ISSSTELEON

Especificación de Diseño del Sistema ISSSTELEON

Versión documental de trabajo v01

Tabla de contenido

1. Introducción 2
1.1 Propósito del documento 3
1.2 Alcance de la especificación 3
1.3 Convenciones y terminología 3
2. Descripción general del sistema 3
2.1 Propósito general de ISSSTELEON 3
2.2 Alcance funcional identificado 3
2.3 Dominios de información: CaCu, CaMa y TEA 4
2.4 Capacidades principales: ingesta, procesamiento, indicadores y consulta de información 4
3. Arquitectura general 5
3.1 Visión arquitectónica 5
3.2 Tecnologías y componentes principales 5
3.3 Responsabilidades de los componentes 5
3.4 Relaciones e integración arquitectónica 6
3.5 Flujos principales de procesamiento de información 7
4. Diseño de componentes 8
4.1 WebApp 8
4.2 WebApi 8
4.3 Ingesta, validación y persistencia 9
4.4 Indicadores y orquestación 11
4.5 Bibliotecas y componentes de soporte 14
5. Datos e interfaces 14
5.1 Persistencia y dominios de información 14
5.2 Interfaces HTTP y comunicación entre componentes 15
5.3 Configuración de conexiones y almacenamiento 16
6. Infraestructura, configuración y despliegue 16
6.1 Entorno de desarrollo 16
6.2 Arquitectura de distribución y contenedores 17
6.3 Entorno de pruebas 18
6.4 Tecnologías, versiones y configuración de ejecución 18
6.5 Controles de seguridad y consideraciones técnicas 19
7. Referencias 19
7.1 Documentación técnica y fuentes del sistema 20
7.2 Documentación de instalación y operación 20
8. Glosario 20

1. Introducción

1.1 Propósito del documento

La presente especificación describe el diseño técnico general del sistema ISSSTELEON y establece una referencia común para comprender sus componentes, responsabilidades y relaciones. Está dirigida al personal técnico e institucional que participe en la revisión, evolución, integración y administración de la plataforma.

1.2 Alcance de la especificación

El documento aborda la organización arquitectónica de ISSSTELEON, sus componentes de presentación y servicios, el tratamiento de información, la persistencia y las interfaces entre sus partes. También comprende la infraestructura y la configuración previstas para apoyar su ejecución y distribución. Los capítulos 2 y 3 presentan el sistema y su arquitectura; los capítulos 4 y 5 desarrollan el diseño de componentes, datos e interfaces; el capítulo 6 describe infraestructura y configuración, y los capítulos 7 y 8 reúnen las referencias y el glosario.

La especificación se refiere a la entrega preliminar y describe el diseño de las capacidades comprendidas en ella. La disponibilidad de datos, conexiones y configuraciones depende del entorno en que se utilice el sistema.

1.3 Convenciones y terminología

Los nombres de los componentes se conservan conforme a su denominación técnica, incluidos **Persistance** y **Orquestacion**. Una *interfaz* es el punto mediante el cual dos componentes intercambian solicitudes o información; una *API* es una interfaz de servicios accesible mediante HTTP. Una *biblioteca* agrupa funcionalidades utilizadas por otros componentes y no implica por sí misma un servicio independiente. En este documento, *dominio* identifica un área de información del sistema y *gobernanza de datos* agrupa las funciones y estructuras destinadas a organizar y administrar información relacionada con los datos. La denominación `governance` se conserva cuando identifica una configuración o ruta técnica de gobernanza. El *trabajador de indicadores* es el proceso separado que atiende solicitudes de recálculo.

2. Descripción general del sistema

2.1 Propósito general de ISSSTELEON

ISSSTELEON organiza capacidades para incorporar, procesar y consultar información relacionada con los dominios CaCu, CaMa y TEA. La plataforma articula una interfaz web, servicios de aplicación y bibliotecas especializadas para apoyar la gestión de datos y la consulta de indicadores.

El proyecto se orienta al aprovechamiento analítico de información clínica y epidemiológica mediante su integración, procesamiento e interpretación. La consulta de indicadores y la visualización de información buscan apoyar la comprensión de los procesos de atención y la toma de decisiones institucionales. El alcance funcional contempla, además, el análisis de riesgos clínicos y el estudio de patrones neurofisiológicos, conforme a los objetivos particulares de cada dominio. La utilidad de estos análisis depende de la calidad, integridad y representatividad de la información disponible, así como de la validación adecuada a su finalidad.

2.2 Alcance funcional identificado

La plataforma dispone de una organización funcional que comprende conexión de fuentes, preparación y validación de datos, persistencia, procesamiento de indicadores y consulta mediante la aplicación web. Incluye espacios y servicios relacionados con gobernanza de datos y reportes. Estas capacidades se articulan mediante componentes especializados; su tratamiento detallado y sus interfaces se describen en los capítulos correspondientes.

La incorporación de información contempla el uso de conectores y el mapeo de campos de origen a campos de destino. Para fuentes documentales, la arquitectura incorpora recursos de extracción de contenido. La validación puede solicitarse antes de la carga; al cargar, el servicio vuelve a leer y validar las fuentes y procesa las filas aceptadas mediante repositorios de persistencia.

2.3 Dominios de información: CaCu, CaMa y TEA

El diseño organiza información y conexiones de persistencia para los dominios CaCu, CaMa y TEA. Adicionalmente, contempla una configuración de persistencia para gobernanza. Esta separación permite identificar el ámbito de las operaciones y organizar los recursos de datos por dominio, sin imponer una distribución física específica de las bases de datos. CaCu corresponde a cáncer cervicouterino; CaMa, a cáncer de mama; y TEA, a trastorno del espectro autista. Son dominios funcionales de la plataforma.

Los objetivos funcionales del proyecto sitúan cada dominio en un ámbito clínico específico:

- **CaCu:** integrar información del proceso de atención desde el tamizaje mediante citología y pruebas de VPH hasta el diagnóstico, tratamiento y seguimiento. Este contexto comprende el análisis de la progresión de lesiones y de la oportunidad de atención, considerando la relación temporal entre los eventos clínicos.
- **CaMa:** relacionar los resultados de mastografías clasificadas como BI-RADS 4, 5 y 6 con antecedentes y datos sociodemográficos. El propósito analítico incluye caracterizar la información clínica disponible y apoyar el análisis del riesgo asociado a este ámbito.
- **TEA:** reunir información clínica y registros electroencefalográficos de población pediátrica con diagnóstico de trastorno del espectro autista. El propósito comprende el estudio de alteraciones en el EEG y su relación con los hallazgos clínicos mediante análisis neurofisiológico.

Estos objetivos describen el ámbito de información y análisis de cada dominio. Las capacidades de integración, cálculo y presentación se articulan conforme a los componentes y recorridos descritos en esta especificación; el alcance de un análisis concreto depende de las fuentes y definiciones aplicables.

Los componentes comunes proporcionan estructuras y mecanismos que pueden utilizarse en los distintos ámbitos. Las entidades, reglas y recorridos particulares se definen en sus respectivos componentes y no se presumen idénticos entre dominios.

2.4 Capacidades principales: ingesta, procesamiento, indicadores y consulta de información

La ingesta comprende la selección de fuentes, el mapeo de campos, la validación y la carga de información. El diseño incorpora conectores para el acceso a fuentes y una biblioteca de extracción para el tratamiento de documentos. La persistencia utiliza modelos, sesiones y repositorios para organizar el acceso a los datos.

