

Universidad San Carlos de Guatemala  
Facultad de Ingeniería  
Escuela de Ciencias y sistemas  
Estructuras de Datos  
Ingenieros:

- Ing. Luis Espino
- Ing. Jesus Guzmán
- Ing. Álvaro Hernández

Auxiliares:

- Alex Lopez
- Walter Mach
- Wilfred Perez



## Proyecto 1 Fase 2 **UDrawing Paper** Implementación de estructuras no lineales

## ÍNDICE

<b>Objetivos</b>	<b>3</b>
Objetivo general	3
Objetivos específicos	3
<b>Descripción</b>	<b>4</b>
<b>    Registro de Usuarios y login</b>	<b>4</b>
<b>    Gestión de imágenes</b>	<b>4</b>
Capa (a nivel de aplicación)	4
Capa (a nivel lógico)	5
Imagen	6
Álbumes	6
Clientes	7
Carga Masiva	7
<b>    Funcionamiento</b>	<b>9</b>
<b>        Generación de imágenes</b>	<b>9</b>
Por recorrido limitado	9
Por Árbol de Imágenes	9
Por capa	9
ABC	10
Clientes	10
Imágenes	10
Visualización del estado de las estructuras	10
Ver árbol de imágenes	10
Ver árbol de capas	10
Ver Listado de álbumes	10
Ver capa	10
Ver imagen y árbol de capas	11
Ver árbol de clientes	11
<b>        UI de la aplicación</b>	<b>11</b>
<b>        Reportes</b>	<b>11</b>
<b>    Resumen de estructuras utilizadas</b>	<b>12</b>
<b>    Restricciones</b>	<b>12</b>
<b>    Observaciones</b>	<b>13</b>
<b>    Entregables</b>	<b>13</b>

# Objetivos

## Objetivo general

- Aplicar los conocimientos del curso Estructuras de Datos en el desarrollo de soluciones de software.

## Objetivos específicos

- Aplicar los conocimientos adquiridos sobre estructuras de datos lineales y no lineales como matrices y árboles
- Implementar una aplicación de escritorio utilizando el lenguaje de programación Java
- Familiarizarse con la lectura y escritura de archivos de JSON.
- Utilizar la herramienta Graphviz para graficar estructuras de datos no lineales.
- Definir e implementar algoritmos de búsqueda, recorrido y eliminación en estructuras de datos.

// Una img está formada por un “bitman” donde tenemos el color y la posición del pixel, esto lo vamos a emular con matrices

// estas capas las vamos a generar antes para poder concatenarlas entre ellas y generar una img después

## 1. Descripción

Se desarrolla una aplicación de escritorio que pueda ejecutarse independientemente del sistema operativo de la pc, por lo que se propone crear una aplicación de escritorio en el lenguaje de programación Java, esta aplicación debe permitir a los clientes de la empresa UDrawing Paper registrar imágenes especiales construidas por capas. Para poder hacer uso de la aplicación el cliente debe registrarse.

La principal funcionalidad de la aplicación consiste en un generador de imágenes por capas, la aplicación contará con un conjunto de capas cargadas previamente y almacenadas en memoria para ser utilizadas, estas capas se utilizarán para generar imágenes hechas con pixeles, cada capa contendrá la información de los distintos pixeles y al colocar una capa sobre otra estas irán formando una imagen más completa.

El sistema es capaz de generar imagen seleccionando las capas deseadas

## 2. Registro de Usuarios y login

Se deberá contar con la funcionalidad de registrar usuario en la aplicación, los datos solicitados son:

- Nombre completo
- DPI
- password

La aplicación contará con una pantalla para el inicio de sesión en la cual un cliente registrado podrá ingresar a la aplicación y gestionar sus imágenes.

## 3. Gestión de imágenes

### 3.1. Capa (a nivel de aplicación)

Las capas se almacenarán utilizando matrices dispersas dado que no existe una dimensión fija para las capas, en cada nodo se almacenará el color del píxel en Hexadecimal.



