Desarrollo de un módulo de gestión de imágenes médicas utilizando el dataset COVID-19 Radiography

E Rodas Banegas, M. Centellas Hinojosa, N. Arce Arnez and E.Miranda
Unidad de Postgrado de la FICCT, Universidad Autónoma Gabriel René Moreno Santa Cruz, Bolivia
elmarcinho23@gmail.com, milcacentellas@gmail.com, ninoskaarcearnez@gmail.com and
emirandaa@soe.uagrm.edu.bo

Resumen - Este proyecto muestra el desarrollo de un módulo para la gestión de imágenes médicas y su metadata, con base al dataset COVID-19 Radiography. El módulo permite registrar, modificar, listar y eliminar imágenes. En su primera versión, utiliza estructuras de datos tipo diccionario, mientras que en la segunda versión implementa almacenamiento en formato JSON aplicando principios de programación orientada a objetos y modularización. La solución fue desarrollada en Python y permite la manipulación de imágenes su metadata desde la terminal en una interacción directa con el usuario. Además, se implementó un repositorio versionado mediante GitHub, siguiendo buenas prácticas de documentación y estilo (PEP8).

Palabras claves: Gestión de imágenes médicas; COVID-19 Radiography, dataset, metadata,Python, programación orientada a objetos, modularización, diccionario, JSON.

Abstract - This project presents the development of a module for managing medical images and their metadata, based on the COVID-19 Radiography dataset. The module allows for registering, modifying, listing, and deleting images. In its first version, it uses dictionary-type data structures, while in the second version, it implements JSON-based storage, applying principles of object-oriented programming and modularization. The solution was developed in Python and enables image and metadata manipulation directly from the terminal through user interaction. Additionally, a version-controlled repository was implemented using GitHub, following best practices for documentation and coding style (PEP8).

.

Keywords: Medical image management; COVID-19 Radiography; dataset; metadata; Python; object-oriented programming; modularization; dictionary; JSON.

I. INTRODUCCIÓN

En los últimos años, el uso de herramientas computacionales en el ámbito médico ha cobrado una importancia notable, sobre todo en lo que respecta al análisis de imágenes médicas. Durante la pandemia de COVID-19, se crearon bases de datos como el COVID-19 Radiography, que alberga miles de radiografías clasificadas en categorías como normales, COVID-19 y neumonía. Este trabajo tiene como objetivo desarrollar un módulo funcional que permita gestionar imágenes médicas, asegurando que su metadata sea organizada y almacenada en formato JSON. La solución esta implementada en Python que permite la manipulación de las imágenes a través de la termina de línea de comandos. Además, el proyecto se encuentra versionado en GitHub, siguiendo buenas prácticas de documentación y estilo de código.

II. METODOLOGÍA

Para el desarrollo de la solución propuesta se aplicaron principios fundamentales de programación orientada a objetos, modularización, con el objetivo de estructurar el código en paquetes independientes, reutilizables y fácilmente escalables tal como se muestra en las figuras [1] y [2]. La implementación fue realizada utilizando Python 3.12, con una arquitectura basada en el patrón de diseño Modelo-Vista-Controlador (MVC).

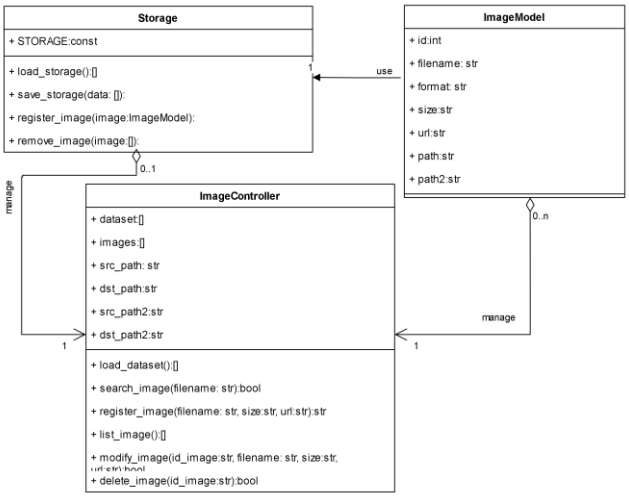

Figura 1 Diagrama de clases para la gestión de imágenes médicas y su metadata en archivo JSON

.