Para indicadores, el trabajador de indicadores coordina ciclos de cálculo con bibliotecas especializadas y repositorios. La API proporciona interfaces para consultar figuras de indicadores y la aplicación web incorpora componentes para solicitar y presentar esa información. La biblioteca

models aporta interfaces y dependencias declaradas para funciones predictivas; estas declaraciones no establecen un flujo de predicción operativo.

Los indicadores permiten organizar la consulta de información clínica y de oportunidad de atención según el ámbito y la medida correspondiente. Las figuras presentan los resultados disponibles para facilitar su interpretación en la aplicación. Su significado depende de la definición de cada indicador, de la población y del periodo considerados y de los datos utilizados en su cálculo. El alcance del proyecto contempla una visión analítica de los tres dominios, cuya presentación se apoya en las interfaces de consulta descritas en este documento.

La gobernanza de datos forma parte de la organización funcional y cuenta con rutas de aplicación y servicios, además de configuración de persistencia propia. Esta especificación recoge su ubicación arquitectónica y la separación de su conexión de datos.

3. Arquitectura general

3.1 Visión arquitectónica

ISSSTELEON se organiza en capas lógicas que separan la interacción con el usuario, la atención de solicitudes y el tratamiento de información:

- **Presentación:** aplicación web que organiza la navegación, la conexión de fuentes y la consulta de información.
- **Servicios de aplicación:** API que expone interfaces HTTP y coordina operaciones de importación, datos, gobernanza e indicadores.
- **Procesamiento y orquestación:** funciones de carga y un trabajador que coordina ciclos de indicadores.
- **Modelos de dominio y persistencia:** entidades, sesiones y repositorios utilizados para representar y almacenar información.
- **Bibliotecas especializadas:** recursos de extracción, cálculo y representación de indicadores, y soporte para modelos.

Estas capas expresan responsabilidades lógicas. Las bibliotecas se integran en los procesos que las utilizan y no representan necesariamente servicios o contenedores independientes.

3.2 Tecnologías y componentes principales

La aplicación WebApp utiliza Angular y TypeScript. WebApi está basada en Python y Flask; los componentes de dominio y persistencia emplean SQLModel y herramientas de acceso a datos y migración. Orquestacion utiliza Apache Hamilton para organizar el ciclo de procesamiento, mientras que indicadores_kit incorpora Plotly para la representación de figuras. El conjunto incluye configuraciones de contenedores Docker para la aplicación web, la API y el proceso de indicadores.

3.3 Responsabilidades de los componentes

Tabla 1. Responsabilidades de los componentes.

Componente	Responsabilidad principal
WebApp	Proporciona la interfaz web y la navegación para conexión de datos, gobernanza, bitácora e indicadores.
WebApi	Expone servicios HTTP y coordina las operaciones de importación, acceso a datos, gobernanza y reportes. Incluye un trabajador separado para recálculo de indicadores.

Componente	Responsabilidad principal
Core	Reúne modelos de dominio compartidos; incluye entidades de información como el modelo de paciente de CaCu.
Persistance	Organiza motores, sesiones y repositorios para el acceso a datos; incluye recursos de migración.
Orquestacion	Coordina ciclos de procesamiento de indicadores y dispone de una interfaz de línea de comandos para su ejecución.
Extraction	Proporciona recursos de extracción de contenido; incluye lectura de documentos PDF con opción de reconocimiento óptico de caracteres.
indicadores_kit	Agrupa funciones de procesamiento y representación de indicadores, incluida la generación de figuras.
models	Agrupa interfaces y soporte declarado para modelos en los dominios del sistema.
dist	Reúne configuración y recursos destinados a la distribución del sistema.

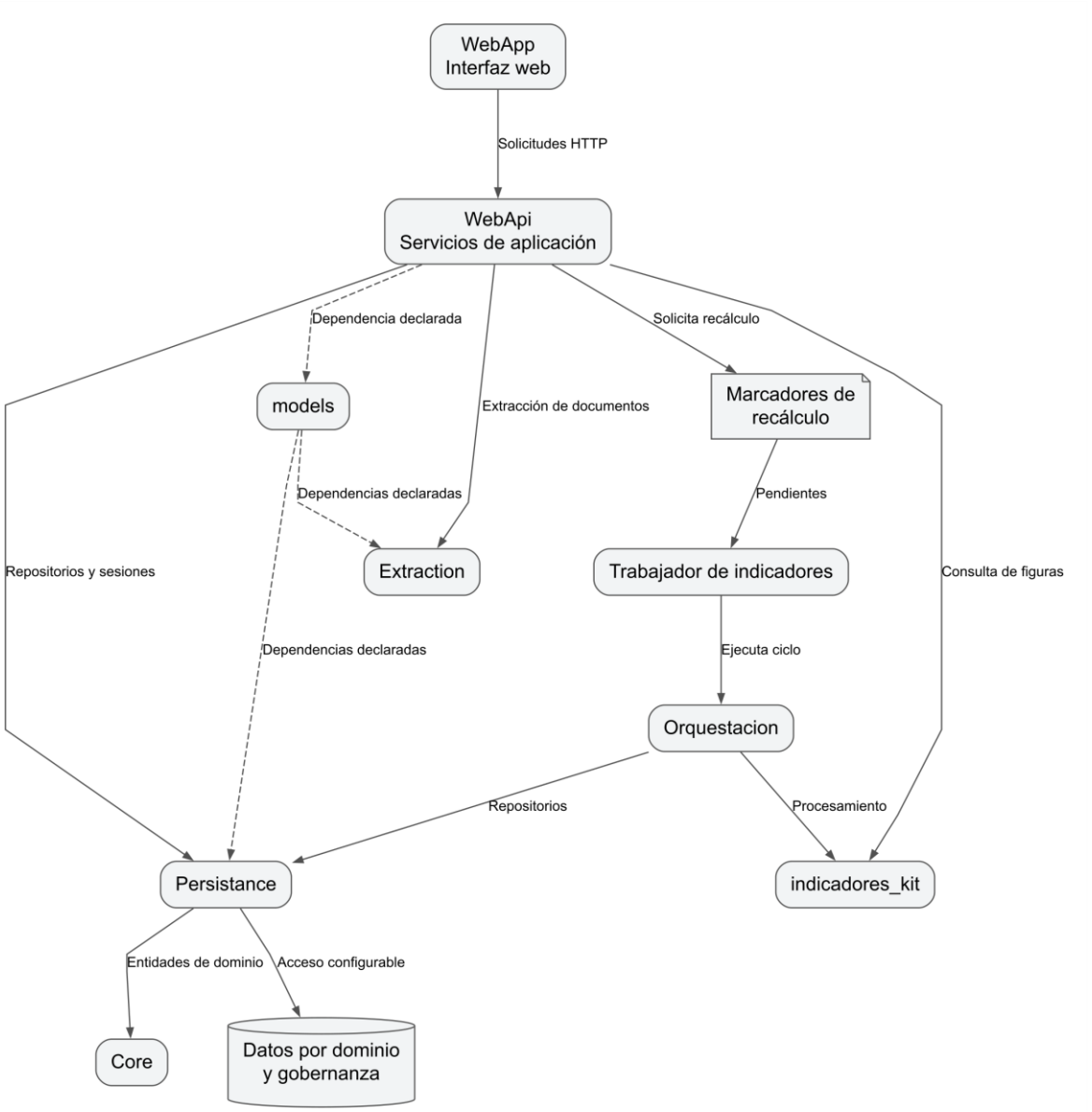
3.4 Relaciones e integración arquitectónica

WebApp se comunica con WebApi mediante solicitudes HTTP para operaciones como validación y carga de fuentes y consulta de figuras. WebApi delega el acceso a datos a repositorios y sesiones de Persistance, que utilizan entidades de Core. En la ingesta documental, los conectores de WebApi emplean Extraction para obtener contenido.

