Documento de Arquitectura de Software

Para

Segmentación Semi-Automática de Tejido Adiposo en el Espacio Parafaríngeo por Medio de Aprendizaje Profundo

Por

David Alejandro Castillo Chíquiza
Oscar David Falla Pulido
Juan Sebastián Ruiz Bulla

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA DE SISTEMAS
BOGOTÁ, D.C.
2022

Tabla de Contenidos

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

En esta sección se describe el proceso de diseño utilizado para garantizar una arquitectura acorde a las necesidades de los stakeholders y mantener las características más deseables del sistema. Además, se da una breve contextualización de los modelos y estándares utilizados para el diseño de la solución y definir los elementos usados para este.

1.1. Propósito

Para realizar la documentación de la arquitectura del sistema se utiliza el modelo 4+1 más una vista adicional, esta última detalla la arquitectura de red neuronal planteada para esta solución. El motivo de las vistas previamente mencionadas es justificar en cada una de estas las decisiones arquitectónicas que se tomaron con el objetivo de obtener una arquitectura coherente, consistente y fiel a los atributos que se desean alcanzar.

En la figura 1.1 se puede apreciar las vistas de modelo, el cómo se relacionan e influyen entre sí y el aspecto y/o abstracción del sistema que representan.

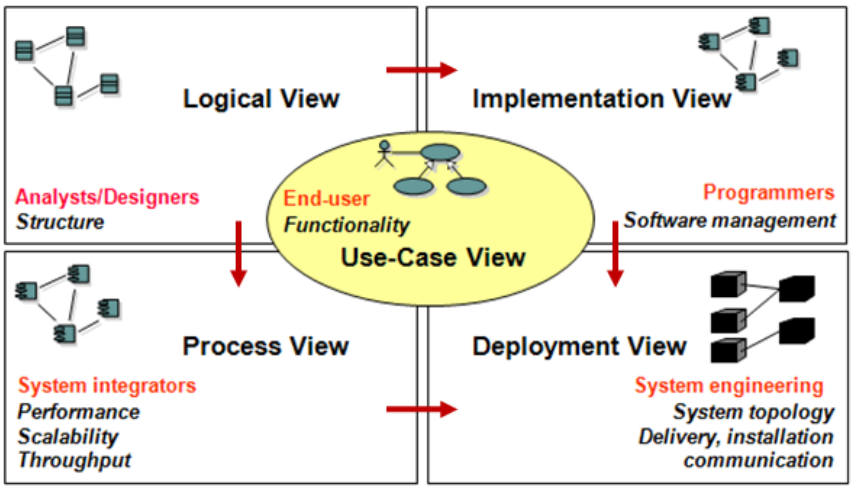

Figura 1.1: Diagrama de Casos de Uso del Sistema.

1.2. Definiciones, acrónimos y abreviaturas

- **Requisitos funcionales (RF):** Describen una función determinada que debe tener el software, es decir, un comportamiento del mismo o algo que este debe hacer a partir de unos datos de entrada determinados. Especifican qué debe hacer cada parte del sistema y los resultados que este debe generar en situaciones determinadas. Pueden incluir cálculos, detalles técnicos, manipulación y procesamiento de datos y otras funcionalidades específicas que definen lo que se supone que debe lograr un sistema. Estos se especifican por lo general en la primera etapa del desarrollo de software, previa a la fase de diseño, y están respaldados por los requisitos no funcionales.
- **Requisitos no funcionales (RNF):** Especifican los criterios que se usan para juzgar la operación del sistema y el cómo el sistema cumple con lo que tiene que llevar a cabo. Mientras los requisitos funcionales describen lo que debe hacer el sistema en términos de funcionalidades, indicando los resultados que se deberían generar en cada parte del mismo, los requisitos no funcionales son transversales a la totalidad del sistema, debido a que buscan definir las propiedades que debe cumplir el sistema como un todo. Estos pueden definir algunos aspectos de calidad del sistema para asegurar que este cumple a cabalidad con las necesidades del usuario.
- **Interfaces de usuario:** Estas se refieren al conjunto de elementos que le permiten al usuario interactuar y comunicarse con el sistema de manera efectiva. El objetivo

de esta interacción es permitir la operación y el control efectivos de la máquina desde el lado humano. Estos pueden incluir la pantalla, el teclado, mouse, etc., así como también elementos informativos y gráficos dentro del software como lo pueden ser texto, botones, tablas, campos de texto, listas, imágenes, etc. El objetivo del diseño de la interfaz de usuario es producir una interfaz de usuario que sea fácil, eficiente y agradable para operar un sistema de manera que produzca el resultado deseado.

- **Interfaces de hardware:** Especifican elementos físicos involucrados con el despliegue y correcto funcionamiento del software, tales como enchufes, cables, señales eléctricas, placas base, USB y otros periféricos, etc., así como también aspectos y elementos referentes a redes.
- **Interfaces de software:** Son los lenguajes, códigos y mensajes que se usan en el sistema para posibilitar la comunicación entre sus distintos componentes y de estos con los elementos de hardware con los que interactúa el sistema. Entre estos se encuentran los sistemas operativos, protocolos de redes, controladores para gestionar los elementos de hardware, tecnologías utilizadas, etc.
- **Atributos de calidad de software:** Definen el grado en el que el sistema se desempeña correctamente de acuerdo con las expectativas y necesidades del cliente y satisface los requerimientos funcionales y no funcionales definidos en la primera fase de diseño.

1.3. Audiencia

Equipo de desarrollo: Se encarga de ejecutar las actividades a realizar durante las fases de implementación, despliegue, pruebas y mantenimiento del software.

