Nez

Servicios de construcción de sistemas de e-Salud

Entregable **2**
Proyecto **41756**

Dr. José Luis González Compeán
Profesor-Investigador, Cinvestav Tamaulipas

Plataforma tecnológica para la gestión, aseguramiento, intercambio y preservación de grandes volúmenes de datos en salud y construcción de un repositorio nacional de servicios de análisis de datos de salud.

Autorizaciones

Fecha: 12/11/2021	*Publicación: 15/11/2021*	*Versión: 2.0*
Seguimiento	**Nombre completo**	**Fecha**
Elaboró:	M. C. Dante Domizzi Sánchez Gallegos	05/11/2021
Revisó:	Dr. Iván López Arévalo	11/11/2021
Autorizó:	Dr. José Luis González Compeán	15/11/2021

Control de cambios

Versión	Fecha	Bitácora
1.0	05/11/2021	Documento Inicial
2.0	12/11/2021	Documento Final

Resumen

En años recientes, el uso de las tecnologías de la información y comunicación (TICs) ha jugado un rol importante en el ámbito médico, en la prevención, diagnóstico y tratamiento de patologías y enfermedades crónicas-degenerativas. En este sentido, organizaciones médicas construyen y administran sistemas de e-salud para transportar, procesar y almacenar datos médicos utilizando las TICs disponibles en la organización. En escenarios reales estos servicios de e-salud deben de hacer frente al reto de manejar grandes volúmenes de datos de forma eficiente. Por ejemplo, hasta 2020 el Instituto Nacional de Rehabilitación (INR) almacenaba 45 millones de imágenes médicas, lo que representa 43 TB de datos, los cuales deben de ser transportados y procesados para ayudar a médicos en la toma de decisiones. Otro reto al que deben de hacer frente las organizaciones para construir estos servicios de e-salud, es el de interconectar aplicaciones heterogéneas para el procesamiento de datos, que además pueden estar distribuidas en diferentes computadoras dentro de una organización, o incluso entre organizaciones. En el presente documento, se describen un conjunto de servicios de construcción de sistemas de e-salud. Estos servicios permiten la encapsulación y manejo de aplicaciones como cajas negras llamadas bloques de construcción, las cuales cuentan con interfaces de entrada y salida para permitir su interconexión para construir sistemas de e-salud. Estos sistemas de e-salud son independientes de la infraestructura, y pueden ser desplegados tanto en computadoras personales, como en clústeres de alto rendimiento y en la nube (por ejemplo, en Amazon EC2 o Google Cloud). Además, los sistemas de e-salud construidos, así como los bloques de construcción, son colocados en un repositorio, para permitir bloques construcción y servicios de e-salud entre organizaciones.

1. Tabla de contenido

2. Introducción

Las tecnologías de la información como el internet de las cosas (IoT, por sus siglas en inglés) [1] [2] así como los algoritmos de análisis de datos y aprendizaje máquina [3], son consideradas por las organizaciones y la comunidad científica como un pilar en la construcción de sistemas de e-salud, los cuales mejoran radicalmente el acceso y eficiencia de los servicios de salud tradicionales. En este sentido, dichos sistemas son utilizados para adquirir datos de los pacientes (por ejemplo, signos vitales o actividad) y ayudar a profesionales de la salud en la toma de decisiones. Además, profesionales de la salud y organizaciones médicas pueden obtener hallazgos importantes al procesar los datos adquiridos desde dispositivos IoT, además del procesamiento de imágenes médicas, estudios, expedientes clínicos y registros históricos de datos [4].

Para encontrar estos hallazgos en los datos, las organizaciones deben de procesarlos a través de un conjunto de tareas a través del ciclo de vida de los datos [5]. En un escenario típico de procesamiento y manejo de datos médicos, estos son adquiridos de una o distintas fuentes (por ejemplo, de tomógrafos o electrocardiogramas), posteriormente procesados con algoritmos y aplicaciones (por ejemplo, para anonimizar imágenes médicas, o procesar los datos utilizando algoritmos de aprendizaje automático), y finalmente los resultados son visualizados utilizando herramientas de visualización de datos. En el flujo descrito anteriormente, son utilizadas un conjunto heterogéneo de aplicaciones, las cuales se encuentran desarrolladas en distintos lenguajes de programación y para diferentes plataformas. Además, regularmente dichas aplicaciones son distribuidas en diferentes equipos dentro de la organización o incluso al exterior de la organización, ya sea utilizando recursos en la nube o de otra organización [6].

En escenarios reales, estas aplicaciones son organizadas en etapas de procesamiento, y deben de ser ejecutadas automáticamente para soportar el proceso de toma de decisiones. Además, el intercambio de datos entre cada una de las etapas se debe de hacer de manera automática y eficiente, desde que los datos son adquiridos hasta que son entregados a los pacientes o profesionales de la salud. En este sentido, durante el intercambio y almacenamiento de los datos, diferentes requerimientos no funcionales deben ser manejados para cumplir con las normas establecidas por la institución o instancias gubernamentales [7] [8].

Crear estas estructuras de forma flexible y portable para permitir la interoperabilidad de aplicaciones heterogéneas, que además puedan ser desplegadas en diferentes infraestructuras (por ejemplo, en la nube o una computadora personal) no es una tarea sencilla.

En el presente documento se presenta un conjunto de servicios de construcción de sistemas de e-salud para el procesamiento y análisis de datos clínicos. Los servicios de e-salud construidos con estos servicios de construcción tienen como principales ventajas las siguientes características:

- **Modularidad**: los sistemas de e-salud se encuentran construidos utilizando abstracciones conocidas como bloques de construcción. Dichos bloques de construcción son autónomos e independientes, por lo que no afectan el funcionamiento de otros bloques, por lo tanto, pueden ser remplazados por otros bloques de construcción sin necesidad de construir un sistema de e-salud nuevo.
- **Agnosticidad**: tanto los sistemas de e-salud, como los bloques de construcción que conforman estos sistemas, son agnósticos. Es decir, estos sistemas pueden ser desplegados en diferentes infraestructuras sin necesidad de hacer cambios en el código de las aplicaciones. Los

sistemas de e-salud pueden ser desplegados en diferentes plataformas y sistemas operativos, por ejemplo, en sistemas Windows o Linux, ya sea en una computadora personal o en la nube.

- **Eficiencia**: los sistemas de e-salud construidos integran patrones de paralelismo implícitos, los cuales mejoran la eficiencia para procesar datos reduciendo los tiempos de respuesta de las aplicaciones.
- **Portabilidad**: los sistemas de e-salud construidos pueden ser trasladados de una infraestructura a otra sin requerir hacer cambios en el sistema o las aplicaciones.
- **Reusabilidad**: mediante el repositorio y catálogo de sistemas y aplicaciones, es posible que una organización reutilicé un sistema de e-salud (o parte de él).

3. Servicios de construcción de sistemas de e-Salud

Nez es un conjunto de servicios que permite a las organizaciones de salud y la comunidad científica crear sistemas de e-Salud agnósticos de la infraestructura para el procesamiento y manejo de grandes volúmenes de datos médicos.

Nez permite crear sistemas de e-Salud de forma automática mediante interfaces gráficas y sin tener conocimientos avanzados de programación. Dichos sistemas de e-Salud se crean mediante el encadenamiento de dos o más aplicaciones para el procesamiento y manejo de datos médicos. Además, los servicios de e-Salud pueden ser manejados internamente por una organización (servicio de e-Salud intra-institucional) o por múltiples organizaciones (servicio de e-Salud inter-institucional).

Nez incluye los siguientes productos:

- Un esquema de construcción de cripto-contenedores de datos y cripto-contenedores de aplicaciones.
- Un esquema de despliegue de e-Servicios independientes de la infraestructura.
- Un esquema de bloques de construcción de flujos de trabajo y servicios de e-Salud basado en mapas de microservicios y nanoservicios.

3.1 Esquema de bloques de construcción de flujos de trabajo y servicios de e-Salud basado en mapas de microservicios y nanoservicios

En la Figura 1 se muestra el proceso de construcción automática de sistemas de e-salud utilizando el esquema de construcción de flujos de trabajo y servicios de e-salud desarrollado como parte del presente proyecto. El proceso de

construcción de sistemas de e-salud se compone de tres etapas principales, las cuales se encuentran ilustradas en la Figura 1. Estas etapas son: creación de bloques de construcción, diseño de la estructura de procesamiento, y despliegue de la estructura.

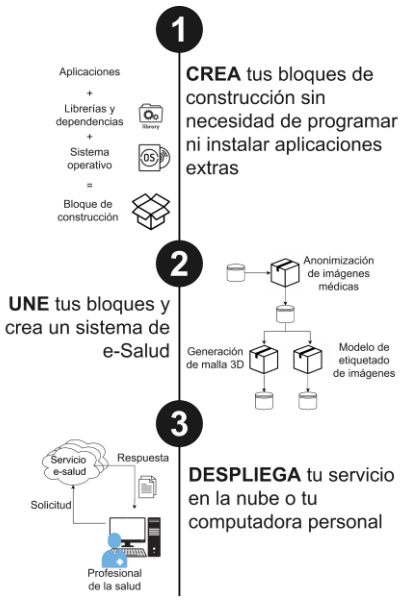


Figura 1. Construcción automática de sistemas e-salud.

La primera etapa es la creación de los bloques de construcción. Este proceso consiste en encapsular en contenedores virtuales el código fuente o ejecutables de la aplicación que será ejecutada en el bloque de construcción. Además, dentro del contenedor virtual son colocadas las dependencias de software (librerías, frameworks, etc.) de la aplicación encapsulada, así como estructuras de control utilizadas durante tiempo de ejecución para invocar al bloque de construcción.

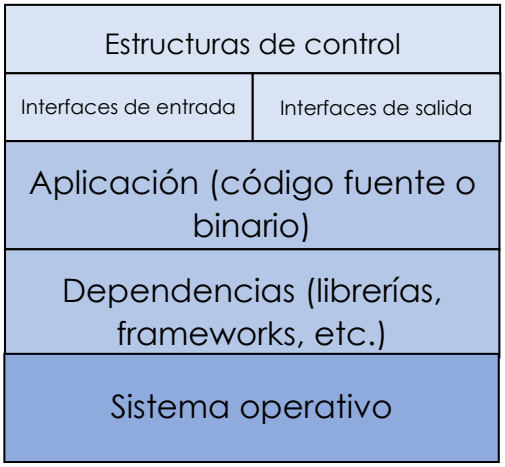


Figura 2. Representación conceptual de un bloque de construcción en Nez.

En la Figura 2, se presenta la representación conceptual de los bloques de construcción manejados en Nez. Como se mencionó anteriormente, los componentes del bloque de construcción son las estructuras de control que permiten la invocación del bloque, así como la comunicación con otros bloques. Lo anterior, mediante interfaces de entrada y salida, ya sea mediante la red o el sistema de archivos. La aplicación y sus dependencias también son agregadas al bloque de construcción. Esta aplicación es utilizada para procesar los datos de entrada que lleguen al bloque durante tiempo de ejecución. Finalmente, al implementar dichos bloques de construcción como contenedores virtuales, un sistema operativo base debe de ser especificado, por ejemplo, Ubuntu o CentOS.

La segunda etapa para la construcción de sistemas de e-salud, es el diseño de la estructura de procesamiento del sistema de e-salud. En este paso, los bloques de construcción son organizados por el diseñador, de tal manera que la salida de una etapa es la entrada de la siguiente etapa en la estructura. La idea básica

de esta etapa es que el diseñador construya un flujo de trabajo[1] interconectado cada uno de los bloques de construcción y eligiendo las fuentes de datos a procesar. En este sentido, diferentes patrones de procesamiento pueden ser diseñados. En la Figura 3 se presentan algunos de los patrones básicos que es posible construir en un flujo de trabajo. Dichos patrones son:

a) **Secuencial:** Una actividad en el proceso del flujo de trabajo es ejecutada inmediatamente después de que la actividad anterior ha sido completada.
b) **Paralelo:** En algún punto del flujo de trabajo se pueden ejecutar diferentes hilos para ejecutar actividades en paralelo.
c) **Sincronización:** Se utiliza cuando dos actividades paralelas convergen en una tercera actividad.
d) **Elección exclusiva:** Se da cuando se debe de elegir cuál es la siguiente actividad en ser ejecutada a partir de un conjunto de actividades candidatas. Para ello se implementa un conjunto de condicionales en el flujo de trabajo.
e) **Mezcla simple:** Sucede cuando en algún punto del flujo de trabajo dos o más actividades, que no se ejecutan paralelamente, se unen sin sincronización.

[1] Un flujo de trabajo es la formalización de un proceso científico o de ingeniería, que permite el despliegue automático de tareas y aplicaciones en una infraestructura de hardware, por ejemplo, en un clúster privado o en la nube **Invalid source specified.**.