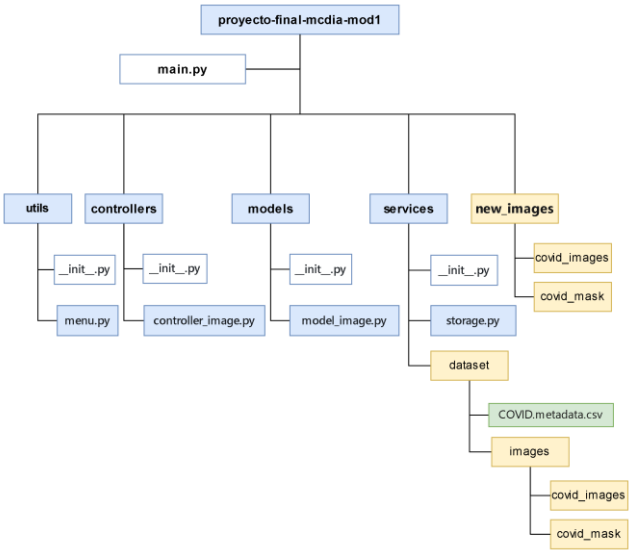

Figura 2 Diagrama de paquetes para la gestión de imágenes médicas y su metadata en archivo JSON

Esta estructura permite separar claramente las responsabilidades de cada componente del sistema: los modelos representan los datos, los controladores gestionan la lógica de negocio y la interacción con los usuarios, y los servicios se encargan de la persistencia de la información. A continuación, se describe la organización del proyecto.

A. *Controllers*

Este paquete contiene se encarga de gestionar la lógica de la aplicación y coordinar la interacción entre el modelo y los servicios:

- __init__.py: archivo de inicialización del paquete.
- *controller_image.py*: incluye la clase *ImageController*, que implementa las funciones CRUD (crear, leer, actualizar y eliminar) sobre las imágenes. Además, se encarga de la carga y persistencia de la metadata en un archivo JSON. También permite la lectura de un conjunto de datos en formato CSV que contiene la metadata de las imágenes, la cual es posteriormente es persistida en el archivo JSON.

B. *Models*

Este paquete contiene las estructuras de datos utilizadas en la aplicación:

- __init__.py: archivo de inicialización del paquete
- *model_image.py*: contiene la clase *ImageModel*, utilizada como DTO (Data Transfer Object) para representar la metadata asociada a una imagén médica.

C. *Services*

Encapsula las funcionalidades para la persistencia y acceso a datos:

- *__init__.py:* archivo de inicialización del paquete.
- *storage.py:* contiene la clase *Storage*, responsable de implementar el CRUD para la gestión de archivos JSON, en donde se almacena la metadata de las imágenes médicas.
- Carpeta Dataset: contiene el archivo .csv con la metadata original de las imágenes.
- Carpeta Images: contiene subcarpetas con las imágenes y máscaras originales:
  - Covid_images
  - Covid_masks

D. *New images*

Carpeta utilizada para almacenar imágenes y máscaras nuevas que son generadas durante las operaciones del sistema, contiene nuevas imágenes y máscaras COVID cargadas o manipuladas.

E. *Utils*

Contiene herramientas de apoyo para la interacción con el usuario con la terminal:

- *__init__.py*: archivo de inicialización del paquete
- *menú.py*: implementa las opciones del menú de opciones en la terminal para el usuario.

F. *Interfaz de Usuario*

El sistema está diseñado para funcionar desde la terminal mediante funciones de entrada y salida (input() y print()), permitiendo al usuario:

- Listar imágenes
- Registrar nuevas imágenes
- Modificar registros existentes
- Eliminar imágenes de la base de datos

G. *Persistencia de Datos*

Se decidió utilizar archivos JSON para la persistencia de la metadata, aprovechando las ventajas que ofrece este formato en términos de flexibilidad, legibilidad y compatibilidad con estructuras de tipo diccionario en Python. Esta decisión permitió una manipulación eficiente y estructurada de la información asociada a las imágenes médicas.