1.4. Alcance

En este documento se diseñan, describen y analizan las estructuras y sus vistas respectivas, justificadas y acompañadas con las decisiones arquitectónicas tomadas, siguiendo el contexto y los requisitos ya discutidos para el sistema. Para realizar esta tarea se va a hacer uso del modelo 4+1, este modelo comprende y define la arquitectura del sistema a través de la vista de casos de uso, expresando las decisiones arquitectónicas del sistema por medio de las vistas: lógica, procesos, despliegue y física, cada una de estas vistas modela

una abstracción de la arquitectura, y están dirigidas a un conjunto de actores involucrados en el desarrollo del proyecto, cabe resaltar que estas vistas no son independientes, por el contrario, se relacionan fuertemente y profundamente, ya que la decisión tomada para una vista puede afectar a otra que sea dependiente de esta. Adicionalmente, se hace uso de una vista extra, esta busca mostrar la arquitectura de red neuronal que se utiliza en el sistema.

Capítulo 2

Representación Arquitectural

Este capítulo tiene como propósito definir las vistas del modelo 4+1 que serán utilizadas para visualizar y entender la arquitectura propuesta, también se propone representar la arquitectura de la red neuronal. Con esto se espera satisfacer las necesidades que los interesados del sistema requieren.

2.1. Vista Lógica

- **Audiencia:** Equipo de desarrollo.
- **Área:** Se encarga de representar las funcionalidades principales, describiendo el modelo de objeto del diseño y las representaciones de casos de uso más importantes.

2.2. Vista Procesos

- **Audiencia:** Equipo de desarrollo.
- **Área:** Se encarga de representar los procesos, la concurrencia, funcionalidad del sistema y su desempeño.

2.3. Vista de Implementación

- **Audiencia:** Equipo de desarrollo.

- **Área:** Se encarga de describir los sistemas, subsistemas, paquetes, componentes y módulos, así como sus relaciones y dependencia entre ellos.

2.4. Vista Física

- **Audiencia:** Equipo de desarrollo.
- **Área:** Se encarga de representar el mapeo de donde está alojado y funcionando el software en el hardware y aspectos de distribución del software.

2.5. Vista de Arquitectura de Red Neuronal

- **Audiencia:** Equipo de desarrollo.
- **Área:** Se encarga de representar la arquitectura de la red neuronal propuesta para satisfacer el sistema. En esta arquitectura, se representan las convoluciones, max pooling, dropout, transpose y concatenate, siendo estos "filtros"por los cuales son procesados los datos con los cuales el sistema es entrenado. Estos filtros tienen el propósito de identificar patrones en los datos que ayuden a identificar la región de interés.

Capítulo 3

Contexto del Problema

Este capítulo tiene como propósito contextualizar al lector sobre las características e interacciones que debe satisfacer el sistema que va a ser propuesto en el presente documento.

3.1. Actores

Como se mencionó en el documento de "Especificación de Requisitos de Software", existen varios actores que pueden hacer uso del sistema como radiólogos, científicos de datos, desarrolladores y entusiastas. Sin embargo, todos interactúan de la misma forma con el sistema y tienen los mismos privilegios. Por tal razón todos se agrupan en uno solo, llamado usuario. el usuario podrá acceder a cada una de las características del sistema y tendrá acceso total, por lo tanto, podrá hacer entrenamiento de nuevos modelos y segmentación de imágenes.

3.2. Entradas

Debido a que el sistema tiene dos subsistemas, entrenamiento de modelo y segmentación de imágenes, se identificaron las entradas de ambos subsistemas, así mismo se identificaron los datos y el tipo de estos. En la tabla 3.1 se listan cada una de las entradas junto con su descripción y el tipo de dato y/o formato.

Entrada	Tipo	Descripción
Ruta (Entrenamiento)	String	El usuario, si lo desea, podrá especificar una ubicación en la cual desa que el sistema guarde el modelo entrenado.
Conjunto de datos	.nii .nii.gz	Es el conjunto de imágenes de resonancias magnéticas y sus respectivas segmentaciones o mascarás, el usuario debe indicarle al sistema la ubicación de estos archivos para que el sistema pueda leerlos y cargarlos.
Ruta (Segmentación)	String	El usuario, si lo desea, puede indicarle la ubicación en donde quiere guardar la segmentación de la imagen
Imagen o Fichero	.nii .nii.gz	Es el conjunto de imágenes sobre las cuales se desea que el sistema realice la segmentación, el usuario debe especificarle al sistema la ubicación del fichero o la imagen.
Modelo prentrenado	.h5	Es el modelo que el usuario desea utilizar para realizar una segmentación sobre un conjunto de imágenes.

Tabla 3.1: Entradas del Sistema

3.3. Salidas

Tal como con las entradas del sistema previamente descritas, las salidas de la ejecución del sistema varían según el subsistema en el que las entradas hayan sido procesadas, dichas salidas fueron identificadas en conjunto con los datos que las componen y sus respectivos tipos. En la tabla 3.2 se listan cada una de las salidas junto con su descripción y el tipo y/o formato de dato.

Salidas	Tipo	Descripción
Modelo Entrenado	.h5	Es el resultado posterior al entrenamiento, un modelo para segmentar MRI.
Métricas	String	Son los resultados e indicadores del modelo, el sistema debe imprimirlos en consola.
Segmentación	.nii.gz	Es la segmentación de la imagen previamente cargada.

Tabla 3.2: Salidas del Sistema

3.4. Diagrama de Contexto

En la figura 3.1 se encuentra el diagrama de contexto, en este se muestran todas las entradas, salidas e interacciones relevantes presentes en el sistema. Como se ha mencionado previamente el sistema está compuesto por 2 subsistemas, entrenamiento de modelo y segmentación de imágenes, en el diagrama podemos visualizar las relaciones entre las entradas y las salidas en cada uno de los subsistemas.