Cuando una carga crea o actualiza filas, la API puede solicitar el recálculo del dominio mediante un marcador de archivo. El trabajador de indicadores atiende los marcadores y llama a Orquestacion, que coordina funciones de indicadores_kit y repositorios de Persistance. Para la consulta, WebApi utiliza funciones de indicadores_kit y sesiones de persistencia para proporcionar figuras a WebApp.

La biblioteca models ocupa una posición de soporte en la arquitectura: WebApi declara su dependencia y models declara dependencias de Persistance y Extraction. Estas relaciones expresan dependencias de componentes; no establecen por sí solas un flujo de predicción.

Figura 1. Arquitectura general y relaciones entre componentes.



El almacenamiento representado identifica accesos lógicos configurables. El trabajador de indicadores pertenece al conjunto de WebApi y se ejecuta por separado del servidor HTTP; las bibliotecas mostradas pueden utilizarse dentro de los procesos que las invocan. Las líneas discontinuas representan dependencias declaradas de models. La coordinación mediante marcadores requiere una ubicación compartida entre la API y el trabajador, conforme al apartado 5.3.

3.5 Flujos principales de procesamiento de información

Ingesta y persistencia. La persona usuaria configura una fuente y el mapeo de sus campos en la interfaz. WebApi recibe la solicitud, utiliza el conector correspondiente y valida las filas contra los esquemas de importación. En una solicitud de carga, las filas aceptadas se procesan mediante los modelos y repositorios del destino. El resultado distingue las filas aceptadas de las rechazadas; la confirmación de escrituras no implica que todas las filas recibidas hayan sido aceptadas.

Cálculo y consulta de indicadores. Una carga con registros creados o actualizados puede generar una solicitud de recálculo. El trabajador de indicadores atiende los pendientes y ejecuta un ciclo coordinado por Orquestacion, que utiliza funciones del kit y repositorios. La API dispone de rutas de consulta de figuras que utilizan sesiones de datos y funciones de representación; WebApp solicita estas figuras para sus paneles de indicadores. La solicitud de recálculo, el cálculo y la generación de la figura son operaciones distintas, cuyo detalle se desarrolla en el apartado 4.4.

4. Diseño de componentes

4.1 WebApp

WebApp es la aplicación de presentación desarrollada con Angular. Su organización combina rutas de navegación, áreas funcionales, componentes reutilizables y servicios que conectan la interfaz con WebApi. Entre sus áreas se encuentran la conexión de datos, gobernanza, bitácora e indicadores. La navegación permite acceder a estas áreas desde una misma aplicación, mientras que los componentes de cada una organizan la interacción correspondiente.

El área de conexión reúne los elementos necesarios para preparar una importación: selección de la fuente, configuración de sus propiedades, identificación del destino y asociación de columnas de origen con campos de destino. El editor de flujo incorpora componentes de mapeo y mantiene la configuración de cada fuente. Los valores predeterminados pueden complementar los campos cuando corresponda. Al cambiar un mapeo se invalida la validación anterior, de modo que la configuración modificada pueda validarse nuevamente.

El estado del espacio de conexión organiza los flujos seleccionados y coordina las solicitudes de validación y carga. Un servicio HTTP construye las peticiones hacia WebApi y recibe los resultados para asociarlos con las fuentes correspondientes. La interfaz presenta las opciones de tratamiento de conflictos y los resultados de la operación; la lectura de fuentes, validación de esquemas y escritura de registros corresponden a los servicios de WebApi y persistencia.

En el área de indicadores, los servicios de Angular solicitan información a la API según el dominio, indicador y figura seleccionados. Un panel gestiona estados de carga, resultado y error, y entrega la figura a componentes de presentación basados en Plotly. Para determinadas vistas, la aplicación también puede adaptar datos recibidos para construir una figura en el cliente. Así, la presentación de una figura publicada y la adaptación de datos son recorridos diferenciados dentro de la interfaz.

4.2 WebApi

WebApi utiliza Flask y organiza sus interfaces mediante *blueprints*, registrados por una fábrica de aplicación. Esta estructura agrupa rutas por áreas funcionales, entre ellas dominios de información, importaciones, datos, gobernanza y reportes. Los blueprints reciben solicitudes HTTP, interpretan sus parámetros y delegan el trabajo a servicios y bibliotecas especializadas.

Para la ingesta, el blueprint de importaciones entrega la configuración recibida a los servicios que seleccionan conectores, resuelven destinos y aplican esquemas de validación. Los destinos se relacionan con modelos y repositorios, de manera que el servicio no necesita incorporar directamente la implementación de cada operación de almacenamiento. Los conectores documentales pueden utilizar Extraction; los repositorios y sesiones se proporcionan mediante Persistance y los modelos de dominio mediante Core.

Los servicios de indicadores atienden consultas de definiciones, reportes y figuras. Para una figura, la API resuelve el indicador y la representación solicitados y utiliza una sesión de dominio junto con funciones de indicadores_kit. La respuesta se entrega a WebApp en un formato adecuado para su

presentación. El servidor HTTP atiende estas solicitudes; el proceso trabajador de indicadores tiene una responsabilidad separada: atender solicitudes de recálculo y coordinar ciclos de procesamiento. Ambos participan en el mismo diseño de aplicación, pero sus actividades y resultados son distintos.

4.3 Ingesta, validación y persistencia

El proceso de importación transforma una configuración de fuente y destino en un conjunto de registros validados y, cuando se solicita la carga, en operaciones de persistencia. El siguiente recorrido utiliza como ejemplo el destino `cacu.Paciente`. Sus reglas concretas de identificación corresponden a esta entidad; otras entidades pueden utilizar configuraciones y reglas diferentes.

Tabla 2. Etapas de ingesta, validación y persistencia.

Etapa	Responsabilidad principal	Resultado
Selección y mapeo	WebApp prepara la fuente; WebApi selecciona el conector y el destino.	Configuración de origen, campos de destino y valores predeterminados.
Validación previa independiente	El servicio lee una fuente y aplica el esquema del destino.	Conteos y detalle de errores para revisar la configuración antes de cargar.
Preparación de la carga	WebApi comprueba fuentes y opciones de un solo dominio, vuelve a leer las filas y puede combinar duplicados antes de validarlas.	Filas aceptadas y rechazos identificados antes de abrir la sesión de escritura.
Escritura	WebApi ordena las fuentes por dependencia; el repositorio utiliza el modelo de destino y la sesión compartida del dominio.	Registros creados o actualizados y conflictos informados.
Confirmación	El servicio confirma las escrituras aceptadas de la sesión.	Datos confirmados, con posibles rechazos individuales en el resultado.
Seguimiento separado	WebApi intenta registrar auditoría y puede solicitar recálculo.	Identificador del intento de auditoría y, cuando corresponde, indicación de solicitud de recálculo.