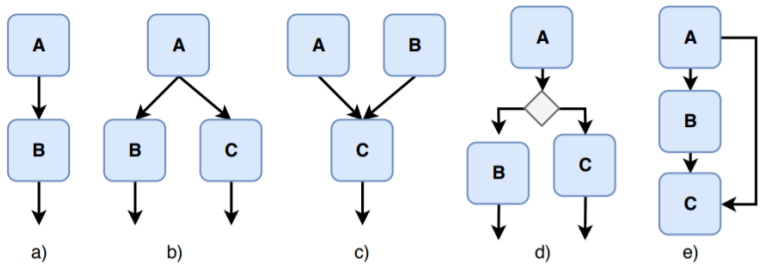


Figura 3. Patrones básicos utilizados en un flujo de trabajo. a) Secuencial. b) Paralelo. c) Sincronización. d) Elección exclusiva. e) Mezcla simple.

En la tercera etapa, la estructura de procesamiento diseñada es puesta en ejecución. Para ello se realiza el despliegue automático de los bloques de construcción (contenedores virtuales) en la infraestructura especificada. En este sentido, la estructura de procesamiento puede ser desplegada en un solo equipo o en múltiples equipos.

3.1.1 Contenerización de aplicaciones para crear bloques de construcción

Para crear un bloque de construcción, el primer paso es crear contenerizar las aplicaciones que se utilizarán para crear el sistema de e-salud. Actualmente, los contenedores son construidos utilizando la plataforma de contenedores Docker[2].

En Docker, uno de los métodos para crear contenedores virtuales es mediante la programación de archivos conocidos como *Dockerfile*. En estos archivos se declara la imagen base de la aplicación, y además se configura la imagen de contenedor virtual indicando la instalación de dependencias de software, así como el directorio de trabajo dentro del contenedor. Además, los archivos Dockerfile incluyen comandos para copiar el código fuente o ejecutables desde el sistema de archivos anfitrión al contenedor virtual.

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

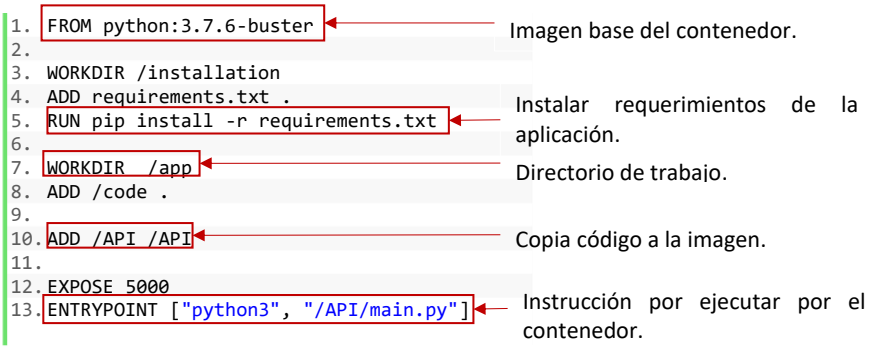


Figura 4. Ejemplo de un archivo Dockerfile.

En la Figura 4 se muestra un ejemplo de un archivo Dockerfile. Como se puede notar, en el inicio del archivo se indica la imagen de contenedor base de la aplicación. Ejemplo de imágenes base incluyen ubuntu:18.04, Python:2.04 o centos:6. Posteriormente, en el archivo se indican los requerimientos que se desean instalar utilizando comandos como *apt-get* en Ubuntu, *yum install* en CentOS, o *pip* en Python.

3.1.2 Creación de microservicios y nanoservicios

En Nez, los bloques de construcción son manejados como microservicios o nanoservicios, dependiendo de las características de la infraestructura donde se despliegue el bloque de construcción. Por ejemplo, si la infraestructura donde se despliega está limitada por el hardware o no es posible instalar una plataforma de contenedores, por ejemplo, en un teléfono inteligente o una computadora personal antigua, entonces el bloque de construcción puede ser desplegado como un nanoservicio, que básicamente es una plantilla que invoca una función para recibir o enviar datos. Mientras, que un microservicio es conveniente utilizarlo cuando el bloque de construcción es desplegado en infraestructura con capacidad para procesar los datos de entrada, por ejemplo, tomografías o datos de electrocardiograma.

En este sentido, el modelo de construcción en Nez permite el acoplamiento de estos microservicios y nano-servicios para construir sistemas de e-salud para el procesamiento de grandes volúmenes de datos. El modelo de construcción está basado en una metáfora de rompecabezas para la construcción de sistemas de e-salud angosticos[3] de la infraestructura [9]. En esta metáfora un bloque de construcción es una *pieza de rompecabezas* que representa una pieza de software que incluye componentes modulares, así como estructuras de control que permiten que la pieza sea independiente del resto, y, por lo tanto, es posible aislar esta pieza del resto para invocarla desde herramientas externas mediante un API, procesar datos y obtener los productos resultantes. Además, las interfaces de entrada y salida (E/S) de la pieza son abstraídas como los *dientes* y *bahías de enchufe* de la pieza de rompecabezas. Estas interfaces de E/S se encargan de manejar los datos recibidos y producidos por las aplicaciones en la pieza de rompecabezas. En este sentido, los resultados son entregados a la siguiente pieza en el sistema de e-salud o a un recurso de almacenamiento, por ejemplo, una red de distribución de contenidos.

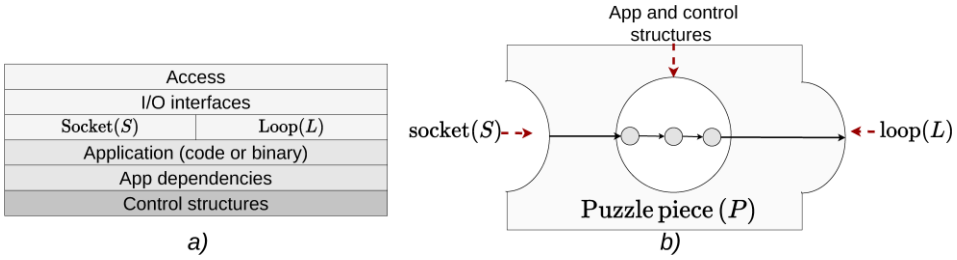


Figura 5. Arquitectura de pila de una pieza (a) y su representación conceptual (b).

[3] Se dice que son agnósticos dado que se encuentran desacoplados de la infraestructura, por lo tanto, es posible migrar los servicios a otra infraestructura sin hacer cambios en las aplicaciones y sin modificar el comportamiento del sistema de e-salud.

En la Figura 5 se muestra la arquitectura en pila (a) de una pieza de rompecabezas y sus componentes, los cuales son:

a) Capa de acceso: se encarga de verificar los tokens y credenciales de acceso para garantizar que solo usuarios, piezas y aplicaciones autorizadas tengan acceso a la pieza y los datos que maneja la pieza, lo cual incluye los datos recibidos, así como los resultados generados.
b) Interfaces E/S: permiten la interconexión de una pieza con otras piezas de rompecabezas, así como con aplicaciones externas.
    a. *Bahías de enchufe:* se encargan de leer los contenidos o datos que recibe la pieza de rompecabezas desde una fuente de datos u otra pieza.
    b. *Dientes:* escriben los resultados producidos en un resumidero de datos o los entregan a otra pieza en el sistema de e-salud.
c) Aplicación: es el software que procesará los datos durante tiempo de ejecución. Este puede ser el código fuente o ejecutable (binarios) de una aplicación.
    a. Dependencias: son las dependencias de software de la aplicación.
d) Estructuras de control: se encargan de manejar y coordinar la ejecución de la aplicación contenida en la pieza de rompecabezas.

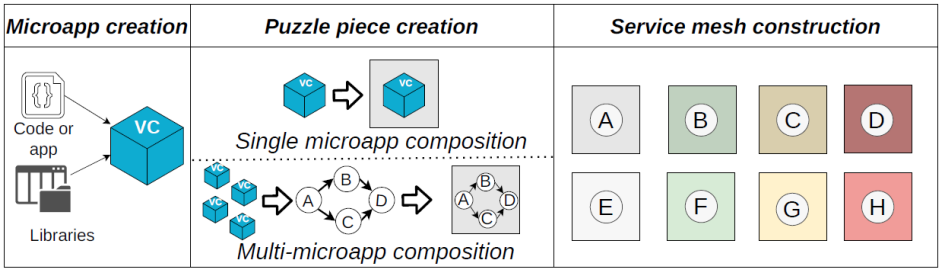


Figura 6. Representación conceptual de la construcción de una malla de servicios basada en la metáfora de rompecabezas.

En la Figura 6, se muestra el proceso de encapsulación de aplicaciones en contenedores, la composición de múltiples aplicaciones (A, B, C y D) en una pieza de rompecabeza, así como la creación de una malla de servicios para almacenar e indexar las piezas construidas por los diseñadores y usuarios de Nez.

En Nez, un rompecabezas representa una estructura de procesamiento que es construido acoplando diferentes piezas, las cuales son seleccionadas de la malla de servicios. Cuando la estructura de procesamiento o rompecabezas es desplegada en una infraestructura se conoce como sistema de e-salud. Un rompecabezas es un patrón (por ejemplo, una tubería o un flujo de trabajo) que maneja el ciclo de vida de los productos digitales (por ejemplo, tomografías, rayos X, o datos de electrocardiograma) producidos por uno o múltiples hospitales. Para exponer una estructura de procesamiento como un servicio, un rompecabezas incluye estructuras de control que implementan una estructura de microservicios para el manejo de las piezas (bloques de construcción y aplicaciones), metadatos (por ejemplo, orden de ejecución), una API Rest usada para intercambiar mensajes entre las piezas, así como un API para invocar una red de distribución de contenidos incluida en Zamná para intercambiar datos entre las piezas.

Commented [DDSG1]: Nombre del entregable 2

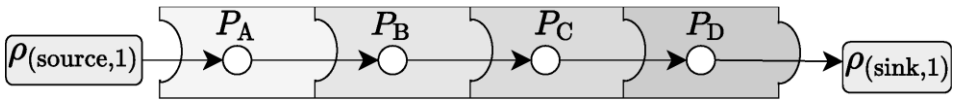


Figura 7. Representación conceptual de una tubería construida como un rompecabezas.

En la Figura 7 se muestra un ejemplo de un rompecabezas construido como una tubería de piezas. Este rompecabezas incluye cuatro piezas (P_A, P_B, P_C, y P_D) conectado a una fuente de datos ($fuente\_1$) que procesa los contenidos entrantes y entregando los resultados a un resumidero de datos ($resumidero\_1$). Como resultado, este rompecabezas es manejado como un servicio. En este

ejemplo, cada una de las piezas puede estar desplegada en una infraestructura diferente e incluso en diferentes organizaciones. Por ejemplo, la fuente de datos, puede ser el sistema PACS de una organización, en donde se almacenan imágenes médicas del Instituto Nacional de Radiología (INR). En este escenario, dentro del INR las imágenes son anonimizadas por la primera pieza (P_A), y enviadas a $P_B$la cual es una pieza en la una instancia en Amazon EC2 y que se encarga de procesar las imágenes para transformarlas en formato JPG, y entregarlas a la pieza P_c para el etiquetado de tumores en la imagen. Finalmente, los resultados son entregados a $P_D$el cual es un visualizador en un centro de salud especializado que permite a los oncólogos consultar las imágenes etiquetadas y descargarlas en su equipo (resumidero de datos).

3.2 Esquema de construcción de cripto-contenedores de datos y cripto-contenedores de aplicaciones

Nez implementa un esquema de construcción de cripto-contenedores de datos y aplicaciones con el objetivo de observar, en forma automática y transparente, las normas oficiales (NOM-024-SSA3-2010 y NOM-004-SSA3-2012) e internacionales (ISO-270001-13, COBIT5, NIST) garantizando privacidad, confidencialidad, integridad, disponibilidad de los contenidos, tolerancia a fallas de servicios/servidores y trazabilidad. En este sentido, estos esquemas son agregados a las piezas de rompecabezas que hacen parte de los servicios de e-salud construidos con Nez. Para distribuir los datos entre diferentes infraestructuras, Nez está conectado con Muyal-Zamná crea redes de cripto-contenedores de datos y blockchain para verificar que los sistemas de e-salud intercambien dato de forma segura.

Los requerimientos no funcionales contemplados en estos esquemas de cripto-contenedores de datos y aplicaciones son: seguridad, confiabilidad y eficiencia. Los servicios de seguridad son agregados para resolver problemas que surgen

cuando los datos e información son manejados y compartidos con múltiples usuarios a través de ambientes no controlados. Este es el caso de la nube y en general de modelos de subcontratación de recursos, donde los usuarios pierden el control físico sobre los datos. En este contexto, la integridad de datos, confidencialidad de datos y control de acceso son aspectos de seguridad importantes a considerar.