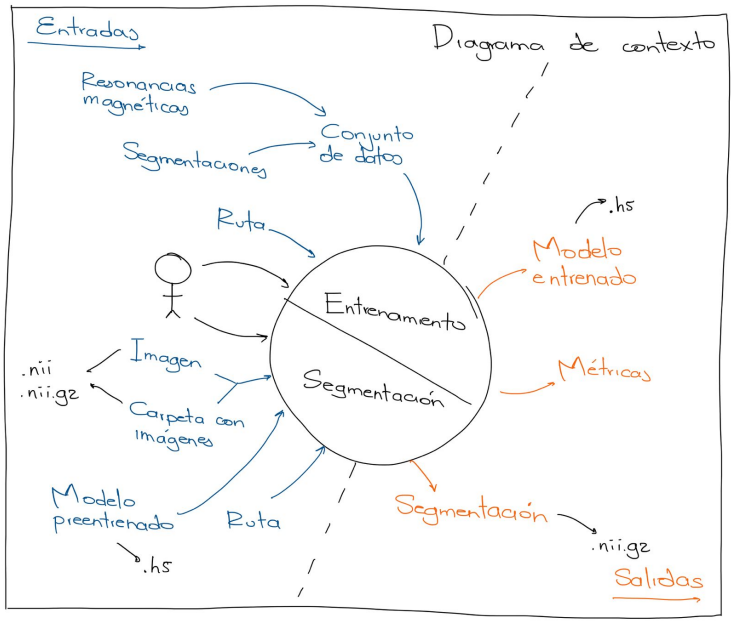

Figura 3.1: Diagrama de contexto

3.5. Casos de Uso

Uno de los resultados del proceso realizado en el documento "Especificación de Requerimientos" es el diagrama de casos de uso, este se puede apreciar en la figura 3.2, como se puede apreciar el sistema está compuesto por 2 casos de uso que son usados directamente por el usuario, esto son: Entrenar Modelo y Segmentar Imágenes.

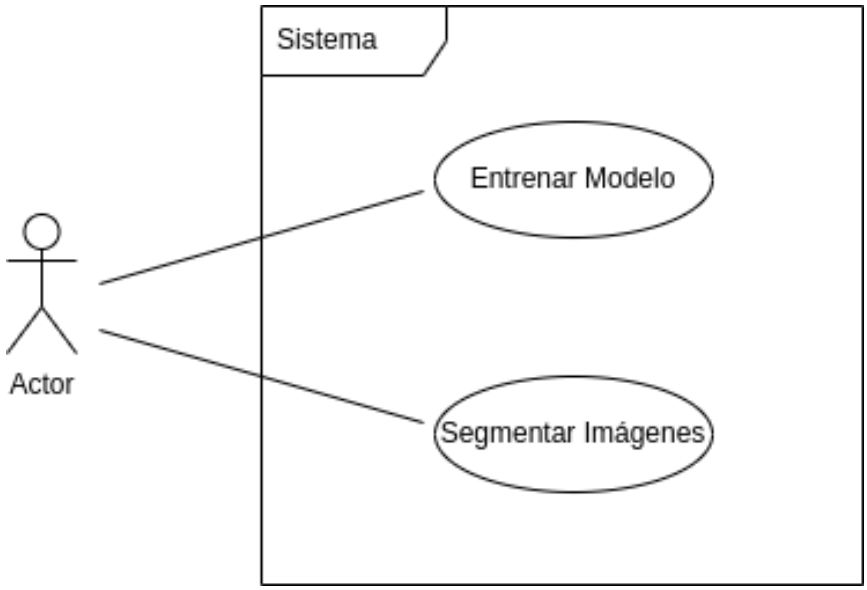

Figura 3.2: Diagrama de Casos de Uso

3.5.1. Tabla de Casos de Uso

ID	Caso de uso	Descripción
CU-1	Entrenar Modelo	El usuario le brinda un conjunto de datos al sistema, dichos datos son procesados por el sistema con el fin de encontrar patrones con los que pueda automatizar la tarea de segmentar imágenes por resonancia magnética.
CU-2	Segmentar Imágenes	El usuario le brinda al sistema imágenes por resonancia magnética, dichas imágenes son procesadas por el sistema haciendo uso de un modelo previamente entrenado, dando como resultado segmentaciones de las imágenes

Tabla 3.3: Casos de Uso

3.6. Drivers Arquitectónicos

En esta sección se encuentran los drivers arquitectónicos del sistema, dichos drivers comprenden los requerimientos funcionales, los requerimientos no funcionales y las restricciones. Adicionalmente, en esta sección también se encuentra un listado priorizado de todos los drivers arquitectónicos.

3.6.1. Requerimientos Funcionales

En la tabla 3.4 se encuentran listados los requerimientos funcionales identificados para el sistema, para mayor detalle del proceso de identificación de estos el proceso se encuentra en el documento anexo “Especificación de Requisitos de Software”.

RF-1	Cargar Imágenes
RF-2	Cargar Modelo
RF-3	Preprocesar Imágenes
RF-4	Entrenar Modelo
RF-5	Evaluar Modelo
RF-6	Segmentar Imágenes
RF-7	Guardar Modelo
RF-8	Guardar Segmentaciones
RF-9	Acciones por Comandos

Tabla 3.4: Requisitos Funcionales del Sistema.

3.6.2. Requerimientos No Funcionales

En la tabla 3.5 se encuentran listados los requerimientos no funcionales identificados para el sistema, para mayor detalle del proceso de identificación de estos el proceso se encuentra en el documento anexo "Especificación de Requisitos de Software".

RNF-1	El sistema implementa el 100 % de los casos de uso identificados.
RNF-2	Los resultados de las segmentaciones realizadas por un modelo deben ser congruentes con la calidad y cantidad de los datos, así como los hiperparámetros usados para su entrenamiento.
RNF-3	El tiempo que requiere el sistema para segmentar una imagen debe ser inferior al que le toma a un experto el realizar la misma segmentación de forma manual.
RNF-4	El sistema debe optimizar el uso de recursos como la memoria RAM y la memoria RAM dedicada de la GPU (VRAM) para que el entrenamiento de modelos se realice de forma satisfactoria.
RNF-5	El sistema no debe realizar segmentaciones y/o modificaciones sobre los archivos originales, sea para los datos que se usen en entrenamiento o en segmentación.