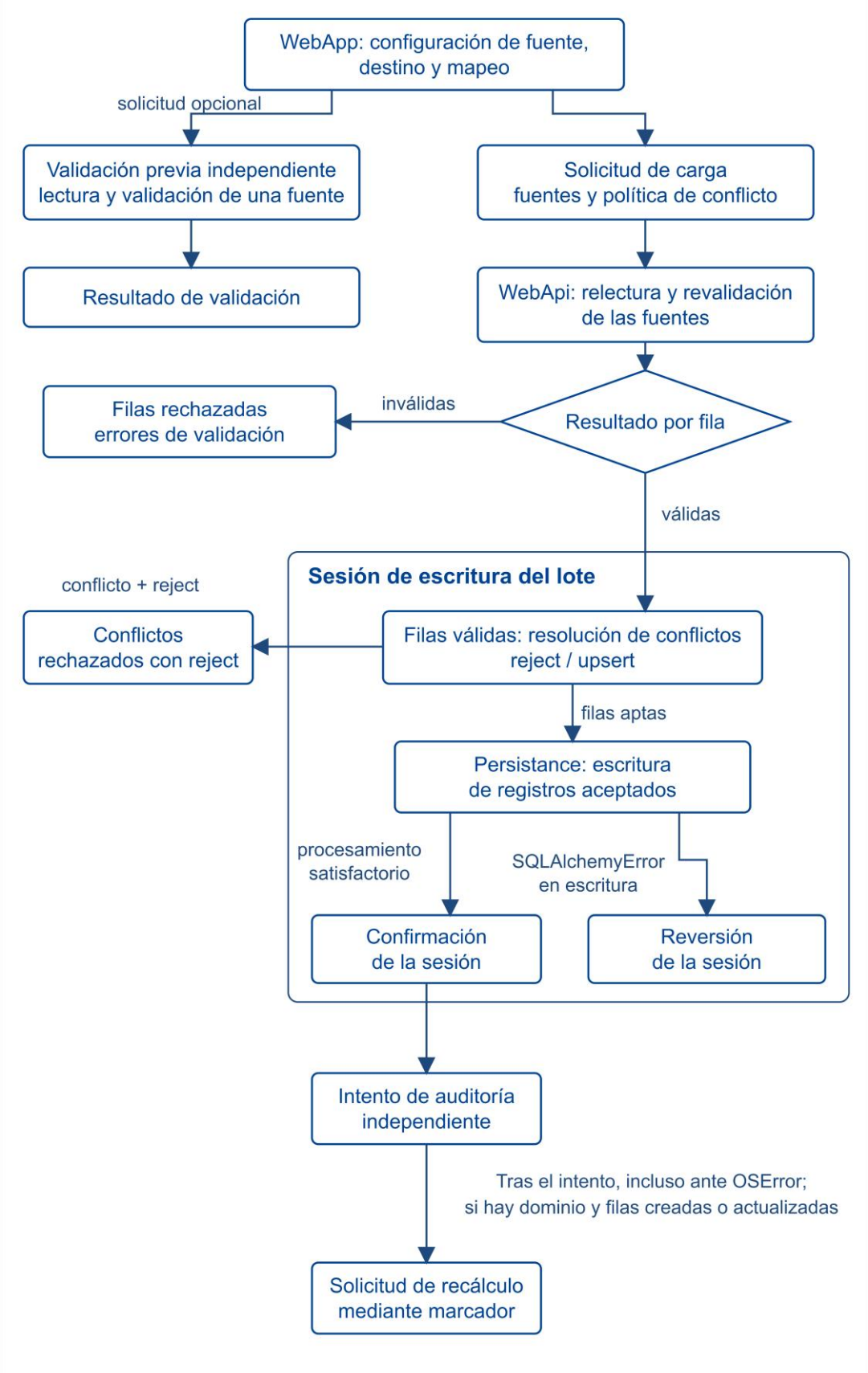


Figura 2. Flujo de ingesta, validación y persistencia.

La configuración de una fuente identifica su tipo, los parámetros propios del conector, el destino, el mapeo y, opcionalmente, una etiqueta y valores predeterminados. El mapeo relaciona una columna de origen con un campo del destino. En el caso SQL, el conector construye las filas utilizando esas relaciones; posteriormente pueden aplicarse valores predeterminados a campos sin valor. El destino `cacu.Paciente` selecciona el esquema de importación, el repositorio de pacientes y la clave de conflicto correspondiente al identificador hospitalario.

La validación es una operación independiente de la carga. El servicio aplica un esquema por fila, normaliza los datos aceptados y reúne errores asociados con la fila y el campo correspondientes. Una fila puede producir más de un error; por ello, el número de errores detallados puede ser mayor que el número de filas rechazadas. El esquema de paciente comprende, entre otros datos, el identificador hospitalario y la fecha de nacimiento. La validación permite revisar la estructura de la información antes de solicitar su incorporación.

Al ejecutar la carga, WebApi vuelve a leer y validar las fuentes antes de abrir la sesión de escritura. Esto permite que el lote se procese con los datos disponibles en ese momento. Las fuentes de una misma solicitud deben pertenecer a un solo dominio. La opción de combinar duplicados permite agrupar filas con la misma clave antes de la validación: conserva la primera, completa valores vacíos y registra advertencias cuando encuentra desacuerdos. Las filas combinadas se identifican por separado en los resultados.

Para conflictos con registros existentes, la opción **omitir** utiliza la política `reject`: el registro existente permanece sin actualización y la fila se informa como rechazada por conflicto. La opción **actualizar** utiliza `upsert`: inserta cuando no hay coincidencia y, cuando la hay, actualiza los campos suministrados, preservando la clave primaria. Los valores predeterminados también pueden aportar campos a la actualización. La disponibilidad de esta estrategia depende de que los destinos del lote cuenten con una clave de conflicto.

Las filas que superan la validación se convierten en objetos del modelo de destino y se entregan al repositorio con una sesión compartida para el lote. El servicio confirma esa sesión después de procesar las fuentes; ante un error de base de datos contemplado por el servicio, solicita la reversión de las escrituras de esa sesión. La unidad de confirmación comprende las escrituras aceptadas del lote: los rechazos individuales de validación o conflicto no implican por sí mismos la reversión de las demás filas. El resultado distingue creaciones, actualizaciones, rechazos y combinaciones; una carga puede confirmar algunas escrituras y, al mismo tiempo, informar rechazos.

Las operaciones de validación y carga generan un intento de registro de auditoría con información agregada de la operación. Después de la carga, si se creó o actualizó al menos una fila, WebApi puede solicitar el recálculo del dominio mediante un marcador. La confirmación de datos, el registro de auditoría y la solicitud de recálculo son operaciones separadas: la respuesta de solicitud de recálculo expresa su envío, no la conclusión del cálculo. Un identificador de auditoría identifica el intento y no equivale por sí solo a una garantía de conservación del registro.

4.4 Indicadores y orquestación

El procesamiento de indicadores se organiza por dominio. Una solicitud de recálculo crea un marcador asociado al dominio; el trabajador identifica los pendientes y ejecuta un ciclo mediante Orquestacion. Las solicitudes acumuladas bajo el mismo marcador pueden agruparse. El trabajador registra el estado del ciclo, de modo que la solicitud de procesamiento se distingue de su resultado. La coordinación mediante marcadores no supone un cálculo independiente por cada carga.