Los servicios de confiabilidad son requeridos para solucionar problemas relacionados con fallas en la infraestructura donde los datos son procesados y almacenados. Lo anterior, dado que las fallas en la infraestructura resultan en que los datos no estén disponibles para su consulta/descarga. Este requerimiento resulta clave para evitar que los usuarios y organizaciones sufran de los efectos de no disponibilidad de los datos.

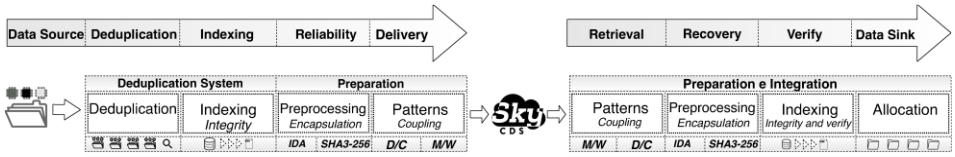


Figura 8. Representación conceptual de esquemas para le preparación y entrega de datos en la nube utilizando cripto-contenedores de datos y aplicaciones.

En la Figura 8 se muestra la representación conceptual de una aplicación cliente que carga contendidos a la nube utilizando los esquemas de cripto-contenedores de datos (ver lado izquierdo de la figura). Además, en la figura también se muestra el cliente de descarga de contenidos (ver lado derecho de la figura). Como se puede observar en la figura, los esquemas incluyen etapas para agregar requerimientos no-funcionales a los datos y ejecutar las aplicaciones en forma segura. En este sentido, las etapas contempladas son: deduplicación e indexamiento para procesar los datos de forma eficiente, confiabilidad utilizando el algoritmo de dispersión de información (IDA, por sus

siglas en inglés), y la entrega de los datos utilizando técnicas de criptografía para agregar seguridad a los datos antes de su transporte. La distribución de los datos entre las diferentes infraestructuras se realiza utilizando una red de distribución de contenidos construida con Muyal-Zamná. Además, estos esquemas permiten agregar y remover etapas para generar diferentes tuberías de preparación de datos. Por ejemplo, agregar un algoritmo de compresión de datos para reducir el volumen de los archivos, o sustituyendo el algoritmo de cifrado de datos o el tamaño de las llaves. El cliente de descarga de datos incluye etapas para recuperar los datos preparados y cargados con el cliente de carga de datos, la verificación de la integridad de los datos recuperados, y la colocación de estos datos en las computadoras de los usuarios autorizados para acceder a los datos.

La construcción de los cripto-contenedores de datos y aplicaciones se compone de tres etapas principales:

i. Definición de los participantes autorizados para acceder a los datos y aplicaciones dentro de los contenedores, así como la secuencia valida de interacciones al compartir e intercambiar datos.
ii. En la segunda etapa, los diseñadores pueden elegir los bloques de seguridad y de requerimientos no-funcionales para construir un cripto-contenedor. Dichos bloques son seleccionados de repositorio de servicios.
iii. En la tercera etapa, los parámetros de los cripto-contenedores son seleccionados. Dentro de estos parámetros se incluyen el nivel de seguridad (e.g., tamaño de la llave), el número de trabajadores para procesar los datos dentro del cripto-contenedor y los parámetros de cada requerimiento no-funcional considerado.

Un cripto-contenedor es creado por cada etapa en la estructura de procesamiento del servicio de e-Salud.

3.3 Esquema de despliegue de sistemas de e-salud independientes de la infraestructura

Una vez que se han construido los bloques de construcción y diseñado el sistema de e-salud, lo siguiente es desplegarlos en la infraestructura en donde será ejecutado el sistema. En este sentido, el sistema y sus bloques pueden ser desplegados tanto en un solo equipo, ya sea en la nube o en una computadora personal, o de forma distribuida en un clúster de cómputo de alto desempeño dentro de una organización o desplegando los bloques en infraestructuras de diferentes organizaciones y/o la nube. En este sentido, dos tipos de sistemas de e-salud pueden ser desplegados: intra-institucionales e inter-institucionales.

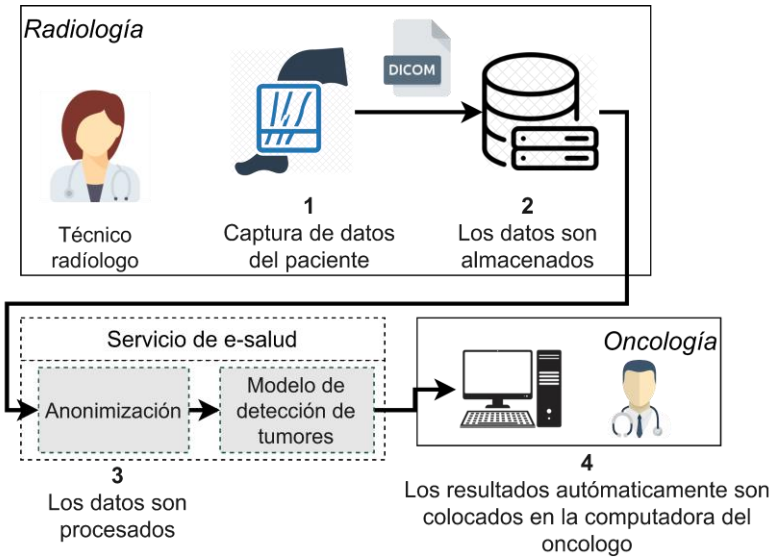


Figura 9. Ejemplo de un sistema de e-salud intra-institucional.

Los sistemas de e-salud intra-institucionales son aquellos permiten a las instituciones y organizaciones de salud procesar e intercambiar datos entre

profesionales de la salud y departamentos dentro la organización u hospital. Por ejemplo, en la Figura 9 se muestra un ejemplo de un sistema de e-salud intra-institucional. En este ejemplo, un técnico radiólogo realiza la captura de tomografías del paciente en el Instituto Nacional de Rehabilitación. Las imágenes son almacenadas, y el radiólogo las comparte con oncólogo del instituto para que las valore y de su diagnóstico. En este sentido, las imágenes antes de ser compartidas con el oncólogo son procesadas mediante un servicio de e-salud automático construido para anonimizar las imágenes y etiquetarlas utilizando un modelo de detección de tumores. Los resultados son entregados al oncólogo de forma automática en su computadora.

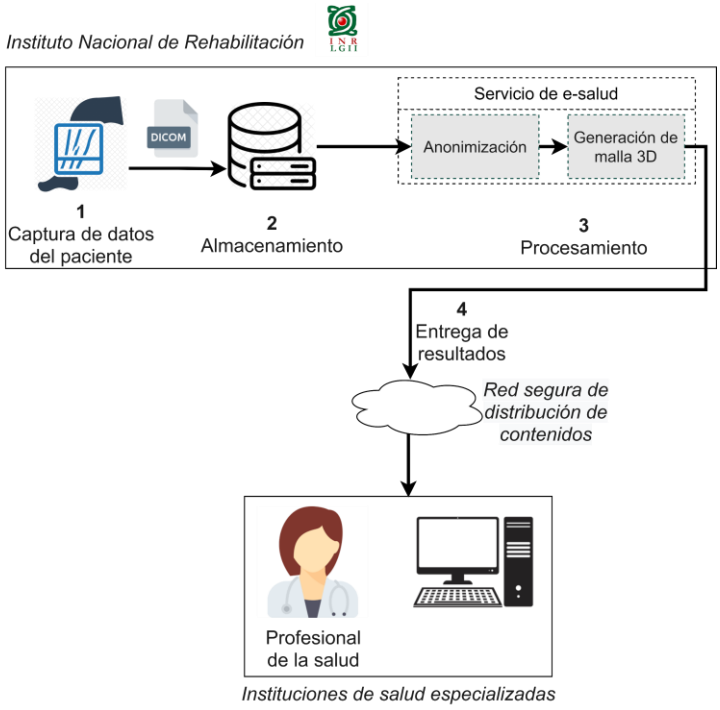


Figura 10. Ejemplo de un sistema de e-salud inter-institucional.

Los sistemas de e-salud inter-institucional son creados para compartir datos entre diferentes instituciones, organizaciones u hospitales. En este tipo de sistemas, los datos son distribuidos utilizando una red segura de distribución de contenidos, la cual automáticamente distribuye los datos a los participantes de las organizaciones que tenga autorización para acceder a los datos. Por ejemplo, en la Figura 10 se muestra un sistema de e-salud inter-institucional creado para intercambiar datos entre el Instituto Nacional de Rehabilitación (INR) y una Institución de salud especializada. Los datos son capturados en el INR y almacenados en su infraestructura. Posteriormente, los datos son inyectados automáticamente en un sistema de procesamiento de e-salud el cual anonimiza los datos y realiza la generación de una malla 3D de las imágenes DICOM. Los resultados, son entregados a la red segura de distribución de contenidos, y posteriormente son automáticamente colocados en las computadoras de los profesionales de salud con acceso a los datos.

3.3.1 Principios de diseño de un prototipo para el manejo de servicios de e-salud

El manejador de servicios de e-Salud se encarga de coordinar y orquestar los servicios en tiempo de ejecución en la arquitectura de microservicios. Para ello, en este bloque se tienen un conjunto de servicios que se encargan de controlar el acceso de los usuarios a los servicios, así como que el intercambio de datos entre las organizaciones y las aplicaciones se lleve a cabo de acuerdo con las Normas Oficiales Mexicanas. Este bloque, además incluye servicios de manejo de catálogos mediante el modelo de publicación/subscripción, un servicio para el intercambio de datos y diagnósticos, servicios para el análisis de datos médicos y servicios para la adquisición de datos desde repositorios médicos utilizando protocolos tales como DICOM y HL7.

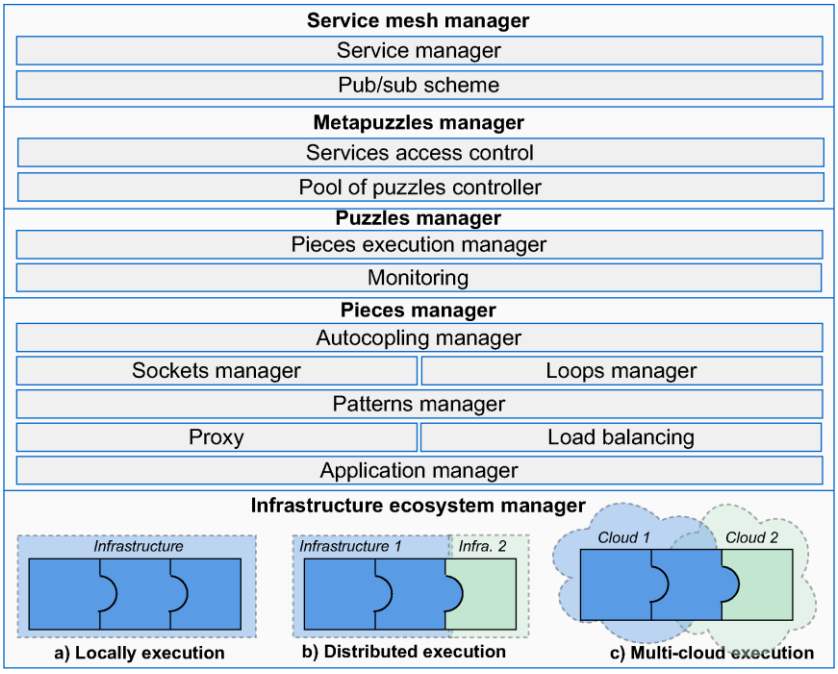


Figura 11. Arquitectura para el manejo de servicios de e-salud y sus componentes (piezas y rompecabezas).

La realización de este manejador de servicios de e-Salud se basa en una arquitectura de jerárquica para el manejo de los servicios de e-salud, rompecabezas, así como las piezas que lo componen. En la Figura 11 se muestra la representación conceptual de esta arquitectura. Como se puede observar, esta arquitectura está compuesta de manejadores para la malla de servicios, rompecabezas, piezas y la infraestructura.

En este sentido, la construcción de un sistema de e-salud utilizando esta arquitectura se compone de las siguientes etapas: etapa de diseño, etapa de construcción, etapa de despliegue y orquestación, etapa de lanzamiento y coreografía, y la etapa de ejecución y operación.

En la fase de diseño, los diseñadores de las organizaciones crean un sistema de e-salud seleccionando un conjunto de piezas y rompecabezas de la malla de servicios (repositorio de servicios). En el caso de que en la malla de servicios no se encuentre la aplicación de procesamiento requerida por la organización, los diseñadores pueden crear una nueva pieza utilizando un esquema declarativo indicando el nombre de la pieza, su comando de ejecución, y el código fuente de la aplicación incluyendo un archivo Dockerfile. Las entradas y salidas para cada pieza, así como su orden de ejecución son establecidas por los diseñadores. Como resultado de esta etapa se produce un grafo acíclico dirigido, el cual es convertido en un archivo de configuración en formato YML.