Tabla 3.5: Requisitos no funcionales del sistema.

3.6.3. Restricciones

En la tabla 3.6 se encuentran listadas las restricciones identificadas para el sistema, para mayor detalle del proceso de identificación de estas el proceso se encuentra en el documento anexo "Especificación de Requisitos de Software".

C-1	El sistema funciona única y exclusivamente para resonancias magnéticas.
C-2	El conjunto de datos utilizado para generar el modelo entrenado debe proceder de fuentes que garanticen la calidad de las imágenes.
C-3	El conjunto de datos de entrenamiento debe incluir tanto las imágenes como las segmentaciones de la región de interés.
C-4	El entrenamiento se hace por medio de técnicas de aprendizaje profundo (CNN).
C-5	Las imágenes de entrada y los resultados de las segmentaciones generadas por el sistema deben poder ser visualizadas en cualquier software que se especialice en su visualización y/o manipulación (formato NIfTI).
C-6	El sistema solo puede generar y cargar modelos de formato "Hierarchical Data Format" en su quinta versión (.h5).

Tabla 3.6: Restricciones del sistema.

3.6.4. Priorización de Drivers Arquitectónicos

En esta sección se encuentran los drivers arquitectónicos priorizados, el método utilizado para realizar la priorización se encuentra explicado a detalle en el documento anexo "Especificación de Requisitos de Software".

RNF-1: El sistema implementa el 100 % de los casos de uso identificados.

- **CU-1:** Entrenar Modelo.
  - **RF-1:** Cargar Imágenes.
    - **C-5:** Las imágenes de entrada y los resultados de las segmentaciones generadas por el sistema deben poder ser visualizadas en cualquier software que se especialice en su visualización y/o manipulación (formato NIfTI).

- **RF-4:** Entrenar Modelo.
  - **C-4:** El entrenamiento se hace por medio de técnicas de aprendizaje profundo (CNN).
  - **RF-5:** Evaluar Modelo
- **RF-7:** Guardar Modelo.
  - **C-6:** El sistema solo puede generar y cargar modelos de formato "Hierarchical Data Format" en su quinta versión (.h5).
- **RNF-4:** El sistema debe optimizar el uso de recursos como la memoria RAM y la memoria RAM dedicada de la GPU (VRAM) para que el entrenamiento de modelos se realice de forma satisfactoria.
  - **RF-3:** Preprocesar Imágenes.

- **CU-2:** Segmentar Imágenes.
  - **RF-1:** Cargar Imágenes.
    - **C-5:** Las imágenes de entrada y los resultados de las segmentaciones generadas por el sistema deben poder ser visualizadas en cualquier software que se especialice en su visualización y/o manipulación (formato NIfTI).
  - **RF-2:** Cargar Modelo.
    - **C-6:** El sistema solo puede generar y cargar modelos de formato "Hierarchical Data Format" en su quinta versión (.h5).
  - **RF-6:** Segmentar Imágenes.
  - **RF-8:** Guardar Segmentaciones.
    - **C-5:** Las imágenes de entrada y los resultados de las segmentaciones generadas por el sistema deben poder ser visualizadas en cualquier software que se especialice en su visualización y/o manipulación (formato NIfTI).

- **RF-9:** Acciones por Comandos

Nota: Los requerimientos no funcionales **RNF-2** y **RNF-3** fueron excluidos de esta priorización, esto dado que si bien son factores determinantes de la calidad del sistema, su naturaleza es de carácter experimental, por ello las estrategias para satisfacerlos se abordan en el plan de pruebas del sistema y en la ejecución del mismo.

Nota 2: El requerimiento no funcional **RNF-5** no está listado en la priorización, ya que pertenece a la categoría de atributos de calidad seguridad y a la subcategoría integridad. Teniendo en cuenta su descripción y las características en común de los elementos priorizados (Datos), se toma como un requerimiento asociado al sistema en todos sus niveles de abstracción y jerarquización.

Nota 3: La restricción **C-2** no será abordada, dado que el mantener control sobre dichos recurso es algo que se escapa del alcance y naturaleza de este proyecto.

Capítulo 4

Metodología de Diseño

En el presente capítulo se explica la metodología usada para el diseño de la arquitectura del sistema, se especifica a detalle cada uno de los pasos y las herramientas utilizadas para llevar a cabo el diseño.

4.1. ADD

ADD es un método para definir una arquitectura de software en la cual el proceso de software está basado en los requisitos de los atributos de calidad de software. Este método sigue un proceso de diseño recursivo para descomponer los elementos o un sistema en sí, aplicando tácticas arquitecturales y patrones que satisfagan los drivers. Esencialmente, ADD sigue el siguiente ciclo [1]:

- **Planear:** Los elementos del sistema tomados como entrada son considerados para elegir cuáles serán usados en la arquitectura.
- **Hacer:** Dichos elementos son instanciados para satisfacer los drivers, es decir, los requisitos funcionales y no funcionales, como así también los atributos de calidad.
- **Revisar:** El resultado es analizado para determinar si efectivamente se está dando cumplimiento a los requisitos.

4.1.1. Entradas y Salidas del ADD

Las entradas para el ADD son requerimientos funcionales, restricciones de diseño y requerimientos de los atributos de calidad, los cuales son priorizados por los stakeholders de acuerdo al negocio y sus objetivos. Los requerimientos especifican cuáles funcionalidades deben estar incluidas en el sistema y son necesarias para los interesados, las restricciones de diseño son las decisiones de diseño que deben ser incorporadas al diseño final del sistema y los requerimientos indican el grado en el cual un sistema debe exhibir varias propiedades. Distinguir entre estos tipos de entradas resulta no ser tan importante como asegurarse la completitud de ese conjunto antes de iniciar ADD. Para el sistema se utilizó como entrada los requisitos no funcionales, los cuales ayudan a identificar los atributos de calidad y otras características importantes que debe incluir el sistema [1].