Orquestacion construye el contexto del dominio y reúne los repositorios necesarios. Apache Hamilton organiza las dependencias del ciclo como un grafo: la preparación de catálogos, carga y vistas se relaciona con la disponibilidad de definiciones y con el cálculo. El grafo permite ramas y pasos condicionales; no representa una única secuencia obligatoria para todos los casos. Durante la preparación, las funciones del kit transforman información de origen y el materializador utiliza repositorios para guardar los datos preparados.

El indicador CaCu `ind1` sirve para ilustrar la continuidad entre definición, cálculo y consulta. Su definición técnica tiene estado **`draft`**, por lo que se identifica como definición en elaboración y no como indicador formalmente aprobado. Describe una medida relacionada con citologías anómalas, con periodos de mes y semestre y una figura de contrato C1. Su cálculo utiliza datos de pacientes y citologías, además de los periodos y parámetros definidos para la medida.

Tabla 3. Recorrido representativo del indicador CaCu `ind1`.

Etapa	Tratamiento de información
Definición	El registro identifica `ind1`; la definición declara la URL interna `cacu/IND1`, la versión `0.3.2`, los periodos y la figura `fig_c1_puntos`.
Preparación	Se leen datos de pacientes y citologías, se transforman y materializan las estructuras necesarias para la consulta del indicador.
Cálculo	La orquestación selecciona definición, periodo y parámetros; indicadores_kit construye las poblaciones y produce un reporte con sus unidades.
Persistencia	El repositorio guarda el reporte y sus filas en una operación de confirmación por reporte; puede reutilizar un reporte anterior equivalente.
Consulta y figura	La API localiza reportes por URL de definición, versión y periodo; indicadores_kit adapta sus datos al contrato C1 y genera JSON Plotly.
Presentación	WebApp solicita la figura y la entrega a sus componentes de presentación.

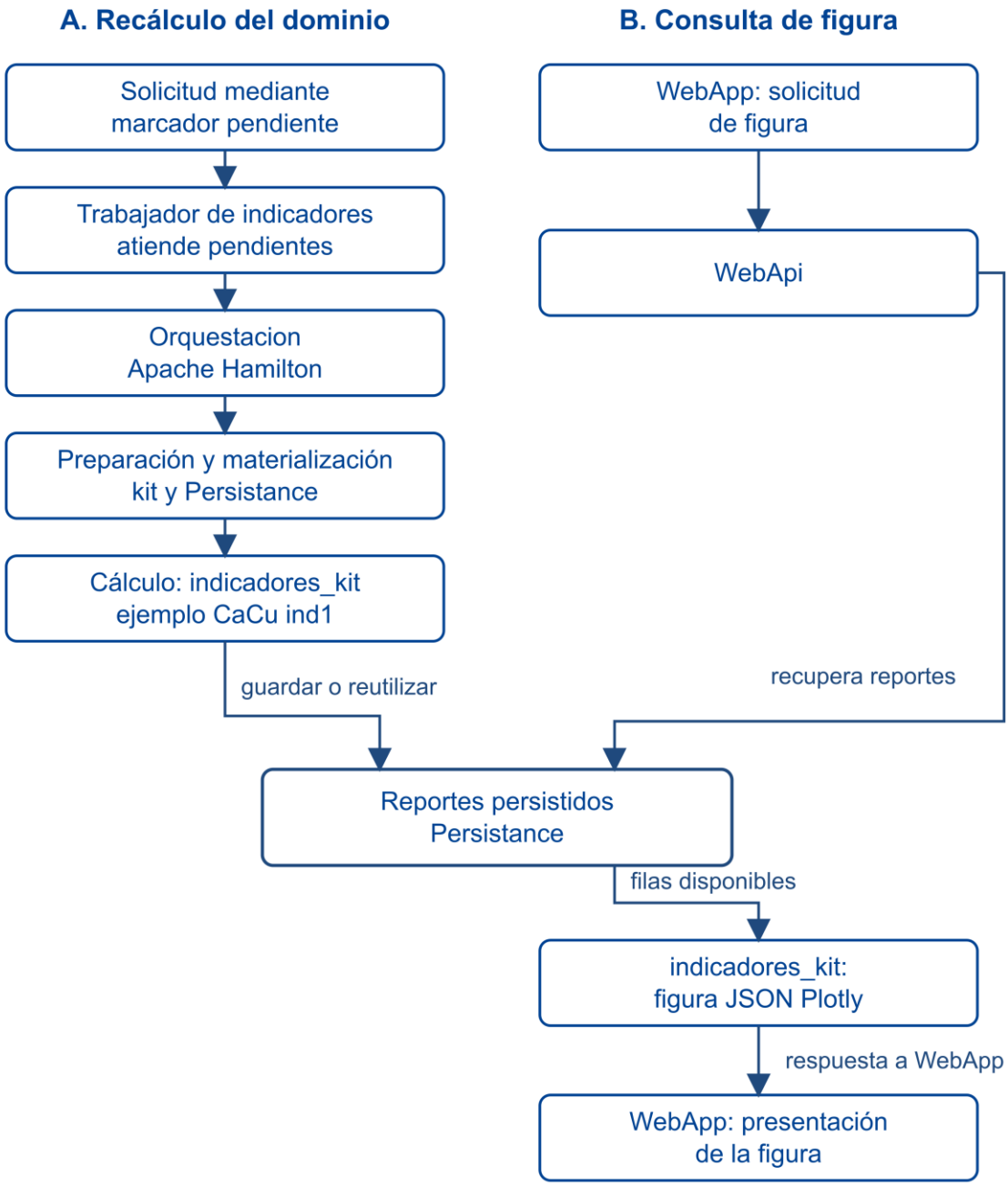


Figura 3. Ciclo de procesamiento y consulta de indicadores.

La URL interna de definición y su versión identifican la medida; no son la dirección HTTP de consulta. El periodo completa la selección del reporte consultado. La escritura de cada reporte con sus filas tiene su propia unidad de confirmación, distinta de la materialización previa y del ciclo completo. En consecuencia, la conclusión o el fallo del ciclo no debe interpretarse como una transacción única de todos sus reportes y etapas.

La generación de la figura ocurre al consultar los reportes disponibles y es distinta del cálculo de la medida. Una importación de pacientes puede solicitar un ciclo del dominio CaCu, pero no determina por sí sola una modificación del resultado de `ind1`: también intervienen las citologías, los periodos y los parámetros aplicables. Este flujo corresponde a indicadores y no implica una invocación a modelos predictivos.

Como criterio para el diseño de indicadores, la documentación del proyecto considera que la unidad de análisis se defina en función de la finalidad de la medida. Según el escenario, un episodio clínico puede organizar información de tamizaje, diagnóstico o tratamiento de una persona, con una delimitación temporal y elementos clínicos pertinentes. La definición de la unidad y la relación entre eventos proporcionan el contexto necesario para interpretar una medida de oportunidad o un análisis de resultados.

Estos criterios metodológicos son generales y se distinguen del recorrido específico de ind1 presentado en este apartado. La selección de población, periodos, parámetros y fuentes corresponde a la definición de cada medida; no se deduce que todos los indicadores utilicen el mismo episodio o la misma regla de cálculo.