Imagen0

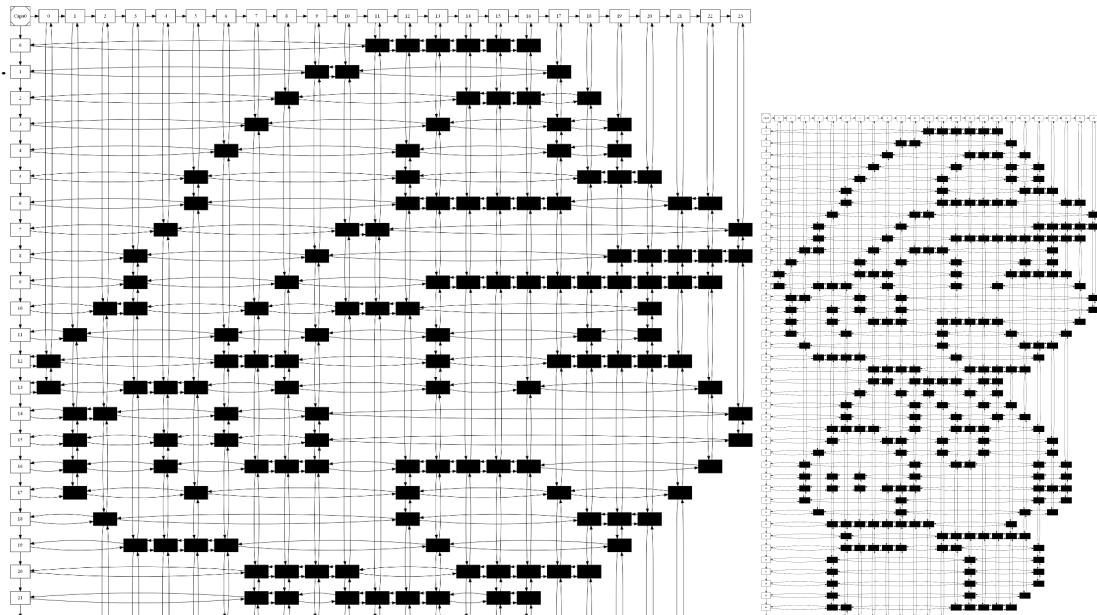
// cada cuadrito sería una posición en nuestra matriz, con un color negro

//método que nos dice cuantas filas y columnas hay maximo, y con eso ya sabemos las dimesiones de la img, así sabemos en donde debemos agregar nodos blancos, en grafica html solo agragamos celdas en blanco

### 3.2. Capa (a nivel lógico)

La capa a nivel lógico estará representada por una estructura matricial, se propone una matriz dispersa o poco densa para dar solución a tal necesidad. Por lo tanto, una imagen como la anterior dentro de la matriz quedaría así:

//Si estamos en el nodo 90. y no hay nodos anteriores, estos nodos estarán en blanco, no es necesario agregar los nodos a la matriz con que se representen en el grafo todo ok