En la fase de construcción, el manejador de malla de servicios recursivamente invoca a los manejadores de piezas y rompecabezas para la construcción del sistema de e-salud siguiendo el archivo de configuración creado por el diseñador.

En la fase de despliegue y orquestación, el manejador de malla de servicios crea las imágenes de contenedor de las piezas consideradas en el archivo de configuración. En un segundo paso, los manejadores de piezas y rompecabezas leen los metadatos de manejo de cada componente para preparar su funcionamiento (agregando aplicaciones y estructuras de control). Además, durante esta fase, un manejador de patrones crea y acopla los componentes requeridos para crear patrones de paralelismo al interior de las piezas, con el objetivo de mejorar su eficiencia. Cabe recalcar, que esta característica puede ser deshabilitada durante tiempo de diseño por el diseñador.

En la etapa de lanzamiento y coreografía, los manejadores de piezas y rompecabezas recursivamente realizan el despliegue de los contenedores correspondientes a cada pieza en el sistema de e-salud. Posteriormente, en esta etapa automáticamente se realiza el acoplamiento de las piezas siguiendo las

entradas y salidas definidas en el archivo de configuración. Como resultado, en esta etapa se genera un conjunto de instancias de contenedor organizadas en la forma de un patrón.

En la fase de ejecución y operación, la estructura de procesamiento (sistema de e-salud) es expuesta como un servicio haciéndola disponible a los usuarios autorizados para su consumo.

Los manejadores de la infraestructura incluidos en la arquitectura, implementan clientes y APIs para la comunicación de los manejadores con la infraestructura en donde se despliegue y opere el sistema de e-salud. Mientras que el manejador de mallas de servicios controla los servicios resultantes utilizando un modelo de publicación/suscripción.

3.3.2 Plataforma web para la construcción de sistemas de e-salud

Con el objetivo de acceder a los servicios previamente descritos para la construcción y despliegue de sistemas de e-salud, se desarrolló una plataforma web. A través de esta plataforma, los diseñadores pueden diseñar sus piezas y rompecabezas para posteriormente materializar un sistema de e-salud desplegando en la infraestructura disponible en la organización.

3.3.2.1 Preliminares

La plataforma web fue desarrollada y probada en un sistema de contenedores Docker, utilizando como sistema operativo base Ubuntu 18.04. El servicio de despliegue y acoplamiento de servicios de e-salud fue desarrollado en C++17, mientras que los clientes de carga y descarga de datos se encuentran desarrollados en Java 8. La lista de requisitos de software para el funcionamiento de la plataforma es:

- Docker CE[4]
    - Docker Compose[5]
    - Docker Swarm[6] (opcional)

Para desplegar la plataforma web, descarga el código fuente de la siguiente ruta: descarga. Navegar a la ruta donde se descomprimió el archivo descargado, y ejecutar el siguiente comando para desplegar la plataforma con Docker Compose:

```
docker-compose up -d
```

Los servicios desplegados por Docker Compose, y que se encuentran declarados en el archivo YML, son los siguientes:

- Servicios de despliegue y construcción de sistemas de e-salud:
    - Interfaz web (nombre del servicio: web-gui).
    - API de construcción (nombre del servicio: build-api).
    - Base de datos de sistemas de e-salud (nombre del servicio: services-db).
    - Servicio de despliegue de servicios de e-salud (nombre del servicio: deployer).
- Servicios de publicación/suscripción y autenticación.
    - Servicio de manejo y redireccionamiento de peticiones (nombre del servicio: apigateway).
    - Servicio de autenticación de usuarios (nombre del servicio: auth).
    - Base de datos de autenticación (nombre del servicio: db_auth).

[4] https://docs.docker.com/get-docker/

[5] https://docs.docker.com/compose/

[6] https://docs.docker.com/engine/swarm/

- Servicio de publicación/suscripción de catálogos (nombre del servicio: pub_sub).
- Base de datos del servicio de publicación/suscripción (nombre del servicio: db_pub_sub).

3.3.2.2 Interfaz gráfica web

La plataforma web tiene una interfaz gráfica en el puerto 22101[7]. Una vez que se ingresa a la plataforma, se debe de iniciar sesión o en caso de no tener una cuenta, se debe de registrar una nueva. Para registrar la cuenta, se debe de llenar un formulario con un nombre de usuario, correo electrónico, contraseña y organización a la que pertenece el usuario. El usuario debe de validar su dirección de correo electrónico. Una vez realizado lo anterior, el usuario podrá ingresar a la plataforma.

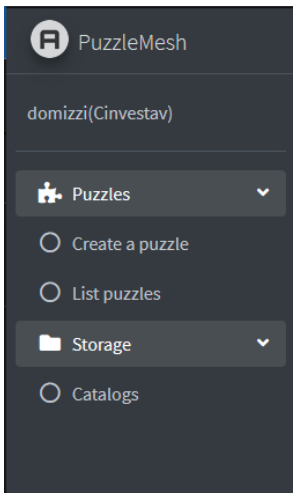


Figura 12. Menú de navegación de la plataforma web.

[7] Este es el puerto declarado por defecto en el archivo YML. Usted puede cambiar el puerto si lo desea.

Al ingresar a la plataforma, se mostrará un panel de administración y manejo de sistemas de e-salud. En la Figura 12. Menú de navegación de la plataforma web. Figura 12 se muestra el menú de navegación de la plataforma web. El menú contiene dos secciones principales. La primera sección permite crear y visualizar los rompecabezas creados (sistemas de e-salud), mientras que la segunda sección permite manejar los catálogos de datos del usuario y/compartidos con el usuario.

3.3.2.3 Creación de un servicio de e-salud

Para crear un servicio de e-salud, se deben de acoplar un conjunto de piezas de rompecabezas, definiendo entradas y salidas. Lo anterior generará un rompecabezas que durante tiempo de ejecución es manejado como un servicio de e-salud, que toma los datos desde la fuente de datos especificado (un catálogo de datos en la red de distribución de contenidos o un directorio en el equipo donde se despliegue el sistema a de e-salud.

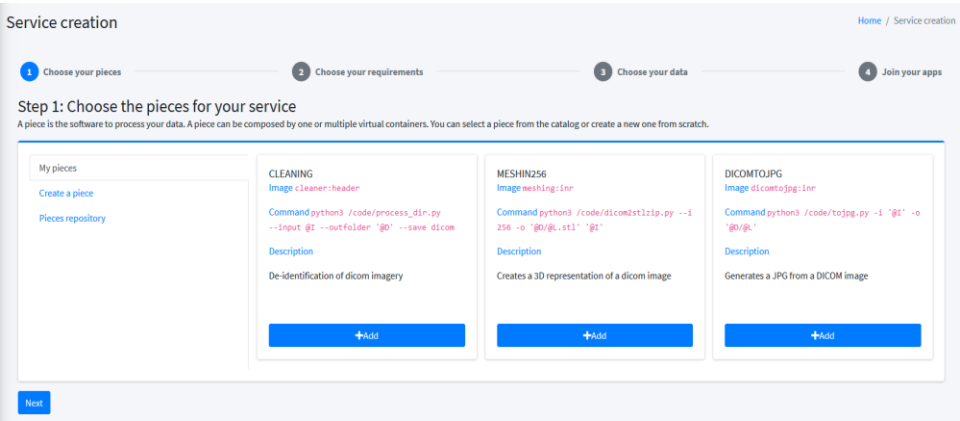


Figura 13. Interfaz gráfica para la creación de sistema de e-salud.

En la Figura 13 se muestra la pantalla para crear un sistema de e-salud. La creación de estos sistemas se compone de cuatro pasos básicos:

1. **Elige tus piezas**: en este paso el diseñador elige las piezas que formarán parte del rompecabezas que será desplegado como un servicio de e-salud. En esta pantalla, también es posible crear una pieza si esta no existe en el repositorio de piezas.

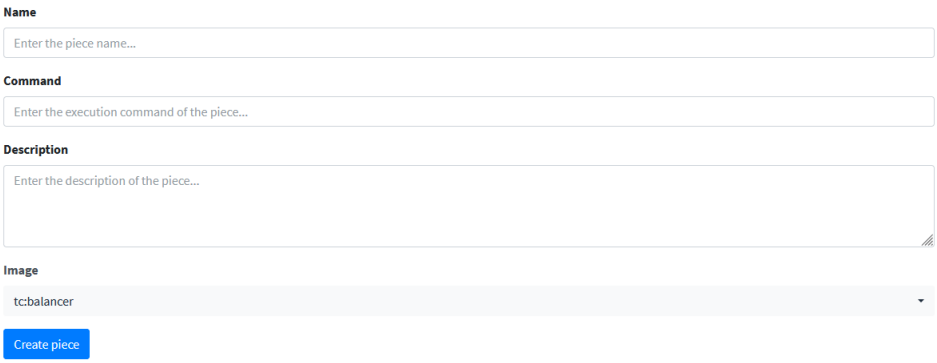


Figura 14. Formulario para crear una nueva pieza de rompecabezas.

En la Figura 14 se muestra el formulario para crear una nueva pieza de rompecabezas, la cual es una aplicación encapsulada en un contenedor virtual y que se utiliza para crear un rompecabezas que al desplegarlo se materializa en un servicio de e-salud. En el formulario se debe de indicar el nombre de la pieza, el comando que ejecutará la pieza durante tiempo de ejecución (por ejemplo, *python3 app.py*), una breve descripción de la pieza, y finalmente la imagen de contenedor que se usará para crear el contenedor de la pieza de rompecabezas.

2. **Elige tus requerimientos**: en este paso el diseñador elige los requerimientos no funcionales que serán agregados a los resultados del servicio de e-salud al almacenarlos.

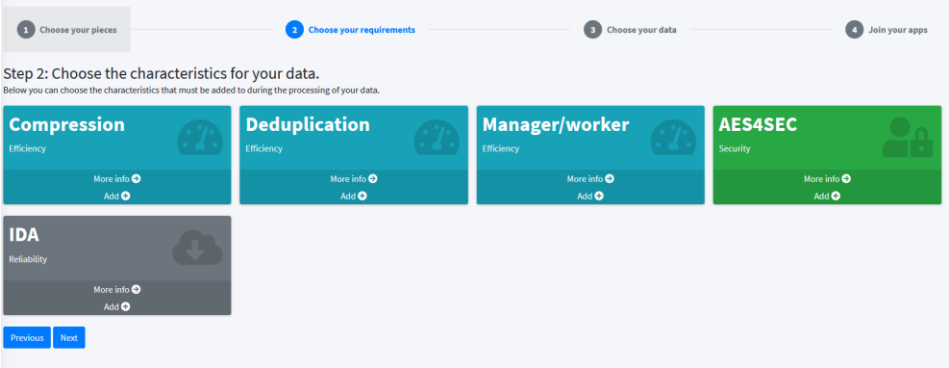


Figura 15. Pantalla de selección de requerimientos no funcionales.

En la Figura 15 se muestra la pantalla de selección de los requerimientos no funcionales para agregarlos al sistema de e-salud. Para agregar un requerimiento no funcional, se debe de agregar la técnica que añade dicho requerimiento, por ejemplo, compresión de datos para agregar eficiencia en el manejo y almacenamiento de datos, el patrón manejador/trabajador (manager/worker) para agregar eficiencia al procesamiento de datos, o AES4SEC para agregar seguridad a los datos. Note que estos requerimientos son agregados a los datos durante tiempo de ejecución para intercambiar datos cumpliendo las normas oficiales en el manejo de datos sensibles.

3. **Elige tus datos**: en este paso el diseñador elige el catálogo de datos que será la fuente de datos del servicio de e-salud.

   En la Figura 16 se muestra la pantalla de selección de catálogos del sistema de e-salud. Los catálogos que se muestran son aquellos creados en el sistema de publicación/suscripción de la red de distribución de contenidos manejada por Zamná. El diseñador puede seleccionar de los catálogos que él ha creado (catálogos publicados) o de los que le han compartido (catálogos suscritos).