4.5 Bibliotecas y componentes de soporte

Core reúne modelos de dominio compartidos, utilizados para representar entidades persistentes. **Persistance** proporciona motores, sesiones, repositorios y recursos de migración que los componentes consumidores emplean para acceder a la información. La separación permite mantener las entidades y el acceso a datos como responsabilidades relacionadas, pero diferenciadas.

Extraction aporta recursos para el tratamiento de documentos, incluida la lectura de PDF y la opción de reconocimiento óptico de caracteres. Los conectores de ingesta pueden utilizar sus funciones para obtener contenido que después se procesa en el flujo correspondiente.

indicadores_kit agrupa funciones de preparación, cálculo y representación de indicadores. **Orquestacion**, además de disponer de una interfaz de línea de comandos, puede utilizarse como biblioteca para construir el contexto y coordinar el ciclo con sus dependencias. La biblioteca **models** ocupa una posición de soporte para interfaces de modelos: WebApi declara una dependencia de ella y models declara dependencias de Persistance y Extraction. Estas relaciones no establecen por sí solas predicciones operativas para los dominios.

La documentación metodológica del proyecto plantea que el desarrollo de modelos parta de cohortes definidas para cada escenario clínico. Una cohorte materializada y versionada permite identificar el conjunto de datos utilizado en un análisis aun cuando las fuentes cambien posteriormente. Su definición considera criterios de elegibilidad y exclusión, junto con el registro de las razones y los conteos correspondientes. Estos elementos aportan contexto para interpretar la población analizada y favorecen la reproducibilidad.

La trazabilidad analítica comprende la identificación de las fuentes, la descripción de las transformaciones y la relación entre la versión de los datos y los artefactos derivados del entrenamiento. Para el uso de modelos en un entorno de operación, los criterios del proyecto contemplan pruebas técnicas documentadas y validación clínica. La calidad y representatividad de los datos forman parte de la interpretación de los resultados. Estos criterios proporcionan un marco metodológico para organizar, documentar y evaluar los procesos analíticos asociados con los modelos del proyecto.

5. Datos e interfaces

5.1 Persistencia y dominios de información

El diseño distingue lógicamente los datos de CaCu, CaMa y TEA e incluye una configuración adicional de persistencia para `governance`. Cada ámbito dispone de una conexión configurable y de funciones para obtener motores y sesiones. Esta organización permite seleccionar el contexto de

datos apropiado para una operación; no exige que cada conexión corresponda a un servidor físico diferente.

Core define modelos que representan entidades del dominio y Persistance los vincula con repositorios y fábricas de sesión. Un repositorio encapsula operaciones de acceso y escritura para su modelo, mientras que la sesión delimita el contexto de interacción con la base de datos. Los servicios consumidores seleccionan el repositorio y la sesión correspondientes al destino, sin que la interfaz web necesite acceder directamente a la base.

Como ejemplo, `cacu.Paciente` representa una entidad de paciente de CaCu. El modelo incluye una identidad interna y un identificador hospitalario; el registro de importación vincula esta entidad con su esquema y repositorio y utiliza el identificador hospitalario para identificar conflictos. Este ejemplo ilustra la relación entre registro de destino, modelo y persistencia sin describir de manera exhaustiva las entidades de los demás dominios.

Los reportes de indicadores constituyen otra clase de información persistida. Un reporte conserva la identidad de la definición y sus resultados por unidades o estratos, de modo que las funciones de consulta puedan recuperar los datos correspondientes a un periodo. Para `ind1`, la selección del reporte vigente utiliza la URL de definición, su versión y el periodo; a partir de sus filas se construye la representación solicitada.

5.2 Interfaces HTTP y comunicación entre componentes

WebApp utiliza solicitudes HTTP con cuerpos y respuestas JSON para interactuar con WebApi. La siguiente selección muestra contratos representativos de validación, carga y consulta de una figura; las interfaces de la plataforma comprenden otras operaciones que se organizan por área funcional.

Tabla 4. Contratos HTTP representativos.

Operación	Entrada principal	Respuesta y tratamiento
`POST /imports/validate`	Objeto `source` con tipo y configuración de fuente, destino, mapeo y opciones aplicables.	Totales de filas, conteos de válidas y con error, y detalle de errores por fila y campo.
`POST /imports/commit`	Lista `sources`, política `on_conflict` y opción `merge_duplicates`.	Resultado general, identificador de auditoría, indicador de solicitud de recálculo y resultados por fuente con conteos y detalle de errores o advertencias.
`GET /cacu/indicadores/ind1/figuras/fig_c1_puntos`	Identificadores en la ruta; tema, periodos y filtro opcional mediante parámetros de consulta.	Figura Plotly en JSON con estructuras `data` y `layout`, construida a partir de reportes disponibles.

En la validación se envía una fuente y se recibe una evaluación de sus filas. La respuesta contiene `total_rows`, `valid_count`, `error_count` y `errors`; el conteo de filas con error y el número de mensajes detallados no son necesariamente iguales. Los errores identifican la ubicación y naturaleza del problema para apoyar su revisión en la interfaz.

La carga recibe una lista no vacía de fuentes de un mismo dominio. `on_conflict` selecciona `reject` o `upsert`; `merge_duplicates` controla la combinación previa de duplicados. La respuesta contiene `ok`, `audit_id`, `indicadores_en_recalculo` y `results`. Cada resultado identifica el destino y presenta, entre otros datos, totales de filas creadas, actualizadas, rechazadas y combinadas, además de errores y advertencias. `ok` indica que no hubo rechazos en las fuentes procesadas; no acredita la finalización

del recálculo. La información de filas rechazadas permite comunicar conflictos o errores sin impedir necesariamente la confirmación de las demás escrituras aceptadas.

Las solicitudes de ingesta deben presentar la estructura requerida para la operación y un destino y configuración de conector válidos. WebApi responde con HTTP 400 para errores de solicitud y ciertas fallas reconocidas de conector, extracción o procesamiento, mediante un objeto con un mensaje de error. Los rechazos de filas pueden formar parte del resultado de una carga procesada y se distinguen de una falla que impide completar la solicitud.

En la consulta de figura, `ind1` identifica el indicador y `fig_c1_puntos` la representación solicitada. La API admite el tema claro u oscuro, periodos separados por comas y, cuando la definición lo permite, un filtro. Para `ind1`, los tipos de periodo definidos son mes y semestre y no se combinan en una misma selección. Si no se especifican periodos, la API selecciona periodos con reportes según la figura. En este recorrido, el servicio de indicadores de WebApp envía tema y periodos; la opción de filtro corresponde al contrato de la API.

La respuesta contiene una figura Plotly en JSON. El servicio de indicadores de WebApp comprueba la estructura esperada y el panel la entrega al componente gráfico. Una solicitud con parámetros inválidos puede recibir HTTP 400; un indicador, figura o reporte no disponible puede producir HTTP 404. La consulta de figura lee reportes: no ejecuta por sí misma un nuevo cálculo de la medida.

5.3 Configuración de conexiones y almacenamiento

Las conexiones de CaCu, CaMa, TEA y `governance` se definen mediante parámetros de configuración por ámbito. Las fábricas de Persistance utilizan la conexión correspondiente para construir el motor y proporcionar sesiones a los repositorios y servicios. El diseño admite cambiar estas conexiones según el entorno sin modificar la responsabilidad de los componentes consumidores.