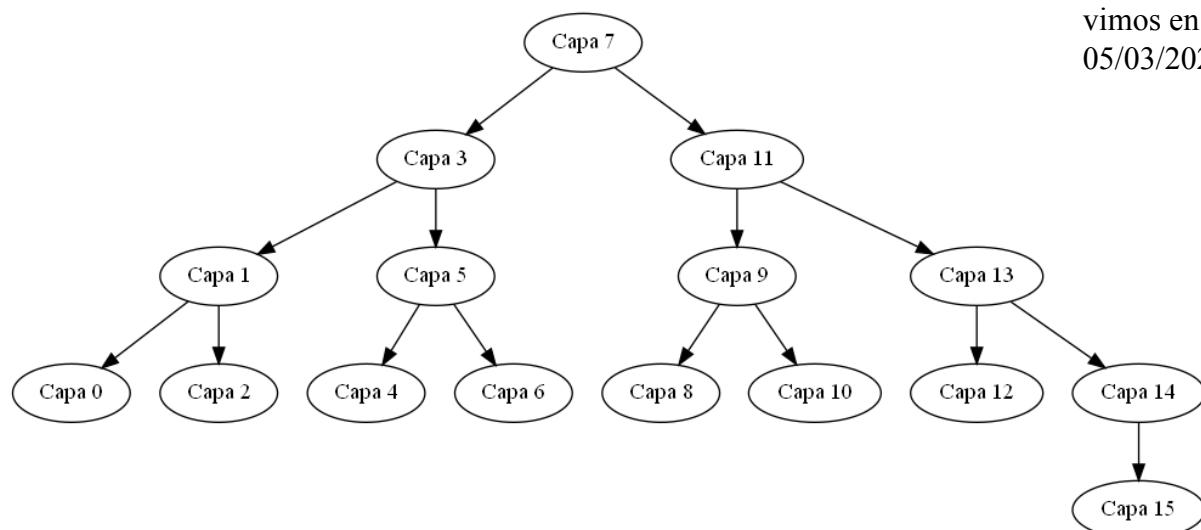


// el aux dijo que una forma de graficar (en graphviz) es por medio de una tabla html

Consideraciones:

- Al generar la imagen en las áreas donde no existe color se mostrará blanco.
- Las capas son únicas por lo cual una capa puede aparecer en cualquier imagen solo una vez estas tendrán un id que las identificará.
- Las capas deben almacenarse en un árbol binario de búsqueda (ABB).

// estos son los que vimos en la clase del 05/03/2022



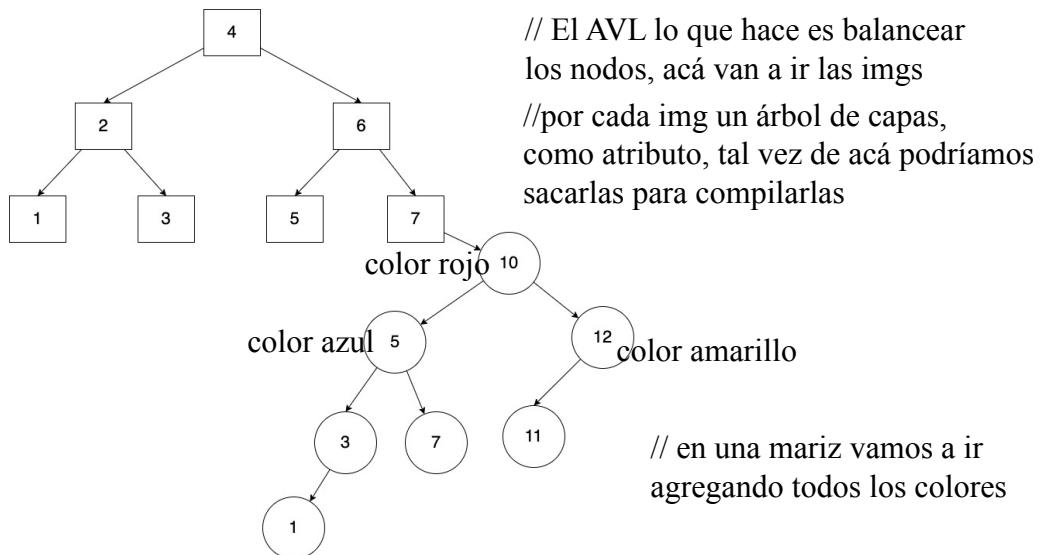
// Podríamos generar una img de solo el contorno del mario solo sería una capa si lo quiero de todos colores ya juntaríamos todas las capas

### 3.3. Imagen

Una imagen es el resultado de superponer una o más capas, las imágenes se pueden generar de diferentes maneras que serán definidas más adelante.

Las imágenes generadas se almacenarán en un árbol AVL donde cada imagen posee a su vez un árbol binario de búsqueda que permite identificar qué capas la conforman.

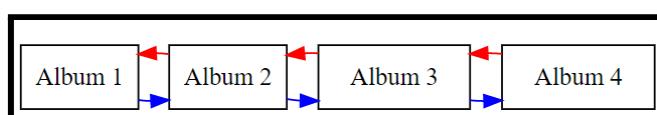
// las entradas van a ser json, vamos a cargar al rededor de 4 jsons  
 •capas  
 •álbumes  
 •usuarios



Representación lógica de las capas que conforman la imagen número 7.

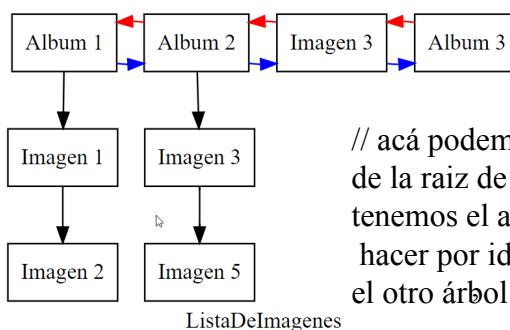
### 3.4. Álbumes // como crear una carpeta

Un álbum es una carpeta la cual almacenará distintas imágenes, cada cliente posee una lista de álbumes la cual queda a discreción del estudiante, además, cada álbum posee una lista simplemente enlazada de imágenes la cual hace referencia a la imagen guardada en el árbol AVL mencionado anteriormente.



Listado de álbumes por cliente

//agregar validación que si hay nulos no los recorra



// podemos tener álbumes vacíos y imgs iguales en distintos álbumes va a parecer que copiamos la pero en realidad solo estamos haciendo referencia a el mismo árbol

// acá podemos guardar las referencias a memoria de la raíz de los árboles, como atributo, así ya tenemos el acceso al árbol, también lo podemos hacer por id, pero acá tendríamos que recorrer el otro árbol

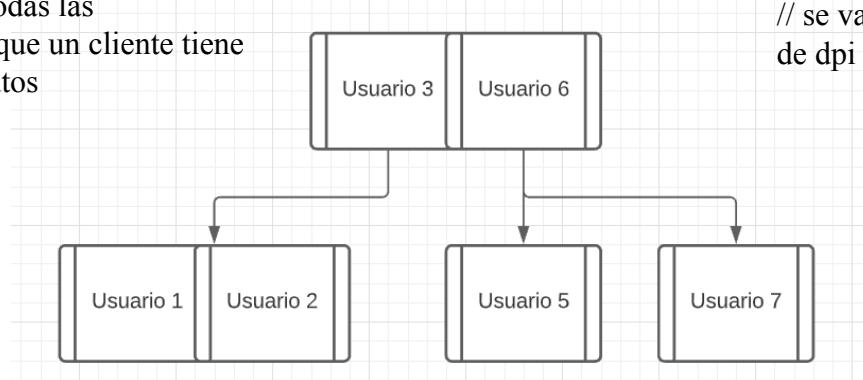
Listado imágenes álbumes donde se pueden identificar sus imágenes

### 3.5. Clientes

Dentro de la aplicación se manejan clientes, los clientes son capaces de agregar capas, imágenes y álbumes al estar logueados, toda su información se almacenará en un árbol B de orden 5 según su número de DPI.

Los clientes deben ser capaces de cerrar la sesión y toda la información que hayan cargado debe persistir siempre y cuando la aplicación se esté ejecutando, es decir, cuando el usuario vuelva a loguearse podrá visualizar el estado de todas sus estructuras.

//acá irian todas las estructuras que un cliente tiene como atributos



// se van a ordenar por el número de dpi

### 3.6. Carga Masiva

El orden de la carga de archivos será:

- Clientes (Usuario Administrador)
- Capas (cuando el cliente está logueado)
- Imágenes (cuando el cliente está logueado)
- Álbumes (cuando el cliente está logueado)

**Cuentas:** este archivo permitirá cargar los clientes en el sistema es decir en el árbol B, por lo cual deberá existir módulo administrador al cual se accede por medio del usuario “admin” y password “EDD2022”, la interfaz queda a discreción del estudiante.

```
[  
  {  
    "dpi": "1234567890123",  
    "nombre_cliente": "AUX EDD",  
    "password": "edd1s2022"  
  },  
  {  
    "dpi": "6745567890123",  
    "nombre_cliente": "cliente 2",  
    "password": "edd1s2022"  
  }  
]
```

// el usuario administrador va a estar definido desde el código, este usuario no debe estar en el árbol B

//el aux nos recomendó hacer 2 una interfaz distinta para el usuario admin y para los usuarios clientes

// porque el admin puede generar ciertos reportes que el cliente no puede

// Los usuarios se pueden o registrar ellos o por la carga masiva

// ya adentro podemos cargar las capas

**Capas:** este archivo permitirá cargar las capas que uno quiere registrar en el sistema, y que posteriormente utilizará para crear imágenes, es decir mediante una carga masiva de imágenes, mediante recorridos, etc. (estas capas no se comparten entre usuarios)

```
[  
  {  
    "id_capa": 0,  
    "pixeles": [  
      {  
        "fila": 25,  
        "columna": 31,  
        "color": "#FFCC33"  
      },  
      {  
        "fila": 38,  
        "columna": 31,  
        "color": "#FFCC40"  
      }  
    ],  
    {  
      "id_capa": 1,  
      "pixeles": [  
        {  
          "fila": 21,  
          "columna": 37,  
          "color": "#FFCC27"  
        }  
      ]  
    }  
  ]
```

// listado de capas  
con una matriz de pixeles

**Imágenes:** este archivo permitirá cargar las imágenes que un usuario quiere registrar en el sistema. Posee el listado de capas por el cual están creadas las imágenes.

```
[  
  {  
    "id":3,  
    "capas":[0,1,2,3,4]  
  },  
  {  
    "id":1,  
    "capas":[3,4]  
  },  
  {  
    "id":2,  
    "capas":[0,2,4]  
  },  
]
```

// objeto img: id y lista de capas  
// siempre en base a las capas  
que ya creamos

**Álbumes:** este archivo permitirá cargar los álbumes que un usuario quiere registrar en el sistema. Posee el listado de imágenes que se encuentran registradas en el álbum.

```
[  
  {  
    "nombre_album":"Album 1",  
    "imgs":[1,3,5]  
  },  
  {  
    "nombre_album":"Album 2",  
    "imgs":[2]  
  },  
  {  
    "nombre_album":"Album 3",  
    "imgs":[]  
  }  
]
```

// objeto, id e imagenes(sus ids)  
// si no existe la capa no debería insertarse en la lista, si no existe  
no se inserta en el arbolS

## 4. Funcionamiento

### 4.1. Generación de imágenes

La salida de la imagen deberá visualizarse a nivel de aplicación como se muestra en el apartado 3.1 // ya como una img compuesta por pixeles, sin las uniones de los nodos en otro apartado si nos piden la estructura como tal, podemos hacer los cambios

#### 4.1.1. Por recorrido limitado

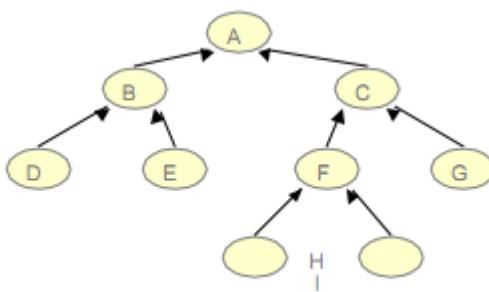
solo para generar cada img

En esta opción se indicará el número de capas a utilizar y el tipo de recorrido que se desea para generar la imagen: Preorden, inorder, postorden, dicho recorrido se realizará sobre el árbol de capas.

