Catedrático: Ing. Luis Espino

Auxiliar: Ricardo Antonio Cutz Hernández

Estructura de Datos
Music++

Objetivo General

- Aplicar los conocimientos del curso de Estructura de Datos en la creación de soluciones de software.

Objetivos Específicos

- Aplicar los conocimientos de estructura de datos lineales
- Aplicar los conocimientos de estructura de datos en el lenguaje de programación de c++
- Utilizar la herramienta graphviz para la generación de reportes gráficos.

Enunciado

Descripción General

Una empresa de desarrollo de software está pensando en el desarrollo de una aplicación que permita el manejo de música del usuario. Se requiere que la aplicación permita manejar una biblioteca de música de forma dinámica. Como estudiante de ingeniería en sistemas se le plantea los requerimientos para la aplicación para su desarrollo utilizando el lenguaje de programación C++. La empresa tiene pensado presentar esta solución para que los usuarios tengan una experiencia diferente en el manejo de su música, siendo una aplicación intuitiva y fácil de utilizar con un diseño atractivo para el usuario.

Descripción de la aplicación

Importing Music Library

La aplicación debe soportar la importación de la biblioteca de música a partir de un archivo de carga que tendrá la descripción de todos los artistas, álbumes y canciones que el usuario desee ingresar. La carga de la biblioteca permitirá al usuario poder utilizar de forma rápida la aplicación llenando la misma con la información necesaria para el manejo de la biblioteca de música. **Es indispensable poder cargar la biblioteca de música mediante el archivo ya que no habrá ingreso manual de los datos. Los archivos a cargar son de tipo JSON.**

Archivo Library.json

Este archivo permite cargar una librería completa de música de forma rápida. El archivo contiene todos los datos relacionados con álbumes, artistas y canciones que se podrán reproducir en la aplicación. El archivo tiene los siguientes atributos:

- Library: Un atributo que contiene un arreglo de artistas
- Artist: Un atributo que contiene la información del artista (Name es un atributo de tipo cadena que tiene el nombre del artista, Albums es un arreglo de discos del artista)
- Albums: Es parte de un atributo de cada artista que contiene un arreglo de los discos que posee el artista. Maneja: (Name que es un atributo de tipo cadena, Month es un atributo de tipo cadena que tiene el mes del disco, Year un atributo de tipo cadena que tiene el año en que el disco fue publicado, Songs que es un arreglo de las canciones que pertenecen al disco)
- Songs: Es un atributo del Album que es un arreglo de canciones que conforman al disco, tiene atributos: Name que es una cadena que representa el nombre de la canción, File que es un atributo que representa el nombre del archivo de la canción, Rating que es una valoración de la canción, este último atributo es importante para calcular la valoración del disco y artista al que pertenece la canción.

Sin la posible carga de este archivo no se calificará el proyecto.

Ejemplo de Archivo:

Descargar Labrary.json de Ejemplo

```
{
  "Library":[
    {
      "Artist" :{
        "Name": "Underoath",
        "Albums": [
          {
            "Name": "They're Only Chasing Safety",
            "Month" : "Julio",
            "Year" : "2004",
            "Songs": [
              {
                "Name": "Young and Aspiring",
                "File": "YoungAndAspiring.mp3",
                "Rating": "8.0"
              }
              ,
              {
                "Name": "Some Will Seek Forgiveness, Others Escape",
                "File": "SomeWillSeekForfiveness.mp3",
                "Rating": "8.0"
              }
            ]
          }
          ,
          {
            "Name": "Erase Me",
            "Month": "Abril",
            "Year": "2018",
            "Songs" : [
              {
                "Name": "On My Teeth",
                "File": "OnMyTeeth.mp3",
                "Rating": "9.0"
              }
              ,
              {
                "Name": "Rapture",
                "File": "Rapture.mp3",
                "Rating": "9.0"
              }
            ]
          }
        ]
      }
    }
    ,
    {
      "Artist" :{
        "Name": "As I Lay Dying",
        "Albums": [
          {
            "Name": "Shaped By Fire",
            "Month": "Septiembre",
            "Year": "2019",
            "Songs": [
              {
                "Name": "Blinded",
                "File": "b.mp3",
                "Rating": "8.0"
```