H. *Versionamiento*

El control de versiones del proyecto se gestionó mediante la plataforma GitHub, permitiendo el trabajo colaborativo entre los integrantes del equipo.

I. *Documentación*

La documentación del proyecto se encuentra disponible en el archivo README.md, ubicado en el repositorio compartido en GitHub. Este documento describe la estructura del proyecto, las instrucciones de instalación y uso.

III. RESULTADOS

La implementación del sistema propuesto permitió verificar la virtud del enfoque modular y orientado a objetos para la gestión de imágenes médicas y su metadata. La arquitectura basada en el patrón MVC demostró ser adecuada para mantener la separación de responsabilidades, facilitando tanto la mantenibilidad como la escalabilidad del proyecto.

El desarrollo de la clase *ImageController* permitió realizar de manera efectiva las operaciones CRUD sobre las imágenes y su metadata, gestionando la carga inicial desde archivos .csv y almacenando la información estructurada en archivos .json.

A través de la terminal, se habilitó una interfaz de usuario simple y funcional que permite:

- Registrar nuevas imágenes con su metadata.
- Modificar campos específicos de la metadata e imágen médica.
- Listar imágenes disponibles en el sistema.
- Eliminar imágenes.

Durante las pruebas del sistema, se utilizaron imágenes de COVID-19 y sus respectivas máscaras, organizadas en carpetas específicas. Estas imágenes fueron manipuladas y su metadata fue sincronizada en el archivo JSON.

Además, se verificó que el módulo storage.py, responsable de la persistencia, maneja adecuadamente operaciones sobre la metadata de las imágenes médicas, permitiendo su reutilización en proyectos similares que requieran gestión de datos no relacionales.

Se implementó control de versiones en GitHub con seguimiento de cambios y documentación técnica adecuada. La solución se encuentra en el repositorio https://github.com/Elmarcinho/proyecto-final-mcdia-mod1.git

Finalmente, se verificó que la estructura modular del sistema facilita su integración con nuevas funcionalidades que incorporen interfaces gráficas o el uso de bases de datos relacionales.

IV. CONCLUSIÓN

El desarrollo de este módulo ha demostrado la utilidad de aplicar principios de la programación orientada a objetos y la modularización en la construcción de soluciones de software funcional, mantenibles y escalables. La adopción de una arquitectura basada en el patrón MVC facilitó una clara separación de responsabilidades. El uso de dataset COVID-19 Radiography permitió simular un entorno real para la gestión de imagenología médica, brindando un contexto valioso para poner en práctica conocimientos adquiridos en modulo fundamentos de ingeniería de software para científicos de datos. Asimismo, el desarrollo promovió la aplicación de buenas prácticas de codificación, documentación y control de versiones colaborativo mediante GitHub.

Como parte del trabajo futuro, se identifican varias líneas de mejora:

- *Incorporación de una interfaz gráfica (GUI)* o el desarrollo de una aplicación web, con el objetivo de mejorar la experiencia del usuario y facilitar la interacción con el sistema.
- *Migrar la persistencia de datos en archivo JSON a una base de datos*, lo cual permitiría una gestión más robusta, segura y escalable de la metadata.
- *Automatización del procesamiento y clasificación de imágenes*, integrando técnicas de aprendizaje automático para la detección y organización de nuevas imágenes médicas sin intervención manual.

Estas propuestas de mejora contribuirían a la evolución del sistema hacia una herramienta más completa, flexible y adaptable a necesidades reales del entorno clínico o investigativo.

REFERENCES

[1] J. P. Cohen, P. Morrison y L. Dao, “COVID-19 Image Data Collection,” arXiv preprint arXiv:2003.11597, 2020. [En línea]. Disponible en: https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database

[2] D. Mertz, Programming Python: Object-Oriented Programming, O’Reilly Media, 2001.

[3] Python Software Foundation, “PEP 8 – Style Guide for Python Code.” [En línea]. Disponible en: https://peps.python.org/pep-0008/

[4] GitHub Inc., “GitHub: Where the world builds software.” [En línea]. Disponible en: https://github.com/

[5] Introducing JSON, “Introducing JSON (JavaScript Object Notation),” JSON.org. [En línea]. Disponible en: https://www.json.org/json-en.html