# Carpeta de diseño: Muyal-Nez

## 1 Modelo conceptual

### 1.1 Modelo de construcción para diseñar, desplegar y ejecutar servicios agnósticos de la infraestructura basados en bloques autosimilares y autocontenidos

En esta sección se describen los principios de diseño de *PuzzleMesh*, un modelo para construir estructuras de procesamiento que soporten el manejo del ciclo de vida de los productos digitales, y desplegarlas como servicios agnósticos de la infraestructura. Este modelo está inspirado en una metáfora de rompecabezas, en donde diferentes piezas independientes son unidas para formar una solución. En este sentido, en este modelo una aplicación puede ser vista como una pieza de un rompecabezas, que al ser unida con otras aplicaciones (piezas) formarán una solución mayor. Un rompecabezas en el caso de las piezas y una estructura de procesamiento en el caso de las aplicaciones. A diferencia de los juegos rompecabezas tradicionales, donde cada pieza tiene un número limitado de interconexiones de entrada y salida, así como que solo puede ser interconectada con un número limitado de piezas, en nuestro modelo las aplicaciones pueden tener un número ilimitado de interconexiones de entrada y salida, permitiendo su interconexión con cualquier aplicación disponible (similar a los rompecabezas basados en bloques de construcción).

#### 1.1.1 Manejo de aplicaciones como piezas de rompecabezas

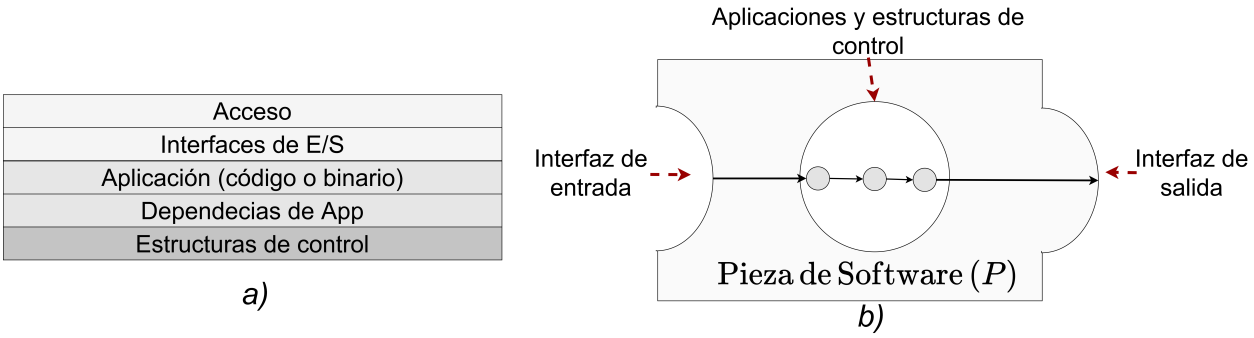



Figure 1: Diseño de una pieza de rompecabezas en PuzzleMesh: *a)* Pila arquitectónica de sus componentes y *b)* su representación conceptual en la metáfora de rompecabezas.

Siguiendo esta metáfora de rompecabezas en PuzzleMesh una aplicación es manejada como una pieza que básicamente representa un artefacto de software que incluye diferentes componentes para garantizar el correcto despliegue y ejecución de esta aplicación, generando un paquete de software autocontenido listo para desplegar y ejecutar en diferentes infraestructuras, sin hacer cambios en la configuración ni en el código fuente de la aplicación. En la Figura 1 muestra la representación de la arquitectura en pila de una pieza, así como su representación conceptual como una pieza de un juego de rompecabezas tradicional. Las capas consideradas en la pila arquitectural de la Figura 1a son las siguientes:

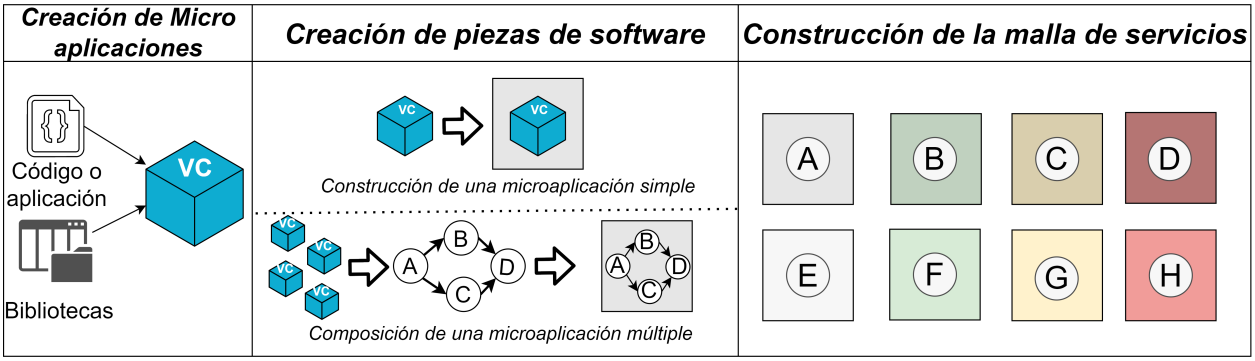



Figure 2: Proceso de diseño y manejo de piezas en PuzzleMesh.

**Capa de acceso.** Verifica que el acceso a la pieza y sus componentes sea válido. Para lo anterior, se utiliza un sistema de tokenización, en el cual cada usuario y aplicación cuenta con un conjunto de tókenes que son validados en esta capa.

**Interfaces de entrada y salida (E/S)** . Permiten la lectura y escritura de datos utilizando la red, el sistema de archivos o la memoria principal. Las interfaces de entrada leen los contenidos desde una fuente de datos, y las interfaces de salida escriben los resultados o datos producidos por una pieza en un destino de datos. Como se puede observar en la Figura 1b, las interfaces de entrada y salida representan, respectivamente, la cuenca y curvatura de una pieza de rompecabezas real.

**Aplicación.** Código fuente o binario de la aplicación que será manejada como una pieza.

**Metadatos.** Incluye los parámetros de entrada y salida de la aplicación, así como la ruta y comando para ejecutar el código fuente o binario de la aplicación.

**Dependencias de la aplicación.** Incluye las librerías y variables de entorno requeridas por la aplicación para ser ejecutada en la misma manera que fue probada y aprobaba para pasar a producción.

**Estructuras de control y manejo.** Controlan la ejecución de estructuras agregadas a una pieza para mejorar su eficiencia, tales como patrones de paralelismo, balanceadores de carga y herramientas de manejo de datos.

En la Figura 1b se muestra como esta pila es convertida en una pieza durante tiempo de ejecución, la cual es manejada como una caja negra.

En la Figura 2 se muestra el proceso de diseño y manejo de piezas software en PuzzleMesh. El proceso inicia con la encapsulación de una aplicación y todos sus componentes y dependencias en un contenedor virtual (CV). El siguiente paso es diseñar una pieza de software siguiendo dos esquemas de composición:

**Composición simple.** Diseño de una pieza de software compuesta solamente por un contenedor virtual, el cual contiene una aplicación para el procesamiento de datos.

**Composición múltiple.** Diseño de una pieza de software mediante el encadenamiento de múltiples contenedores virtuales.

