Proyecto Estructura de Datos

Entrega Inicial

Grupo 81.02
Alfons Baiges Parada - 1744020
Héctor Berger Martínez - 1751751
Diogo da Costa Lã Cabrita Teixeira - 1751533

Calificación en Caronte - 71

Implementación utilizada para cada estructura

En este proyecto hemos intentado aplicar las estructuras de datos vistas en clase no solo para ganar experiencia en la utilización de las mismas y familiarizarnos con ellas, sino también para ganar cierto nivel de eficiencia aunque no sea lo fundamental de esta parte del trabajo. Dicho lo anterior, podemos empezar a explicar qué estructuras hemos utilizado para las diferentes clases.

1. Image Files

En nuestro caso, esta clase tiene tres atributos: prev, added y removed, hemos optado por un set de python para prev y listas para los otros. Existe este contraste entre las estructuras de datos porque en el método reload_fs() tenemos que actualizar nuestra clase con una nueva serie de archivos. Como consecuencia, nos va bien utilizar los operadores de conjuntos de python para realizar operaciones entre ellas. Por otro lado added y removed son listas porque el enunciado indica que deben devolverse listas, no conjuntos y al mismo tiempo no las usamos para búsquedas intensivas, esta combinación permite mejorar la eficiencia en el cálculo de cambios siendo a la vez compatible con los formatos esperados.

2. Image ID:

En la clase ImageID hemos utilizado dos diccionarios, uno para asociar el archivo a su uuid y el otro para asociar el uuid al archivo. Elegimos diccionarios porque proporcionan acceso en tiempo constante tanto al consultar el UUID a partir del archivo como a la inversa. Esta estructura es ideal para garantizar que cada imagen tenga un identificador único y para evitar colisiones mediante comprobaciones por ambos lados. Además, simplifica la eliminación de un UUID concreto sin necesidad de recorrer estructuras completas.

3. **Image Data:**

ImageData gestiona toda la información asociada a cada UUID, almacenando el path del archivo, todas las metadades (prompt, modelo, pasos, etc.) y las dimensiones de la imagen. Para ello utilizamos un diccionario principal, donde la clave es el UUID y el valor es otro diccionario con los campos asociados. Esta estructura nos ayuda a agrupar toda la información de manera compacta, permite acceso rápido con getters y facilita la actualización de metadatos sin tener que reescribir todo.

4. **Image Viewer:**

La clase Image Viewer no requiere de ninguna estructura compleja, ya que la función es recuperar información desde Image Data, mostrarla por pantalla y mostrar la imagen mediante PIL, por lo tanto no implementa nuevas colecciones propias más allá de la referencia a Image Data.

5. **Search MetaData :**

En esta clase solo utilizamos listas para almacenar resultados de búsqueda.
Es importante resaltar el uso de una lista para guardar los uuids de las imágenes que tienen la cadena que se está buscando en su metadata. También implementamos los operadores lógicos AND y OR basados en operaciones entre listas, sin estructuras adicionales, ya que el volumen de datos es reducido y el objetivo es claridad y simplicidad.

6. **Gallery :**

Para gestionar la lista de imágenes dentro de una galería, hemos utilizado la estructura deque de la librería collections, que actúa como una lista doblemente enlazada. Hemos elegido esta opción, ya que esta permite inserciones y borrados, facilita operaciones como add_image_at_end(), remove_first_image() o remove_last_image() y nos ha permitido trabajar con una estructura distinta a las listas estándar, tal como se sugiere en la asignatura. Deque nos ha parecido una opción excelente para representar una galería que puede crecer o reducirse.

Pruebas Realizadas

Hemos realizado múltiples tests de pruebas para asegurar el funcionamiento del sistema.

- Inicialización de todas las clases, comprobando que no generen errores.

Flujo complejo típico:

- Cargar la colección de imágenes con Image Files
- Generar UUID para todas las imágenes con ImageID
- Leer metadatos de algunas imágenes con ImageData
- Visualizar imágenes con Image Viewer
- Cargar y mostrar galerías con Gallery
- Hacer búsquedas y mostrar galerías con Search Metadata

También hemos hecho test para ver donde puede fallar (archivos inexistentes, UUIDs repetidos, galerías vacías…)

Búsquedas realizadas

Prompt:

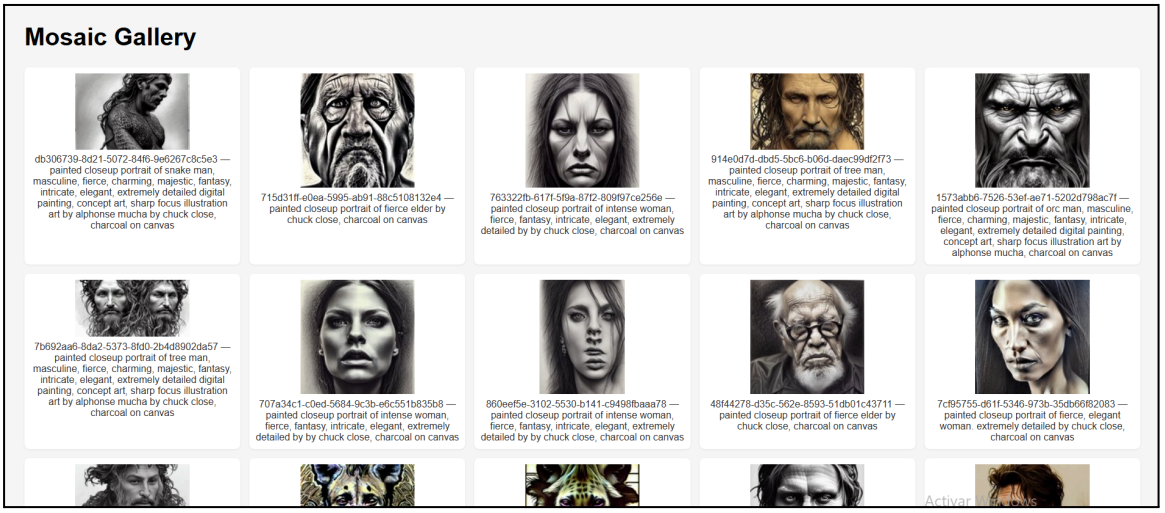
- George Washington: 9 imágenes encontradas
- Painted Closeup: 38 imágenes con este prompt
- Dog: 8 imágenes con este prompt

Steps:

- 1000 imágenes con 50 steps

Galerias Creadas

Hemos creado una galería con el resultado de “painted closeup”



Anotaciones

Git Repository: https://github.com/HectorBerger/Projecte_Estructura_Dades

En este repositorio tenéis la versión más reciente del código que no es la misma versión que la que se encuentra en el caronte ya que la versión de caronte era la que tenía la puntuación máxima. Aquí podéis consultar los test que hemos hecho.

Diagrama Estrutural: