

Plan Operativo LOD0 – v1.1

Clasificación Asistida de Patologías con SAM-3

1. Objetivo y alcance

El objetivo del proyecto es construir una herramienta de **clasificación asistida de patologías en imágenes de auscultación de edificios**, que:

- Reduzca drásticamente el tiempo de etiquetado manual de grandes volúmenes de imágenes.
- Permita una **segmentación jerárquica** (Niveles 1, 2 y 3) sobre un único modelo fundacional: **SAM-3**.
- Funcione en entorno **local**, aprovechando GPU, y ofrezca una **UX guiada y profesional** para usuarios técnicos pero no especialistas en IA.

El presente plan LOD0 (versión 1.1) define:

1. La **arquitectura lógica** (frontend, backend, motor de inferencia).
2. Los **requisitos de SAM-3** (software, hardware, checkpoints).
3. El **modelo de datos mínimo**.
4. Las **fases funcionales** (0 a 4) con responsabilidades BE/FE.
5. El **diseño de los endpoints de backend**.
6. Las **pantallas clave de frontend** (rutas y payloads principales).

2. Arquitectura general

2.1. Capas y responsabilidades

Se define una arquitectura en tres capas:

1. **Frontend (UX)**

Aplicación web (SPA) que ofrece:

- a. Selección de datasets
- b. Definición de prompts y ejemplares.
- c. Visualización de métricas y muestras.
- d. Validación masiva y ajustes de umbrales.

No interactúa directamente con la GPU.

2. **API / Orquestador (Backend ligero)**

Expuesto vía HTTP/REST.

Responsabilidades:

- a. Gestión de **datasets**.
- b. Gestión de **conceptos** (niveles 1, 2, 3).
- c. Creación y seguimiento de **jobs** de inferencia.
- d. Acceso a estadísticas, muestras y resultados.

Health checks y descubrimiento de capacidades.

3. **Motor de Inferencia SAM-3 (Backend pesado en GPU)**

Uno o varios workers que:

- a. Cargan **SAM-3** en memoria GPU.
- b. Ejecutan inferencia PCS por texto o por ejemplar.
- c. Optimizan el uso de GPU (batching, resolución, etc.).

Se comunican con el orquestador vía colas internas o RPC.

3. Requisitos de SAM-3

3.1. Requisitos de software

Entorno recomendado:

- Sistema operativo: Linux (Ubuntu LTS o similar).
- Python: ≥ 3.10 (idealmente 3.12 si la librería lo soporta de forma estable).
- PyTorch:
 - Versión moderna con soporte CUDA 12.x.
 - `torchvision`, `torchaudio` compatibles.
- Librerías asociadas:
 - `numpy`, `opencv-python`, `pillow`, `pycocotools` (o equivalente RLE), etc.
 - Librerías oficiales de SAM-3 (clonado del repo y `pip install -e .`).
- Manejador de colas (opcional pero recomendado):
 - Redis + Celery / RQ, o sistema equivalente para jobs de inferencia.

Se definirá un entorno dedicado, por ejemplo:

- Carpeta `envs/sam3_env/`.
- Ficheros:
 - `pyproject.toml` o `requirements.txt` con versiones fijadas.
 - `environment.yml` (si se usa Conda).

3.2. Requisitos de hardware

Configuración objetivo:

- GPU NVIDIA con ≥ 16 GB VRAM (recomendado 24 GB para margen).
- CPU con 8–16 cores.
- RAM ≥ 32 GB (ideal 64 GB para buffering de imágenes y metadatos).
- Almacenamiento ≥ 1 TB SSD (datasets + checkpoints + cachés + resultados).

Limitaciones:

Si la GPU no cumple el mínimo, el sistema sigue siendo funcional, pero se reduce batch size y se avisa al usuario de que el rendimiento será inferior.

3.3. Checkpoints y licencias

- Checkpoints de SAM-3 obtenidos desde Hugging Face / repositorio oficial.
- Requisitos:
 - Token de Hugging Face configurado (variable de entorno o fichero local de credenciales).
 - Aceptación de la licencia del modelo.
- El backend debe:
 - Descargar los pesos bajo demanda en el primer arranque.
 - Cachearlos en una ruta configurable (`checkpoints/sam3/`).
 - Registrar versión del modelo (tag/commit) en logs y en cada job.

4. Modelo de datos mínimo

Se propone un esquema conceptual (no ligado aún a una tecnología concreta: SQL/NoSQL).

4.1. Tabla **images**

- **image_id** (PK)
- **file_path** (ruta local)
- **hash** (checksum para detectar duplicados)
- **width, height**
- **Flags:**
 - **processed_level1** (bool)
 - **processed_level2** (bool)
 - **processed_level3** (bool)
- **Metadatos:**
 - **dataset_id**
 - **created_at**

4.2. Tabla **datasets**

- **dataset_id** (PK)
- **name**
- **root_path**
- **num_images**
- **status** (imported, indexing, ready)
- **created_at, created_by**

4.3. Tabla **concepts**

- **concept_id** (PK)
- **level** (1, 2, 3)
- **type** (text | exemplar)
- **name** (ej. “Fachada”, “Balcón”, “Grieta vertical”)
- **prompt_text** (para niveles 1 y 2, principalmente en inglés)
- **description** (texto libre)
- **dataset_id** (opcional: si un concepto está restringido a un dataset)
- **model_version**
- **created_at, created_by**

4.4. Tabla **examples** (ejemplos de nivel 3)

- `example_id` (PK)
- `concept_id` (Nivel 3)
- `image_id`
- `bbox` (x, y, w, h) en coordenadas de imagen.
- `comment`
- `created_at`, `created_by`

4.5. Tabla **regions** (segmentaciones detectadas)

- `region_id` (PK)
- `image_id`
- `concept_id` (con qué concepto se detectó)
- `level` (1, 2, 3)
- `mask_ref` (ruta a la máscara, RLE o similar)
- `bbox` (x, y, w, h)
- `confidence` (score [0,1])
- `model_version`
- `job_id` (job de inferencia que la generó)
- `created_at`

4.6. Tabla **evidence** (acumulación de evidencia nivel 3)

- `evidence_id` (PK)
- `region_id`
- `example_id`
- `score` (score [0,1] para ese ejemplo)
- `created_at`

A partir de **evidence** se calculará un **score acumulado** por región:

- Ejemplo de agregación probabilística:
 - $p_r = 1 - \prod_e (1 - s_{\{r,e\}})$
- Se almacenará un **aggregated_score** en:
 - Tabla **regions** o tabla derivada **region_scores**.

5. Fases funcionales

5.1. Fase 0 – Configuración y Carga del Motor de Inferencia

En la Fase 0 se aborda la preparación del motor de inferencia y la verificación del entorno antes de cualquier procesamiento masivo. En el backend, el módulo de inferencia de SAM-3 se encarga de cargar los checkpoints del modelo, fijar una resolución objetivo para las imágenes (por ejemplo, reescalarlas a un máximo de 1024 píxeles en su lado mayor) y ejecutar una o varias inferencias de prueba a modo de warm-up, de forma que se validen la carga del modelo, la compatibilidad con la GPU y la latencia base del sistema.

En paralelo, el frontend ofrece una pantalla de “Estado del sistema” que muestra, de manera clara y resumida, si el modelo SAM-3 está correctamente cargado o se encuentra en error, qué hardware de cómputo está disponible (incluyendo la presencia o no de GPU y la memoria VRAM total y libre) y qué versión concreta del modelo se está utilizando. Esta misma pantalla debe advertir explícitamente al usuario cuando no se detecta ninguna GPU o cuando la VRAM disponible es inferior a la recomendada, anticipando así posibles degradaciones de rendimiento antes de iniciar las fases de clasificación.

Backend

- **Módulo de Inferencia SAM-3:**
 - Carga del modelo (checkpoints).
 - Config de resolución objetivo (p.ej. 1024 px lado mayor).
 - Pre-cálculo de warm-up (inferencias de prueba).

Frontend (UX)

- Pantalla “**Estado del sistema**” (ver sección de pantallas):
 - Estado del modelo (cargado / en error).
 - Información de hardware (GPU, VRAM).
 - Versión del modelo.
- Avisos al usuario si:
 - No hay GPU detectada.
 - VRAM insuficiente para la configuración recomendada.

5.2. Fase 1 – Clasificación Jerárquica I (Nivel 1: Detección Masiva)

Objetivo: etiquetar imágenes completas con conceptos amplios: “Fachada”, “Cubierta”, etc.

En la Fase 1 comienza la clasificación masiva del dataset utilizando conceptos amplios como “Fachada” o “Cubierta”. El backend procesa cada imagen realizando un pretratamiento básico, extrayendo características y aplicando el mecanismo PCS de SAM-3 con los prompts definidos para este nivel. A partir de esta inferencia se generan máscaras y cajas delimitadoras que quedan registradas como regiones de Nivel 1. El proceso está orquestado mediante un sistema de *jobs* asíncronos que incorpora la información necesaria para reanudar tareas interrumpidas, gestionar estados de ejecución y aplicar configuraciones específicas como umbrales o tamaños de lote. Paralelamente, se calculan estadísticas que permiten conocer cuántas imágenes han sido clasificadas por concepto y cómo se distribuyen los niveles de confianza según los buckets derivados del umbral elegido por el usuario.

En el frontend, la pantalla dedicada a esta fase permite seleccionar el dataset y definir los prompts de Nivel 1 mediante un formulario sencillo. Los resultados se presentan de forma agregada mediante tablas y gráficos que muestran el volumen de imágenes clasificadas y la distribución de los niveles de confianza, acompañados de una galería de ejemplos visuales que ilustran el desempeño del modelo en cada rango. La interfaz proporciona además controles directos para lanzar nuevos procesos, monitorizar su progreso estimado y reanudar o cancelar ejecuciones en curso, ofreciendo al usuario una experiencia clara y operativa durante esta primera etapa de clasificación jerárquica.

Backend

- **Módulo de Inferencia – Nivel 1:**

Para cada imagen del dataset:

- Preprocesar (reescalado, normalización).
- Extraer features de imagen.
- Ejecutar PCS con lista de prompts de Nivel 1.

Almacenar en **regions** las máscaras y bboxes, nivel 1.

- **Módulo de Jobs:**

job_type = level1_inference

Parámetros:

- **dataset_id**
- Lista de **concept_id** de nivel 1.
- Config de umbrales y batch size.

Estados: pending, running, completed, failed, cancelled.

Reanudación: Cursor por **image_id** para retomar tras interrupciones.

- **Módulo de Estadísticas:**

Cálculo de N° de imágenes con al menos una región por concepto.

Distribución de regiones por buckets de confianza:

- (Max) = 0.9
- $0.9 - ((0.9 - \text{“User_Confidence”})/3)$
- $0.9 - (2*(0.9 - \text{“User_Confidence”})/3)$
- (Min) = User_Confidence”

Frontend (UX)

• Pantalla “Clasificación Nivel 1”:

Formulario:

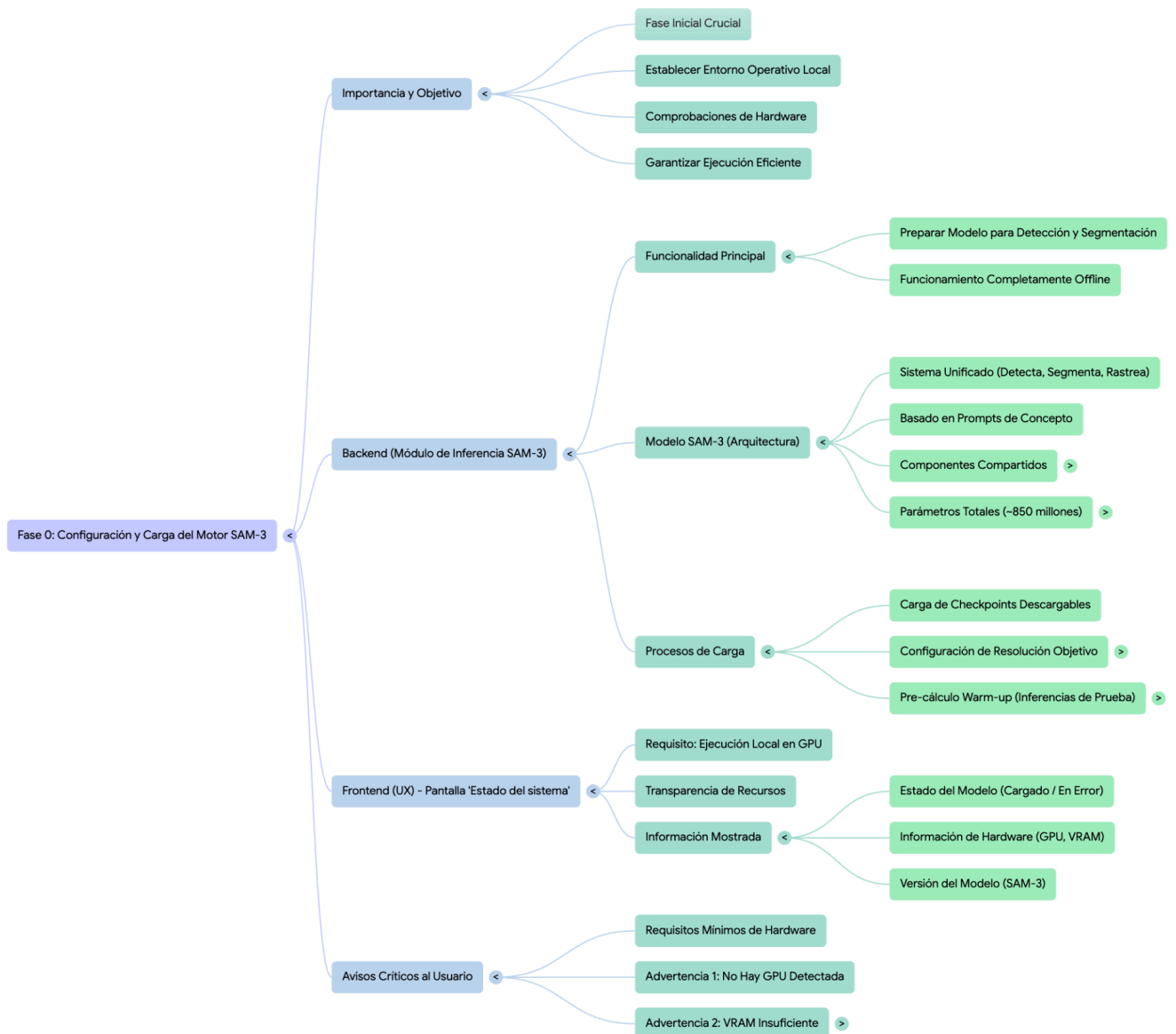
- Seleccionar **dataset**.
- Especificar prompts de nivel 1 (posiblemente preconfigurados en inglés, p.ej. “building facade”, “roof”).

Panel de resultados:

- Tabla de conceptos con N° de imágenes afectadas y Gráfico de barras por buckets de confianza
- Galería de muestras con Thumbnails representativos de cada bucket.

Controles:

- Botón “Lanzar job Nivel 1”.
- Visualización de progreso (porcentaje, ETA aproximado).
- Posibilidad de cancelar/reanudar.



5.3. Fase 2 – Clasificación Jerárquica II (Nivel 2: Subcomponentes)

Objetivo: detectar subcomponentes (aleros, balcones, cornisas, etc.) dentro de una clase de Nivel 1

En la Fase 2 el sistema profundiza en la estructura de las imágenes clasificadas previamente como pertenecientes a una clase de Nivel 1, como “Fachada”, con el objetivo de identificar subcomponentes más específicos —aleros, balcones, cornisas u otros elementos arquitectónicos relevantes. El backend ejecuta este nivel únicamente sobre las imágenes que ya cuentan con una detección válida de Nivel 1 y, si se desea, sólo sobre aquellas que superen un umbral mínimo de confianza. Para llevar a cabo esta detección más fina, se generan recortes a partir de las máscaras de fachada y se aplica nuevamente PCS, ahora con los prompts específicos de Nivel 2. Los resultados se almacenan como nuevas regiones asociadas al nivel jerárquico correspondiente, y el módulo de estadísticas presenta resúmenes calculados en relación con las regiones de fachada procesadas y las detecciones obtenidas.

En paralelo, el frontend ofrece una pantalla dedicada a la clasificación de Nivel 2 que permite seleccionar el dataset, elegir la clase de Nivel 1 sobre la que se desea refinar la búsqueda y definir los conceptos de Nivel 2 que se quieren detectar, junto con el umbral mínimo de confianza heredado del nivel anterior. La interfaz muestra una tabla con el número de imágenes y regiones detectadas por cada subcomponente, su distribución de confianza y una galería de ejemplos visuales en la que se superponen las máscaras de fachada con las nuevas regiones identificadas, facilitando así la validación rápida del proceso.

Backend

- **Módulo de Inferencia – Nivel 2:**

Scope:

- Solo imágenes con regiones de Nivel 1 de un concepto dado, p.ej. “Fachada”.
- Opcionalmente, solo aquellas regiones con `confidence_level1 ≥ threshold`.

Estrategia: Generar crops a partir de las máscaras de fachada y ejecutar PCS de Nivel 2 sobre esos crops.

La elección se tomará en prototipo, pero el modelo de datos ya soporta ambas (siempre se guardan masks y bboxes).

Los resultados se almacenan en `regions` con `level = 2`.

- **Módulo de Jobs:**

`job_type = level2_inference`

Parámetros:

- `dataset_id`
- `parent_concept_id` (ej. “Fachada”)
- Lista de `concept_id` de Nivel 2 (ej. “Balcón”, “Alero”).
- `min_confidence_level1` para filtrar fachada.

- **Módulo de Estadísticas:**

Análogo al de Nivel 1, pero, los totales se expresan respecto a:

- N° de regiones de Nivel 1 procesadas.
- N° de regiones de Nivel 2 detectadas.

Frontend (UX)

- Pantalla “Clasificación Nivel 2”:

Formulario:

- Seleccionar dataset.
- Seleccionar concepto de Nivel 1 (p.ej. “Fachada”).
- Especificar prompts de Nivel 2 (ej. “balcony”, “eaves”, “cornice”).
- Umbral mínimo de confianza de Nivel 1.

Panel de resultados:

- Tabla de conceptos de Nivel 2 con:
 - N° de imágenes y regiones donde aparecen.
 - Distribución de confianza.
- Muestras visuales:
 - Imagen con la máscara de fachada y las regiones de Nivel 2 superpuestas.



5.4. Fase 3 – Clasificación Jerárquica III (Nivel 3: Patologías por Ejemplar)

Objetivo: localizar patologías específicas (grietas, humedades, desprendimientos...) a partir de **imágenes de ejemplo** marcadas por el usuario.

En la Fase 3 el sistema introduce la capacidad más especializada del flujo: la identificación de patologías concretas a partir de ejemplos proporcionados por el usuario. Para ello, el backend permite definir conceptos de Nivel 3 basados en ejemplares, creando una categoría de patología y asociándole una o varias imágenes del propio dataset en las que el usuario delimita la región representativa mediante un recorte. A partir de estos ejemplos, la inferencia se realiza aplicando PCS de manera focalizada sobre un subconjunto del dataset —habitualmente restringido a clases detectadas en niveles previos— con el fin de localizar regiones similares. Cada coincidencia identificada queda registrada como una región de Nivel 3, junto con su score individual por ejemplo, lo que alimenta un módulo de agregación encargado de combinar estas evidencias en un score acumulado para cada región. Dicho módulo genera estadísticas globales que permiten observar la distribución de confianza del concepto y anticipar el grado de fiabilidad alcanzado.

En el frontend, la pantalla dedicada a esta fase guía al usuario a través de un flujo progresivo. Primero se define la patología, asignándole un nombre descriptivo. Después se añaden los ejemplos mediante un visor que permite elegir una imagen del dataset y marcar visualmente la zona de interés. Con los ejemplos introducidos, el usuario decide el ámbito de búsqueda —el conjunto completo de imágenes o un subconjunto filtrado por niveles anteriores— y lanza el proceso de inferencia. Los resultados iniciales se muestran como una galería de imágenes con sus respectivas máscaras y puntuaciones, facilitando la validación temprana y permitiendo señalar tanto aciertos evidentes como falsos positivos, lo que constituye la base para la etapa posterior de consolidación y automatización del etiquetado.

Backend

- **Módulo de Conceptos de Nivel 3:**

Creación de un `concept_id` con `level = 3`, `type = exemplar`.

Asociar uno o varios `examples`:

- Imagen del dataset.
- Bounding box que delimita la patología de referencia.

- **Módulo de Inferencia – Nivel 3 (Exemplar PCS):**

Para cada ejemplo:

- Ejecutar PCS en un subconjunto de imágenes, Inicialmente, quizá sólo en una clase de Nivel 2 (ej. “Aleros”) para reducir el espacio.

Para cada imagen:

- Obtener candidatos (regiones) con sus scores.
- Insertar en `regions` (`level = 3`).
- Insertar en `evidence` una fila por (`region`, `example`).

- **Módulo de Jobs:**

`job_type = level3_inference`

Parámetros:

- `dataset_id`
- `concept_id` de Nivel 3.
- Lista de `example_id` activos.
- Scope: todas las imágenes, o sólo las que cumplan un filtro (ej. fachada + balcón).

- **Módulo de Agregación de Evidencia:**

Cálculo del score acumulado por región `S_r` o `p_r`.

Actualización de `regions.aggregated_score`.

Exposición de estadísticas agregadas por concepto:

- Histograma de `aggregated_score`.
- Número de regiones por rangos (ej. ≥ 0.85 , $0.5-0.85$, < 0.5).

Frontend (UX)

- **Pantalla “Búsqueda por Ejemplo (Nivel 3)”:**

Paso 1: Definir patología

Formulario:

- Nombre de la patología (ej. “Grieta vertical en fachada”).