La ingesta configura sus orígenes por fuente: el tipo de conector determina los parámetros necesarios para acceder a la información y el mapeo especifica cómo relacionarla con el destino. Una fuente SQL y una fuente documental tienen configuraciones propias; el servicio de importación las trata mediante el conector seleccionado y el esquema del destino. Las credenciales y valores concretos de conexión pertenecen a la configuración del entorno y no forman parte de esta descripción.

El almacenamiento lógico comprende los datos de dominio, la información de gobernanza, los datos preparados y reportes de indicadores, y los archivos utilizados para coordinar el recálculo. La API escribe marcadores de dominio y el trabajador los consulta mediante la ubicación configurada para ese propósito. Para que ambos procesos intercambien esos marcadores deben utilizar una ubicación compartida efectiva. Las configuraciones de contenedores y almacenamiento compartido se desarrollan en el apartado 6.2.

6. Infraestructura, configuración y despliegue

6.1 Entorno de desarrollo

El entorno de desarrollo permite trabajar con WebApp y WebApi a partir de sus respectivos componentes de código fuente. WebApp utiliza Angular y TypeScript para la interfaz, con Node.js y npm como herramientas de ejecución y administración de paquetes. WebApi utiliza Python y Flask para los servicios de aplicación; uv administra las dependencias de Python de la API y de las bibliotecas que participan en la composición.

La preparación de WebApi comprende la sincronización de dependencias, la aplicación de migraciones de base de datos mediante Alembic y el arranque de la aplicación Flask. Las migraciones se organizan para CaCu, CaMa, TEA y gobernanza. En paralelo, WebApp instala sus dependencias y ejecuta el servidor de desarrollo Angular. La documentación de entrega identifica `uv sync` y `uv run python main.py` para la API, junto con las migraciones correspondientes, y `npm install` y `npm start` para la aplicación web.

En esta modalidad, los puertos predeterminados comunicados son **4200** para WebApp y **5050** para WebApi. La configuración del servidor de desarrollo Angular incorpora un proxy que dirige al puerto local de la API las solicitudes correspondientes a sus rutas. Los puertos y el proxy de esta modalidad se distinguen de las publicaciones previstas en las configuraciones de contenedores.

6.2 Arquitectura de distribución y contenedores

El diseño de contenedores organiza la aplicación en tres servicios: `webapp`, `webapi` e `indicadores`. El primero sirve la interfaz; el segundo atiende solicitudes HTTP; el tercero ejecuta el trabajador que procesa solicitudes de recálculo. WebApi y el trabajador utilizan la misma imagen de aplicación con funciones de arranque distintas. Core, Persistance, Orquestacion, Extraction, indicadores_kit y models se incorporan como bibliotecas de la aplicación Python y no requieren un contenedor independiente por cada biblioteca en esta configuración.

La imagen construible de WebApp compila la aplicación Angular y utiliza Nginx para servir el contenido y dirigir a WebApi las solicitudes correspondientes. Gunicorn proporciona el servidor HTTP de la API en la configuración de contenedor. El trabajador de indicadores se inicia como proceso separado, atiende marcadores y llama al ciclo de Orquestacion; no actúa como servidor HTTP de la API.

Docker Compose describe los servicios, sus relaciones de arranque y su almacenamiento. La configuración de desarrollo construye las imágenes a partir de los Dockerfiles del proyecto; la de distribución referencia imágenes con una etiqueta de versión configurable. Ambas representan modalidades de configuración para los mismos papeles generales de presentación, servicios y procesamiento.

Tabla 5. Configuraciones de Docker Compose.

Aspecto	Compose de desarrollo (`codigo/docker-compose.yml`)	Compose de distribución (`codigo/dist/docker-compose.yml`)
Imágenes	Construcción desde los Dockerfiles de WebApp y WebApi; el trabajador reutiliza la imagen de WebApi.	Referencias a imágenes de WebApp y WebApi mediante una versión configurable; el trabajador reutiliza la imagen de WebApi.
Puertos publicados	WebApp `8080:80` y WebApi `5050:5000`.	WebApp publica un puerto configurable, `8080` por defecto; el archivo no publica un puerto para WebApi.
Almacenamiento	Volumen de datos compartido por WebApi y el trabajador.	Ruta de datos configurada para el almacenamiento compartido por ambos servicios.
Dependencias	El trabajador espera el estado saludable de WebApi; WebApp declara dependencia de WebApi.	WebApp y el trabajador esperan el estado saludable de WebApi.

La comprobación de salud consulta la interfaz `/health` de WebApi para coordinar el arranque previsto. El punto de entrada de la imagen construible de la API prepara directorios para SQLite, aplica las migraciones de los cuatro ámbitos y después inicia Gunicorn. El proceso trabajador sigue una rama de arranque distinta y no ejecuta esas migraciones en su ciclo. Los archivos de recálculo

y estado utilizan una ubicación configurada dentro del almacenamiento compartido por la API y el trabajador.

La configuración de distribución contempla políticas de reinicio y rotación de los registros de contenedor. Gunicorn envía sus registros de acceso a la salida estándar; los registros propios de auditoría y estado tienen su tratamiento y ubicación de configuración. La rotación de registros de contenedor y la conservación de archivos de auditoría son aspectos distintos.

La entrega se organiza en `codigo/`, con código fuente y recursos de desarrollo, y `deploy/`, con materiales preparados para distribución, incluidos artefactos de imágenes Docker. La documentación de distribución contempla la configuración de `.env` y el uso de `instalar.sh`; el puerto de publicación predeterminado de WebApp es 8080. El capítulo describe la organización y configuración previstas para distribución, mientras que los valores concretos dependen del entorno donde se utilicen.

6.3 Entorno de pruebas

El equipo de desarrollo utilizó Ubuntu Server 26.04 como entorno de pruebas y realizó pruebas de ejecución mediante herramientas locales y contenedores Docker. Este entorno proporciona el contexto tecnológico para las modalidades de ejecución descritas en este capítulo; su configuración es independiente de la infraestructura que pueda utilizarse en otra instalación.

6.4 Tecnologías, versiones y configuración de ejecución

Los manifiestos y configuraciones establecen las tecnologías de la composición. Las versiones de la tabla representan requisitos o versiones declaradas para los contextos indicados; la configuración efectiva de un entorno puede incluir otros parámetros de ejecución.

Tabla 6. Tecnologías y requisitos declarados.

Tecnología	Función y versión o requisito declarado
Angular y TypeScript	Interfaz web. El manifiesto de WebApp declara Angular `^22.1.0` y TypeScript `~6.0.2`.
Node.js y npm	Herramientas de WebApp. Para desarrollo se recomienda Node.js 22 o posterior; el Dockerfile de WebApp utiliza Node 22 para compilación. El manifiesto declara npm `11.17.0`.
Python y uv	Ejecución y administración de dependencias Python. Los manifiestos de WebApi, Orquestacion e indicadores_kit exigen Python `>=3.12`, requisito que se aplica a la composición completa; el Dockerfile de WebApi declara Python 3.12. uv se utiliza para sincronizar dependencias en desarrollo.
Flask y Gunicorn	Flask `>=3.0` está declarado para WebApi; Gunicorn sirve la API en el arranque configurado de contenedor.
Nginx	El Dockerfile de WebApp declara Nginx `1.27` para servir la interfaz y actuar como proxy HTTP.
Docker y Docker Compose	Empaquetado y definición de los servicios de contenedor. La configuración descrita no fija aquí un número de versión de estas herramientas.
SQLModel, SQLAlchemy y Alembic	Modelos, motores y sesiones de datos, y migraciones de base de datos, respectivamente.
Apache Hamilton y Plotly	Orquestación del grafo de procesamiento y representación de figuras. Orquestacion declara Apache Hamilton `>=1.90`.

