que los descriptores seleccionados lo permitan, el usuario puede editar sus hiperparámetros para ajustar el procesamiento a las características del experimento.
4) **Seleccionar resultados de descriptores** – El sistema genera una imagen por cada descriptor aplicado, representando en un mapa de calor o imagen pseudo coloreada el nivel de actividad detectado. Estas imágenes pueden visualizarse en detalle, descargarse en formatos PNG, JPG, SVG o PDF, o bien exportar sus matrices en formatos TXT, CSV o JSON. El usuario debe seleccionar aquellas imágenes que considere útiles para el entrenamiento del modelo.
5) **Seleccionar algoritmos de *clustering*** – Se permite elegir entre 5 métodos de *clustering* (*Ver sección 4.6.5*).
6) **Editar parámetros de *clustering*** – El usuario puede modificar los parámetros específicos del método de agrupamiento seleccionado, optimizando el proceso según las características de los datos.

7) **Seleccionar resultados de *clustering*** – Se presentan las imágenes generadas por los algoritmos de *clustering* seleccionados, con las mismas opciones de visualización, descarga y exportación que en pasos anteriores. El usuario debe seleccionar una de ellas para continuar con el proceso de entrenamiento.
8) **Editar parámetros de la red neuronal** – El usuario define parámetros generales del entrenamiento, tales como el número de épocas, *batch size* y criterios de *early stopping*.
9) **Editar capas de la red neuronal** – Se permite especificar la cantidad de capas, el número de neuronas por capa, y aplicar técnicas como *dropout* y *batch normalization*, ajustando la arquitectura de la red según las necesidades del experimento.
10) **Resultado del entrenamiento** – El sistema ejecuta el entrenamiento del modelo y devuelve la matriz de confusión (*Ver sección 4.6.15*). Finalmente, se ofrece la opción de guardar la configuración del entrenamiento junto al modelo entrenado, permitiendo su posterior consulta.

5.1.3. Módulo de consulta

El módulo de consulta permite al usuario utilizar modelos entrenados para evaluar nuevos videos y obtener clasificaciones basadas en dichos modelos.

El acceso a este módulo se realiza desde el *header*, a través de su sección correspondiente, o mediante la barra lateral, donde se muestra el listado de modelos de entrenamiento guardados. El usuario debe seleccionar el modelo sobre el cual desea efectuar la consulta.

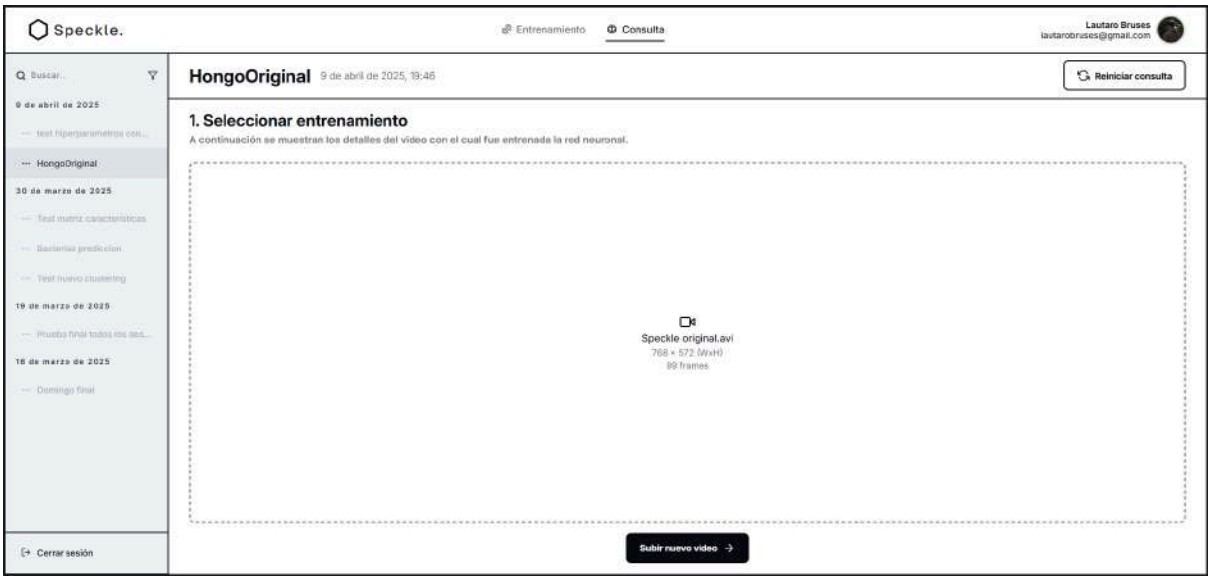

Figura 17. Pantalla inicial del módulo de consulta con modelos de entrenamiento guardados en la barra lateral.

Es importante destacar que, en caso de no disponer de un modelo de entrenamiento previamente guardado, el usuario no podrá avanzar más allá del primer paso del flujo de consulta, ya que la operación depende directamente de la existencia de un modelo entrenado.

El flujo consta de 6 pasos secuenciales:

1) **Seleccionar entrenamiento** – En este primer paso, el usuario debe seleccionar un modelo de entrenamiento previamente guardado. El sistema muestra las propiedades del video utilizado en la generación de dicho modelo, incluyendo detalles como el formato y sus dimensiones. Este paso actúa, además, como un mecanismo de control, impidiendo que el usuario avance en caso de no haber seleccionado un modelo guardado, garantizando así la integridad del proceso de consulta.
2) **Subir nuevo video** – El usuario carga un nuevo video, el cual será procesado por el sistema. Al igual que en el módulo de entrenamiento, se presentan al usuario las propiedades del archivo, tales como el tamaño, las dimensiones y el número de frames.
3) **Descriptores utilizados** – El sistema muestra los descriptores que fueron aplicados durante el entrenamiento del modelo seleccionado.

4) **Hiperparámetros utilizados** – En el caso que los descriptores seleccionados lo permitan, se presentan los hiperparámetros que fueron configurados en la etapa de entrenamiento, permitiendo al usuario conocer los valores con los que se procesa el nuevo video.
5) **Resultados de descriptores para la consulta** – El sistema genera una imagen por cada descriptor aplicado al nuevo video, al igual que en la fase de entrenamiento, permitiendo también la visualización en detalle dentro de la plataforma, la descarga en formato PNG, JPG, SVG o PDF, o bien exportar sus matrices asociadas en formato TXT, CSV o JSON.
6) **Visualizar resultado de la consulta** – Finalmente, el sistema presenta el resultado de la consulta como una imagen, que consiste en las clasificaciones generadas por el modelo entrenado a partir del análisis del nuevo video, pudiendo descargar tanto la imagen como la matriz asociada.

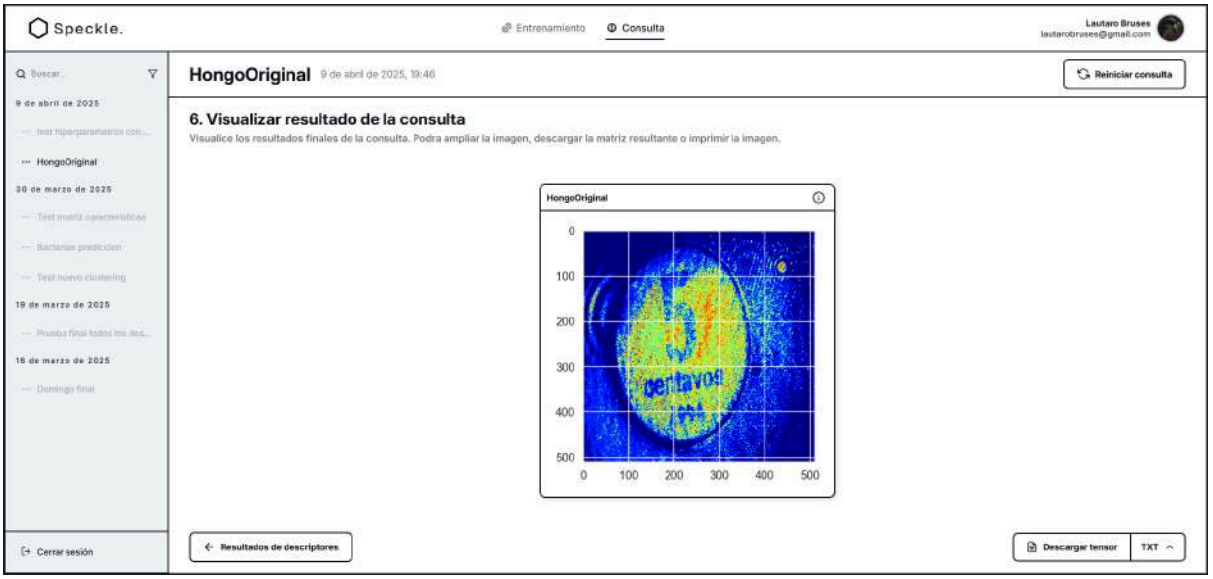

Figura 18. Consulta realizada a partir del modelo guardado “HongoOriginal”.

5.2. Oportunidades de mejora y expansión

A lo largo del desarrollo surgieron oportunidades de mejora y expansión, como la posibilidad de procesar múltiples videos de forma automatizada para entrenar la red neuronal con un volumen mayor de datos, normalizar los descriptores de forma grupal en lugar de individual y tener una configuración personalizable de los hiperparámetros. Estas ideas, si bien fueron descartadas por exceder el alcance

actual, abren la puerta a futuras mejoras del sistema con funcionalidades más robustas.

Además de estas posibles mejoras funcionales, también se identificaron oportunidades en el plano arquitectónico del sistema, principalmente de la API de cálculo. En particular, podría evaluarse desacoplar las tareas más exigentes (por ejemplo, el procesamiento de clustering y descriptores) mediante una arquitectura que permita gestionar estos procesos de forma independiente y asincrónica. Este enfoque contribuye a una mejor distribución de recursos, evita bloqueos en los puntos críticos del sistema y mejora la experiencia general del usuario bajo condiciones de alta demanda.

6. Testing

Para el proceso de *testing*, se optó por restringir todas las entradas del usuario desde el frontend —como los hiperparámetros de los descriptores, los parámetros de los algoritmos de clustering y las configuraciones de la red neuronal—, el conjunto de datos a validar fue más acotado y controlado. Esto permitió realizar pruebas manuales sobre las funciones desarrolladas tanto en el backend como en la API de cálculo, a medida que se completaban las implementaciones. Como resultado, se logró depurar la mayoría de los errores detectados durante el desarrollo.

6.1. Pruebas unitarias

Para llevar a cabo las pruebas unitarias, se desarrollaron módulos de prueba específicos que simulan el comportamiento de componentes reales (mockups). Esta estrategia permitió validar el funcionamiento de cada módulo de forma aislada, sin depender del desarrollo completo del sistema. Al ensayar distintos flujos de ejecución, se logró verificar y depurar las funcionalidades de manera eficiente, agilizando el proceso de validación.

Como criterio de aceptación en el módulo de cálculo, se definió que debía ser evaluado al menos con tres escenarios diferentes. Estos incluían los extremos del rango permitido de entrada, así como valores intermedios. De este modo, fue posible ensayar bajo condiciones límite —que al tratarse de un proyecto de investigación no son límites absolutos— y validar la robustez del módulo. En los casos en que se involucraban múltiples variables de entrada, se diseñaron combinaciones específicas que pudieran representar situaciones críticas para el sistema, además de cubrir los límites y el centro del rango aceptado por cada una de las variables.

Por otro lado, la integración entre el frontend y el backend se realizó en las etapas finales del desarrollo, el proceso resultó ágil y sin complicaciones dado que, en las fases previas, se llevaron a cabo pruebas manuales. Dichas pruebas fueron realizadas mediante Postman, una herramienta eficaz para probar y documentar

APIs. Con su ayuda, se simuló un cliente frontend que interactuaba directamente con los endpoints del backend, los cuales, a su vez, gestionaban la comunicación con la API de cálculo. Durante estas simulaciones, se enviaron solicitudes HTTP que incluían los tokens de autenticación necesarios, lo que permitió verificar el correcto funcionamiento tanto de los mecanismos de seguridad y como el acceso a los recursos protegidos.

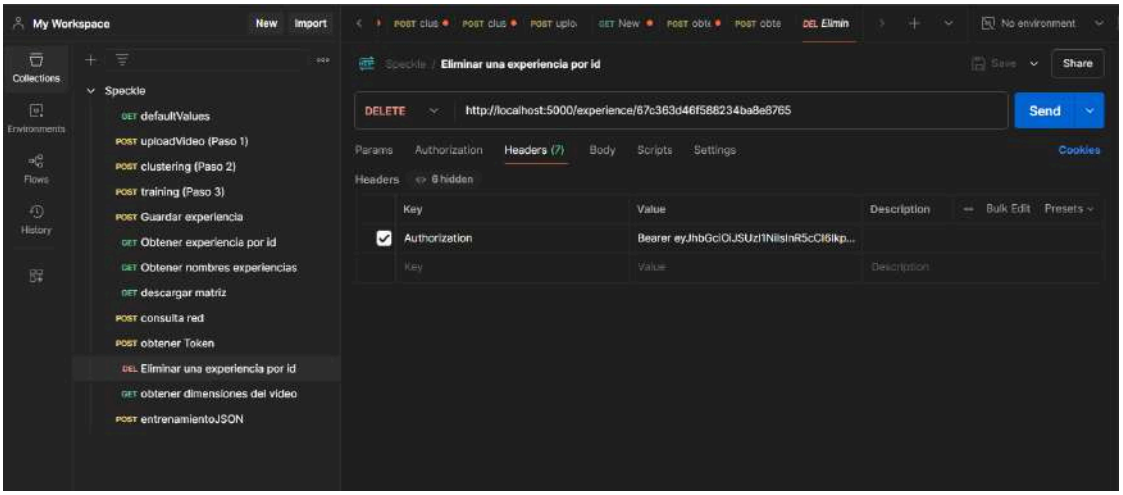

Figura 19. Batería de prueba utilizada para testear los endpoints del backend.

En cada solicitud, se incluían los *keys* de autenticación necesarios dentro de los *headers* y, en el *body*, se enviaban los parámetros requeridos por cada endpoint específico. Esto permitió validar tanto la correcta autenticación como el procesamiento de datos en cada uno de los flujos previstos.

6.2. Pruebas de rendimiento en API de cálculo

Ante la falta de requerimientos acerca de los requisitos mínimos necesarios de la infraestructura se realizaron pruebas de rendimiento de la API de cálculo utilizando Apache JMeter, una herramienta utilizada para medir la capacidad de respuesta, la cantidad de usuarios concurrentes y la estabilidad de aplicaciones web. El análisis se basó en los reportes de resumen generados por la herramienta (*Ver anexo III*).

El objetivo de estas pruebas fue evaluar las especificaciones requeridas para alojar la API de cálculo en un servidor web.

Las pruebas se ejecutaron en un entorno local, utilizando una computadora perteneciente a uno de los integrantes del equipo de desarrollo, con las siguientes especificaciones: procesador AMD Ryzen 7 5700U y 12GB de memoria RAM.

Se presentaron restricciones significativas al intentar escalar el número de hilos (usuarios concurrentes). En particular, se evidenció que el entorno no era adecuado para manejar cargas pesadas, ya que al superar los 15 hilos concurrentes o ejecutar más de 4 procesos simultáneamente, se producían errores de falta de memoria que impedían la continuidad de las pruebas.

Se realizaron pruebas con distintas cantidades de hilos concurrentes para analizar cómo respondía el sistema ante cargas progresivamente mayores.

El análisis del uso de memoria y CPU reflejó que el sistema es altamente intensivo en recursos. Entre los puntos más relevantes se destacan:

- **Elevado consumo de memoria**: La API de consulta requiere una cantidad considerable de memoria, lo que está alineado con los errores de falta de memoria detectados al intentar ejecutar un mayor número de procesos o hilos concurrentes.
- **Monitor de memoria y CPU**: Durante las pruebas el uso de la memoria promedio el 62,5%, con picos de 100%, mientras que el uso de CPU promedio 31,5% con picos también del 100%. Sin embargo mientras las pruebas no estaban en ejecución el uso de CPU rondó el 0,5% y el de la memoria 44%.
- **Bajo crecimiento del throughput**: No se observó un aumento proporcional en el throughput (número de peticiones procesadas por segundo) a medida que se incrementaron las muestras, lo que sugiere limitaciones en la capacidad de escalabilidad del sistema en el entorno local.

A partir de estas evaluaciones, se identificaron los siguientes observaciones:

- **Dificultades de escalabilidad**: A medida que aumentaba la carga de trabajo, el sistema evidenció limitaciones para escalar de forma eficiente, especialmente en los *endpoints* de descriptores y *clustering*.

- **Incremento en los tiempos de respuesta**: El aumento en el volumen de muestras procesadas generó una subida considerable en los tiempos de respuesta, afectando principalmente los endpoints mencionados.
- **Alta desviación estándar en los tiempos de respuesta**: Aunque no se produjeron errores críticos, se observó una elevada desviación estándar en los tiempos de respuesta, lo que indica un comportamiento menos predecible del sistema bajo cargas elevadas.

Los resultados indican que la API es estable bajo cargas moderadas, pero presenta limitaciones cuando se incrementa el número de hilos concurrentes debido a restricciones de memoria y CPU en el entorno de prueba. El comportamiento observado en los *endpoints* de descriptores y *clustering* sugiere que estos módulos requieren un entorno con mayores recursos para poder escalar eficientemente.

Para futuras implementaciones en producción, es recomendable realizar pruebas en un entorno con mayores recursos computacionales. Además, realizar un análisis de costo-beneficio entre la performance del sistema y la precisión requerida por los cálculos y resultados obtenidos por el módulo. Lo cual nos permitirá obtener conclusiones más detalladas de los pasos a seguir.

7. Despliegue

El despliegue es un proceso mediante el cual un sistema se pone en funcionamiento en un ambiente de producción. Esto implica la transferencia de código fuente, archivos y otros recursos necesarios hacia un servidor para que el sistema se ejecute en él.

Inicialmente se proyectó realizar el despliegue en los servidores de la facultad de Ingeniería de la Universidad Nacional de Mar del Plata. Esta alternativa fue descartada porque su complejidad excedía los tiempos y los plazos del proyecto.

Como alternativa se evaluaron diversas opciones, incluyendo Render, Railway y PythonAnywhere. Finalmente, se seleccionó Render por su curva de aprendizaje accesible, especialmente considerando la poca experiencia previa en tareas similares. La plataforma demostró ser intuitiva y sencilla de utilizar.

Debido a limitaciones presupuestarias, se utilizó la versión gratuita de la herramienta para llevar a cabo el despliegue. En este entorno se realizaron pruebas destinadas a evaluar los tiempos de respuesta de procesos habituales en la aplicación.

7.1. Despliegue en Render

La versión gratuita de Render ofrece recursos bastante limitados: específicamente, 0,1 vCPU y 512 MB de memoria RAM. Además, presenta restricciones adicionales importantes como:

- Desconexión automática tras períodos de inactividad.
- Ausencia de acceso mediante SSH.
- Imposibilidad de escalado automático.
- No permite la ejecución de tareas programadas.
- Falta de discos persistentes.

Estos recursos limitados sugerían que la API podría enfrentar problemas de rendimiento y seguridad. Aun así, se decidió realizar este despliegue para evaluar la

viabilidad del sistema en un entorno remoto básico, sin incurrir en costos mensuales.

Se realizaron los ajustes para llevar a cabo este despliegue:

- Se creó un repositorio nuevo y dedicado exclusivamente al código de la API de cálculo, facilitando su vinculación con Render.
- Se renombró el archivo principal de ejecución de *api.py* a *main.py*, cumpliendo con la convención exigida por Render.
- Se generó un archivo *requirements.txt* que incluyó todas las dependencias necesarias para garantizar la ejecución correcta en producción.

Además, debido a que el módulo de cálculo forzaba el uso de HTTPS, en el entorno local de desarrollo se emplearon certificados autofirmados. Sin embargo, en producción, Render proporciona automáticamente certificados HTTPS válidos, eliminando la necesidad de gestionar certificados adicionales.

Para lanzar el servidor en producción se utilizó Gunicorn, un servidor HTTP WSGI (*Web Server Gateway Interface*) optimizado para aplicaciones Python. Esto representó un cambio respecto al entorno de desarrollo, donde se utilizó Uvicorn para facilitar pruebas y depuración.

El despliegue permite al módulo operar correctamente y exponer todos sus endpoints públicamente en la URL: https://speckleapi.onrender.com/.

Tras el despliegue de la API de cálculo, se procedió con los módulos de backend y frontend, que también fueron desplegados en Render, debido a su simplicidad.

Se creó un nuevo repositorio independiente para estos módulos, facilitando su integración con Render. Para lograr un despliegue compacto, el módulo frontend fue construido utilizando el comando *"npm run build"* y luego integrado directamente en el módulo backend.

Estos ajustes requirieron modificaciones menores en el backend, solucionando inicialmente algunos errores en el despliegue para servir correctamente los archivos estáticos generados por el frontend. Además, fue necesario autorizar las IP

proporcionadas por Render en la base de datos, actualizar la configuración de autenticación en *Auth0* con la URL de producción y configurar correctamente las variables de entorno para asegurar la comunicación efectiva entre módulos.

De esta manera, el sistema quedó disponible públicamente en la URL: https://specklesystem.onrender.com/.

Una vez desplegado el sistema se realizaron pruebas de rendimiento sobre el endpoint de cálculo de descriptores. Los resultados evidenciaron un muy bajo rendimiento. Por ejemplo, el cálculo de un descriptor que en un entorno local se ejecuta en menos de 1 segundo (*Ver sección 6.2.*), en el servidor web tarda más de 2 minutos. Además, al intentar procesar tres o más descriptores simultáneamente, el servicio supera el límite de memoria disponible e imposibilita la continuidad del procesamiento.

Los resultados confirman que los recursos disponibles en la versión gratuita de Render no son suficientes para ejecutar la API de manera eficiente, debido al alto consumo de memoria y procesamiento requerido por el sistema. Por este motivo, se descartó el uso de dicha versión. Como alternativa, se contempla la posibilidad de optar por planes pagos que ofrecen recursos más acordes a las necesidades del sistema. Estos planes tienen un costo inicial de aproximadamente 85 USD mensuales, lo cual representa una inversión significativa que deberá ser evaluada por el laboratorio en futuras etapas del proyecto.

7.2. Despliegue local a partir de un archivo ejecutable

Aunque la versión desplegada en Render quedó implementada, se observó que la aplicación resultó lenta y prácticamente inutilizable debido a las limitaciones de recursos del plan gratuito empleado.

Como alternativa, se optó por la creación de un archivo que se ejecuta de manera local, utilizando los recursos disponibles en el equipo del usuario. Esta solución permite el uso del sistema sin depender de servicios externos eliminando costos adicionales asociados.

Como resultado se obtuvieron tiempos de respuesta acordes a las necesidades del proyecto. Gracias a esta decisión, los usuarios del Laboratorio de Bioingeniería pueden hacer uso del software con fluidez, lo que permitió una retroalimentación final que contribuyó a las etapas de validación y finalización del sistema.

8. Memorias del proyecto

Si bien no está directamente relacionado con este proyecto, es importante mencionar que el desarrollo de éste surgió luego del intento fallido de una propuesta de proyecto final anterior. En dicho proyecto inicial, el principal obstáculo fue la dificultad para acotar un problema que excedía tanto en complejidad como en la cantidad de horas requeridas para su implementación. Como consecuencia, no se logró definir una solución viable y aplicable en un contexto real, lo que llevó a descartar el proyecto y dar lugar a la presente propuesta.

Tras desestimar esta primera idea, se estableció contacto con el Dr. Ing. Gustavo Meschino, quien, junto con el Dr. Ing. Marcelo Guzmán y la Ing. Estefany Cujano Ayala, integrantes del Laboratorio de Bioingeniería de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata, presentaron la problemática que dio origen a este desarrollo.

El laboratorio no disponía de un software integral dedicado para el cálculo de descriptores y su posterior integración con técnicas de inteligencia artificial y aprendizaje automático. Hasta ese momento, utilizaban herramientas dispersas y de difícil integración, lo que motivó la necesidad de desarrollar una solución unificada que optimizara su flujo de trabajo.

Dentro de las etapas iniciales del proyecto surgió un cambio de rumbo en la planificación. La metodología secuencial propuesta inicialmente no se ajustaba a las necesidades reales, debido a la naturaleza dinámica de los requerimientos y la inexperiencia del equipo en el proceso de elicitación. Este ajuste metodológico tuvo como consecuencia que los tiempos reales invertidos difirieran considerablemente de la planificación preliminar, provocando que no resulten comparables con lo planteado inicialmente.

Con el objetivo de relatar el proceso vivido durante el desarrollo del proyecto, en los siguientes subcapítulos se realizará una retrospección de cada una de las etapas. Esta revisión permitirá describir cómo evolucionó el trabajo en cada fase, los

cambios que se fueron incorporando, así como las decisiones tomadas y los aprendizajes obtenidos a lo largo del proyecto.

8.1. Análisis y diseño

El producto esperado evolucionó a lo largo del tiempo. Inicialmente, se planteó como un sistema capaz de procesar videos de experiencias de Speckle Láser Dinámico mediante el cálculo de descriptores e integrar estos datos con alguna técnica de aprendizaje automático (*machine learning*). El sistema se redefinió a un modelo semi-supervisado, capaz no sólo de procesar videos de experiencias de laboratorio, sino también de aplicar diversas técnicas de agrupamiento, definir y entrenar una red neuronal en tiempo real, y utilizar ese entrenamiento para clasificar zonas de mayor o menor actividad en nuevas experiencias.

El cambio de enfoque, hizo necesario el establecimiento de reuniones internas semanales en lugar de quincenales lo cual fue una decisión clave para la organización del equipo. Esta dinámica surgió como respuesta a las distintas disponibilidades horarias de los integrantes, y permitió no depender del trabajo presencial simultáneo.

Estas reuniones no solo brindaron flexibilidad al proceso, sino que también facilitaron la alineación del equipo, asegurando que todos los miembros se mantuvieran actualizados sobre el progreso de las tareas asignadas, la resolución rápida de problemas y la adaptación a cambios durante el desarrollo.

Paralelamente, se programaron encuentros presenciales cada dos semanas con los directores del proyecto, como parte del proceso de análisis y validación del producto. Estas instancias permitieron identificar y redefinir necesidades, asegurando que el desarrollo del proyecto se mantuviera enfocado en los objetivos planteados.

Desde las primeras reuniones, si bien la necesidad de desarrollar un software integral estaba establecida, surgieron nuevas necesidades y prioridades planteadas por los directores y referentes funcionales. Esto llevó a realizar un cambio significativo respecto de la planificación secuencial inicialmente prevista, migrando

hacia una metodología iterativa e incremental con enfoque ágil. Este cambio de paradigma resultó acertado, ya que permitió que los requerimientos fueran ajustados conforme avanzaba el desarrollo. Como consecuencia, la metodología secuencial inicialmente planificada fue completamente reemplazada: haberla mantenido habría implicado una mayor cantidad de trabajo y refactorizaciones constantes del sistema, incrementando el costo en horas de desarrollo.

Este cambio de paradigma permitió desarrollar el sistema en ciclos cortos, lo que resultó esencial para adaptarse a las necesidades emergentes del proyecto. Además, favoreció una toma de decisiones flexible y una gestión más efectiva del conocimiento, ya que, a medida que avanzaba el desarrollo, se adquirió una comprensión más profunda de los objetivos del sistema, lo que impactó positivamente en la calidad de las implementaciones.

Aunque no se adoptó completamente un enfoque iterativo-incremental en el desarrollo —dado que no se planificaron entregables funcionales desde el inicio—, se trabajó con una estrategia de ajuste continuo. En cada reunión quincenal se presentaban los avances y se acordaban nuevas funcionalidades; no obstante, la adopción de una planificación de entregables funcionales desde el inicio podría haber optimizado aún más el flujo de validación del producto. Este enfoque permitió recibir retroalimentación temprana y realizar correcciones oportunas, lo que ayudó a evitar desvíos costosos y a asegurar que el proyecto se mantuviera en el rumbo adecuado.

Dado que el trabajo se organizó dividiendo los tres módulos principales entre los integrantes del equipo, el análisis y diseño de bajo nivel dentro de cada uno de ellos quedó a cargo del responsable asignado a cada módulo. En este contexto, la selección de tecnologías no sólo respondió a criterios de eficiencia y adecuación al problema, sino que también se basó en los conocimientos previos de los desarrolladores. Con el avance del proyecto, se advirtió que la estrategia adoptada resultó acertada ya que permitió reducir significativamente los tiempos de aprendizaje donde cada integrante pudo apoyarse en su experiencia previa para implementar soluciones efectivas y enfocarse en resolver los desafíos propios del proyecto.

Como se mencionó en los párrafos anteriores, un punto de mejora en el desarrollo del proyecto hubiese sido la adopción completa de una metodología ágil que incluyera la entrega de prototipos funcionales durante el proceso ya que la retroalimentación más valiosa por parte de los directores —y potenciales usuarios del sistema— se obtuvo recién en las últimas semanas de desarrollo, cuando se integraron todos los módulos y pudieron interactuar con el sistema de forma unificada.

8.2. Desarrollo e implementación

Para registrar y gestionar los cambios en el proyecto, se utilizó el sistema de control de versiones GitHub. Se creó un repositorio con una estructura organizada en carpetas separadas para cada uno de los módulos principales, lo que permitió a cada integrante contar con un espacio de trabajo aislado. Esta organización facilitó la colaboración, optimizó la gestión de los cambios y redujo los conflictos durante las fusiones de código.

Como se mencionó anteriormente, esta etapa se desarrolló de manera paralela al dividir los tres módulos principales entre los integrantes del proyecto, lo que permitió que cada miembro se concentrara en la implementación específica de su parte. Este enfoque también requirió la definición de interfaces claras y bien delimitadas para garantizar la correcta interacción entre los módulos. En este contexto, las reuniones semanales resultaron un acierto clave, ya que facilitaron una comunicación fluida, la coordinación de avances y el ajuste continuo de dichas interfaces ante nuevos requerimientos o modificaciones, evitando así bloqueos y permitiendo que el trabajo avanzara de forma independiente.

La naturaleza iterativa de la metodología adoptada dificultó la generación de documentación completa durante el desarrollo. Además, la especialización de los integrantes en módulos concretos redujo la necesidad inmediata de contar con documentación detallada para coordinar el trabajo interno. Debido a que elaborar dicha documentación habría excedido los plazos del proyecto, se decidió excluirla del alcance actual. Sin embargo, representaría una buena práctica para futuras extensiones o mantenimiento del sistema, por lo que consideramos recomendable

aprovechar herramientas modernas de inteligencia artificial para generar documentación detallada en futuras etapas.

Si bien inicialmente se había planificado un tiempo menor para esta fase —aproximadamente un 30 % menos—, la adopción del nuevo enfoque metodológico permitió que la cantidad de horas requeridas no se incrementara aún más respecto de lo estimado inicialmente con la metodología secuencial.

8.3. Testing

Aunque esta etapa estaba originalmente planificada para ejecutarse una vez finalizada la implementación, en la práctica el proceso de pruebas convivió con el desarrollo durante gran parte de la codificación. Esto se debió a que, luego de implementar cada segmento de código, se desarrollaron módulos adicionales o scripts específicos para verificar su correcto funcionamiento.

Aunque las pruebas automáticas unitarias y de integración son prácticas muy recomendadas en el desarrollo de software de calidad, al tratarse de un proyecto de investigación y debido a los cambios frecuentes en los requerimientos las prioridades del desarrollo no se centraran en la implementación de pruebas automáticas. En lugar de ello, se priorizó la flexibilidad y rapidez en la entrega de funcionalidades para su consecuente retroalimentación, lo que permitió adaptarse rápidamente a las nuevas exigencias.

Dada la naturaleza del desarrollo, se advirtió que la implementación y el testing tienden a integrarse de manera continua, ya que los nuevos bloques de código se validan en el momento de su creación. De esta experiencia se concluyó que es difícil separar completamente ambas actividades y que esto complica la realización de estimaciones precisas sobre su duración. Como aprendizaje, se reconoce la importancia de planificar considerando esta superposición natural, así como de adoptar prácticas que permitan integrar las pruebas de forma temprana y continua en el ciclo de desarrollo, mejorando así la calidad del producto final.

Durante la fase de testing se verificó que el sistema cumplió con los objetivos establecidos en cuanto a los tiempos de respuesta observados bajo condiciones

normales de uso, aun cuando no se habían definido explícitamente parámetros máximos aceptables en las etapas iniciales del proyecto. Sin embargo, el problema surgió al analizar la posibilidad de escalar en la cantidad de usuarios concurrentes, ya que, si bien no se había contemplado originalmente un escenario de alta concurrencia, esta limitación nunca fue formalmente especificada ni documentada.

Asimismo, se evidenció que los requerimientos mínimos de infraestructura, la arquitectura seleccionada y la decisión respecto de la plataforma tecnológica —en este caso, una aplicación web— no estuvieron directamente fundamentados en criterios claros de eficiencia ni sustentados en un análisis detallado de carga, concurrencia o crecimiento futuro. Esta situación expuso una falencia en la planificación inicial del proyecto, que relegó aspectos críticos como la performance, la escalabilidad y la definición precisa de las necesidades de infraestructura.

8.4. Despliegue

Como se mencionó anteriormente, ninguno de los integrantes del equipo contaba con experiencia en el despliegue de una aplicación web en producción. Inicialmente, se consideró realizar el despliegue en el servidor de la facultad, ya que esta opción permitía anticiparse a posibles problemas de lentitud en el procesamiento de los modelos de inteligencia computacional. Esta alternativa, sin embargo, implicaba un método de implementación más complejo y requería tiempos de puesta en producción más prolongados.

Al descartar la primera opción por la limitación de los plazos del proyecto, se optó por emplear un servicio de hosting gratuito. Sin embargo, debido a las restricciones de rendimiento que presentó esta plataforma, se decidió finalmente desarrollar un ejecutable que pudiera correr localmente en la computadora del usuario. Como aspecto positivo, este recorrido reflejó un proceso de mejora continua y evolutiva, donde se buscaron soluciones frente a los errores surgidos en la planificación inicial.

Como aprendizaje de esta etapa, se concluye que es clave realizar un análisis técnico más detallado de la solución a implementar al momento de estimar la duración del despliegue. Esto implica investigar la infraestructura disponible y

desglosar con mayor precisión las tareas necesarias para anticiparse a posibles inconvenientes y así obtener una estimación más realista del tiempo requerido para la puesta en producción.

8.5. Redacción de informe y finalización

Aunque inicialmente se había planificado realizar redacciones parciales al finalizar cada una de las etapas del ciclo de vida del proyecto, la dinámica de trabajo adoptada no permitió cerrar las etapas. Por este motivo, la redacción del presente informe comenzó recién durante los últimos meses del desarrollo.

Para reconstruir cronológicamente las decisiones y actividades llevadas a cabo, se recurrió principalmente a dos fuentes: las minutas de reunión, que registraban los temas tratados y las conclusiones alcanzadas; y una bitácora de horas, donde cada integrante anotaba las tareas realizadas y el tiempo dedicado. Sin embargo, en ambos casos, la información resultó insuficiente. Las minutas documentaron sólo los aspectos más relevantes, omitiendo detalles que habrían enriquecido el análisis posterior. Por su parte, la bitácora indicaba de manera general a qué módulo se destinaba el trabajo, sin describir en profundidad las tareas concretas ni los avances logrados, lo que dificultó la asignación precisa de esas horas a etapas específicas del proyecto.

Ante el cambio metodológico que implicó pasar de un enfoque secuencial a uno iterativo, la falta de documentación progresiva generó un efecto acumulativo. Inicialmente, se comprendía al informe como un documento orientado al producto, cuando en realidad su objetivo principal era reflejar también la evolución y los cambios de gestión ocurridos durante el desarrollo. Este desfase hizo evidente la importancia de comenzar la redacción en paralelo con el avance del proyecto.

Como principal aprendizaje, se concluye que postergar la resolución de problemas sólo agrava su impacto a futuro. Resulta fundamental abordar los inconvenientes de inmediato y contar con un plan de contingencia que permita anticiparse a imprevistos. La implementación de un registro más detallado y minucioso de los hitos, decisiones y tareas desarrolladas no solo facilita la etapa de documentación y

la redacción posterior, sino que mejora significativamente la trazabilidad, el análisis crítico y la calidad de la gestión del proyecto en su conjunto.

8.6. Tiempo de entrega

Aunque la fecha de finalización estaba prevista originalmente para fines de febrero, la entrega del proyecto y la redacción del informe se vieron demoradas aproximadamente 60 días. Esta postergación fue producto de varios factores, los cuales se detallan a continuación.

El factor más determinante fue el cambio en la metodología de trabajo adoptada, ya que la metodología secuencial planificada inicialmente resultó inadecuada para un contexto en el que los requerimientos no estaban completamente definidos desde el comienzo. Por esta razón, la estimación inicial de tiempos y esfuerzos se basaba en un modelo de trabajo que no se ajustaba a la naturaleza real del proyecto. Bajo esta dinámica, las aproximadamente 80 horas adicionales invertidas respondieron a las necesidades concretas del desarrollo; manteniendo la planificación inicial, es probable que el tiempo requerido hubiera resultado considerablemente mayor.

Por otro lado, al momento de proyectar las horas semanales de trabajo, no se tuvo en cuenta el período específico en el que se llevaría a cabo el desarrollo. Este coincidió con las festividades de Navidad y Año Nuevo, así como con las vacaciones personales del mes de enero. Durante estas semanas, la dedicación horaria al proyecto se redujo considerablemente, dificultando la continuidad de las tareas. Además, en ese mismo período no fue posible coordinar reuniones con los directores, ya que también se encontraban de licencia.

Adicionalmente, los meses finales de desarrollo coincidieron con una disminución en la cantidad de horas disponibles destinadas al proyecto. Por ejemplo, el hecho que dos integrantes del equipo estuvieran en proceso de búsqueda laboral, afectó de manera directa la disponibilidad de tiempo para el proyecto limitando la posibilidad de incrementar las horas destinadas al proyecto para cumplir con los plazos establecidos, e incluso, en algunos casos, obligó a reducirlas. Como enseñanza podemos describir que estos hechos debieron ser señalados en el análisis FODA

como una debilidad para ser tenidos en consideración a la hora de realizar la planificación del proyecto.

A continuación se detallan las horas dedicadas individualmente por Lautaro, Ignacio y Matías, así como las correspondientes a tareas realizadas en conjunto (Equipo), en el período comprendido entre septiembre de 2024 y abril de 2025.

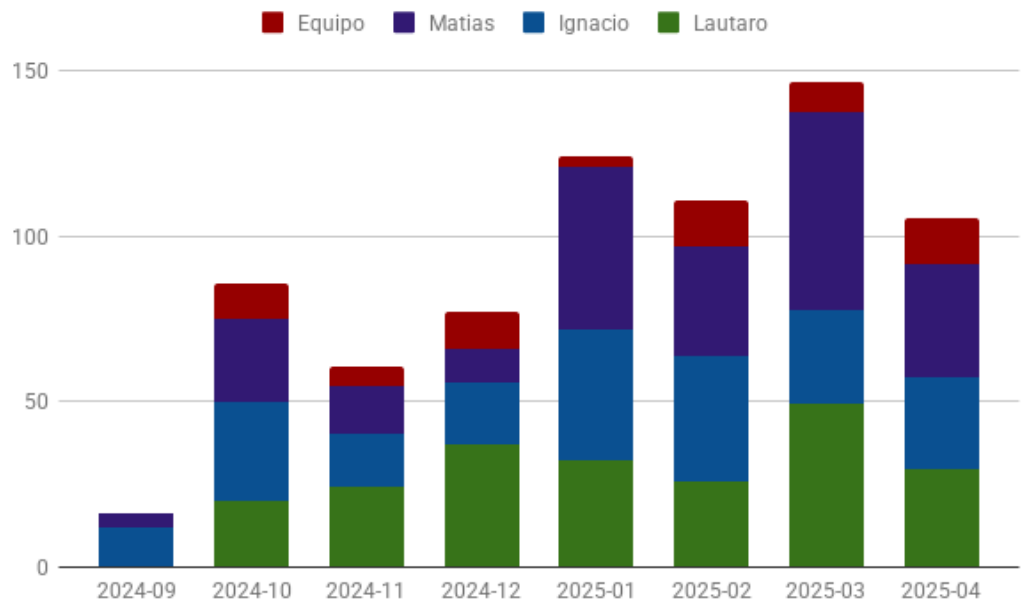

Figura 20. Distribución mensual de horas invertidas por integrante del equipo durante el desarrollo.

Esta experiencia nos permitió reflexionar sobre la importancia de planificar no solo en función de los tiempos técnicos, sino también considerando los aspectos personales y contextuales del equipo. Para futuros proyectos, consideramos clave realizar una planificación más consciente, que contemple posibles interrupciones y márgenes de contingencia.

8.7. Ejecución real del proyecto

La siguiente imagen muestra dos diagramas de Gantt: el superior corresponde a la propuesta inicial desarrollada al comienzo del proyecto, mientras que el inferior refleja la ejecución real de las tareas, reconstruida a partir de los registros documentados en la bitácora de trabajo. Si bien no es posible realizar una comparación directa entre ambos —dado que la planificación inicial respondía a una

metodología secuencial con etapas fijas, mientras que la ejecución real se basó en un enfoque iterativo e incremental—, el contraste permite identificar diferencias significativas no sólo en los tiempos estimados, sino también en la distribución y superposición de las etapas a lo largo del ciclo de vida del proyecto.

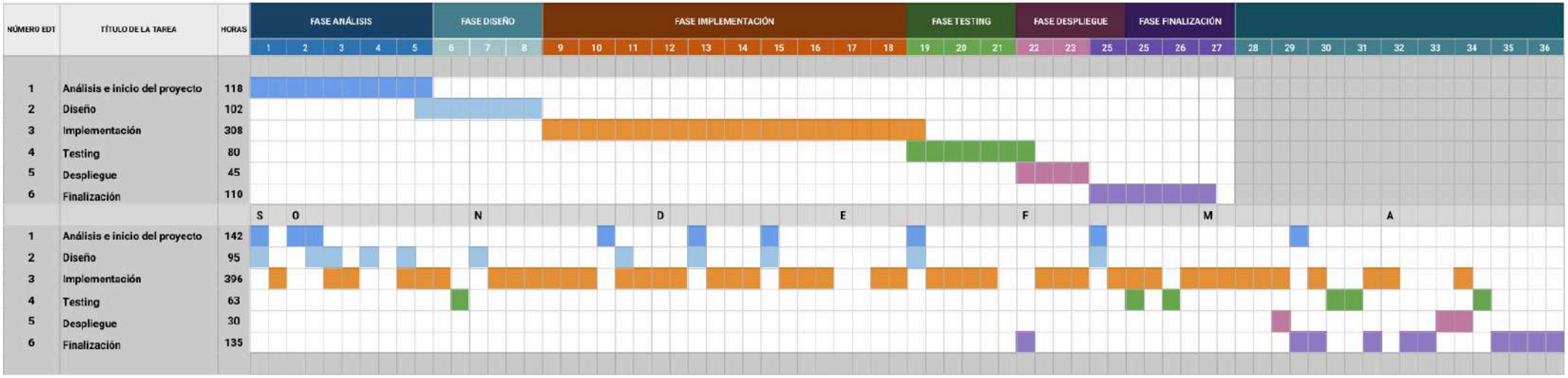

Figura 21. Comparativa entre la planificación original y la ejecución real del proyecto. En el gráfico se presentan dos diagramas de Gantt: el primero refleja la distribución inicial de horas por cada fase del proyecto, mientras que el segundo muestra la distribución real de las horas ejecutadas en las distintas etapas.

8.7.1. Distribución mensual de horas ejecutadas

En esta sección se presenta la ejecución real mes a mes junto con la distribución mensual de horas dedicadas a cada una de las etapas del desarrollo —análisis, diseño, implementación, testing, despliegue y finalización—.

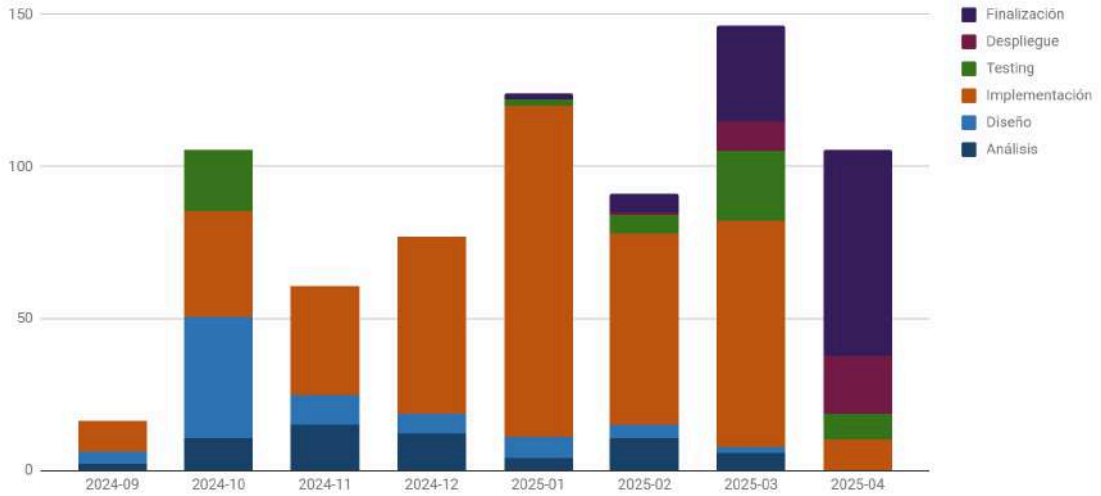

Figura 22. Distribución mensual de horas invertidas según las distintas etapas del proyecto. Se detallan las horas dedicadas a análisis, diseño, implementación, testing, despliegue y finalización entre septiembre de 2024 y abril de 2025.

9. Conclusiones

La entrega de un producto funcional que satisface las necesidades del Laboratorio de Bioingeniería de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata demuestra que se cumplió de forma exitosa el objetivo general planteado al inicio del proyecto.

Desde el punto de vista formativo, este proyecto representó un desafío significativo para el equipo de desarrollo, al brindar la oportunidad de aplicar los conocimientos teóricos adquiridos a lo largo de la carrera en el diseño y construcción de una solución de software con impacto real. Durante el proceso, surgieron diversos obstáculos que, lejos de ser un freno, se transformaron en valiosas fuentes de aprendizaje. Estos desafíos permitieron fortalecer competencias técnicas y metodológicas, así como incorporar lecciones que enriquecieron nuestra formación profesional de un modo que trasciende lo estrictamente académico.

Una de las principales lecciones aprendidas fue que la implementación constituye sólo una parte del proceso. Llevar adelante un proyecto real permitió comprender que el desarrollo forma parte de un ciclo más amplio que incluye planificación, estimación, análisis, diseño, validación y documentación. Como aprendizaje fundamental, se hizo evidente la importancia de dimensionar adecuadamente cada etapa, prever márgenes de contingencia y establecer mecanismos de control de cambios que permitan sostener el rumbo frente a ajustes durante el proceso.

Un cambio significativo durante el desarrollo del proyecto fue la adopción de una metodología ágil, en contraposición a la metodología secuencial que se había planteado inicialmente. La metodología definida al comienzo no se ajustaba completamente a las necesidades del equipo demandante ni a la naturaleza dinámica del problema. Este ajuste nos permitió ser más flexibles, obteniendo un producto final más alineado con las necesidades reales de los demandantes y permitió comprender que la capacidad de reaccionar ante cambios es una de las competencias más valiosas en nuestra profesión.

Durante el proyecto se identificaron algunas limitaciones en el análisis FODA realizado inicialmente. Hubo factores que se enfocaron en aspectos del producto final, como la facilidad de uso o la especialización de software, en lugar de centrarse en el proyecto. Además, aparecieron elementos que no se habían considerado en el análisis, como la activa participación de los demandantes y usuarios finales, que terminó siendo una fortaleza clave ante el cambio de metodología. Asimismo, ciertos puntos fueron clasificados incorrectamente, como la disponibilidad horaria de los integrantes, que se consideró una amenaza cuando en realidad era una debilidad interna vinculada a la organización personal del equipo.

Un aspecto clave fue la ausencia de una matriz de riesgos y un plan de contingencia, lo que generó incertidumbre frente a imprevistos. Por ejemplo, la planificación de la redacción del informe se pensó inicialmente para realizarse al finalizar cada etapa, pero al no avanzar estas de manera secuencial, el trabajo se acumuló en la fase final y afectó el nivel de detalle alcanzado.

La extensión del plazo de entrega estuvo vinculada tanto a estimaciones iniciales poco precisas como a la falta de un análisis técnico detallado al planificar cada etapa; esto se evidenció con mayor claridad en la fase de despliegue, donde se subestimó la complejidad de implementar la solución planteada.

Se comprendió también que la falta de declaración formal de ciertos requerimientos clave —ya sea por parte de los demandantes o del equipo de desarrollo— impacta directamente en atributos esenciales para la verificación y validación del sistema. Por ejemplo, la ausencia de especificaciones sobre usuarios concurrentes, requisitos de rendimiento y criterios de aceptación dificultó dimensionar correctamente costos y plazos.

Estos aprendizajes refuerzan la importancia de destinar el tiempo necesario a una planificación detallada, elaborar matrices de riesgos y planes de contingencia, y formalizar los requerimientos desde el inicio para poder gestionar de forma eficaz los cambios e imprevistos que puedan surgir. Al mismo tiempo, se reconoce como un logro el haber mantenido una orientación constante hacia el objetivo, sumado a una capacidad de adaptación y trabajo colaborativo que permitió responder de

forma ágil a las dificultades y entregar un producto funcional alineado con las necesidades reales de los usuarios.

En conclusión, este proyecto representó mucho más que el cumplimiento de objetivos académicos. Se logró desarrollar un sistema funcional con aplicación en el ámbito del Laboratorio de Bioingeniería de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata consolidando una solución que responde a necesidades concretas. Pero, además, este proceso significó nuestra primera inmersión en un entorno de trabajo similar al profesional, donde enfrentamos desafíos propios del desarrollo de software en contextos reales, como la gestión de tiempos, adaptación a cambios, toma de decisiones y la colaboración y negociación con los demandantes del producto.

La combinación entre el desarrollo de un producto operativo y la experiencia vivida nos permitió transformar conocimientos teóricos en resultados tangibles, al mismo tiempo que fortalecimos competencias claves que trascienden lo técnico, como la organización, la comunicación efectiva y la capacidad de resolver problemas en equipo.

Cerramos esta etapa con la satisfacción de haber construido una herramienta útil, pero sobre todo con la certeza de que este proyecto marcó un antes y un después en nuestra formación. Nos permitió dar el paso de estudiantes a jóvenes profesionales, mejor preparados para afrontar nuevos retos con una visión integral de lo que implica crear soluciones de valor en el mundo real.

10. Anexos

Anexo I. Endpoints backend

Endpoint	Método HTTP	Descripción	Parámetros	Salida
/uploadVideo	POST	Recibe un video de la experiencia Speckle junto a los descriptores preseleccionados por el usuario.	Authorization: Bearer <token> (Header), video (archivo AVI), selectedDescriptors (archivo JSON)	JSON con las imágenes obtenidas en el cálculo de descriptores en base64
/clustering	POST	Recibe los métodos de clustering a aplicar junto a los descriptores seleccionados por el usuario	Authorization: Bearer <token> (Header), selectedDescriptors (array), selectedClustering (array con id y parámetros)	JSON con las imágenes obtenidas en el cálculo de los métodos de clustering en base64
/training	POST	Entrena una red neuronal a partir de matrices de descriptores y de clustering.	Authorization: Bearer <token> (Header), neuralNetworkLayers (array con neuronas, batchNorm y dropout), neuralNetworkParams (épocas, batch size, early stopping), selectedClustering (string)	Imagen de la matriz de confusión en base64
/defaultValues	GET	Obtiene los valores por defecto	Authorization: Bearer <token> (Header)	Archivo JSON con valores por defecto
/experience	POST	Guarda una experiencia para su consulta	Authorization: Bearer <token> (Header), name (string), video (objeto con nombre), selectedDescriptors (array con id y parámetros)	Mensaje de éxito
/experience/:id	GET	Obtiene una experiencia por ID	Authorization: Bearer <token> (Header), id (string)	Información de la experiencia guardada
/experience/:id	DELETE	Eliminar una experiencia por ID	Authorization: Bearer <token> (Header), id (string)	Mensaje de éxito
/experience/user/all	GET	Obtiene todas las experiencias de un usuario	Authorization: Bearer <token> (Header)	Array de las experiencias del usuario, especificando id, nombre y fecha en que se realizó
/prediction	POST	Realiza una consulta sobre una experiencia guardada	Authorization: Bearer <token> (Header), experienceId (string), video (archivo AVI)	Imagen producto de la consulta

/matrices/?type=<type>&method=<name>	POST	Permite la descarga de las matrices obtenidas	Authorization: Bearer <token> (Header),type (descriptorRaw,descriptorNormalized,clustering,prediction), method (id del descriptor, método de clustering, tensor/matrix)	Matriz seleccionada para su descarga
/dimensions	POST	Obtener alto, ancho y cantidad de frames de un video	Authorization: Bearer <token> (Header), video (archivo AVI)	Cantidad de frames, alto y ancho del video

Anexo II. Endpoints API

Endpoint	**Método HTTPS**	**Descripción**	**Parámetros**	**Salida**
/descriptores	POST	Recibe un video de la experiencia Speckle y calcula las matrices de los descriptores asociados.	x_api_key (str, Header) - Clave de autenticación. video_experiencia (UploadFile) - Archivo de video de experiencia Speckle en formato avi. datos_descriptores (str, Form) - Configuración para el cálculo de descriptores.	Archivo con: -Matrices de descriptores calculadas. -Matrices de descriptores calculadas y normalizadas -Imágenes en mapa de calor de los descriptores calculados
/clustering	POST	Recibe archivo con matrices de descriptores y calcula las matrices de los métodos de clustering asociados.	x_api_key (str, Header) - Clave de autenticación. matrices_descriptores (UploadFile) - Archivo con las matrices de descriptores. datos_clustering (str, Form) - Parámetros de los métodos de clustering. video_dimensiones (str, Form) - Parámetros con las dimensiones del video original.	Archivo con: -Matrices de clustering calculados. -Imágenes en mapa de calor de los clustering calculados.
/entrenamiento	POST	Entrena una red neuronal a partir de matrices de descriptores y de clustering.	background_tasks (BackgroundTasks) - Permite ejecutar la tarea de eliminar archivos temporales en segundo plano. x_api_key (str, Header) - Clave de autenticación. matriz_carateristicas (UploadFile) - Archivo con las matrices de descriptores y con matriz de clustering. parametros_entrenamiento (str,	Archivo con: -Modelo entrenado -Imagen de matriz de confusión

			Form) - Configuración del entrenamiento.	
/prediccion	POST	Recibe un modelo previamente entrenado y nuevas matrices de descriptores para realizar una predicción	background_tasks (BackgroundTasks) - Permite ejecutar la tarea de eliminar archivos temporales en segundo plano x_api_key (str, Header) - Clave de autenticación. modelo_entrenado (UploadFile) - Archivo con el modelo previamente entrenado. matrices_descriptores (UploadFile) - Archivo con las matrices de descriptores a analizar. video_dimensiones (str, Form) - Parámetros con las dimensiones del video original.	Archivo con: -Matriz de consulta -Tensor con probabilidades de pertenencia a cada centro de cluster -Imágen en mapa de calor de la consulta