La configuración de una pieza se realiza mediante un esquema declarativo en el cual los diseñadores deben de agregar cada contenedor virtual que desea manejar con dicha pieza de software.

Finalmente, el proceso de diseño de una pieza termina cuando esta es agregada a una malla de servicios, la cual se encarga del almacenamiento e indexamiento de todas las piezas construidas por los diseñadores y usuarios finales.

### 1.1.2 Acoplamiento de piezas de software para crear estructuras de procesamiento

En PuzzleMesh, la unión de un conjunto de piezas de software crean un rompecabezas que básicamente representa una estructura de procesamiento (e.g., un flujo de trabajo o una tubería) que permite el manejo del ciclo de vida de los productos digitales. El orden de ejecución de las piezas y su unión es definido durante tiempo de diseño mediante un grafo acíclico dirigido (DAG, por sus siglas en inglés), el cual determina como los datos son intercambiados entre las piezas utilizando sus interfaces de E/S. El inicio de un DAG es una o múltiples fuentes de datos que contienen los productos digitales que serán procesados a través del rompecabezas. Por otra parte, los nodos finales del DAG son destinos de datos, los cuales representan el destino de los datos (e.g., equipos de almacenamiento o los usuarios finales).

En este sentido, un rompecabezas es conveniente para manejar el ciclo de vida de los productos digitales. Para producir un servicio agnóstico de la infraestructura a partir de una estructura de procesamiento manejada como un rompecabezas, este implementa los siguientes componentes:

**Arquitectura de microservicios.** Permite el manejo de las piezas consideradas en el rompecabezas como servicios verticales que pueden ser consumidos por los usuarios finales y otras aplicaciones.

**Metadatos.** Contienen las rutas de acceso a las fuentes y destinos de datos, así como el DAG que establece el orden de ejecución de las piezas.

**API REST.** Permite que los usuarios consuman los resultados producidos por un rompecabezas, así como el estado de este.

**Red de distribución de contenidos.** Para manejar la entrega de datos entre las piezas consideradas en el rompecabezas. Esta red está basada en un modelo de publicación/suscripción, lo cual permite diseñar diferentes patrones de intercambio y compartición de datos, lo cual permite dar sincronizar los datos intercambiados entre las piezas del rompecabezas.

Estos componentes se encuentran autocontenidos en un rompecabezas, lo cual significa que tanto las piezas como los rompecabezas manejan el despliegue de sus componentes, así como su acoplamiento con otros componentes.

En PuzzleMesh, es posible reutilizar rompecabezas previamente creados para unirlos y crear una nueva estructura de software llamada metarompecabezas, el cual implementa un componente para coordinar la ejecución de los rompecabezas considerados en este metarompecabezas. Esta construcción de estructuras de procesamiento basada en piezas, rompecabezas y metarompecabezas crea estructuras autosimilares, las cuales permiten el manejo del ciclo de vida de los productos digitales a diferentes niveles (aplicación, sistema y servicio).

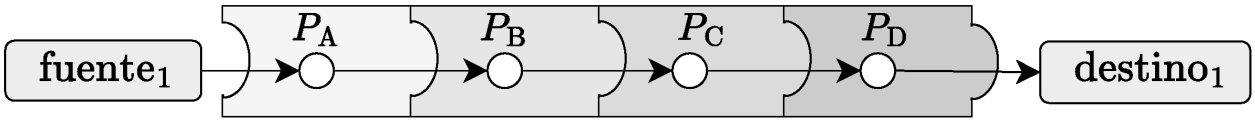



Figure 3: Representación conceptual de un rompecabezas construido en la forma de una tubería.

### 1.2 Modelo basado en rompecabezas para diseñar servicios agnósticos de la infraestructura

En esta sección se presenta un modelo para formalizar el diseño de estructuras de procesamiento como rompecabezas y desplegarlos como servicios agnósticos de la infraestructura.

Para modelar una pieza ($P$), primero se define una funcionalidad ($F$) que puede estar compuesto por una sola aplicación (composición simple) o múltiples aplicaciones (composición múltiple) organizadas en la forma de una tubería. Por lo tanto, esta funcionalidad se puede definir de la siguiente manera:

$$F = \begin{cases} A_1, & \text{Composición simple} \\ A_1 \rightarrow A_2 \rightarrow A_n, & \text{Composición múltiple} \end{cases} \tag{1}$$

donde $n$ es el número de aplicaciones consideradas para ser manejadas.

Esta funcionalidad $F$ es agregada a una pieza ($P$) junto con un conjunto de interfaces de entrada ($\mathbf{E}$) y salida ($\mathbf{S}$), las dependencias de las aplicaciones y un conjunto de estructuras de control para la creación de piezas de software autocontenidas. Una pieza puede modelarse como a continuación:

$$P = [\mathbf{E}, F, \mathbf{S}]. \tag{2}$$

donde $F$ es la funcionalidad de la pieza de software, $\mathbf{E}$ y $\mathbf{S}$ son las interfaces de entrada y salida que pueden ser implementadas utilizando la red, memoria o el sistema de archivos ($\mathbf{E} \wedge \mathbf{S} = \{\text{red} \vee \text{memoria} \vee \text{sistema\_de\_archivos}\}$).

La interconexión de dos o más piezas generan un rompecabezas, el cual básicamente es una estructura de procesamiento para el manejo de datos y productos digitales. En la Figura 3 se muestra una tubería construida mediante el acoplamiento de cuatro piezas ($P_A, P_B, P_C$ y $P_D$) conectadas a una fuente y un destino de datos. El rompecabezas procesa los datos de la fuente y los deposita en el destino.

Como se mencionó anteriormente, un rompecabezas es diseñado como un DAG mediante la interconexión de las interfaces de entrada y salida de un conjunto de piezas de software. Los nodos de este DAG representan las piezas utilizadas para diseñar el rompecabezas, y los vértices son las dependencias de datos que existen entre estas piezas.

La interconexión de dos o más piezas para formar un rompecabezas se realiza utilizando sus interfaces de entrada y salida. En PuzzleMesh, esta interconexión es denotada de la siguiente manera:

$$P_i \rightarrow P_{i+1}. \tag{3}$$

Esto representa que $P_{i+1}$ consumirá, utilizando su interfaz de entrada, los datos entregados por la interfaz de salida dé $P_i$.

Por lo tanto, un rompecabezas ($R$) diseñado en la forma de un DAG es representado de la siguiente manera:

$$R = [\text{fuente}_x \rightarrow P_1], [P_1 \rightarrow P_2], [P_2 \rightarrow P_3], \ldots, [P_{n-1} \rightarrow P_n], [P_n \rightarrow \text{destino}_y], \tag{4}$$

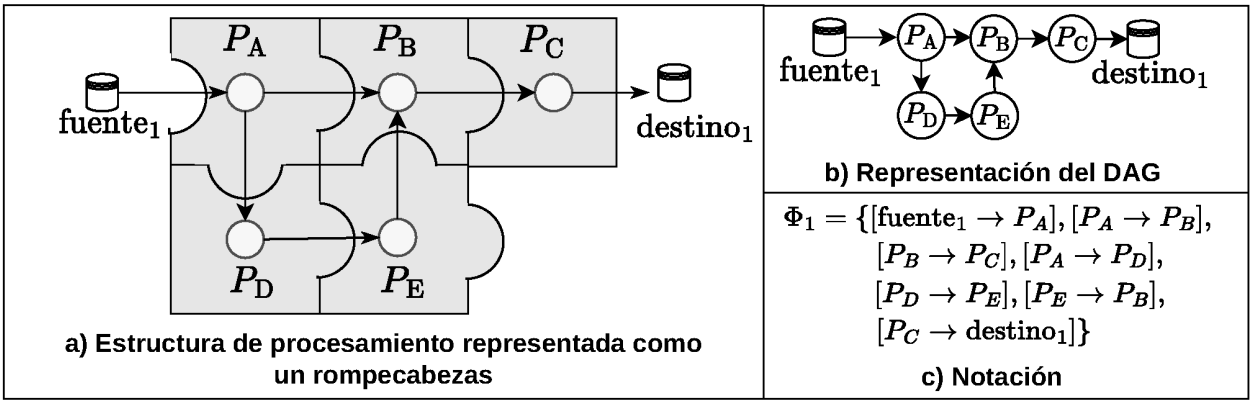



Figure 4: Representación conceptual de *a)* un rompecabezas creado mediante el acoplamiento de piezas, *b)* su representación como un DAG y *c)* y su notación en PuzzleMesh.

donde fuente$_x$ es la fuente de datos con identificador $x$, $n$ es el total de piezas en el rompecabezas $R$ y destino$_y$ es el destino de datos con identificador $y$. En este rompecabezas, el proceso inicia cuando $P_1$ adquiere productos digitales desde fuente$_x$, y finaliza cuando $P_n$ escribe los resultados del procesamiento en destino$_y$.

En la Figura 4 se presenta un ejemplo de un rompecabezas diseñado con cinco piezas, así como su representación como DAG y su notación.

## 2 Modelo abstracto

Un rompecabezas puede ser encadenado con otros rompecabezas para crear estructuras de procesamiento complejas llamadas metarompecabezas ($\Omega$). La creación de un metarompecabezas es conveniente cuando se desean crear flujos de datos que permitan el intercambio de datos entre departamentos de una organización (flujos intrainstitucionales) o incluso entre diferentes organizaciones (flujos interinstitucionales), así como la compartición de recursos de software e infraestructura.

El proceso de creación de un metarompecabezas es similar al de un rompecabezas. Dos rompecabezas se pueden acoplar utilizando las interfaces de entrada y salida de sus piezas, lo cual se denota de la siguiente manera:

$$(P_1 \in R_1) \Rightarrow (P_2 \in R_2), \tag{5}$$

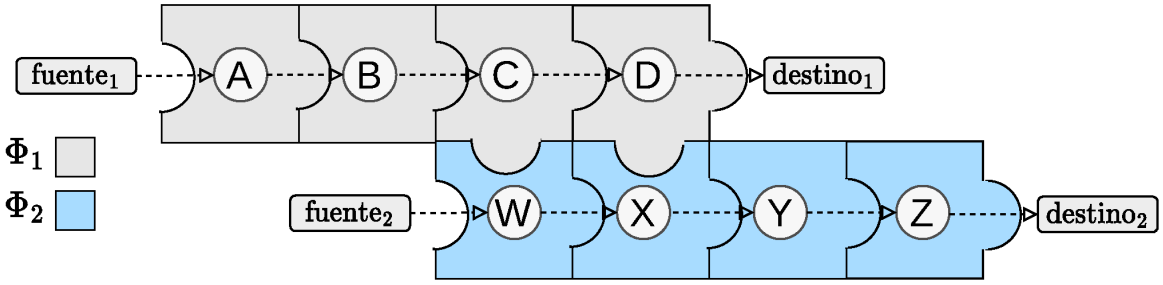
lo que significa que la pieza $P_1$ del rompecabezas $R_1$ está acoplada con la pieza $P_2$ del rompecabezas $R_2$, permitiendo el intercambio de datos entre los rompecabezas $R_1$ y $R_2$.

Formalmente, un metarompecabezas está compuesto por un conjunto de fuentes, destinos y piezas, de la siguiente manera:

$$\begin{aligned}\Omega = &\text{fuente}_{1,\ldots,n)}, \{[(P_1 \in R_1) \Rightarrow (P_2 \in R_2)], \ldots, \\ &[(P_a \in R_x) \Rightarrow (P_b \in R_y)]\}, \text{destino}_{1,\ldots,m}\end{aligned} \tag{6}$$

donde $n$ representa el número de fuentes de datos, $m$ es el número de destinos de datos, $x$ e $y$ son los identificadores de dos rompecabezas acoplados utilizando las piezas $a$ y $b$.

En la Figura 5 muestra una representación conceptual de un metarompecabezas integrado por dos rompecabezas ($R_1$ y $R_2$), los cuales se encuentran unidos mediante el acoplamiento de las piezas $P_C$ y $P_D$ de $R_1$ con $P_W$ y $P_X$ de $R_2$.




$$\Omega_1 = \text{fuente}_{(1,2)}, \{[(C \in \Phi_1) \Rightarrow (W \in \Phi_2)], [(D \in \Phi_1) \Rightarrow (X \in \Phi_2)]\}, \text{destino}_{(1,2)}$$

Figure 5: Representación conceptual de un metarompecabezas compuesto de dos rompecabezas.

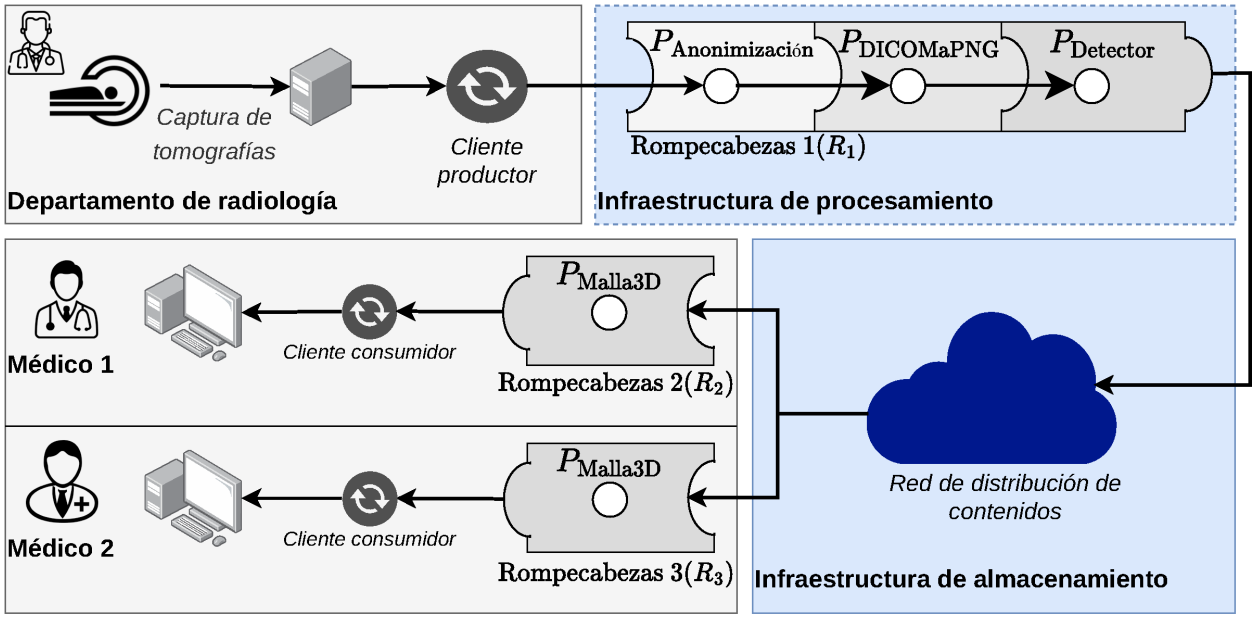



Figure 6: Ejemplo de un flujo de datos intrainstitucional en un hospital.