La salida del ADD es un diseño del sistema en términos de roles, responsabilidades, y relaciones entre elementos de software. El diseño del resultado de ADD es documentado usando diferentes vistas de la tecnología, para este caso se utiliza 4+1 para la construcción de las vistas. En general, ADD produce una arquitectura de software inicial que describe y justifica un conjunto de decisiones que muestran [1].

- Como particionar un sistema en elementos de desarrollo.
- Qué elementos serán parte de las múltiples estructuras, su tipo, propiedades y las relaciones estructurales que paseen.
- Cuáles interacciones ocurren entre elementos, los mecanismos y propiedades de esas interacciones.

4.1.2. Pasos del ADD

A continuación en términos generales se describe cada uno de los pasos del ADD. Además, en la figura 4.1 se muestran detalles adicionales como las entradas, salidas, junto con el flujo y los pasos del proceso de diseño recursivo de ADD [1].

1. Se debe disponer de una lista de requisitos estables y priorizados que incluyan atributos funcionales, de restricciones y de calidad. Los requisitos funcionales se expresan mediante casos de uso, y los atributos de calidad se expresan mediante plantillas de escenarios de atributos de calidad.

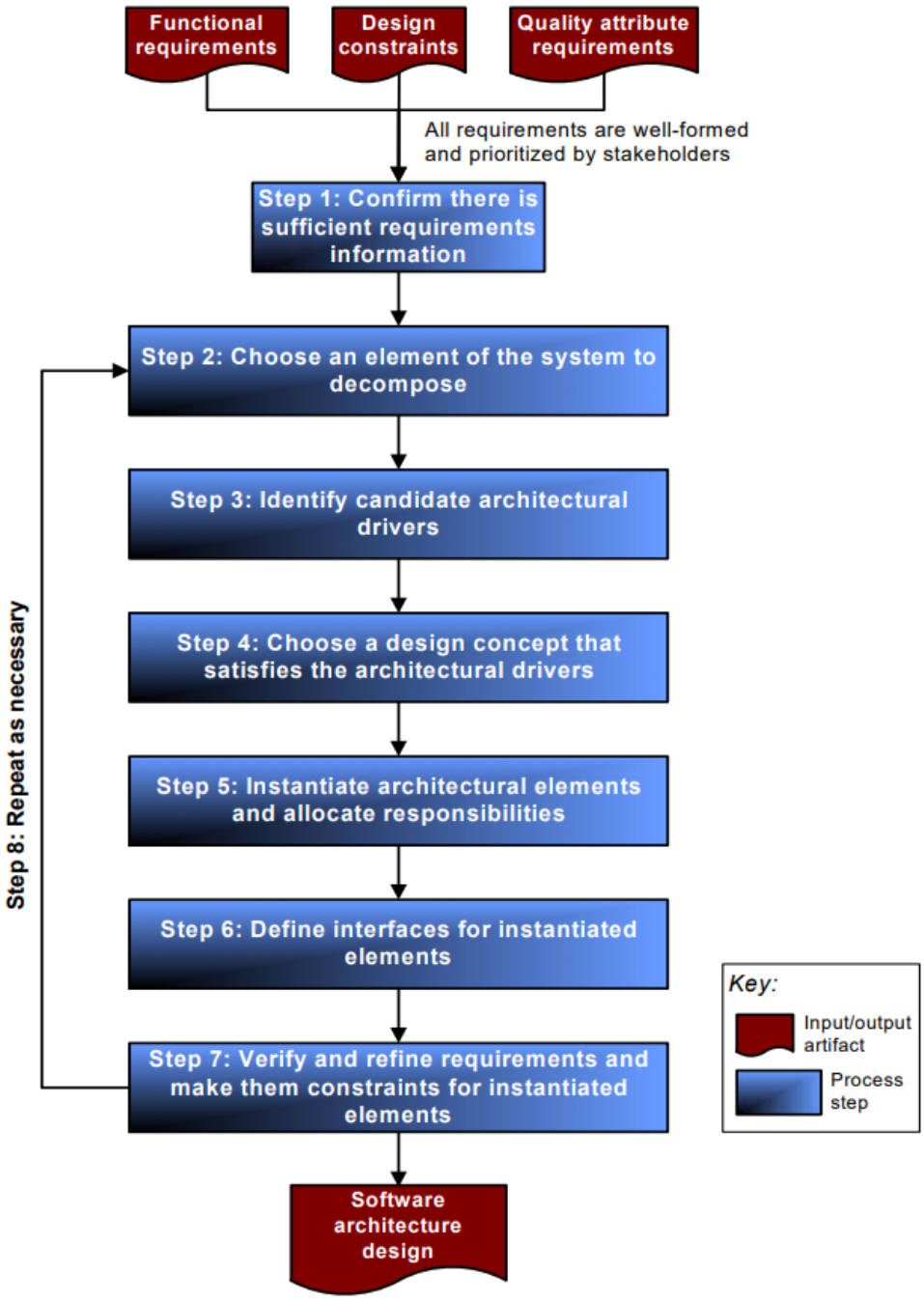

Figura 4.1: Pasos del ADD [1].

2. Si es la primera iteración, el sistema es el elemento a descomponer. Si no, se elige un elemento arquitectónico para empezar a descomponer.

3. La clasificación de los requisitos realizada por las partes interesadas se combina con las clasificaciones basadas en su impacto en la arquitectura. Se eligen entre cinco y seis requisitos de alta prioridad y se denominan candidatos a impulsores de la arquitectura.

4. Se eligen los tipos de elementos arquitectónicos y sus relaciones. Para identificarlos, las restricciones de diseño y los atributos de calidad se utilizan para identificar patrones y tácticas que los satisfacen, y las vistas arquitectónicas se capturan parcialmente.