Anexo III. Pruebas de rendimiento API

	EndPoint	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Rec KB/sec	Sent KB/sec	Avg. Bytes
Prueba 1	Descriptores	2	94674	89302	100046	5372	0,00%	0,00098	46,75	34,33	49050858
	Clustering	2	59578	57907	61250	1671,5	0,00%	0,00099	1,48	21,71	1520118
	Entrenamiento	2	92177	91646	92708	531	0,00%	0,00098	0,15	21,51	152750,5
	Consulta	2	10429	10352	10507	77,5	0,00%	0,00102	5,9	22,29	5933373
	TOTAL	8	64214	10352	100046	34117,23	0,00%	0,00361	49,98	91,14	14164274,9
Prueba 2	Descriptores	5	134314	97186	275863	70781,47	0,00%	0,01807	864,78	635,71	48998682
	Clustering	5	119184	69338	171075	34643,01	0,00%	0,02011	32,33	438,93	1646256
	Entrenamiento	5	130627	62782	217631	52072,25	0,00%	0,02281	3,53	501,77	158609,4
	Consulta	5	7994	6116	13146	2607,35	0,00%	0,04534	262,69	992,34	5933373
	TOTAL	20	98030	6116	275863	70465,42	0,00%	0,04645	643,35	1171,46	14184230,1
Prueba 3	Descriptores	8	270581	137676	520164	131515,34	0,00%	0,01536	736,45	540,22	49103034
	Clustering	8	460796	67503	1259982	456812,27	0,00%	0,00577	7,85	125,83	1394102
	Entrenamiento	8	573435	107948	1319691	463938,93	0,00%	0,00606	0,94	133,37	158272,4
	Consulta	8	11059	8639	14219	2139,02	0,00%	0,00664	38,45	145,25	5933373
	TOTAL	32	328968	8639	1319691	394599,76	0,00%	0,01874	258,85	472,58	14147195,3
Prueba 4	Descriptores	10	229592	107071	445733	127036,87	0,00%	0,02241	1073,92	788,27	49071728,4
	Clustering	10	264458	61240	443764	119639,37	0,00%	0,01533	21,63	334,67	1444571,6
	Entrenamiento	10	219894	75329	485138	136661,91	0,00%	0,01501	2,31	330,22	157665,2
	Consulta	10	9042	374	16710	3825,78	0,00%	0,02404	125,35	526,13	5340055
	TOTAL	40	180746	374	485138	149625,87	0,00%	0,04748	649,24	1197,44	14003505,1
Prueba 5	Descriptores	13	295790	92693	736263	216542,05	0,00%	0,01764	845,75	620,4	49103034
	Clustering	13	363499	96620	711463	239110,67	0,00%	0,01404	19,12	306,46	1394144,9
	Entrenamiento	13	232723	69499	765967	187322,16	0,00%	0,01448	2,24	318,52	158615,2
	Consulta	13	15256	6131	71841	17027,19	0,00%	0,01755	101,67	384,06	5933373
	TOTAL	52	226817	6131	765967	227857,37	0,00%	0,045	621,72	1135,03	14147291,8

Anexo IV. Glosario

- **API (*Application Programming Interface*)**: es un conjunto de reglas, protocolos y herramientas que permite que diferentes programas de software se comuniquen y se integren entre sí.

- **Atributos de calidad**: son características medibles de un sistema que indican cuán bien cumple con las necesidades de las partes interesadas.
- **Cámara CCD (Dispositivo de Carga Acoplada)**: es una videocámara que utiliza un sensor de luz transistorizado para capturar imágenes.
- **Mapas Autoorganizados (*SOM, Self Organizing Maps*)**: son una técnica de aprendizaje no supervisado utilizada en redes neuronales para reducir la dimensionalidad de datos complejos.
- ***JavaScript Object Notation* (*JSON*)**: es un formato de texto ligero y fácil de leer para el intercambio de datos.
- **Punto final (*Endpoint*)**: es cualquier dispositivo que está conectado a una red informática y que se utiliza para intercambiar datos con ella.
- ***Mockup***: es una representación visual de alta fidelidad de un diseño, a menudo utilizada para presentar un proyecto final a un cliente o para obtener retroalimentación.
- **Aprendizaje automático (*Machine learning*)**: Es una rama de la inteligencia artificial que se enfoca en el desarrollo de algoritmos y modelos estadísticos que permiten a las computadoras aprender de datos sin ser programadas explícitamente para realizar tareas específicas.
- **WSGI (*Web Server Gateway Interface*)**: es una especificación que define cómo los servidores web se comunican con las aplicaciones web escritas en Python.
- **Desarrollo iterativo-incremental**: es un enfoque de desarrollo de software que combina la creación de productos en fases, llamadas iteraciones, con la entrega de incrementos funcionales
- ***Camelcase***: es una convención para escribir palabras compuestas sin espacios, donde la primera letra de cada palabra (excepto la primera, en algunos casos) se escribe con mayúscula.

11. Bibliografía

I. **Auth0**. (s.f.). Auth0: secure access for everyone. But not just anyone. Recuperado de https://auth0.com/docs

II. **Base64**. (s.f.). Base64 encode & decode. Recuperado de https://www.base64decode.org/es/

III. **Express.js**. (s.f.). Express - Node.js web application framework. Recuperado de https://expressjs.com/routing.html

IV. **FastAPI**. (s.f.). FastAPI documentation. Recuperado de https://fastapi.tiangolo.com/

V. **Full Stack Open**. (s.f.). Full stack open. Recuperado de https://fullstackopen.com/es/

VI. **Géron, A.** (2023). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media. ISBN: 978-1-098-12597-4.

VII. **GitHub, Inc.** (s.f.). Acerca de GitHub y Git. Recuperado de https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

VIII. **Goodfellow, I., Bengio, Y., & Courville, A.** (2016). Deep learning. MIT Press. Recuperado de http://www.deeplearningbook.org

IX. **Guzmán, M. N., Nisenbaum, M., Cujano Ayala, E., Murialdo, S., & Meschino, G. J.** (2023). Preliminary study of the application of dynamic speckle pattern analysis for toxicants detection based on bacterial motility changes. ICYTE, Facultad de Ingeniería, Universidad Nacional de Mar del Plata - CONICET.

X. **Jain, A. K., Murty, M. N., & Flynn, P. J.** (1999). Data clustering: a review. ACM Computing Surveys, 31(3), 264-323.

XI. **JSON**. (s.f.). JSON: un formato ligero de intercambio de datos. Recuperado de https://www.json.org/json-es.html

XII. **Keras**. (s.f.). Keras API documentation. Recuperado de https://keras.io/api/

XIII. **MathWorks**. (s.f.). Matlab. Recuperado de https://www.mathworks.com/

XIV. **MathWorks**. (s.f.). Subtractive clustering (Subclust). Recuperado de https://la.mathworks.com/help/fuzzy/subclust.html#bvm9zpz-5

XV. **MongoDB Atlas**. (s.f.). Atlas database. Recuperado de https://www.mongodb.com/products/platform/atlas-database/features

XVI. **Nielsen, M. A.** (2015). Neural networks and deep learning. Determination Press.

XVII. **Node.js**. (s.f.). Node.js documentation. Recuperado de https://nodejs.org/es

XVIII. **NumPy**. (s.f.). NumPy documentation. Recuperado de https://numpy.org/doc/stable/

XIX. **Python Software Foundation**. (s.f.). Python documentation. Recuperado de https://www.python.org/

XX. **Render**. (s.f.). Deploy for free. Recuperado de https://render.com/docs/free

XXI. **Scikit-learn**. (s.f.). Scikit-learn user guide. Recuperado de https://scikit-learn.org/stable/

XXII. **SciPy**. (s.f.). SciPy documentation. Recuperado de https://docs.scipy.org/doc/

XXIII. **Vite**. (s.f.). ¿Por qué Vite? Recuperado de https://vite.dev/guide/why.html

XXIV. **Wikipedia**. (s.f.). Numba. Recuperado de https://es.wikipedia.org/wiki/Numba

XXV. **Wikipedia**. (s.f.). Speckle dinámico. Recuperado de https://es.wikipedia.org/wiki/Speckle_dinámico