### 2.0.1 Flujos de datos intrainstitucionales

Un flujo de datos intrainstitucional es creado cuando un metarompecabezas interconecta múltiples usuarios dentro de una misma organización. Estos flujos automáticamente procesa, entrega y recupera datos a los miembros autorizados para acceder a los datos y resultados en una organización. En este sentido, para automatizar la entrega y recuperación de estos datos, los usuarios finales tienen acceso a un cliente, el cual puede realizar operaciones de carga de datos y la recuperación de resultados. Este cliente puede tomar dos roles:

**Productor.** Un cliente productor carga los datos o productos digitales para ser procesados a través del metarompecabezas.

**Consumidor.** Adquiere los resultados producidos por el metarompecabezas.

En la Figura 6 se ilustra un ejemplo de un flujo de datos intrainstitucional para el manejo de datos médicos un hospital. En este flujo, un técnico radiólogo captura imágenes de tomografía de un

paciente, las cuales son almacenadas en una computadora ubicada en el departamento de radiología. Los datos son adquiridos por el *cliente productor* y procesados a través de un metarompecabezas que considera dos rompecabezas:

1. El primero para la identificación automática de tumores en las radiografías. Para este fin, este rompecabezas incluye tres piezas para la anonimización de las tomografías eliminando datos personales de los pacientes, la conversión de las imágenes de formato DICOM a PNG y finalmente una red neuronal convolucional para la identificación de tumores en las imágenes. Este rompecabezas se despliega en un servidor en las instalaciones del hospital. La última pieza en el rompecabezas prepara los resultados agregándoles requerimientos no funcionales tales como confidencialidad utilizando el algoritmo de AES-256 [1] y confiabilidad utilizando el algoritmo de dispersión de información [2]. En este sentido, estos algoritmos realizan las operaciones de cifrado y codificación sobre los datos, respectivamente. Los datos preparados son entregados a una red de distribución de contenidos para su almacenamiento.

2. El segundo rompecabezas toma de entrada los datos almacenados en la red de distribución de contenidos para su procesamiento con una pieza que realiza la generación de una representación en malla 3D de las tomografías, para que el médico las pueda visualizar. Por lo tanto, la pieza en su interfaz de entrada recupera los datos aplicando las operaciones inversas (decodificación y descifrado) que se le aplicaron a los datos durante su preparación, con el objetivo de que los datos puedan ser procesados por esta pieza. Este rompecabezas se despliega en la computadora de los médicos interesados en consumir los resultados (e.g., oncólogos), los cuales pueden acceder a los resultados a través de un *cliente consumidor*.

#### 2.0.2 Flujos de datos interinstitucionales

Los flujos de datos interinstitucionales permiten la interconexión de usuarios pertenecientes a múltiples instituciones. Las metas de estos servicios son:

1. Permitir que los usuarios de diferentes instituciones puedan compartir datos.

2. Permitir que las organizaciones puedan compartir recursos de procesamiento y almacenamiento de datos para manejar datos de forma distribuida.

En la Figura 7 se muestra un ejemplo de un flujo de datos interinstitucional para el manejo de tomografías adquiridas por un técnico radiólogo en un Hospital A. Utilizando un cliente productor, el radiólogo entrega los datos a un rompecabezas el cual procesa los datos a través de las piezas de anonimización, conversión de DICOM a PNG y una red neuronal para la identificación de tumores en las tomografías. Los resultados son preparados y entregados a la red de distribución de contenidos, desde donde los hospitales B y C recuperan los datos. En la infraestructura de estos hospitales, se encuentra desplegado un rompecabezas para la recuperación de los datos y su procesamiento con una pieza para la generación de una malla 3D de las tomografías. Finalmente, utilizando un cliente consumidor, los médicos de estos hospitales pueden acceder a los resultados.

### 2.1 Malla de servicios para el manejo de servicios

Los metarompecabezas, rompecabezas y piezas son incorporados en una malla de servicios ($\Psi$), que además incluye las fuentes y destinos de datos. Esta malla produce un repositorio desde el

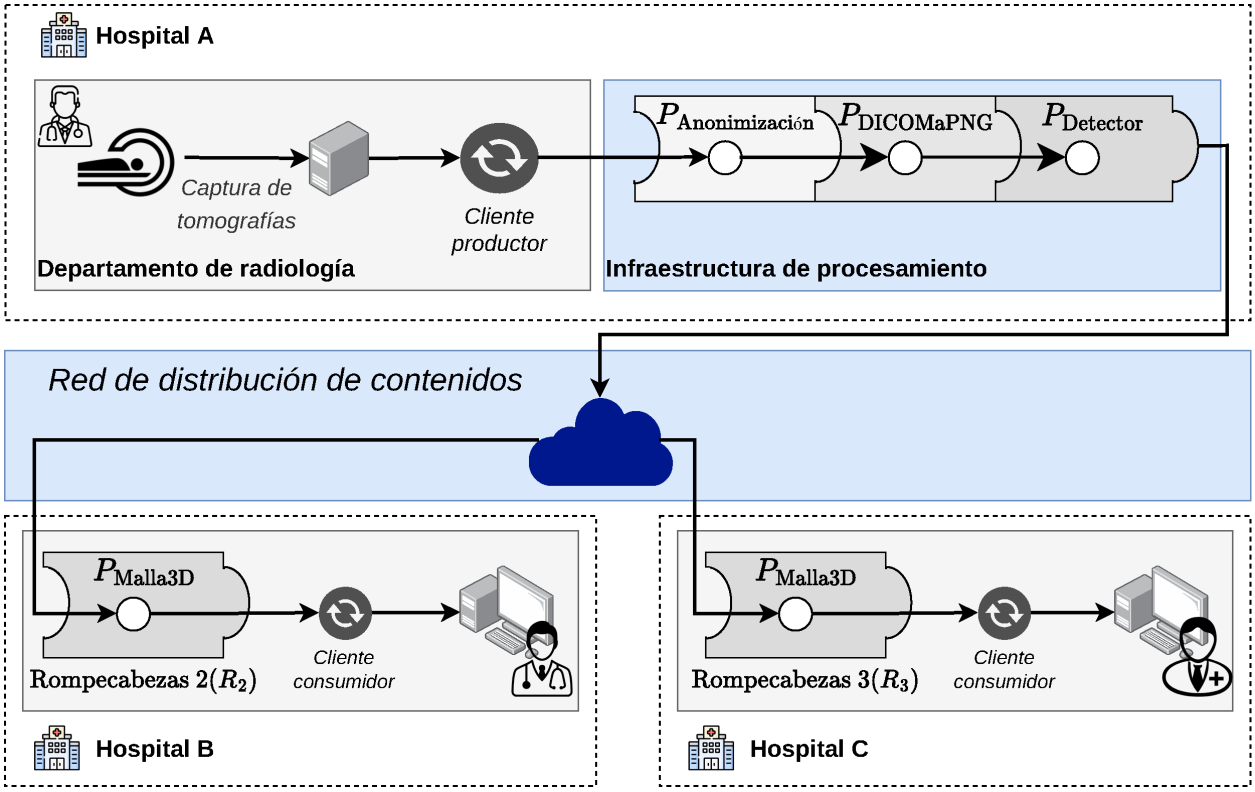



Figure 7: Ejemplo de un flujo de datos interinstitucional construido con PuzzleMesh para el intercambio de datos entre hospitales.