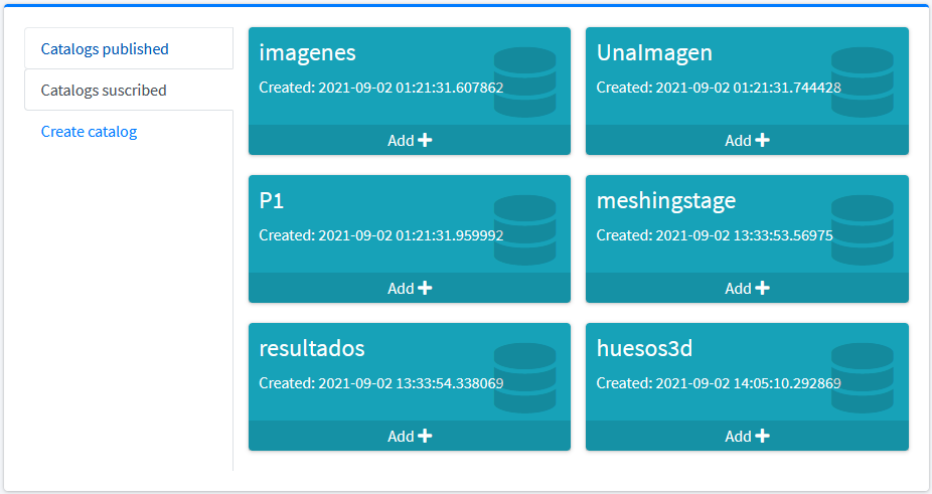


Figura 16. Pantalla de selección de catálogos para un sistema de e-salud.

En la Figura 17 se muestra la pantalla de creación de un catálogo en la red de distribución de contenidos de Zamná. Para crear el catálogo solo se debe de especificar el nombre del catálogo, así como al grupo al que pertenece.

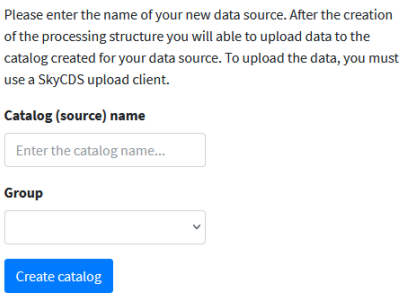


Figura 17. Pantalla de creación de un catálogo.

4. **Une tus aplicaciones**: en este paso el diseñador ordena y une sus aplicaciones para definir su orden de ejecución. Los diseñadores pueden crear diferentes patrones de procesamiento. Al guardar el servicio de e-salud, este puede ser

desplegado ya sea en una computadora personal, en la nube o en un clúster de computadoras con Docker Swarm.

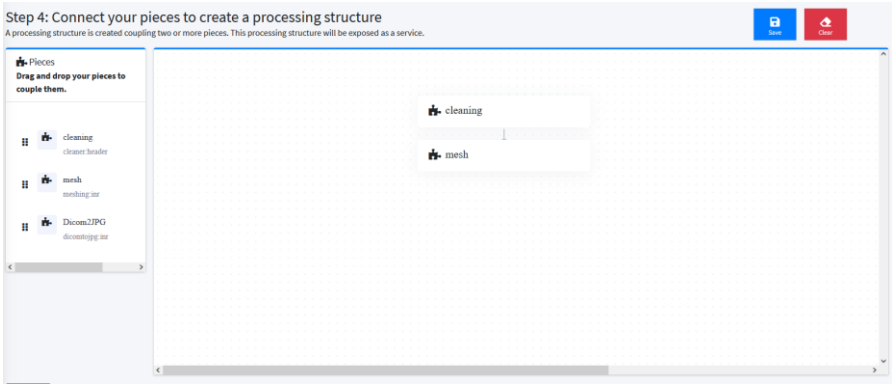


Figura 18. Interfaz gráfica para el acoplamiento de las piezas.

En la Figura 18 se muestra la pantalla en la que los diseñadores pueden acoplar sus piezas para crear un rompecabezas que posteriormente será desplegado como un sistema de e-salud. El diseñador solo debe de arrastrar las piezas y unirlas creando el patrón que dese, por ejemplo, una tubería de procesamiento o un flujo de trabajo con bifurcaciones. Al guardar el rompecabezas, la plataforma solicitará un nombre para identificarla.

En la Figura 19 se muestra la pantalla de despliegue del sistema de e-salud. Actualmente hay dos opciones de despliegue, con Docker Compose para desplegarlo en un solo equipo o con Docker Swarm para desplegarlo en un clúster de computadoras.

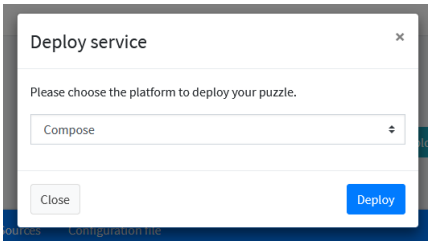


Figura 19. Pantalla de despliegue de un sistema de e-salud.

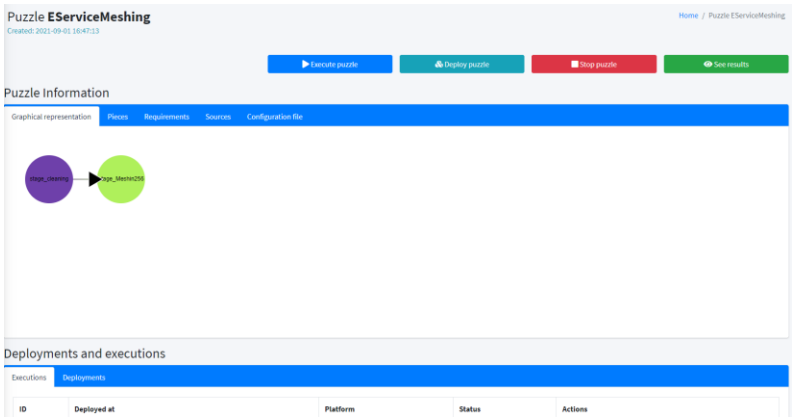


Figura 20. Pantalla de manejo de un sistema de E-salud construido utilizando un modelo basado en rompecabezas.

Una vez construido rompecabezas, y desplegado como un sistema de E-salud, podrá manejar el sistema desde la pantalla que se muestra en la Figura 20. Para acceder a dicha pantalla, se debe navegar a través del menú e ir a *Puzzles>List,* en donde se enlistarán los rompecabezas creados (ver Figura 21). Posteriormente, dar clic en "See puzzle". En la pantalla de manejo del sistema de E-Salud (Figura 20) es posible ejecutar, desplegar y detener el sistema, así como visualizar los resultados producidos por el sistema en forma de catálogos (ver Sección 3.4), y publicar el sistema a un usuario para que pueda reutilizar el sistema de E-Salud y desplegarlo en su infraestructura.

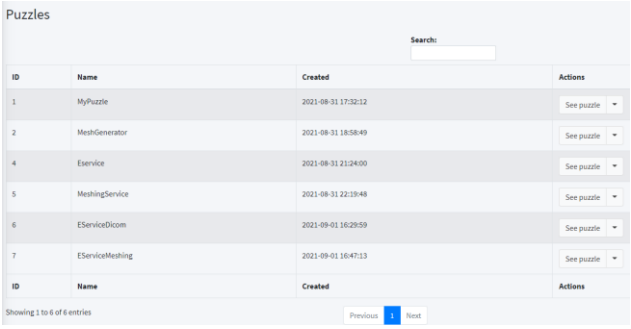


Figura 21. Lista de rompecabezas creados.

3.4 Repositorio de servicios de e-Salud

El repositorio de servicios y aplicaciones se encarga de manejar y mantener los diferentes servicios de e-Salud, así como los contenedores de datos y aplicaciones en un repositorio que puede ser consultado por los desarrolladores mediante las APIs de programación del Servicio de construcción de servicios. Además, este catálogo se encarga de manejar el acceso de los desarrolladores a sus servicios y aplicaciones previamente desarrollados y/o desplegados, con el objetivo de que estos puedan modificar y reutilizar servicios previamente existentes.

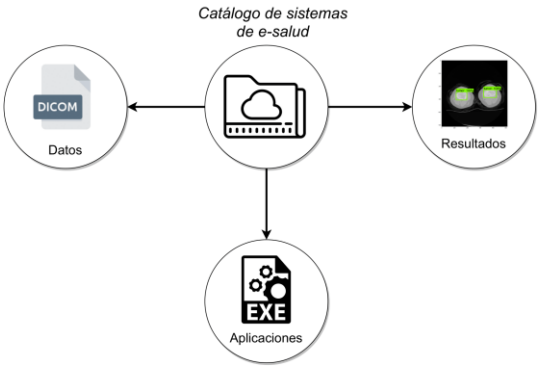
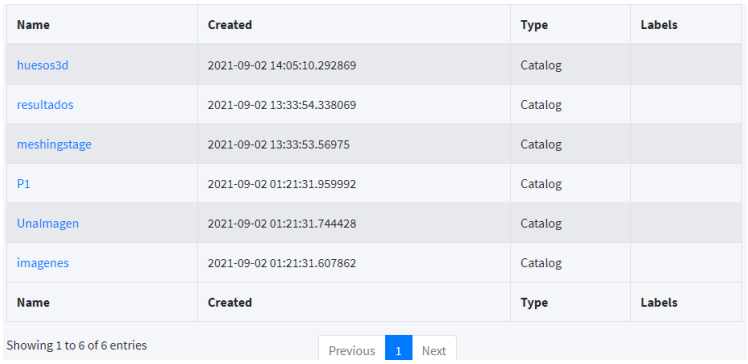


Figura 22. Catálogo de sistemas de e-salud, datos, aplicaciones y resultados.

El catálogo de sistemas de e-salud permite a los diseñadores y organizaciones reutilizar sistemas y piezas creados dentro de la organización o incluso creados por otras organizaciones cuando estos comparten sus catálogos. Como se puede observar en la Figura 22, dichos catalogo también son utilizados para manejar datos, aplicaciones y resultados generados en un sistema de e-salud.

Un catálogo de datos contiene aquellos productos producidos por las organizaciones. Por ejemplo, imágenes médicas, señales de electrocardiogramas o expedientes clínicos electrónicos. Dichos catálogos de datos son construidos utilizando Muyal-Painal, el cual permite a las instituciones compartir, intrainstitucional e interinstitucionalmente, bases de datos, resultados/información y sistemas de e-Salud mediante un modelo de publicación/suscripción, en privado y/o público, de catálogos. a través de intra/internet. Para obtener la fuente de datos de entrada de un sistema de e-Salud desde Muyal-Painal, Nez se apoya en Muyal-Zamná, el cual implemente una red de distribución segura y confiable de datos. En Nez, es posible observar los catálogos disponibles (ya sea publicados por el usuario o compartidos por otros usuarios) que un usuario puede utilizar como fuente de un sistema de e-Salud. En la Figura 23 se muestra el listado de catálogos en Nez.



Name	Created	Type	Labels
huesos3d	2021-09-02 14:05:10.292869	Catalog	
resultados	2021-09-02 13:33:54.338069	Catalog	
meshingstage	2021-09-02 13:33:53.56975	Catalog	
P1	2021-09-02 01:21:31.959992	Catalog	
Unalmagen	2021-09-02 01:21:31.744428	Catalog	
imagenes	2021-09-02 01:21:31.607862	Catalog	
Name	**Created**	**Type**	**Labels**

Showing 1 to 6 of 6 entries Previous 1 Next

Figura 23. Lista de catálogos disponibles para el usuario.

Nez automáticamente crea un catálogo en Muyal-Painal para cada conjunto de resultados producidos por cada etapa en el sistema de e-Salud. En la Figura 24 se muestra un ejemplo de catálogos producidos por Nez. Como se puede observar, el nombre de cada catálogo producido contiene el nombre de la etapa, así como una estampa de tiempo para identificar al catálogo. Note que cada catalogo es etiquetado (ver columna *Labels*) con la etapa en la que los resultados fueron producidos.

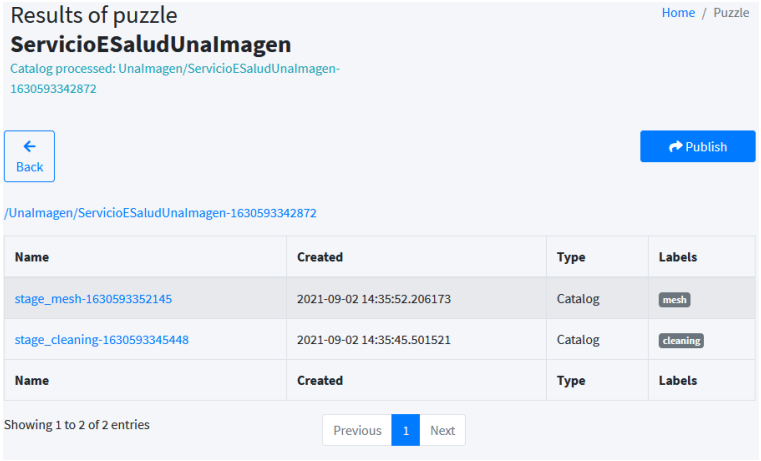


Figura 24. Catálogos de resultados producidos por un sistema de E-salud.

Finalmente, Nez maneja catálogos de sistemas de e-Salud y aplicaciones, para permitir que los usuarios y organizaciones compartan estructuras de procesamiento o parte de ellas. En la Figura 25 se muestra un listado de servicios de e-Salud (rompecabezas) creado con Nez y manejados como catálogos.