5. El elemento arquitectónico elegido se descompone en elementos hijos. Las responsabilidades se asignan a los elementos hijos utilizando casos de uso, y las vistas arquitectónicas del paso 4 se complementan.

6. Se definen los servicios y propiedades que se requieren para cada elemento.

7. Se verifica que los requisitos funcionales, los atributos de calidad y las restricciones de diseño se han cumplido en la descomposición del elemento y que los requisitos funcionales y de atributos de calidad de los elementos hijos están asignados.

Después de la última iteración, se inicia un proceso iterativo a partir del paso dos con el objetivo descomponer los elementos arquitecturales. El número de iteraciones es arbitraria del diseñador o arquitecto y depende del nivel de abstracción y detalle que requiera el sistema.

Capítulo 5

Arquitectura del Sistema

5.1. Vista Lógica

En la figura 5.1 se puede apreciar la vista lógica, en esta muestra la estructura funcional del sistema.

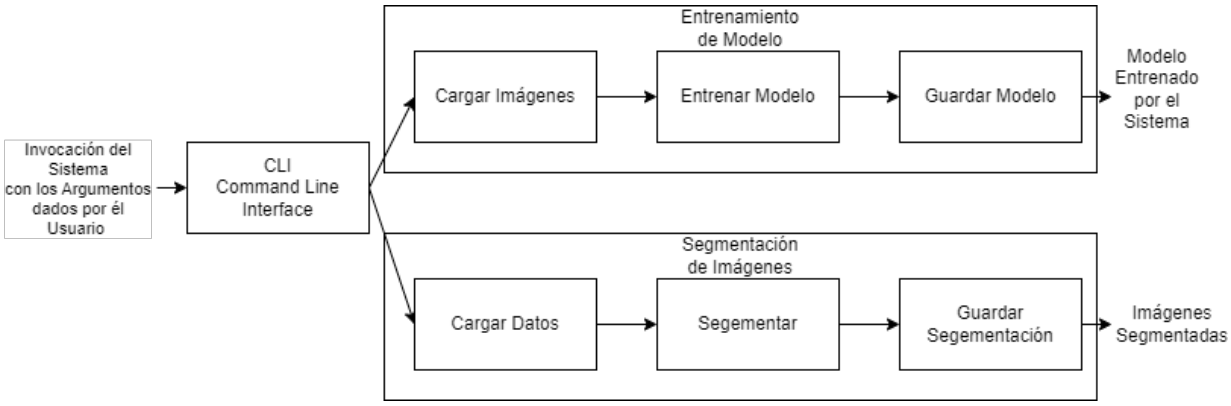

Figura 5.1: Vista Lógica del Sistema

Dado que esta vista determina la estructura funcional del sistema, sobre esta se tomaron y realizaron las primeras decisiones arquitectónicas, inicialmente se determinó el patrón/estilo arquitectónico, dado que se identificó que el sistema se basa en el uso y procesamiento de datos se exploraron diversas arquitecturas que suelen ser usadas para estos fines, dichas opciones fueron los patrones: Repository (Repositorio), Black Board

(Pizarrón) y Pipes and Filters (Tubos y filtros), siendo este último el seleccionado, los dos principales criterios de esta elección son su intuitiva implementación en comparación con las otras opciones y aún más relevante para el objetivo de este sistema, la nula interacción del usuario en el procesamiento de los datos en el sistema, como resultado de esta decisión se identificaron 2 subsistemas, estos son: Entrenamiento de Modelo y Segmentación de Imágenes.

Otra decisión arquitectónica relevante tomada en esta vista fue la descomposición de los subsistemas en unidades funcionales más pequeñas conocidas como "Componentes", dichos componentes se determinaron considerando las acciones principales que tiene que realizar cada subsistema para obtener el resultado que se proponen satisfacer, para el subsistema de "Entrenamiento de Modelo" se determinaron que las acciones son cargar las imágenes que se usan en entrenamiento, entrenar el modelo y guardar el modelo, por parte del subsistema de "Segmentación de Imágenes" las acciones son cargar los datos que permiten realizar las segmentaciones, segmentar los datos y guardar las segmentaciones.

5.2. Vista Procesos

En la figura 5.2 se puede apreciar la vista de procesos, dado el alcance y el propósito del sistema sobre esta vista no hay decisiones arquitectónicas relevantes. Sin embargo, en esta vista se puede apreciar claramente que cada ejecución del sistema está contenida en un único proceso, agregado a esto cabe resaltar que es posible ejecutar múltiples veces el sistema de forma simultánea, de este modo generando diferentes procesos.

Figura 5.2: Vista de Procesos del Sistema

5.3. Vista de Implementación

En la figura 5.3 se puede apreciar la vista de implementación, en esta se muestra con gran detalle el dónde y el cómo se relacionan diversas partes y otros aspectos relevantes del sistema que son de especial interés para los desarrolladores de este.