Ejemplo:

- *Preorder:* A B D E C F H I G
- *Inorder:* D B E A H F I C G
- *Postorder:* D E B H I F G C A

// de todo mi arbol  
yo solo quiero que me grafique 2  
capas y en cierto orden



// podríamos utilizar una matriz auxiliar para graficar, y usar el método insertar, ya en el ejemplo que nos dio el aux si viene uno ya registrado lo sustituye

Cada capa posterior en el recorrido superpone a la anterior, es decir apilando una sobre otra. El recorrido debe mostrarse en la pantalla de visualización de la gráfica (mediante un label, textbox...etc).

// aparte es el recorrido por grafica y el label de que escrito diga el recorrido de las capas

#### 4.1.2. Por Árbol de Imágenes

Se ingresará el id de la imagen el cual deberá buscarse y según el árbol de capas que posea se deberá generar la imagen **utilizando un recorrido en amplitud.**

// la idea es la misma, apilar las capas y dar un resultado al final

#### 4.1.3 Por capa

Se ingresará el id de una o varias capas a graficar y se buscarán en el árbol binario correspondiente para luego generar su imagen apilando dichas capas.

// acá es dónde nos dijo el aux que odiamos graficar solo el contorno negro del mario o el mario en blanco y negro y rojo

## 4.2. ABC

### 4.2.1. Clientes // esto lo va a hacer un admin

La aplicación debe ser capaz de registrar nuevos clientes, eliminar clientes y modificar clientes.

### 4.2.2. Imágenes

#### Registrar imagen

Al momento de registrarla se debe tomar en cuenta que el id es único y por lo tanto de existir el id la imagen no será agregada.

// podemos eliminar en el arbol colocarle nulo y si no está solo se ignora en el resto de procesos que le llaman

#### Eliminar imágenes

Se debe seleccionar la imagen, al momento de eliminarla se debe actualizar el árbol AVL del usuario y si existe dentro de un álbum también se deberá actualizar.

## 4.3. Visualización del estado de las estructuras

Dentro de la interfaz gráfica del programa se debe incluir un módulo de visualización de estructuras, se debe mostrar al cliente el conjunto de estructuras disponibles, al seleccionarse una de ellas se generará un reporte gráfico (imagen de la estructura), la cual se mostrará en la pantalla.

las opciones serán las siguientes:

### 4.3.1. Ver árbol de imágenes

Deberá mostrar el árbol AVL de las imágenes.

// podemos hacer un cambio en graphviz en los adges de color negro a color del fondo

### 4.3.2. Ver árbol de capas

Deberá mostrar el árbol de capas.

### 4.3.3. Ver Listado de álbumes

Deberá mostrar la lista doblemente enlazada de álbumes y sus sublistas de imágenes.

// no es necesario que sea doblemente enlazada, es opcional

### 4.3.4. Ver capa

Se indicará el número de capa a mostrar y se graficara su matriz correspondiente como en el apartado 3.2

// Acá vamos a mostrar las estructuras como las vemos en este enunciado, sin color a linea negra aunque si les colocamos el color tampoco hay problema

// podemos hacer 2 metodos para imprimir, uno como lo hicimos en IPC2, con graphviz puro y otro con tabla html que explicó el aux, aún debemos investigar esto

#### 4.3.5. Ver imagen y árbol de capas

Se debe seleccionar una imagen, se mostrará el arbol de imágenes y el árbol de capas que lo conforman como se visualiza en el apartado 3.3

#### 4.3.6. Ver árbol de clientes

Mostrará el árbol B de clientes registrados en el sistema, esta gráfica solo puede realizar el usuario administrador.

### 5. UI de la aplicación

El diseño de la interfaz gráfica de la aplicación queda a discreción del estudiante con la única observación que debe de poder realizarse lo siguiente:

- Módulo administrador
  - Árbol B de usuarios (Gráfico)
  - Operaciones sobre los usuarios (insertar, modificar y eliminar)
  - Operaciones de carga masiva de usuarios.
- Módulo de usuario:
  - Visualizar reportes de las estructuras
  - Navegación y gestión de imágenes
  - Opciones de carga masiva, según lo especificado en los apartados anteriores excluyendo la carga de clientes.
- Inicio de Sesión
- Registro de usuarios

// las del apartado 4.3

// para actualizar los reportes podemos agregar un botón recarga es eso o usar hilos

### 6. Reportes

El módulo de reportes debe de permitir ver los estados de las estructuras en cualquier momento durante la ejecución de la aplicación. Los reportes tendrán validez siempre y cuando estos sean elaborados utilizando la herramienta de Graphviz. Los reportes deben de mostrarse y actualizarse dentro de la aplicación.