ID	Name	Owner	Created	Actions
11	Test123	domizzi	2021-10-26 02:02:21	See puzzle
12	Test123	domizzi	2021-10-26 02:22:51	See puzzle
13	Test123	domizzi	2021-10-26 02:30:12	See puzzle
14	Test123	domizzi	2021-10-26 02:30:23	See puzzle
15	Test123	domizzi	2021-10-26 02:30:37	See puzzle

Figura 25. Lista de servicios de e-Salud.

3.5 Sistema de manejo de catálogo de servicios para que los desarrolladores puedan acceder a cada producto del servicio

Como se mención o anteriormente, Nez maneja un sistema de catálogo de servicios. Este permite a los diseñadores y desarrolladores acceder a los sistemas de e-salud creados, así como a los datos y piezas (aplicaciones) creados previamente dentro de la misma organización, e incluso por otras organizaciones.

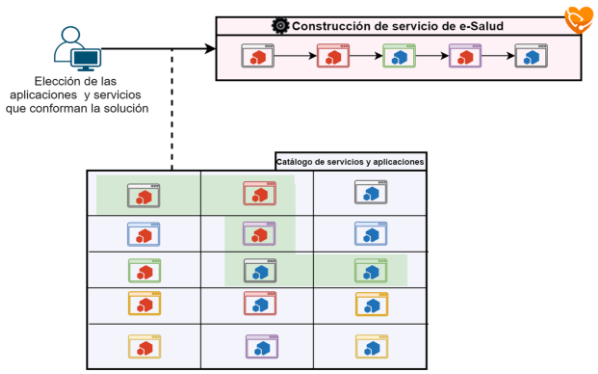


Figura 26. Representación conceptual del sistema de manejo de catálogos de servicios.

En la Figura 26 se muestra una representación conceptual del sistema de manejo de catálogo de servicios. Como se puede observar en la figura, los diseñadores y desarrollares pueden elegir del catálogo los servicios y aplicaciones que harán parte de su sistema de e-salud, y que durante tiempo de ejecución serán ejecutadas como piezas y rompecabezas. En la figura, los servicios coloreados de verde son aquellos que el diseñador/desarrollador eligió para crear el servicio de e-salud. De igual manera, es posible seleccionar las fuentes de datos del sistema de manejo de catálogos de datos.

Tanto los sistemas de e-Salud como los catálogos de datos y resultados pueden ser compartidos con otros usuarios mediante un sistema de publicación/suscripción.

Para publicar un catálogo de datos/resultados, se debe de navegar a *Storage>Catalogs.* Posteriormente, se debe de hacer clic en el catálogo que se desea compartir y dar clic en el botón *Publish.*

Mientras, que para publicar un sistema de e-Salud, se debe de navegar a *Puzzles>List,* y posteriormente dar clic en el botón desplegable de cada rompecabezas, y seleccionar *Publish* (ver Figura 27).

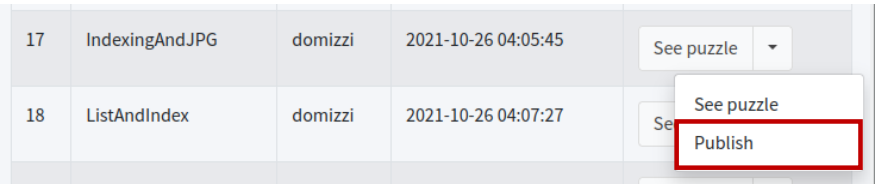


Figura 27. Compartir sistema de e-Salud (rompecabezas).

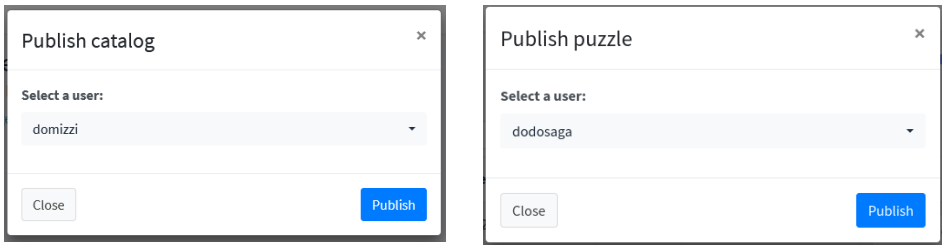


Figura 28. Pantallas de publicación de catálogos y sistemas de e-Salud (rompecabezas).

En la Figura 28 se muestran las pantallas para compartir tanto catálogos de datos/resultados como de rompecabezas (sistemas de e-Salud). En dichas pantallas se debe de seleccionar el usuario al cual se publicará el catálogo o el sistema de e-Salud. Una vez compartido el catálogo, los usuarios deberán suscribirse a los catálogos que le han compartido.

3.5.1 Interfaz de programación de aplicaciones y servicios

Nez permite a los desarrolladores acceder a un conjunto de APIs para la creación y manejo de sistemas de e-Salud. Además, dichas APIs pueden ser consumidas por aplicaciones desarrolladas por las organizaciones para obtener el estado de un sistema de e-Salud o consumir sus resultados, por ejemplo.

3.6 Servicio de descubrimiento, indexamiento y monitoreo de cripto-contenedores de sistemas de e-Salud

Nez incluye un método para la creación y consumo de representaciones de cripto-contenedores virtuales construidos utilizando Docker. La idea principal es obtener una representación de un contenedor virtual Docker que pueda ser consumida por usuarios diversos (humanos, dispositivos IoT, aplicaciones, contenedores virtuales, etc.) y que sirva como una herramienta de apoyo para la toma de decisiones en la gestión de los contenedores virtuales. Para ello es

necesario crear un DST (*Digital Sentinel Twin -DST-*), este es un objeto de software generado a partir del método propuesto. La conceptualización de un *DST* se ilustra en la Figura 29, a continuación, son explicados los conceptos[8].

Una $CApp$ (*Container Application -$CApp$-*) representa aquella aplicación que se encapsula en uno o más cripto-contenedores virtuales. Un vc (*virtual container -vc-*) es una pieza de software que ofrece un mecanismo para la encapsulación lógica de aplicaciones, en el que éstas son independientes del entorno en el que se ejecutan.

Figura 29. Conceptualización de un DST.

Un VCS (*Virtual Container System -VCS-*) representa un conjunto de vc_i construido como una única solución (servicio) para realizar una tarea en un flujo de datos. Un VD (*Virtual Device -VD-*) representa aquellos dispositivos virtuales que forman parte una $CApp$. Esto significa que un VD comprende componentes como VC o VCS.

Por lo tanto, un DST es un objeto de software para interactuar de manera sencilla con la estructura compleja y detallada de los VD y su $CApp$. Esto se debe a la flexibilidad de la *WoT card* que las representa, la cual cumple con las recomendaciones de W3C (ver Sección 2.3). Esta información proviene de una DfE (*Dataflow Entity -DfE-*), la cual captura información de cada componente

[8] Es importante remarcar que en el resto del documento se emplean acrónimos que se derivan de conceptos en inglés. Se ha decidido emplear estos acrónimos en inglés para mantener el vínculo con los conceptos empleados en la publicación de este trabajo en una revista.

interno (cualquier $CApp \in vc$ o $vc_i \in VCS$). DfE es básicamente una estructura de datos que incluye información sobre la estructura, el comportamiento y la función de un VD. La estructura, el comportamiento y la función se utilizan para modelar el flujo de datos desde el VD hasta los procesos de toma de decisiones. Se consideró una capa adicional para hacer más accesible la representación de una DfE utilizando las pautas de WoT; esto produce lo que se denomina *WoT card*. Esto significa que una *WoT card* representa la información de DfE a través de conceptos utilizados en el dominio de contenedores virtuales. Estos conceptos provienen de una ontología[9] basada en la norma ISO/IEC JTC 1/SC 38[10]. Siguiendo estos estándares de WoT, una *WoT card* puede representar a un VD de una manera definida y única.

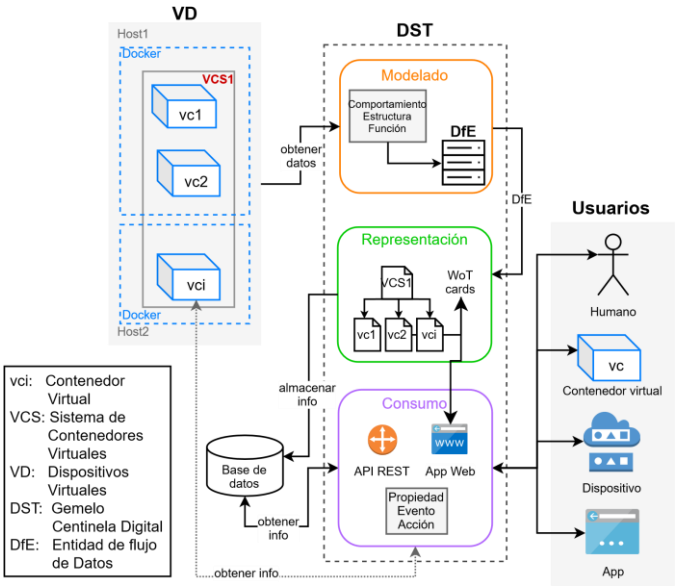


Figura 30. Contexto de un DST.

[9] Una ontología es una definición formal de tipos, propiedades, y relaciones entre entidades que fundamentalmente existen para un dominio de discurso en particular.

[10] www.iso.org/committee/601355.html

En la Figura 30 se ilustra el método para la construcción y consumo de un DST. Se propone un método de tres fases para la construcción y el consumo de un DST, tomando en cuenta el flujo de datos desde los VD hasta los procesos de toma de decisiones. En la Figura 30 también se muestra la vista conceptual de las etapas del método: *Modelado* (etapa 1), donde se adquieren los datos de un VD y se modelan sus elementos teniendo como resultado una DfE; luego, en la *Representación* (etapa 2) esta DfE se representa en *WoT cards* y se almacenan en una base de datos, para posteriormente ser utilizadas en el *Consumo* (etapa 3). A continuación, se describe con mayor detalle cada una de las etapas.

3.6.1 Etapa 1: Modelado funcional para construir una DfE

Un VD se puede modelar como un proceso para lograr un objetivo. El *modelado funcional* [10] [11] es adecuado para crear una representación de la estructura, comportamiento y función. Este enfoque de modelado se ha utilizado, durante los últimos años, para representar con éxito procesos en múltiples escenarios [12] [13] [14].

En el método propuesto todos los participantes del flujo de datos se modelan como objetos compuestos de partes de bajo nivel; el objeto tiene un objetivo y sus componentes contribuyen a lograr juntos ese objetivo al realizar sus tareas. El modelado funcional permite modelar a todos los participantes en la producción de estos flujos de datos (como vc_i, VCS o $CApps$), que, de hecho, tienen un comportamiento de procesos encadenados. Este modelo se captura en una DfE, que describe el comportamiento (propiedades y eventos), la función (tareas) y la estructura (interconexiones) de todos los participantes en el flujo de datos.

Como paso de preparación, se asume la existencia de un vc_i (ver VD en la Figura 30) ejecutando una transformación de datos (tarea);

independientemente del número de vc_i internos en un flujo de datos, esto se modela como un DST. Consideremos el caso más simple, donde un vc_i se descompone de función, estructura y comportamiento y se almacena en una DfE. Esta descomposición se representa mediante *WoT cards*, lo que hace que la DfE sea un DST listo para consumo. Para el caso de un $V\ CS$, ocurre el mismo proceso por cada vc_i, integrando funciones individuales como la función general del DST.

El objetivo de esta fase es obtener los tres elementos principales de modelado de un vc:

- *Comportamiento*, donde se especifican los valores de los atributos de los componentes, según la función del vc.
- *Función*, donde se especifica el objetivo principal de un vc y las tareas necesarias para lograrlo.
- *Estructura*, donde se especifican los componentes de un vc y las conexiones entre estos componentes.

Esta fase comienza recibiendo el archivo de configuración de vc, en formato *YAML*[11]. De este archivo se adquieren todos los datos necesarios para representar el vc.

3.6.2 Etapa 2: Acceso universal a *DST* mediante *WoT*

En este punto, una DfE proporciona una representación de los datos necesarios del vc. Sin embargo, necesitamos una representación útil para interactuar con un vc; dicha interacción puede ser de computadora a computadora o de humano a computadora. Para lograr esta flexibilidad, esta representación se basa en los principios de Web of Things (WoT) [15]. Esta representación de un vc se denomina *WoT card*. Además de la información capturada por una DfE, también se agregan metadatos del vc a la *WoT card*. Estos son metadatos que corresponden

[11] YAML *(*Ain't Markup Language) es un formato de serialización de datos legible por humanos, usado por Docker para configurar los servicios de su aplicación (mediante *docker-compose*)

a la descripción técnica del *vc*: direcciones IP, volúmenes, puertos, espacios de nombres, entre otros.