Los valores de versión de Angular y TypeScript se expresan con los rangos del manifiesto; no representan necesariamente una versión instalada o resuelta. Para Gunicorn, uv, SQLModel, SQLAlchemy, Alembic y Plotly, esta descripción identifica su función sin fijar una versión concreta.

La configuración se organiza por funciones. Las conexiones de CaCu, CaMa, TEA y gobernanza seleccionan las bases utilizadas por sus fábricas de motores y sesiones. En distribución, otras variables identifican la versión de las imágenes, la ubicación de datos persistentes y el puerto de publicación. Los servicios admiten además parámetros de ejecución, como puerto, número de trabajadores HTTP y tiempos de espera.

El trabajador de indicadores dispone de parámetros para la ubicación de marcadores y estado, el intervalo de revisión, la solicitud al inicio y el rango de años. La configuración de registros comprende ubicaciones de auditoría y parámetros para el tratamiento agregado de información. Los archivos y variables de entorno pueden contener datos de conexión; sus valores concretos corresponden a cada entorno y no forman parte de esta especificación.

6.5 Controles de seguridad y consideraciones técnicas

WebApi configura un límite de tamaño para el cuerpo de las solicitudes y devuelve una respuesta de tamaño excedido cuando corresponde. La configuración de Nginx también establece un límite para el cuerpo recibido por el proxy. Estos parámetros pertenecen a capas distintas y pueden configurarse en función de las necesidades del entorno.

La API admite una lista opcional de orígenes CORS para el acceso desde navegadores. Esta configuración regula el tratamiento de solicitudes entre orígenes en el navegador y no constituye un mecanismo de autenticación. Para las conexiones SQL de ingesta, la política del conector comprueba los dialectos permitidos, puede restringir hosts y compara el destino con las conexiones de las bases propias. Las consultas personalizadas están sujetas a comprobaciones de su forma; esas comprobaciones, por sí solas, no garantizan permisos de solo lectura en el servidor de origen.

Las operaciones de importación generan intentos de auditoría agregada que reúnen información de la operación y sus resultados. El trabajador registra estados de procesamiento y el servidor produce registros de acceso. La auditoría de importación se trata separadamente de la confirmación de las escrituras y de la solicitud de recálculo, conforme al diseño descrito en el capítulo 4.

Nginx organiza el acceso HTTP a la aplicación y remite a WebApi las rutas configuradas. Las respuestas de la API distinguen errores de solicitud y de consulta mediante códigos HTTP y mensajes; los componentes registran información para el seguimiento de las operaciones. Los valores de conexión y las credenciales pertenecen a la configuración de cada entorno.

7. Referencias

7.1 Documentación técnica y fuentes del sistema

Las siguientes referencias identifican los recursos técnicos de la entrega utilizados para describir la organización del sistema:

Tabla 7. Referencias técnicas del sistema.

Recurso	Contenido pertinente
Árbol `ISSSTELEON/`	Componentes WebApp, WebApi, Core, Persistance, Extraction, Orquestacion, indicadores_kit y models.
`ISSSTELEON/WebApp/package.json`	Dependencias y herramientas declaradas para la interfaz web.
`pyproject.toml` en `ISSSTELEON/WebApi/`, `ISSSTELEON/Core/`, `ISSSTELEON/Persistance/`, `ISSSTELEON/Extraction/`, `ISSSTELEON/Orquestacion/`, `ISSSTELEON/indicadores_kit/` y `ISSSTELEON/models/`	Requisitos y dependencias declarados por cada biblioteca o aplicación.
`ISSSTELEON/WebApp/angular.json`, `ISSSTELEON/WebApp/proxy.conf.json` y `ISSSTELEON/WebApp/nginx.conf`	Configuración de la aplicación Angular y redirección de solicitudes.
`ISSSTELEON/indicadores_kit/README.md`	Descripción técnica del kit de indicadores.
`ISSSTELEON/models/README.md`	Descripción de las interfaces y organización de la biblioteca de modelos.
Manual plataforma de ingesta	Guía de uso para la selección de fuentes, mapeo, validación y carga de información.

7.2 Documentación de instalación y operación

Los archivos `ISSSTELEON/docker-compose.yml` y `ISSSTELEON/dist/docker-compose.yml`, junto con los Dockerfiles de WebApp y WebApi, describen las configuraciones de servicios, construcción y distribución. Las configuraciones de Alembic en Persistance organizan las migraciones de los ámbitos de información.

Los archivos `deploy/README.md` y `deploy/MANUAL.md` se incluyen como referencias de instalación y distribución. La configuración y los procedimientos de instalación se consultan en esos materiales junto con los recursos de desarrollo de `ISSSTELEON/`.

8. Glosario

Tabla 8. Glosario del sistema.

Término	Definición en el contexto del sistema
API	Interfaz mediante la cual WebApp y otros consumidores solicitan servicios o información a WebApi mediante HTTP.
Blueprint	Unidad de Flask que agrupa rutas y servicios relacionados de la API.
Dominio	Ámbito lógico que organiza información y operaciones, como CaCu, CaMa o TEA.
Ingesta	Proceso de incorporación de información desde una fuente hacia un destino del sistema.
Mapeo	Asociación entre campos de una fuente y campos del destino de importación.
Persistencia	Organización y conservación de información mediante modelos, sesiones, repositorios y bases de datos.
Repositorio	Componente que encapsula operaciones de acceso y escritura para modelos o conjuntos de información.
Orquestación	Coordinación de tareas y dependencias de un ciclo de procesamiento.
Indicador	Medida definida que se calcula a partir de datos y parámetros para generar resultados consultables.
Gobernanza de datos	Funciones y estructuras destinadas a organizar y administrar información relacionada con los datos.
Reconocimiento óptico de caracteres (OCR)	Técnica para extraer texto de imágenes o páginas documentales.
Marcador de recálculo	Archivo asociado a un dominio que comunica una solicitud de procesamiento al trabajador de indicadores.
Trabajador de indicadores	Proceso separado del servidor HTTP que atiende marcadores y ejecuta ciclos mediante Orquestacion.
Reporte de indicador	Resultado persistido de una definición, con sus unidades o estratos; en el recorrido de `ind1` se consulta por URL de definición, versión y periodo.
Figura	Representación construida al consultar reportes disponibles y entregada como JSON Plotly para su presentación.
`reject`	Política de conflicto que conserva el registro existente y comunica el rechazo de la fila entrante.
`upsert`	Política que inserta un registro sin coincidencia o actualiza los campos suministrados de uno existente, preservando su clave primaria.
Contenedor	Entorno de ejecución empaquetado a partir de una imagen y configurado como servicio.
Migración de base de datos	Cambio organizado en la estructura de una base de datos mediante herramientas de evolución de esquemas.
CaCu	Cáncer cervicouterino.
CaMa	Cáncer de mama.
TEA	Trastorno del espectro autista.