cual las organizaciones pueden seleccionar las piezas requeridas para crear servicios para manejar sus productos digitales. En PuzzleMesh el repositorio de servicios puede ser consultado por los desarrolladores mediante un conjunto de APIs de programación disponibles en la malla de servicios.

Esta malla incluye mecanismos de control de acceso para solo permitir el acceso a los servicios a usuarios autorizados, con el objetivo de que estos puedan modificar y reutilizar servicios previamente existentes. Además, la malla de servicios incluye un mecanismo de paso de mensajes basado en un esquema de publicación/suscripción permite sincronizar los metadatos y datos entre las piezas de un rompecabezas.

En la Figura 8, se muestra la representación conceptual de la construcción de un rompecabezas a partir de piezas disponibles en la malla de servicios, la cual permite a los diseñadores y organizaciones reutilizar rompecabezas y piezas creados dentro de la organización o incluso creados por otras organizaciones cuando estos comparten sus servicios.

Además, esta malla de servicios incluye un repositorio de catálogos que contienen aquellos productos producidos por las organizaciones. Por ejemplo, imágenes médicas, señales de electrocardiogramas o expedientes clínicos electrónicos. Dichos catálogos de datos son construidos utilizando los servicios de transporte y almacenamiento de datos, los cuales permiten a las instituciones compartir bases de datos, resultados/información y servicios.

En este contexto, una malla de servicios ($\Psi$) se compone como el conjunto de piezas, rompecabezas, y metarompecabezas creados, como se denota a continuación:

$$\Psi = [\mathbf{F}, \mathbf{R} \wedge \mathbf{\Omega} \wedge \mathbf{P}, \mathbf{RS}], \tag{7}$$

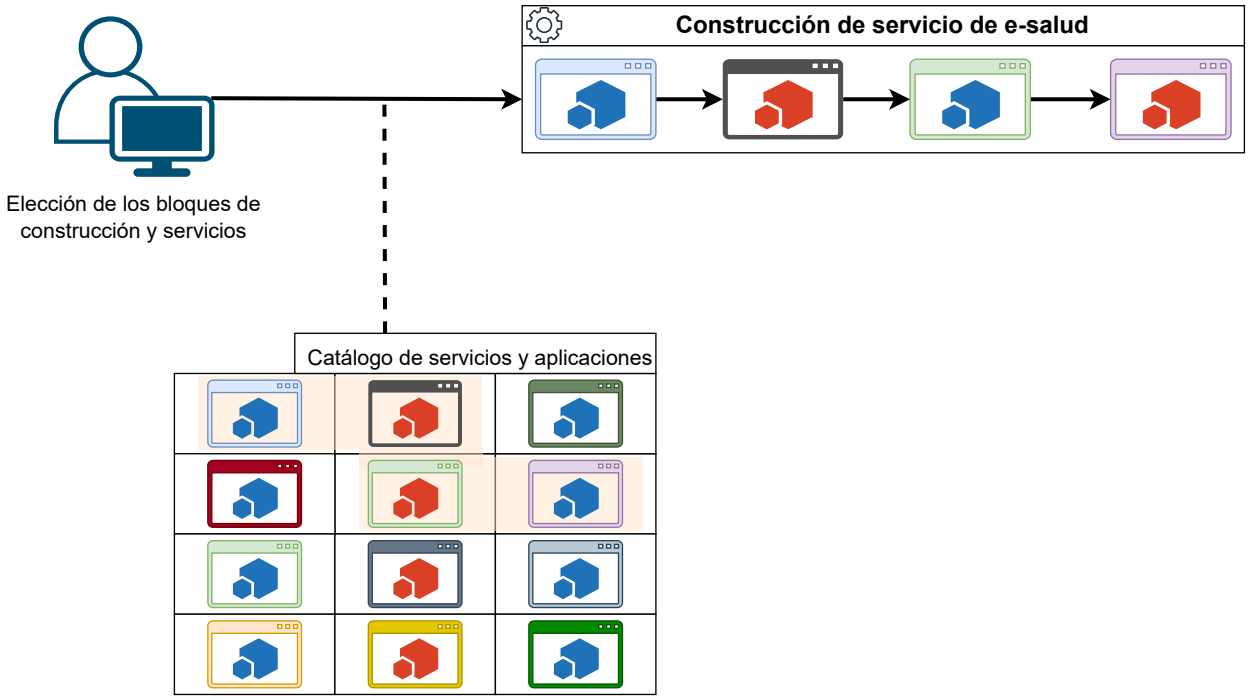



Figure 8: Representación conceptual del repositorio de piezas disponibles en la malla de servicios.

donde **F** es el conjunto de fuentes de datos disponibles, **RS** es el conjunto de destinos de datos disponibles, **R** es el conjunto de rompecabezas creados, **P** el conjunto de piezas y **Ω** el conjunto de metarompecabezas diseñados.

El despliegue de estos artefactos de software en la malla de servicios es posible gracias a que son paquetes de software autocontenidos y autosimilares. Tres características importantes pueden ser identificados de este método autocontenido y autosimilar:

1. El manejo de la heterogeneidad logrado al abstraer las aplicaciones como piezas de software para crear rompecabezas (estructuras de procesamiento). Esto permite que las organizaciones unan aplicaciones desarrolladas en diferentes lenguajes de programación, así como asimilar cambios el ciclo de vida de los productos digitales, simplemente reajustando las interfaces de entrada y salida de una pieza y acoplándola con otras piezas en el rompecabezas.

2. La reutilización de rompecabezas acoplándolos con otros rompecabezas, creando un metarompecabezas que permite la generación de flujos de datos intra e inter institucionales.

3. La creación automática de flujos de datos a través de las piezas, rompecabezas y metarompecabezas. Lo anterior es logrado a que cada pieza está a cargo de la entrega de sus datos de salida, así como de la recuperación de sus datos de entrada.