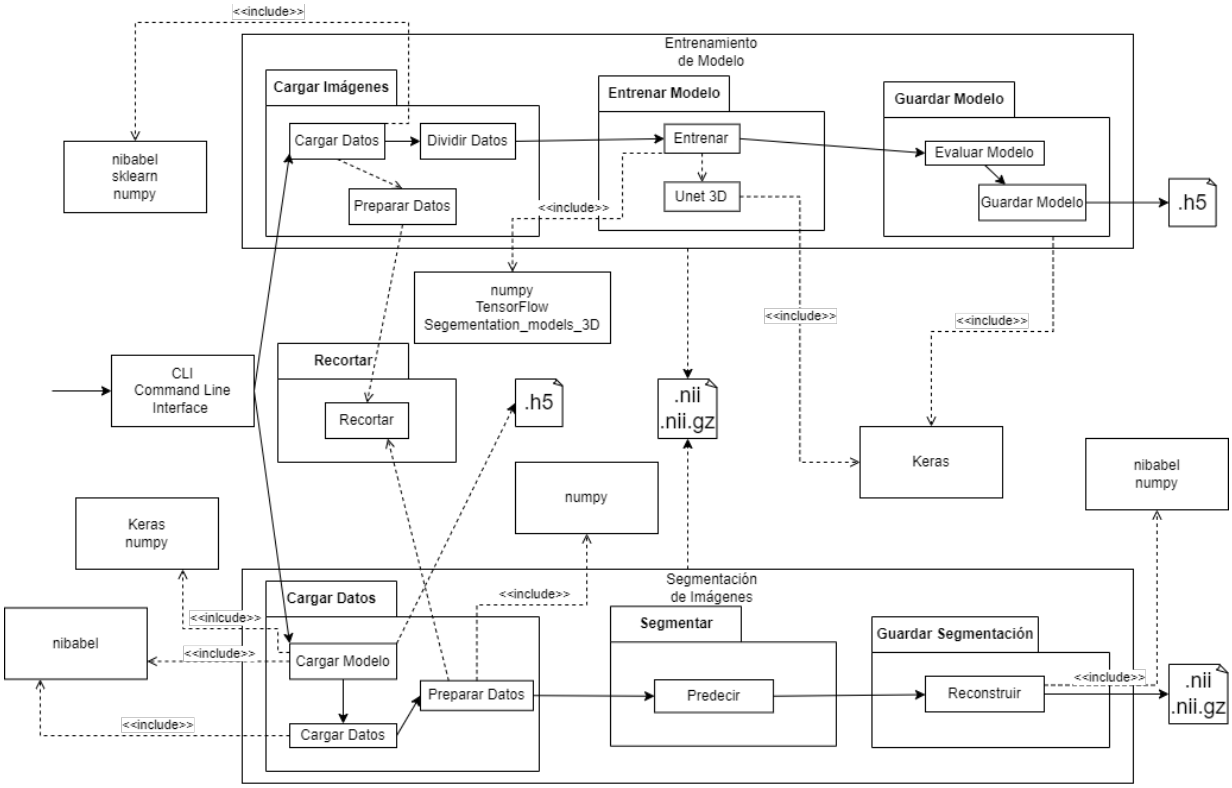

Figura 5.3: Vista de Implementación del Sistema

Al igual que en el caso de la vista lógica, en la vista de implementación también se tomaron decisiones arquitectónicas relevantes para el sistema. La primera de estas decisiones fue el seguir descomponiendo las unidades funcionales obtenidas de la vista lógica del sistema, esto con el fin de descomponer los componentes en unidades funcionales aún más pequeñas conocidas como "Módulos" y asignar a dichos módulos acciones aún más especificas de lo que deben realizar para satisfacer la función principal del componente que los contiene.

Una decisión asociada a la descomposición de los componentes en módulos es el cómo

estos módulos se relacionan en la totalidad del sistema, naturalmente dichas relaciones no son todas iguales, por ello es necesario determinar y diferenciar dichas relaciones, como proceso de la toma de decisiones se determinaron que existen 3 tipos de relaciones en el sistema propuesto, dichas relaciones son:

- **Uso:** Esta relación está identificada por una flecha negra, esta indica que un módulo hace uso de la funcionalidad de otro, sin embargo, no requiere de un retorno de información como resultado para cumplir con su propósito.
- **Dependencia funcional:** Esta relación está identificada por una línea punteada con una flecha abierta, esta indica que un módulo hace uso de la funcionalidad de otro, la diferencia de esta radica en que la funcionalidad principal del módulo que realiza el llamado requiere de un retorno de información para cumplir con su propósito.
- **Dependencia de formato:** Esta relación está identificada por una línea puenteada con una flecha negra al final, esta indica que la unidad funcional asociada solo puede hacer uso de recursos en un formato especificado.

Finalmente, la última decisión relevante tomada en esta vista es el determinar las librerías de las cuales hace uso o incluso implementa cada una de las unidades funcionales del sistema, dichas librerías son las siguientes: Numpy, Nibabel, SkLearn, Keras, TensorFlow y Segementation_Models_3D, resultado de esta decisión se generó un caso especial para el componente de guardar modelo, a pesar de estar representado en la vista de implementación, este no será implementado por el equipo de desarrollo y la funcionalidad que este satisface será realizada por funciones ya implementadas por Keras, sin embargo, se decidió dejar dicho componente y sus módulos asociados en la vista para ejemplificar como es su funcionamiento y que el lector pueda ver e intuir el flujo de procesamiento y datos del sistema.

5.4. Vista Física

En la figura 5.4 se puede apreciar la vista física, en esta se muestra el dónde y qué procesos corren el hardware que ejecuta el sistema.

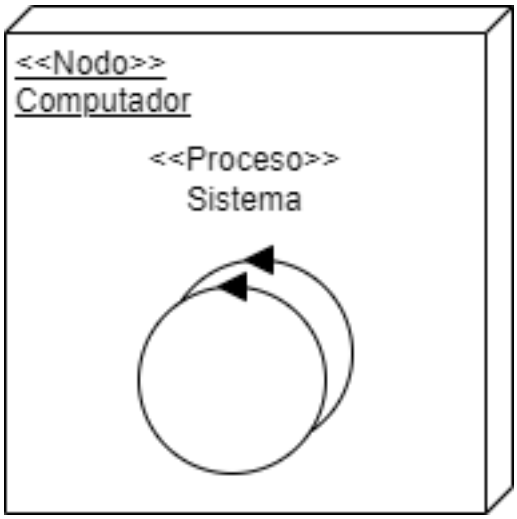

Figura 5.4: Vista de Física del Sistema

Al igual que en la vista de procesos, sobre esta no hay decisiones relevantes, sin embargo, es importante aclarar que, si bien es posible correr el sistema propuesto en cualquier dispositivo, quizás sea necesario (dependiendo del hardware), el modificar la implementación de algunos módulos para que partes del sistema puedan ser ejecutados y/o reducir el volumen de datos dados para la ejecución de las funcionalidades del sistema. Otro punto importante respecto a esta vista es él cuantos procesos de la ejecución del sistema pueden ser ejecutados en el dispositivo, en teoría es posible hacerlo, pero esto dependerá de las especificaciones de el dispositivo, la función que se ejecute en el sistema, el volumen de datos que se empleara en el sistema y aspectos de la implementación de la red neuronal.