Paso 2: Añadir ejemplos

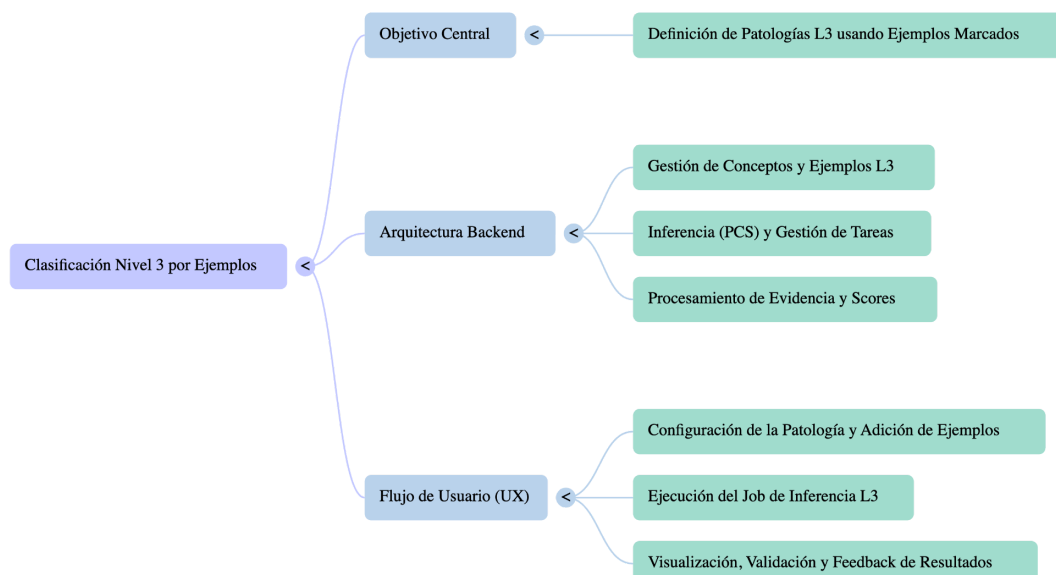
- Widget de selección de imagen del dataset.
- Herramienta para dibujar bounding box sobre la patología.
- Botón “Guardar ejemplo”.
- Lista de ejemplos actuales (miniaturas).

Paso 3: Lanzar búsqueda

- Selección del scope, todo el dataset o filtrado (por Nivel 1/2).
- Botón “Lanzar job Nivel 3”.

Paso 4: Visualización inicial de resultados

- Muestra de imágenes con máscaras y scores.
- Posibilidad de marcar ejemplos como “claramente correctos” o “falsos positivos” para análisis posterior.



5.5. Fase 4 – Automatización, Evidencia y Validación Masiva

Objetivo: aplicar reglas automáticas de decisión sobre los scores acumulados, para etiquetar masivamente con control.

En la Fase 4 el sistema consolida todo el conocimiento acumulado en los niveles anteriores para automatizar la clasificación masiva de patologías con un grado elevado de control por parte del usuario. El backend aplica reglas de decisión basadas en los scores agregados de Nivel 3, permitiendo establecer umbrales de aceptación y rechazo, así como un rango intermedio destinado a revisión manual. A partir de estos criterios, cada región pasa a un estado coherente con su nivel de confianza —aceptada automáticamente, descartada o pendiente de revisión— y se generan informes que indican la cobertura alcanzada y el volumen de elementos aún sin clasificar. Paralelamente, el sistema prepara muestras estratificadas de la zona gris y ofrece capacidades de exportación en múltiples formatos, desde listados simples hasta datasets estructurados aptos para futuros entrenamientos.

En el frontend, la interfaz organiza este proceso en una pantalla específica donde el usuario puede configurar los umbrales por patología mediante controles intuitivos y visualizar de inmediato el impacto de esas decisiones a través de gráficos y resúmenes numéricos. La zona gris se presenta como una galería de imágenes seleccionadas estratégicamente para facilitar una validación rápida y representativa, con opciones para aceptar en bloque o descartar elementos identificados como falsos positivos. Finalmente, la interfaz permite ejecutar acciones globales —como aplicar las reglas al conjunto completo del dataset o recalcular scores tras incorporar nuevos ejemplos— de forma coherente y eficiente, cerrando así el ciclo de clasificación asistida con un alto grado de automatización y control técnico.

Backend

- **Módulo de Reglas de Decisión:**

Parámetros por patología (`concept_id` Nivel 3):

- `threshold_accept` (ej. ≥ 0.85).
- `threshold_reject` (ej. ≤ 0.2).
- Tamaño de muestra para revisión manual (ej. 100 imágenes de la zona gris).

Marcar regiones como:

- `status = accepted_auto`
- `status = rejected_auto`
- `status = pending_review`

Generar:

- Lista de imágenes para muestreo.
- Informe de cobertura (cuántas imágenes/regiones quedan sin clasificar).

- **Módulo de Muestreo y Exportación:**

Muestreo estratificado por rangos de score.

Exportación de resultados:

- CSV/JSON con paths, bboxes, scores, estado.
- Opcionalmente, dataset listo para entrenamiento futuro.

Frontend (UX)

- Pantalla “Validación Masiva y Reglas”:

Configuración por patología:

- Sliders o campos para `threshold_accept` y `threshold_reject`.

Panel de resumen:

- N° de regiones en cada categoría (accepted, rejected, pending).
- Histograma de scores.

Módulo de muestreo:

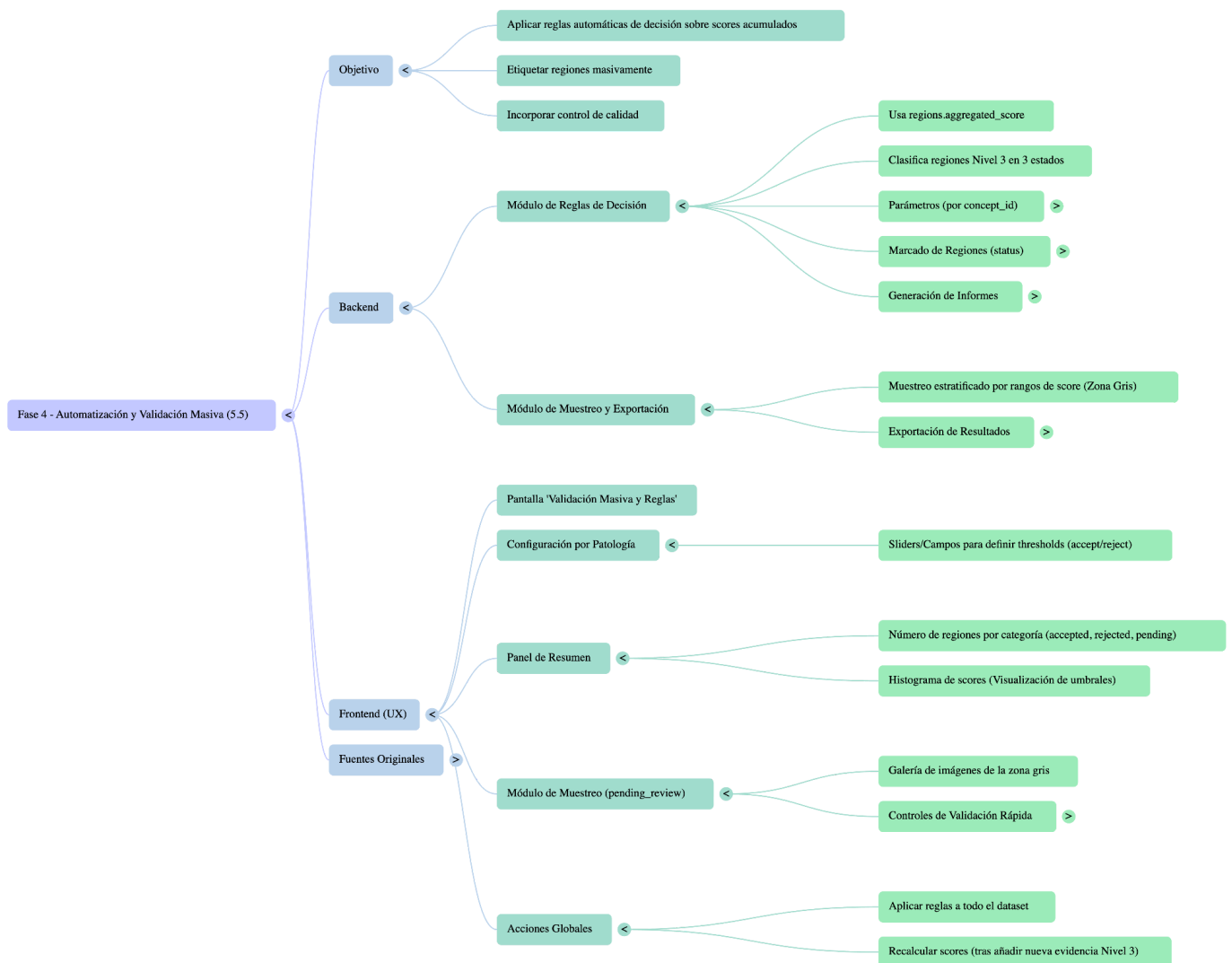
- Galería de imágenes de la zona gris.

Controles:

- “Aceptar todas en esta muestra”.
- “Marcar como falsos positivos”.

Acciones globales:

- “Aplicar reglas a todo el dataset”.
- “Recalcular scores tras añadir nuevos ejemplos”.



6. API de Backend: Endpoints y payloads

A continuación se define un diseño inicial de endpoints REST. Es un diseño conceptual: nombres y estructuras podrán ajustarse, pero sirven como contrato FE-BE.

6.1. Convenciones generales

Prefijo base: `/api/v1`.

Formato: JSON.

Autenticación: a definir (token, basic, etc.) – fuera del alcance de este LOD0.

Todos los endpoints de jobs son **asíncronos**: devuelven un `job_id`.

La API de backend establece el contrato formal entre la interfaz de usuario y el motor de procesamiento, definiendo cómo se gestionan datasets, conceptos, ejemplos, trabajos de inferencia y resultados. Todo el servicio se organiza bajo un prefijo único (`/api/v1`), utiliza JSON como formato estándar y expone operaciones asíncronas para los procesos de análisis, que devuelven siempre un identificador de *job* para su seguimiento.

El sistema comienza con un endpoint de salud que permite conocer el estado del modelo SAM-3, la disponibilidad de GPU y la configuración del entorno, lo que habilita al frontend para verificar que la infraestructura está en condiciones antes de iniciar cualquier operación pesada. La gestión de datasets se articula mediante endpoints para registrar nuevas rutas locales y recuperar los conjuntos disponibles, preparando así el espacio de trabajo para las fases de clasificación.

El módulo de conceptos permite crear categorías de análisis en cualquiera de los niveles jerárquicos, desde conceptos amplios definidos por texto hasta patologías basadas en ejemplos. Para estas últimas, existe la capacidad de adjuntar imágenes de referencia y bounding boxes que servirán de guía al modelo durante la búsqueda basada en similitud.

Las tareas de clasificación se orquestan mediante endpoints específicos para lanzar *jobs* de Nivel 1, Nivel 2 y Nivel 3. Cada invocación describe el dataset sobre el que se trabajará, los conceptos involucrados, los ejemplos cuando corresponda y el ámbito exacto de búsqueda. Una vez creados, estos trabajos pueden consultarse en cualquier momento para conocer su estado y su progreso.

La API también proporciona mecanismos para extraer estadísticas relevantes, ya sea sobre la detección masiva del Nivel 1 o sobre las regiones identificadas en niveles posteriores. Estas estadísticas incluyen distribuciones de confianza y permiten visualizar muestras representativas filtradas por concepto, nivel o rango de puntuación. Finalmente, el módulo de reglas ofrece una vía para automatizar decisiones basadas en scores acumulados, aplicando umbrales de aceptación y rechazo y generando muestras de revisión para garantizar la calidad del etiquetado. Con ello, el backend proporciona un marco robusto y coherente que soporta todo el flujo de clasificación asistida.

6.2. Sistema y configuración

GET /api/v1/health

- Descripción: estado del sistema y capacidades.
- Respuesta (ejemplo):

```
{
  "status": "ok",
  "sam3_model_loaded": true,
  "sam3_model_version": "sam3-large-v1",
  "gpu": {
    "available": true,
    "name": "NVIDIA RTX 4090",
    "total_vram_gb": 24,
    "free_vram_gb": 18
  },
  "env": {
    "python_version": "3.12.1",
    "pytorch_version": "2.3.0"
  }
}
```

6.3. Datasets

POST /api/v1/datasets

- Descripción: registrar un nuevo dataset local.
- Request:

```
{
  "name": "Auscultacion_2025_Q1",
  "root_path":
  "D:/datasets/auscultacion_2025_q1"
}
```

- Respuesta:

```
{
  "dataset_id": "ds_00123",
  "num_images": 485732,
  "status": "ready"
}
```

GET /api/v1/datasets

- Descripción: listar datasets.
- Respuesta (simplificada):

```
[
  {
    "dataset_id": "ds_00123",
    "name": "Auscultacion_2025_Q1",
    "num_images": 485732,
    "status": "ready"
  }
]
```

6.4. Conceptos y ejemplos

POST /api/v1/concepts

- Descripción: crear un concepto (nivel 1, 2 o 3).
- Request (ejemplo Nivel 1):

```
{
  "level": 1,
  "type": "text",
  "name": "Fachada",
  "prompt_text": "building facade",
  "dataset_id": "ds_00123"
}
```

- Respuesta:

```
{
  "concept_id": "c_0001",
  "level": 1,
  "name": "Fachada"
}
```

POST

/api/v1/concepts/{concept_id}/examples

- Descripción: añadir un ejemplo a un concepto de Nivel 3.

- Request:

```
{
  "image_id": "img_123456",
  "bbox": {
    "x": 230,
    "y": 140,
    "w": 120,
    "h": 80
  },
  "comment": "Grieta vertical muy clara"
}
```

- Respuesta:

```
{
  "example_id": "e_0456",
  "concept_id": "c_0030"
}
```

GET /api/v1/concepts/{concept_id}

- Descripción: obtener detalle de un concepto (incluyendo ejemplos si Nivel 3).

6.5. Jobs de inferencia

POST /api/v1/jobs/level1

- Descripción: lanzar un job de Nivel 1 sobre un dataset.

- Request:

```
{
  "dataset_id": "ds_00123",
  "concept_ids": ["c_0001", "c_0002"],
  "batch_size": 8,
  "max_images": null
}
```

- Respuesta:

```
{
  "job_id": "job_l1_0001",
  "status": "pending"
}
```

POST /api/v1/jobs/level2

- Descripción: lanzar un job de Nivel 2 dentro de un concepto padre de Nivel 1.

- Request:

```
{
  "dataset_id": "ds_00123",
  "parent_concept_id": "c_0001",
  "concept_ids": ["c_0101", "c_0102"],
  "min_confidence_level1": 0.7
}
```

POST /api/v1/jobs/level3

- Descripción: lanzar un job de Nivel 3 (búsqueda por ejemplo).

- Request:

```
{
  "dataset_id": "ds_00123",
  "concept_id": "c_0200",
  "example_ids": ["e_0456", "e_0457"],
  "scope": {
    "level1_concepts": ["c_0001"],
    "level2_concepts": ["c_0101"]
  }
}
```

GET /api/v1/jobs/{job_id}

- Descripción: consultar estado del job.

- Respuesta:

```
{
  "job_id": "job_l1_0001",
  "type": "level1_inference",
  "status": "running",
  "progress": {
    "processed_images": 15000,
    "total_images": 485732
  },
  "started_at": "2025-12-11T10:35:00",
  "finished_at": null
}
```

6.6. Estadísticas, muestras y evidencia

GET /api/v1/stats/level1

- Query params:

- dataset_id
- concept_id

- Respuesta:

```
{
  "dataset_id": "ds_00123",
  "concept_id": "c_0001",
  "num_images_with_regions": 50000,
  "confidence_buckets": [
    { "min": 0.9, "max": 1.0, "num_images": 1000 },
    { "min": 0.7, "max": 0.9, "num_images": 5000 },
    { "min": 0.6, "max": 0.7, "num_images": 44000 }
  ]
}
```

GET /api/v1/samples

- Query params:

- concept_id
- level
- bucket_min
- bucket_max
- limit

- Respuesta: lista de imágenes y regiones (simplificada).

```
{
  "samples": [
    {
      "image_id": "img_123456",
      "file_path": "D:/datasets/.../img_123456.jpg",
      "regions": [
        {
          "region_id": "r_987",
          "bbox": { "x": 200, "y": 100, "w": 300, "h": 200 },
          "confidence": 0.92
        }
      ]
    }
  ]
}
```

POST /api/v1/rules/apply

- Descripción: aplicar reglas de decisión para un concepto de Nivel 3.

- Request:

```
{
  "concept_id": "c_0200",
  "threshold_accept": 0.85,
  "threshold_reject": 0.2,
  "sample_size_pending": 100
}
```

- Respuesta:

```
{
  "concept_id": "c_0200",
  "num_accepted_auto": 1200,
  "num_rejected_auto": 5000,
  "num_pending_review": 800,
  "sample_for_review": [
    { "image_id": "img_111", "region_id": "r_2001" },
    { "image_id": "img_222", "region_id": "r_2002" }
  ]
}
```

7. Pantallas clave de Frontend (rutas y endpoints asociados)

El frontend se articula en un conjunto de pantallas diseñadas para guiar al usuario a lo largo de todo el proceso de clasificación, desde la verificación del entorno hasta la validación masiva de resultados. La interfaz comienza con una vista de estado del sistema, donde se informa de la disponibilidad del modelo SAM-3, la salud de la GPU y cualquier advertencia relacionada con limitaciones de hardware. Esta primera pantalla actúa como punto de control previo a cualquier operación intensiva.

A continuación, la sección de gestión de datasets permite registrar nuevas colecciones de imágenes y consultar las ya existentes. El usuario puede introducir la ruta local del dataset y visualizar inmediatamente cuántas imágenes contiene y en qué estado se encuentra, lo que establece la base de trabajo para las fases posteriores.

La pantalla de clasificación de Nivel 1 habilita el lanzamiento de procesos masivos de detección de conceptos generales. Desde ella se selecciona el dataset, se configuran los prompts iniciales y se supervisa el progreso del job en ejecución. Los resultados se presentan mediante gráficos y una galería de ejemplos que muestran la calidad de la detección por rangos de confianza.

Para el Nivel 2, la interfaz se centra en refinamientos locales dentro de una clase de Nivel 1. El usuario elige previamente un concepto amplio —por ejemplo, fachada— y define los subcomponentes que desea detectar, disponiendo también de un umbral de confianza que acota las regiones válidas. La vista presenta los resultados de forma visual, superponiendo detecciones específicas sobre la región original de fachada.

La pantalla de Nivel 3 introduce el flujo de búsqueda por ejemplo. En ella se crean patologías, se añaden ejemplares mediante recortes manuales en imágenes del dataset y se ejecutan búsquedas basadas en similitud. Los resultados iniciales se presentan como máscaras con sus correspondientes puntuaciones, proporcionando un primer control de calidad.

Finalmente, la sección de validación masiva permite aplicar reglas automáticas y revisar de forma eficiente los casos dudosos. Aquí se configuran los umbrales de aceptación y rechazo, se visualiza un resumen global del impacto de esas reglas y se trabaja sobre una muestra de la zona gris para confirmar o descartar detecciones. Esta fase culmina el proceso combinando automatización con control manual allí donde es necesario.

A nivel LOD0 +1, se propone el siguiente mapa de pantallas:

7.1. **/system/status** – Estado del sistema

- Objetivo: mostrar salud del sistema y capacidades.

Endpoints:

- `GET /api/v1/health`

Elementos:

- Card “Modelo SAM-3”:
 - Versión, estado.
- Card “GPU”:
 - Nombre, VRAM total/libre.
- Alertas:
 - “GPU no detectada”.
 - “VRAM inferior a la recomendada”.

7.2. **/datasets** – Gestión de datasets

- Objetivo: registrar y listar datasets.

Endpoints:

- `GET /api/v1/datasets`
- `POST /api/v1/datasets`

Elementos:

- Tabla de datasets.
- Formulario para alta de nuevo dataset (ruta local + nombre).

7.3. **/classification/level1** – Clasificación Nivel 1

- Objetivo: lanzar y supervisar jobs de Nivel 1.

Endpoints:

- `GET /api/v1/datasets`
- `POST /api/v1/concepts` (crear “Fachada”, “Cubierta”... si no existen).
- `POST /api/v1/jobs/level1`
- `GET /api/v1/jobs/{job_id}`
- `GET /api/v1/stats/level1`
- `GET /api/v1/samples`

Elementos:

- Selector de dataset.

- Listado de prompts activos de Nivel 1 (conceptos).
- Botón “Lanzar clasificación”.
- Widget de progreso global.
- Gráficos por concepto (barras por bucket de confianza).
- Galería de muestras.

7.4. **/classification/level2** – Clasificación Nivel 2

- Objetivo: refinar dentro de una clase de Nivel 1 (aleros, balcones, etc.).

Endpoints:

- GET /api/v1/datasets
- GET /api/v1/concepts?level=1
- POST /api/v1/concepts (crear conceptos Nivel 2 si no existen).
- POST /api/v1/jobs/level2
- GET /api/v1/jobs/{job_id}
- GET /api/v1/samples

Elementos:

- Selector de dataset.
- Selector de concepto de Nivel 1 (ej. “Fachada”).
- Formulario para prompts de Nivel 2.
- Umbral mínimo de confianza de Nivel 1.
- Panel de resultados con muestras superpuestas.

7.5. **/classification/level3** – Búsqueda por Ejemplo (Patologías)

- Objetivo: definir patologías y lanzar búsquedas por ejemplar.

Endpoints:

- POST /api/v1/concepts (Nivel 3).
- POST /api/v1/concepts/{concept_id}/examples
- GET /api/v1/concepts/{concept_id}
- POST /api/v1/jobs/level3
- GET /api/v1/jobs/{job_id}
- GET /api/v1/samples

Elementos:

- Formulario de creación de patología.
- Selector de imágenes del dataset.
- Herramienta de marcado (bounding box).
- Lista de ejemplos.
- Panel de resultados con máscaras y scores iniciales.

7.6. **/validation/bulk** – Validación masiva y reglas

- Objetivo: aplicar reglas de decisión y revisar muestras.

Endpoints:

- GET /api/v1/concepts?level=3
- GET /api/v1/stats/level3 (endpoint futuro a añadir, análogo a stats/level1).
- POST /api/v1/rules/apply
- GET /api/v1/samples (para zona gris).

Elementos:

- Selector de patología (concepto Nivel 3).
- Sliders para threshold_accept y threshold_reject.
- Panel de resumen (nº aceptados, rechazados, pendientes).
- Galería de muestra para revisión manual.

Acciones:

- Aceptar lote.
- Marcar falsos positivos.