## 3 Arquitectura

En esta Sección se describe una arquitectura que describe el método presentado en esta tesis para desarrollar servicios agnósticos de la infraestructura para el manejo del ciclo de vida de los

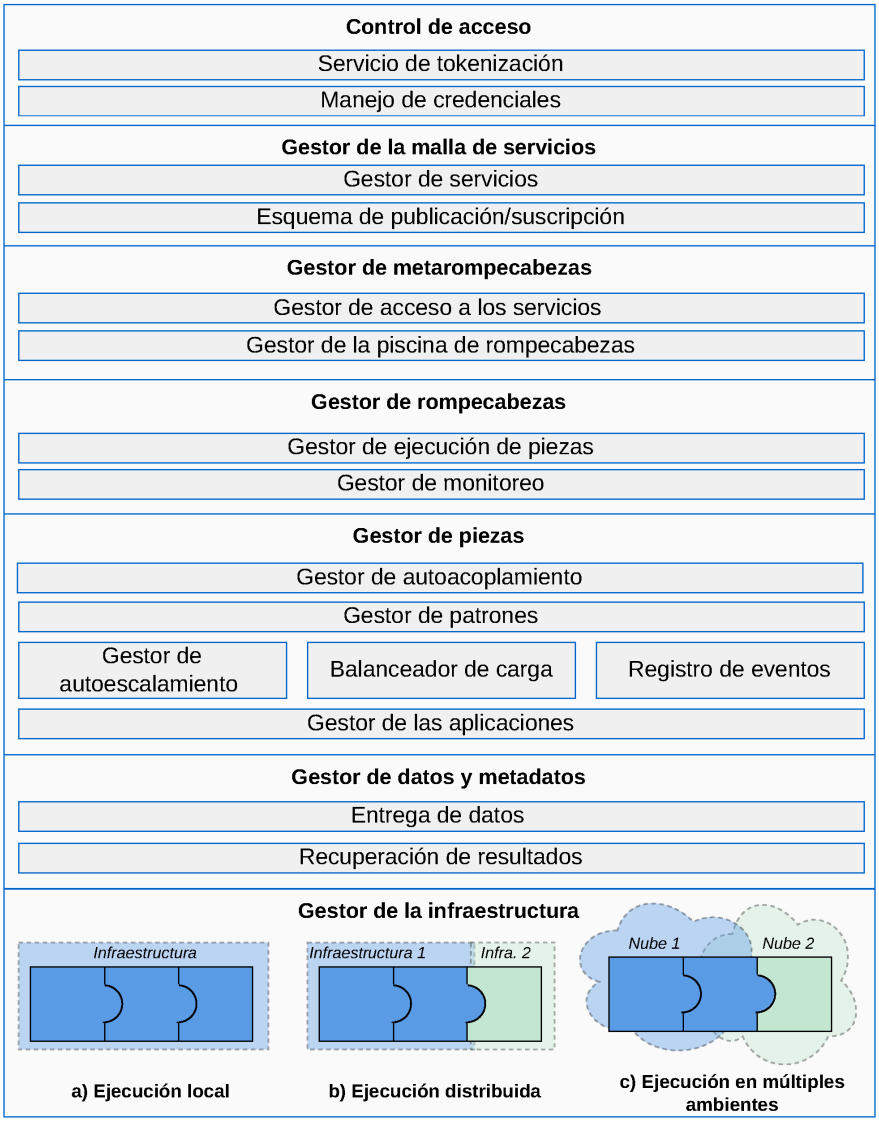



Figure 9: Arquitectura para el manejo de piezas, rompecabezas y metarompecabezas.

productos digitales. Esta arquitectura fue desarrollado como una composición de servicios basada en estructuras autocontenidas y autosimilares. La arquitectura propuesta permite alcanzar las siguientes metas:

- Permitir el despliegue de estructuras de procesamiento (diseñadas como rompecabezas), conformadas por un conjunto de piezas autosimilares y autocontenidas, en la infraestructura disponible.
- Crear un flujo de datos a través de todos los componentes de la estructura de procesamiento para el manejo del ciclo de vida de los productos digitales.
- Mitigar los cuellos de botella que puedan surgir durante tiempo de ejecución en una estructura de procesamiento.

- Exponer las estructuras de procesamiento y sus componentes como servicios agnósticos de la infraestructura que puedan ser consumidos por los usuarios autorizados.

Es importante recalcar que esta arquitectura ha sido diseñada para manejar cada nivel de composición de un servicio agnóstico de la infraestructura, los cuales son:

- Metarompecabezas: flujos de datos inter e intrainstitucionales.
- Rompecabezas: estructuras de procesamiento de datos para el manejo del ciclo de vida de los productos digitales.
- Pieza de software: aplicaciones para procesar y manejar los datos en un rompecabezas.

En la Figura 9 se presenta la arquitectura en forma de pila para el manejo de piezas, rompecabezas y metarompecabezas. Como se puede observar, la pila considera las siguientes capas:

**Control de acceso.** Esta capa es el *front-end* que recibe las solicitudes realizadas por agentes externos (clientes y aplicaciones de terceros). Esta capa implementa un sistema de autenticación basado en tokenización, donde a cada agente externo se le asigna token que es validado cada que este hace una solicitud. Cabe resaltar que este sistema simple de tokenización puede ser remplazado por un sistema de autenticación basado en llaves pública/privada.

**Tenencia múltiple.** Esta capa permite manejar las solicitudes realizadas por los múltiples clientes que acceden a los servicios y que han sido validados en la capa anterior. Dado que múltiples clientes van a acceder a los mismos servicios, esta capa se encarga de aislar estar solicitudes, asignándoles espacios virtuales y privados de ejecución. Esto evita accesos no autorizados a los datos y resultados producidos por los servicios ejecutados por un cliente.

**Gestor de la malla de servicios.** Esta capa se encarga de manejar los servicios disponibles en la malla de servicios. Esta capa incluye un esquema de publicación/suscripción que permite a los diseñadores y desarrolladores hacer disponibles sus piezas a otros usuarios para que las puedan utilizar. En este sentido, los desarrolladores publican sus piezas en la malla de servicios para que otros usuarios se suscriban a ellas y las puedan utilizar.

**Gestor de metarompecabezas.** Esta capa se encarga de manejar el acceso a los rompecabezas creados. Incluye una componente para el control de acceso a los rompecabezas manejados como servicios. En este sentido, este servicio redirecciona las peticiones al rompecabezas que debe de atenderla.

**Gestor de rompecabezas.** Esta capa se encarga de manejar la ejecución y monitoreo de las piezas que conforman un rompecabezas.

**Gestor de piezas.** En esta capa se manejan las piezas desplegadas. Para esto incluyen componentes para manejar el autoacoplamiento de las piezas utilizando sus interfaces de entrada y salida, manejo de patrones y el autoescalamiento de estos patrones. Además, en esta capa se realiza el manejo de la ejecución de las aplicaciones consideradas en la pieza.

**Gestor de datos y metadatos.** Esta capa maneja la entrega y manejo los datos requeridos por los componentes de las capas superiores. En este sentido, esta capa está implementada utilizando una red de distribución de contenidos basada en abstracciones llamadas catálogos y un esquema de publicación/suscripción [3, 4].

**Gestor de la infraestructura.** En esta capa se maneja el acceso a la infraestructura disponible par desplegar los servicios agnósticos de la infraestructura.

### 3.1 Diseño y manejo de rompecabezas con la arquitectura

En la Figura 9 se muestra la representación conceptual de una arquitectura jerárquica compuesta por un conjunto de gestores para el control de piezas, rompecabezas y metarompecabezas, así como de la malla de servicios y de la infraestructura.