En el caso de un VCS, dichos elementos se definen de forma recursiva para capturar datos sobre estructuras y transformaciones utilizadas y realizadas por todo el VCS y sus componentes (vc_i), respectivamente. Según las recomendaciones de WoT, la generación de las *WoT cards* debe basarse en ontologías.

En este sentido, se definió y creó una ontología (llamada *VC Docker FU Ontology*[12]), que se puede adaptar al contexto de cualquier *WoT card* en varios escenarios. La *VC Docker FU Ontology* se utiliza como referencia en toda la generación de *WoT cards* durante la representación de un *vc*. Esta ontología extiende de dos ontologías, primero extiende de la *VC Docker Ontology*[13], la cual a su vez extiende de *VC ISO Ontology*. VC ISO Ontology fue formalizada por nosotros de acuerdo con la norma ISO/IEC JTC 1/SC 38[14], que define todos los conceptos y restricciones de la norma de una manera abstracta. La *VC Docker Ontology*, en su versión original, ya define conceptos y restricciones de contenedores virtuales en el entorno Docker, fue adaptada siguiendo la *VC ISO Ontology*; se incluyeron algunos conceptos y restricciones adicionales para cumplir con la norma ISO. *VC Docker FU Ontology* agrega conceptos sobre el comportamiento relacionado con los recursos de infraestructura -CPU, MEM, FS y NET- (como niveles de utilización y propiedades de dichos valores), y la función de los contenedores virtuales (como funciones y tareas).

[12] Disponible en github.com/adaptivez/VirtualContainerOntology
[13] github.com/langens-jonathan/docker-vocab/blob/master/docker.md#config
[14] www.iso.org/committee/601355.html

Técnicamente, una *WoT card* se basa en una clase abstracta llamada *Thing*, que es el objeto base para el modelado en el enfoque WoT. Se basa en la estructura de representación de *Thing*

Description (TD)[15]. Por lo tanto, una *WoT card* se compone de tres elementos: *i) metadata* (de $Thing$), que contiene interacciones (cómo se puede usar o consumir $Thing$); *ii) vocabulary*, que contiene definiciones de conceptos que se utilizan en la estructura *Thing Description* y en la ontología (*VC Docker FU Ontology*), útil para interacciones; y *iii) URIs*, que son útiles para identificar recursos en *Thing Description*, éstos son enlaces de Internet que permiten tener acceso a los metadatos de $Thing$.

3.6.3 Etapa 3: Consume de DST

Una vez generada y almacenada la *WoT card*, está lista para el consumo mediante un DST. Para que el DST sea accesible y consumido, debe convertirse en un intermediario entre el objeto modelado (vc) y el consumidor. Esto es posible mediante el uso de un sistema RESTful, que puede procesar solicitudes mediante las acciones HTTP más comunes: GET, POST, PUT, DELETE. De esta manera, cualquier consumidor que realice solicitudes de tipo REST puede consumir el DST. El consumo puede ser sobre propiedades, acciones o eventos.

Cada elemento de la *WoT card* es universalmente identificado y aceptado por otras entidades físicas y/o abstractas (p. ej., otros vc, $V\ CS$, aplicaciones, dispositivos, peticiones de humanos, etc.) mediante un URI[16].

3.6.4 Prototipo del servicio

Esta sección se describe la implementación de un prototipo para construir y consumir un DST, basado en el método propuesto. Los componentes de este

[15] Es el modelo base para describir cualquier cosa de IoT en el enfoque de Web of Things del W3C. *Thing Description* describe los metadatos y las interfaces de Things. www.w3.org/TR/wot-thing-description

[16] URI (*Identificador de recurso universal*) identifica en Internet un recurso (página web, imagen, audio, video, archivo, dispositivo IoT, etc.) de una manera única y universal.

prototipo y sus interacciones se muestran en la Figura. Los componentes se implementaron como microservicios (encapsulados en contenedores virtuales), codificados mediante Python 3.0, excepto el componente *Observación*, que se implementó mediante JavaScript y PHP debido a la naturaleza de las tareas que se ejecutan. El prototipo fue desarrollado para la plataforma Docker, pero el DST puede ser creado para otras plataformas de contenedores, como LXC[17], FreeBSD jail[18], o Solaris Zones[19], donde un vc es creado a partir de un archivo YML o $YAML$.

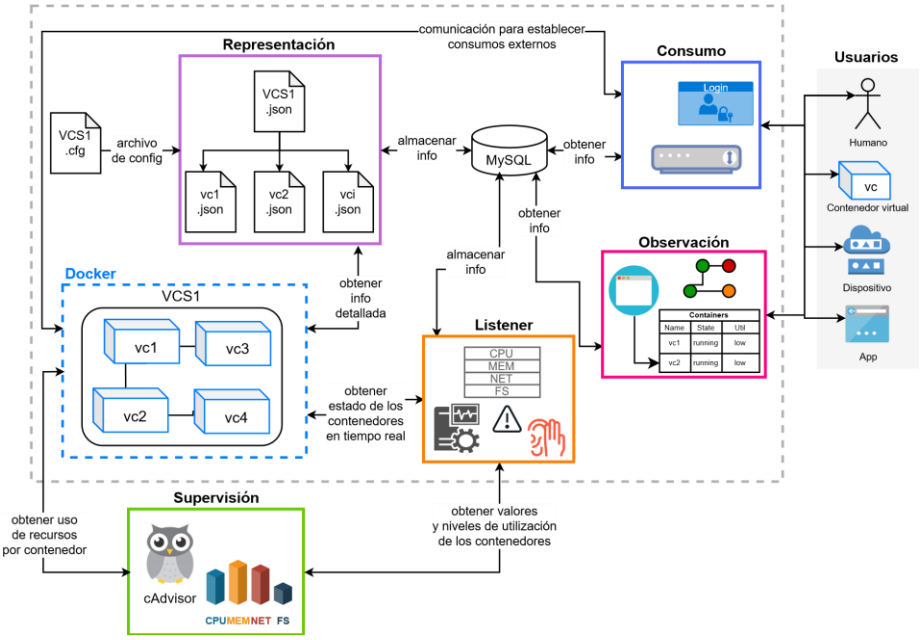


Figura 31. Componentes del prototipo.

[17] https://linuxcontainers.org/lxc

[18] https://docs.freebsd.org/en/books/handbook/jails/

[19] https://docs.oracle.com/cd/E37929_01/html/E36580/zonesoverview.html

4. Grupo de trabajo y colaboraciones institucionales

A continuación, se listan las instituciones participantes en el proyecto, así como el equipo de trabajo intrainstitucional e interinstitucional.

4.1 Instituciones participantes

Tabla 1. Lista de instituciones participantes en el desarrollo de Nez.

Institución	Tipo de entidad	Área
CINVESTAV Tamaulipas	Entidades entregantes	Grupo de Gestión de Datos y Redes de Computadoras
Universidad Carlos III de Madrid (UC3M)		Grupo de Arquitectura de computadoras y tecnologías (ARCOS)
Universidad Autónoma Metropolitana (UAM)		Departamento de Ing. Eléctrica, áreas de Redes y Telecomunicaciones/ Procesamiento Digital de Señales e Imágenes Biomédicas
Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE)		Departamento de Electrónica y Telecomunicaciones de la división de física aplicada de Ensenada y Unidad Monterrey.

4.2 Grupo de trabajo intrainstitucional

Tabla 2. Lista del grupo de trabajo intra-institucional.

Institución	Participante	Tipo de participación en el proyecto	Nivel SNI
Cinvestav Tamaulipas	Dr. José Luis González Compeán	Responsable Técnico del proyecto. Participa en todas las etapas del proyecto.	**1**
	Dr. Miguel Morales Sandoval	Diseño de cripto-contenedores.	**2**
	MC. Dante Domizzi Sánchez Gallegos	Diseño y desarrollo de servicio de construcción de sistemas de e-salud Muyal-Nez: Servicio de construcción de sistemas de e-salud (Entregable 1).	**N/A**

	MC. Mariana Magdalena Hinojosa Tijerina	Realizó su tesis de maestría sobre el proyecto. (Terminada). Monitoreo de cripto-contenedores del servicio Muyal-Nez	**N/A**

4.3 Grupo de trabaja interinstitucionales

Tabla 3. Grupo de trabajo inter-institucional.

Institución	Participante	Tipo de participación en el proyecto	Nivel SNI
UC3M	Dr. Jesús Carretero Pérez	Revisión de arquitectura y diseño.	**N**
UAM	Dr. Ricardo Marcelín Jiménez	Sistemas de distribuidos y preparación y almacenamiento de datos.	**1**

5. Estado de los productos de Nez

La Figura 32 muestra el resumen de los productos de Nez. En él se encuentran el estado de cada uno de los productos: comprometido y no comprometido, así como un resumen de su estado.

Entregable	ID	Producto	Estado					Etapa		Comprometido	NMT
			Diseño	En desarrollo	En Evaluación	Prototipado	En Producción	1	2		
Nez	Muyal Nez P1	Un esquema de bloques de construcción de flujos de trabajo y servicios de e Salud basado en mapas de microservicios y nanoservicios.	*	*	*	*		✓		Si	6
	Muyal Nez P2	Un esquema de construcción de cripto contenedores de datos y cripto contenedores de aplicaciones.	*	*	*	*		✓		Si	5
	Muyal Nez P3	Un esquema de despliegue de e Servicios independientes de la infraestructura.	*	*	*	*		✓		Si	6
	Muyal Nez P4	Servicio de descubrimiento, indexación y monitoreo de cripto contenedores de sistemas de e salud	*	*	*	*		✓		No	5

Simbología	
*	Terminado
X	En proceso
NMT	Nivel de Madures Tecnológica

Figura 32. Resumen y estado de los productos de Nez.

6. Anexos

6.1 Descarga de software

En el siguiente enlace se puede descargar los contenedores virtuales que contienen los servicios de Nez:

Descarga de software

6.2 Manual de instalación de servicios y descripción de las APIs para crear sistemas de e-Salud

El presente documento busca describir el desarrollo de un repositorio de servicios de e-salud. Esto mediante un Sistema Gestor de Servicios (SGS) donde un servicio representa las aplicaciones contenerizadas ya sea de forma individual, así como trabajando en conjunto para formar cadenas de valor o flujos de trabajo. Presenta el objetivo con los alcances del sistema, los endpoints de las APIs y los diagramas necesarios para un mejor entendimiento de sus componentes. Finalmente se muestra el proceso de instalación.

Ver reporte

6.3 Productos de investigación conseguidos

6.3.1 Tesis maestría

- **Modelo Funcional de Contenedores Virtuales Docker**

  Autor: Mariana Magdalena Hinojosa Tijerina

  Institución: Cinvestav Tamaulipas

  Año: 2021

  Directores: Dr. José Luis González Compeán y Dr. Iván López Arévalo

  Resumen: En la última década, la tecnología de contenedores virtuales ha demostrado ser una gran opción al problema de portabilidad de las aplicaciones de software. Los contenedores virtuales facilitan el desarrollo rápido y ágil de aplicaciones, siendo Docker la plataforma más usada. Un

reto en el dominio de contenedores virtuales (incluido Docker) es la toma de decisiones en tiempo real sobre su gestión, asociada a tareas de migración, monitoreo, clasificación, descubrimiento, entre otras, las cuales permiten la manipulación de los contenedores para tener un mejor desempeño en el entorno. En cualquiera de las tareas mencionadas anteriormente es deseable contar con los datos esenciales de los contenedores de forma rápida y precisa. En esta tesis se propone un método para la representación de contenedores virtuales Docker, la cual facilita la manipulación de los datos y la invocación del funcionamiento de los contenedores. El método propuesto consta de tres etapas: Modelado, Representación y Consumo. En la primera etapa (Modelado) se obtiene el modelo del contenedor virtual mediante un enfoque de modelado funcional que resalta las propiedades asociadas a la estructura, comportamiento y función del contenedor. En la segunda etapa (Representación) se utiliza el modelo creado para realizar la representación del contenedor virtual siguiendo los principios de la W3C Web of Things. Y, por último, en la tercera etapa (Consumo) se da acceso a usuarios externos para el consumo de la representación generada, por medio de un servicio RESTful. La evaluación experimental reveló la factibilidad de utilizar el método propuesto. Mediante un estudio de caso se demuestra que es posible representar tanto dispositivos virtuales como dispositivos físicos IoT utilizando el método propuesto.