5.5. Vista de Arquitectura de Red Neuronal

Teniendo en cuenta las vistas anteriormente mencionadas y en pro de cumplir con los requerimientos del sistema, así como con sus restricciones, se hicieron decisiones relevantes que ayudaron a una óptima construcción de la arquitectura de la red neuronal convolucional presentada en la imagen 5.5. Esta arquitectura se divide en dos partes:

- **Trayectoria de contracción:** Utilizada para capturar el contexto en la imagen; tanto la región de interés como sus demás partes.

- **Trayectoria de expansión:** Utilizada para permitir una localización precisa, en este caso de la región de interés, mediante convoluciones transpuestas.

El número de convoluciones usadas en las capas de las trayectorias fue decidido con el propósito de abarcar la mayor información de la imagen en las primeras capas, optimizando el uso de los recursos, para realizar filtros de mayor calidad (a lo largo de las trayectorias) que ayudaron a una identificación exitosa de la región de interés en la imagen.

Dentro de la trayectoria de contracción se comienza con dos capas de 64 convoluciones. Estas convoluciones se encargan de dividir la imagen en volúmenes de 3∗3∗3 con el propósito de encontrar patrones en la imagen para la identificación de la región de interés. Luego, se utiliza un *max pooling* de $2 * 2 * 2$ con el fin de reducir estas divisiones maximizando el patrón encontrado. Después de la trayectoria de contracción, y copiando el contexto encontrado en esta, se encuentra la trayectoria de expansión. Esta trayectoria empieza con un *transpose* de $2 * 2 * 2$ que ayuda a reconstruir la imagen dividida en la trayectoria de contracción copiando el contexto de las etapas de esta trayectoria. Luego, se tienen dos capas de 512 convoluciones de $3 * 3 * 3$ que ayudan a localizar la región de interés en el contexto copiado de la imagen. Estos filtros de patrones, reconstrucción y localización, se siguen haciendo a lo largo de las dos trayectorias para al final obtener una capa de 2 convoluciones de $1 * 1 * 1$ que clasifica la región de interés en cada vóxel de la imagen.

La trayectoria de contracción cuenta con la función de activación sigmoide dentro de sus capas. Esta función permite flexibilidad en el filtro de las imágenes para aceptar más características que ayudan a diferenciar la región de interés, y sus partes similares, de las otras partes de la imagen que no tienen importancia. De ese modo, teniendo estos filtros, en la trayectoria de expansión se utiliza la función de activación ReLU para usar un filtro más estricto que ayuda a separar esta región de interés de las demás partes filtradas. Al final de la trayectoria de expansión se tiene una capa de 2 neuronas con activación sigmoide que realizan la clasificación de esta región de interés en todos los vóxeles de la imagen.

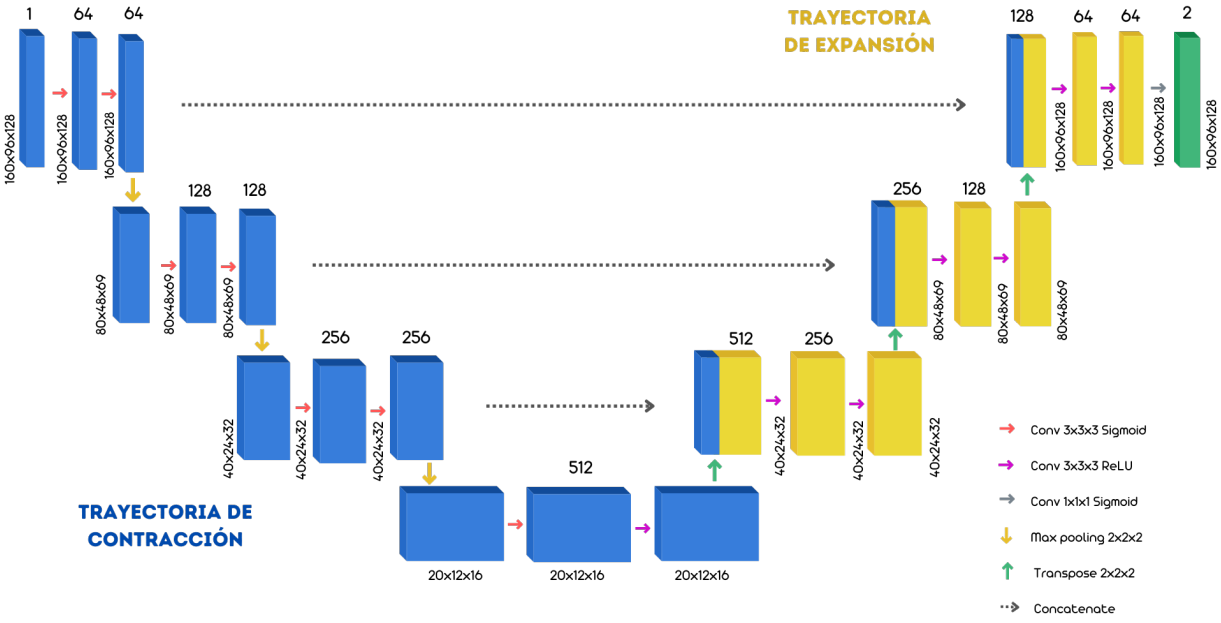

Figura 5.5: Arquitectura red neuronal convolucional.

Referencias

[1] Rob Wojcik, Felix Bachmann, Len Bass, Paul Clements, Paulo Merson, Robert Nord, and Bill Wood. Attribute-driven design (add), version 2.0 software architecture technology initiative, 11 2006.