En la etapa de diseño, los desarrolladores definen los servicios y estructuras de procesamiento seleccionando un conjunto de piezas, rompecabezas y metarompecabezas de la malla de servicios. Además, los desarrolladores pueden crear una nueva piezas utilizando un esquema declarativo que incluye la imagen de contenedor a utilizar y un comando de ejecución. Las interfaces de entrada y salida, así como el acoplamiento de piezas, es especificado por los programadores en un archivo de configuración, que al ser interpretado por PuzzleMesh produce un DAG, el cual es convertido en un archivo de configuración YML que contiene la declaración de todos los servicios de un rompecabezas que posteriormente será materializado como un conjunto de contenedores virtuales.

En la fase de construcción, el gestor de la malla de servicios invoca recursivamente, siguiendo la arquitectura jerárquica, los gestores de piezas, rompecabezas y metarompecabezas siguiendo el archivo de configuración creado por el desarrollador en tiempo de diseño. En las fases de despliegue y orquestación, el gestor de servicios primero crea las imágenes de contenedor de las piezas consideradas en el archivo de configuración. Dichas piezas serán desplegadas como microservicios, y el conjunto de piezas de un rompecabezas serán manejados como una arquitectura de microservicios. En un segundo paso, los gestores leen los metadatos de cada artefacto desplegado para preparar su funcionalidad, utilizando las aplicaciones y estructuras de control (APIs, balanceadores de carga y estructuras de intercambio de daots) embebidas en las imágenes de contenedor. Además, durante esta fase, un gestor de patrones crea y acopla los componentes requeridos para implementar patrones paralelos.

En las fases de despliegue y coreografía, los gestores primero realizan recursivamente el despliegue de las instancias de contenedor utilizando las imágenes previamente creadas, y posteriormente se realiza el acoplamiento automático siguiendo la configuración de las interfaces de entrada y salida declaradas en el archivo de configuración. Los contenedores virtuales son creados utilizando la plataforma de contenedores Docker, incluído Docker Compose para el manejo de sistemas multicontenedores, así como Docker Swarm y Kubernetes para manejar servicios distribuidos en clústeres de contenedores. El resultado de esta fase es un conjunto de contenedores virtuales organizados en la forma de una estructura de procesamiento, siguiendo el diseño de piezas y rompecabezas declarado.

En las fases de ejecución y operación, la estructura de procesamiento (conformada por piezas, rompecabezas y metarompecabezas) primero es expuesta como un servicio, para posteriormente estar disponible para los usuarios autorizados para operarla.

El gestor de la infraestructura incluye un cliente que implementa una API para la comunicación con la infraestructura donde los servicios serán desplegados. Cabe resaltar que cada gestor establece no solo control sobre el acoplamiento de las fases en la fase de construcción, sino que además durante el manejo de datos en tiempo de ejecución y operación.

En resumen, la construcción de servicios es realizada siguiente la arquitectura jerárquica de arriba hacia abajo, mientras que los controles de ejecución y operación son establecidos desde el fondo hacia el tope de la arquitectura. Los gestores de la malla de servicios controlan los servicios resultantes utilizando un esquema de publicación/suscripción.

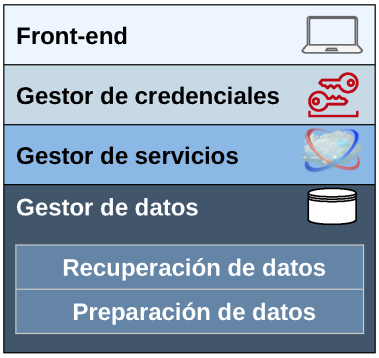



Figure 10: Pila de servicios del cliente implementado para interactuar con la arquitectura.

### 3.2 Clientes de acceso a la arquitectura

Se diseñó un cliente que permite a los usuarios interactuar con la arquitectura previamente descrita. En este sentido, la arquitectura puede recibir cuatro tipos de *solicitudes* para manejar cada uno de estos componentes:

- Crear una nueva piezas de rompecabezas y hacerla disponible en la malla de servicios.
- Crear un rompecabezas o metarompecabezas a partir de las piezas disponibles en la malla de servicios.
- Desplegar un rompecabezas en la infraestructura y hacerlo disponible a los usuarios como un servicio.
- Entregar datos a los servicios.
- Recuperar los datos crudos y los resultados producidos por los servicios.

En la Figura 10 se muestra la pila de módulos de este cliente, el cual incluye las siguientes capas:

**Front-end.** Implementa una interfaz gráfica para que los usuarios puedan acceder al resto de componentes del cliente.

**Gestor de credenciales.** Esta capa implementa un esquema de autenticación de usuarios basado en credenciales usuario/contraseña. Además, esta capa implementa un módulo para emitir credenciales de acceso a usuarios y organizaciones.

**Gestor de servicios.** Esta capa se encarga de manejar las solicitudes que hacen los usuarios a la arquitectura.

**Gestor de datos.** Esta capa maneja la entrega de datos a los servicios de la arquitectura para su procesamiento, así como la recuperación de los resultados producidos por la arquitectura. Por lo tanto, esta capa incluye módulos para la recuperación y preparación de datos, para manejar los requerimientos no funcionales considerados por el usuario u organización. Por ejemplo, si se incluye el requerimiento de confidencialidad, este módulo cifrará los datos entregados a la arquitectura y descifrará los resultados obtenidos de la arquitectura.

# 4 Casos de uso

En el siguiente enlace se encuentra un video demostrativo del despliegue y funcionamiento de Muyal-Nez.

`https://youtu.be/EhSGPeGAqPg`

# References


[1] Miguel Morales-Sandoval, Jose Luis Gonzalez-Compean, Arturo Diaz-Perez, and Victor J Sosa-Sosa. A pairing-based cryptographic approach for data security in the cloud. *International Journal of Information Security*, 17(4):441–461, 2018.

[2] Michael O Rabin. The information dispersal algorithm and its applications. In *Sequences: Combinatorics, Compression, Security, and Transmission*, pages 406–419. Springer, 1990.

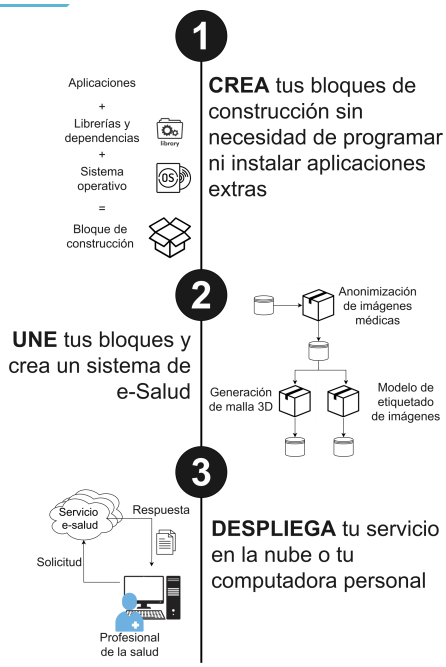
[3] J. Gonzalez, J. Perez, V. Sosa, Luis Sanchez, and B. Bergua. Skycds: A resilient content delivery service based on diversified cloud storage. *SIMPAT*, 54:64–85, 2015.

[4] Diana Carrizales-Espinoza, Dante D Sanchez-Gallegos, JL Gonzalez-Compean, and Jesus Carretero. Fedflow: A federated platform to build secure sharing and synchronization services for health dataflows. *Computing*, pages 1–19, 2022.

# Nez

# Servicios de construcción de sistemas de e-salud

## Servicio de construcción automático de sistemas e-salud para el manejo de datos y contenidos en la práctica médica




*Construcción automático de sistemas e-salud*

## Maneja y comparte tus servicios de e-salud

### Sistemas de e-salud **intra-institucionales**

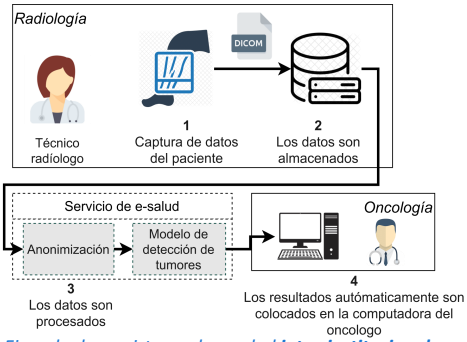
Puedes compartir tu servicio de e-salud, y los datos generados, entre:

**Múltiples usuarios**

- Sistemas de compartición de imagenología.
- Medición de signos vitales.
- Datos de múltiples fuentes (sensores, y bases de datos).

**Múltiples niveles de atención**

- Departamentos compartiendo contenidos y datos.
  - Diagnósticos
  - Imagenología
  - Métricas de pacientes

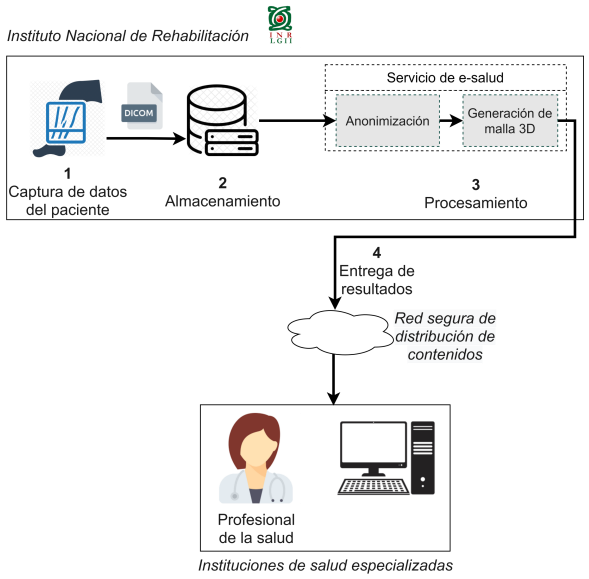



*Ejemplo de un sistema de e-salud **intra-institucionales***

http://adaptivez.org.mx/e-SaludData/e1.html

### Sistemas de e-salud **inter-institucionales**

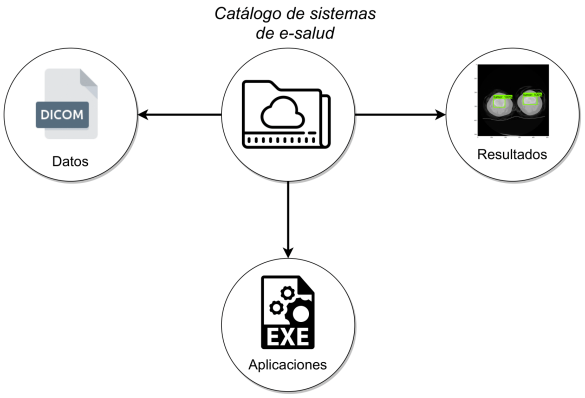
Múltiples instituciones de salud.




*Ejemplo de un sistema de e-salud inter-institucional*

## Sistema de manejo de catálogos de servicios

Interfaces para el control y acceso al catálogo de sistemas de e-salud.




*Catálogo de sistemas de e-salud, datos, aplicaciones y resultados*

**Reutiliza**

Datos, aplicaciones y sistemas de e-salud.

**Comparte**

Tus resultados con profesionales de la salud de tu organización o de otra organización.

**Accede**

A tus datos, aplicaciones y sistemas de e-salud en cualquier lugar y momento.

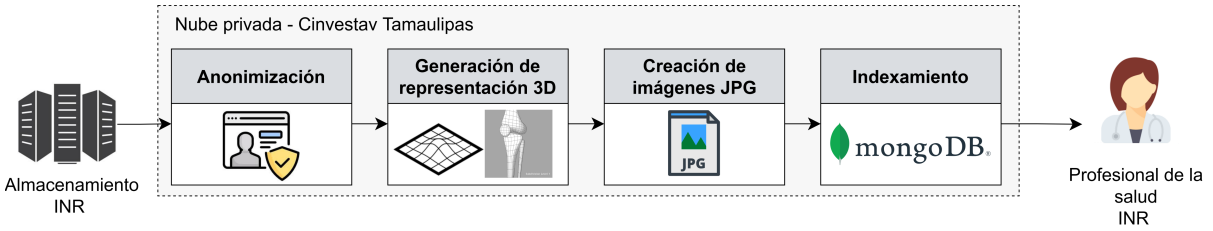


Figure 11: Póster cualitativo.

# Nez

# Servicios de construcción de sistemas de e-salud

## Ejemplo de un sistema de e-salud




*Nez construye y despliega este servicio de e-salud en **6 minutos.***

La construcción de este sistema de e-salud requiere de conocimientos avanzados de programación y arquitectura de sistemas.

**Nez** elimina estas barreras y te permite desplegar **automáticamente** sistemas de e-salud.

Nez procesa los datos en paralelo lo que reduce el tiempo de procesamiento comparado con soluciones tradicionales.

En un equipo con 8 núcleos de procesamiento: **sin paralelismo**, *25 estudios* (**37 GB** de datos) son procesados en **5.8 horas**.

En el mismo equipo: Nez procesa los 25 estudios en **1.1 horas**.
**4.9x** de aumento en la velocidad de procesamiento.

En la nube esto significa un ahorro monetario de aproximadamente **34%**. Esto al pasar de pagar 5.30 dólares aproximadamente a 3.47 dólares aproximadamente.

**45** millones de imágenes médicas almacenaba el Instituto Nacional de Rehabilitación en 2020

**43** TB de tomografías, resonancias magnéticas, rayos X, medicina nuclear, ultrasonidos

Almacenar este volumen de datos en la nube supondría un inversión aproximada de *3,440 dólares*, sin contar costos de procesamiento.

## Catálogo de aplicaciones y servicios disponibles en Nez

**Procesamiento de imágenes médicas**
- Generador de representaciones 3D
- Modelo de detección y etiquetado de imágenes
- Anonimización de datos
- Conversión de imágenes en JPG/PNG
- Indexamiento de metadatos

**Almacenamiento:**
- SkyCDS
- Algoritmos de dispersión de información (IDA)
- Algoritmos de compresión de datos (LZ4, ZIP)
- Algoritmos de balanceo de carga

**Seguridad:**
- CPABE
- AES4Sec
- Búsquedas cifradas

http://adaptivez.org.mx/e-SaludData/e1.html



Figure 12: Póster cuantitativo.