Los reportes deben de generarse en una tabla dentro de la aplicación y mostrar lo siguiente:

#### Reportes de usuario

- Top 5 de imágenes con más número de capas // if lista > 5 mostrar solo los 5 de arriba else mostrar toda la lista
- Todas las capas que son hojas
- Profundidad de árbol de capas
- Listar las capas en: preorden, inorden, postorden // listado en texto

#### Reportes de administrador

- Buscar un cliente y mostrar su información:
  - Nombre, DPI y Password
  - Cantidad de álbumes y sus imágenes.
  - Cantidad de imágenes totales (puede que no pertenezcan a un álbum)
  - Cantidad de capas totales
- Listar clientes utilizando un recorrido por niveles
  - Mostrar su nombre, dpi y cantidad de imágenes totales

## 7. Resumen de estructuras utilizadas

- **Árbol B:** Esta estructura se utilizará para guardar los clientes registrados en el sistema.
- **Matriz Dispersa:** Esta estructura se utilizará para guardar la información de una capa, es decir las posiciones y colores de los píxeles utilizados para generar imágenes.
- **Árbol binario de búsqueda (ABB):** Esta estructura se utilizará para guardar todas las capas con las cuales se pueden generar distintas imágenes.
- **Árbol AVL:** Esta estructura se utilizará para guardar cada una de las imágenes de un cliente
- **Lista circular doblemente enlazada:** Esta estructura se utilizará para guardar el listado de álbumes por cada cliente, a su vez cada álbum posee una sublista de imágenes el tipo de sublista queda a discreción del estudiante.

## 8. Restricciones

Las estructuras deben ser desarrolladas por los estudiantes sin el uso de ninguna biblioteca o estructura predefinida en el lenguaje a utilizar.

- Se calificará desde el ejecutable, el cual estará dentro de su repositorio y dentro del archivo en UEDI. // Acá ya va a ser el ejecutable, creo que es el punto jar, esto subimos a uedi
- Se validará que los reportes estén correctos ya que únicamente se calificarán las estructuras graficadas, sin importar que exista el código de la implementación.
- Se deberán realizar un mínimo de 3 commit coherentes por semana para validar la continuidad de su flujo de trabajo. Se penalizará con el 25% de la nota final si no se cumple este requisito.

## 9. Observaciones

- Lenguaje de programación a utilizar: JAVA //En el manual técnico debe decir los primeros 3 creo o solo el 1 y 3
- Sistema Operativo: Libre
- IDE: Libre.
- Las gráficas deben mostrarse dentro de la aplicación, no buscarse en carpetas ajena. La sección de la aplicación que no genere sus gráficas correspondientes no podrá ser calificada, se debe utilizar Graphviz.
- Durante la calificación se harán preguntas para validar que el estudiante realizó el proyecto, de no responder correctamente anulará la nota obtenida en la o las secciones en la que se aplique tal concepto.
- Cada estudiante deberá utilizar el repositorio de la fase anterior separando por carpetas cada fase del proyecto, verificar que su auxiliar esté agregado como colaborador para poder analizar su progreso.
- Deberá colocar el Link de su repositorio y un archivo comprimido con el ejecutable de su proyecto en el apartado de entrega en la plataforma UEDI de lo contrario no tendrá validez su entrega y tendrá nota de 0.
- Fecha y hora de entrega: 27 de marzo a las 23:59 horas.
- Las copias serán penalizadas con una nota de 0 y castigadas según lo indique el reglamento.

## Entregables

- Manual de Usuario
- Manual Técnico
- Link a repositorio
- Código fuente
- Ejecutable