Ver tesis completa

6.3.2 Artículos de revista

- **A WoT-Based Method for Creating Digital Sentinel Twins of IoT Devices**

  Autores: Lopez-Arevalo, Ivan, Jose L. Gonzalez-Compean, Mariana Hinojosa-Tijerina, Cristhian Martinez-Rendon, Raffaele Montella, and Jose L. Martinez-Rodriguez

  Estado: Publicado

  Año: 2021

  Revista: Sensors

  DOI: https://doi.org/10.3390/s21165531

  Resumen: The data produced by sensors of IoT devices are becoming keystones for organizations to conduct critical decision-making processes. However, delivering information to these processes in real-time represents

two challenges for the organizations: the first one is achieving a constant dataflow from IoT to the cloud and the second one is enabling decision-making processes to retrieve data from dataflows in real-time. This paper presents a cloud-based Web of Things method for creating digital twins of IoT devices (named sentinels). The novelty of the proposed approach is that sentinels create an abstract window for decision-making processes to: (a) find data (e.g., properties, events, and data from sensors of IoT devices) or (b) invoke functions (e.g., actions and tasks) from physical devices (PD), as well as from virtual devices (VD). In this approach, the applications and services of decision-making processes deal with sentinels instead of managing complex details associated with the PDs, VDs, and cloud computing infrastructures. A prototype based on the proposed method was implemented to conduct a case study based on a blockchain system for verifying contract violation in sensors used in product transportation logistics. The evaluation showed the effectiveness of sentinels enabling organizations to attain data from IoT sensors and the dataflows used by decision-making processes to convert these data into useful information.

Ver artículo

- **PuzzleMesh: A puzzle model to build mesh of agnostic services for edge-fog-cloud**

  Autores: Dante D. Sanchez-Gallegos, J. L. Gonzalez-Compean, Jesus Carretero, Heidy M. Marin-Castro, Andrei Tchernykh, and Raffaele Montella

  Estado: Enviado y en revisión

  Año: 2020

  Revista: IEEE Transactions on Services Computing

  Resumen: This paper presents the design, development, and evaluation of PuzzleMesh, an agnostic service mesh composition model to process large volumes of data in edge-fog-cloud environments. This model is based on a puzzle metaphor where pieces, puzzles, and metapuzzles represent self-

contained autonomous and reusable software artifacts encapsulated into containers and published as microservices. A piece represents the integration of applications with I/O interfaces (loops/sockets), parallel processing, and management software. A puzzle represents processing structures (pipelines, patterns, or workflows) built by pieces coupled through loops and sockets. Puzzles integrate structures with a microservice architecture, implicit continuous dataflows, and transparent data exchange management software. A metapuzzle represents a recursive assemble of combinations of puzzles. A mesh represents a pool of pieces, puzzles, and metapuzzles available for designers to choose artifacts to build services. A prototype developed by using the PuzzleMesh model was evaluated through case studies about the automatic construction of processing services for the acquisition, pre-processing, manufacturing, preserving, and visualizing of satellite imagery. A qualitative evaluation revealed that PuzzleMesh provides a flexible way to build reusable and portable services and to improve the usability of the services. The case study also revealed that PuzzleMesh yielded better performance results than other state-of-the-art tools.

Ver artículo

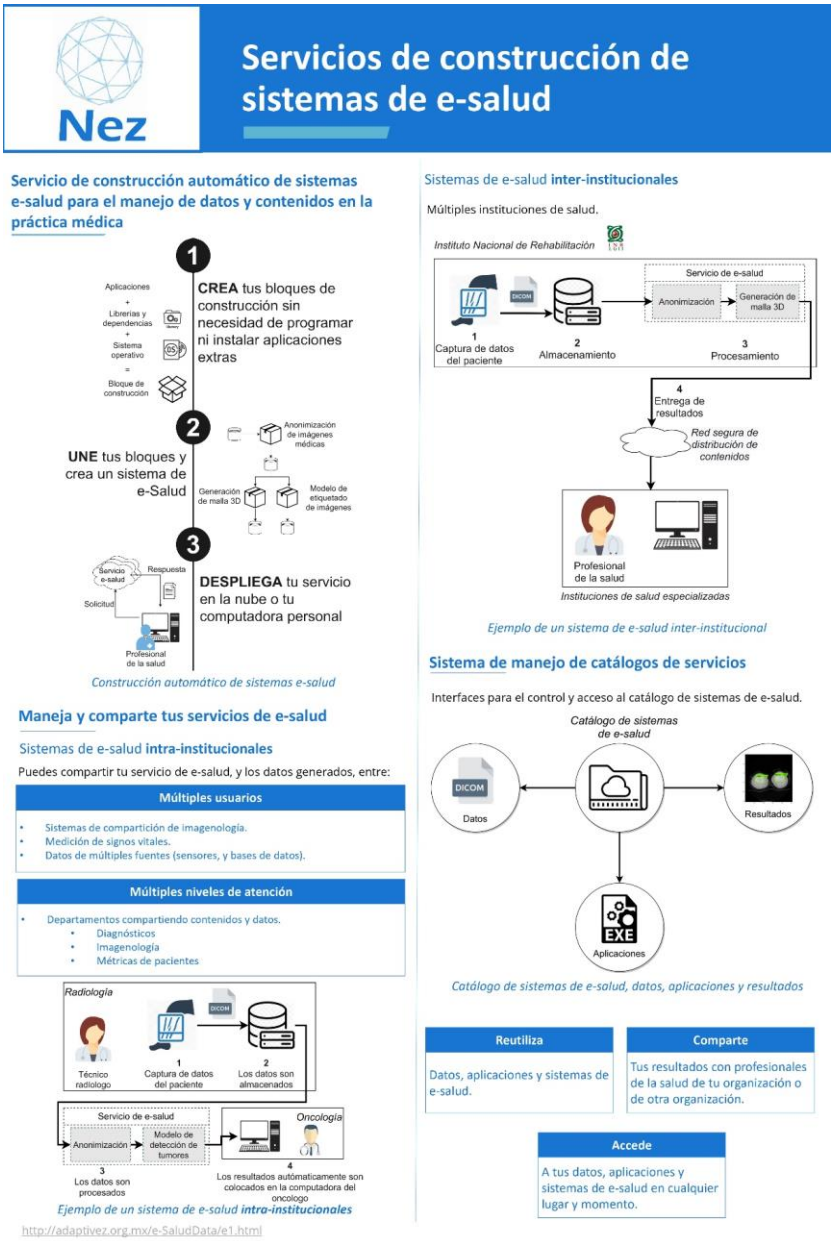
6.4 Infografías de los sistemas de Nez

En los siguientes enlaces se pueden acceder a infografías en donde se describen de manera general y técnica los componentes de Nez:

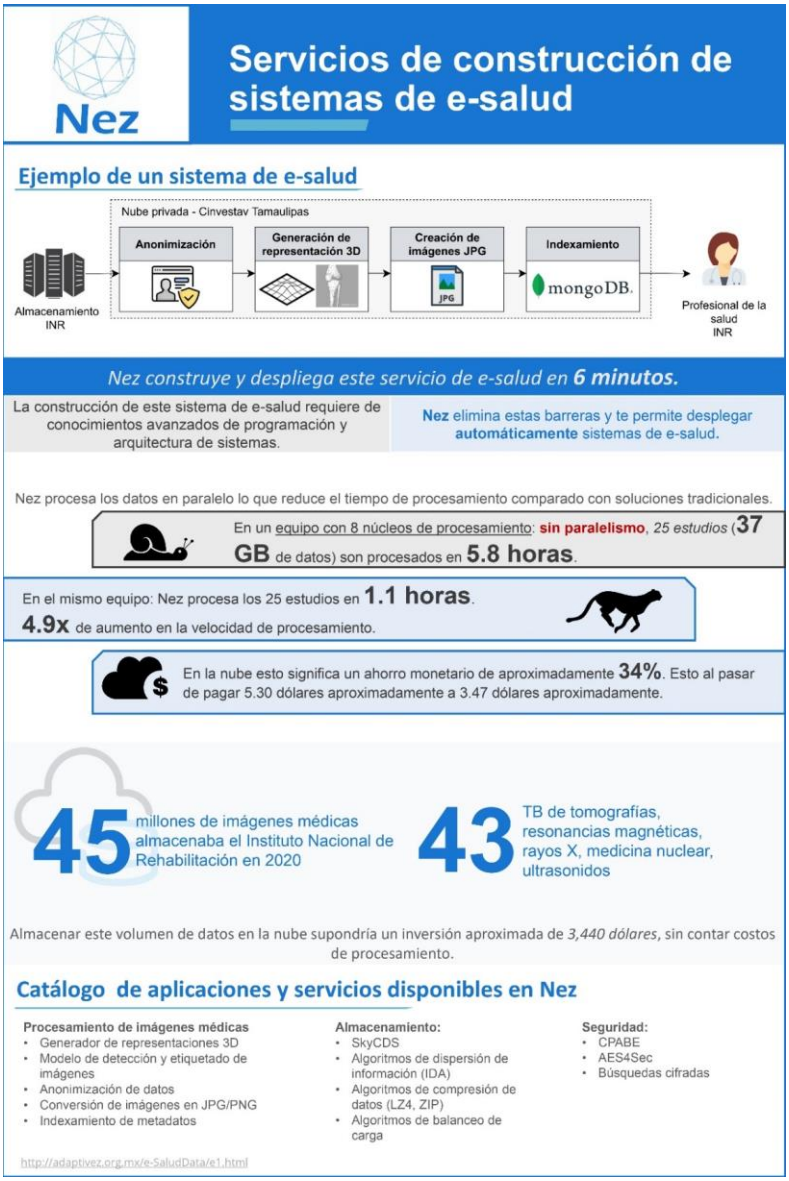
Ver infografía general

Ver infografía técnica

6.5 Poster de divulgación cualitativo



6.6 Poster de divulgación cuantitativo



7. Referencias

[1] A. a. Q. Y. A. a. A. M. a. Z. Y. B. a. A. M. K. a. K. S. W. Nauman, «Multimedia Internet of Things: A comprehensive survey.,» *IEEE Access,* vol. 8, pp. 8202-8250, 2020.

[2] R. P. J. M. H. A. &. S. R. Singh, «Internet of things (IoT) applications to fight against COVID-19 pandemic,» *Diabetes & Metabolic Syndrome: Clinical Research & Reviews,* vol. 14, nº 4, pp. 521-524, 2020.

[3] S. B. S. B. A. K. K. S. P. J. M. H. A. &. S. R. P. Kushwaha, «Significant applications of machine learning for COVID-19 pandemic,» *Journal of Industrial Integration and Management.,* 2020.

[4] M. G. K. &. F. S. Stanfill, «Health information management best practices for quality health data during the COVID-19 global pandemic.,» *Perspectives in health information management,* 2020.

[5] E. R. &. J. H. Berchick, «Data Processing Improvements for Estimates of Health Insurance Coverage in the Current Population Survey Annual Social and Economic Supplement.,» *Medical Care Research and Review,* 2021.

[6] D. D. D. L. D. K. S. G.-C. J. L. &. M. R. Sánchez-Gallegos, «An efficient pattern-based approach for workflow supporting large-scale science: The DagOnStar experience.,» *Future Generation Computer Systems,* vol. 122, pp. 187-203, 2021.

[7] D. D. G.-M. A. G.-C. J. L. V.-R. S. P.-R. A. E. C.-E. D. &. C. J. Sánchez-Gallegos, «On the continuous processing of health data in edge-fog-cloud computing by using micro/nanoservice composition,» *IEEE Access,* vol. 8, pp. 120255-120281, 2020.

[8] D. D. C.-E. D. R.-A. H. G. G.-C. J. L. C. J. M.-S. M. &. G.-M. A. Sánchez-Gallegos, « From the edge to the cloud: A continuous delivery and preparation model for processing big IoT data,» *Simulation Modelling Practice and Theory,* vol. 105, p. 102136, 2020.

[9] J. L. G.-C. J. C. H. M. M.-C. A. T. a. R. M. Dante D. Sanchez-Gallegos, «PuzzleMesh: A puzzle model to build mesh of agnostic services for edge-fog-cloud,» *Under revision,* 2021.

[10] L. G. G. T. C. &. T. E. Chittaro, «Functional and teleological knowledge in the multimodeling approach for reasoning about physical systems: a case study in diagnosis.,» *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 23, nº 6, pp. 1718-1751., 1993.

[11] M. &. Z. X. Lind, « Functional modelling for fault diagnosis and its application for NPP,» *Nuclear Engineering and Technology,,* vol. 46, nº 6, pp. 753-772, 2014.

[12] B. &. J. J. R. Chandrasekaran, «Function in device representation.,» vol. 16, nº 3-4, pp. 162-177, 2000.

[13] U. C. F. &. W. H. Yildirim, «Function modeling using the system state,» *AI EDAM,* vol. 31, nº 4, pp. 413-435, 2017.

[14] Y. T. H. T. T. &. Y. H. Umeda, «Function, behaviour, and structure.,» *Applications of artificial intelligence in engineering* , vol. 1, pp. 177-193, 1990.

[15] D. D. &. T. V. M. Guinard, «Building the web of things: with examples in node. js and raspberry pi,» *Simon and Schuster.,* 2016.