Figura 1. Ejemplo de archivo Library.json

Archivo Playlist_#Nombre.json

Ya que se ha cargado la biblioteca de música mediante el archivo Library.json, se pueden crear las playlist mediante archivos (Sino se ha cargado una biblioteca entonces no se podrá crear playlists), el formato de nombre de estos archivos es el siguiente: Playlist_#NombrePlaylist.json donde #NombrePlaylist es el nombre que se le dará a la lista de reproducción. Ejemplo: Playlist_Rock.json, Playlist_HeavyMetal.json. Este archivo contendrá los siguientes atributos que permiten saber el tipo de playlist que se importará:

- Type: Este atributo permite saber el tipo de playlist que se va a crear. Los tipos permitidos son los siguientes:
    - Stack: Crea una playlist utilizando una estructura de tipo Pila.
    - Queue: Crea una playlist utilizando una estructura de tipo Cola.
    - Shuffle: Crea una playlist utilizando una lista doblemente enlazada en donde las canciones son insertadas de forma aleatoria (queda a discreción del estudiante la forma de inserción)
    - Circular: Crea una playlist utilizando una lista doblemente enlazada circular que inserta las canciones en el orden que se van leyendo del archivo.
- Songs: Es una arreglo de canciones que hacen referencia a las existentes en la biblioteca previamente importada. los atributos que tiene cada una de las canciones de este arreglo son:
    - Year: Año en que fue lanzado el disco donde pertenece la canción
    - Month: Mes en el que fue lanzado el disco donde pertenece la canción
    - Album: Nombre del disco al que pertenece la canción
    - Song: Nombre de la canción que será agregada a la playlist.
    - Artist: Nombre del artista que es autor de la canción.

El usuario puede importar cuantas playlist desea por medio de este tipo de archivos. **Es indispensable contar con la importación de playlists para poder calificarse el proyecto ya que no existe otra forma de crear playlists.**

Ejemplo de Archivo:

Descargar Archivo Playlist_Rock.json

```
{
  "Type": "Stack",
  "Songs": [
    {
      "Year": "2004",
      "Month": "Julio",
      "Album": "They're Only Chasing Safety",
      "Song": "Young and Aspiring",
      "Artist": "Underoath"
    }
    ,
    {
      "Year": "2018",
      "Month": "Abril",
      "Album": "Erase Me",
      "Song": "On My Teeth",
      "Artist": "Underoath"
    }
    ,
    {
      "Year": "2019",
      "Month": "Septiembre",
      "Album": "Shaped By Fire",
      "Song": "Blinded",
      "Artist": "As I Lay Dying"
    }
  ]
}
```

Figura 2. Archivo ejemplo Playlist

Estructuras para el almacenamiento de la Información

La información que maneja la aplicación será almacenada en memoria utilizando las siguientes estructuras de datos:

- Artistas: los artistas serán almacenados utilizando una **lista doblemente enlazada**, esta lista debe de ordenarse alfabéticamente.
- Albums: cada artista va almacenar su discografía utilizando un **cubo disperso**. Cada disco tiene un año y mes de lanzamiento, estos datos se utilizarán como cabecera para poder ordenar el cubo, en donde las cabeceras en el eje “x” serán los años y las cabeceras del

eje “y” serán los meses, el nodo en la intersección de año y mes almacena el disco que fue lanzado. En caso de haber más discos lanzados en un año y mes ya existente este se coloca en el eje “z” para formar el cubo.

- Canciones: Cada uno de los discos contienen canciones, estas canciones serán almacenadas utilizando una **lista simplemente enlazada**.
- Playlists: La aplicación soporta la importación de playlists por lo que se debe de tener una estructura que almacene todas las playlists importadas. Para almacenar las playlists se utilizará un **árbol binario de búsqueda**, tomando como criterio de inserción el nombre de la playlist.

A continuación se presenta un gráfico para entender de mejor forma la estructura de la información en memoria.

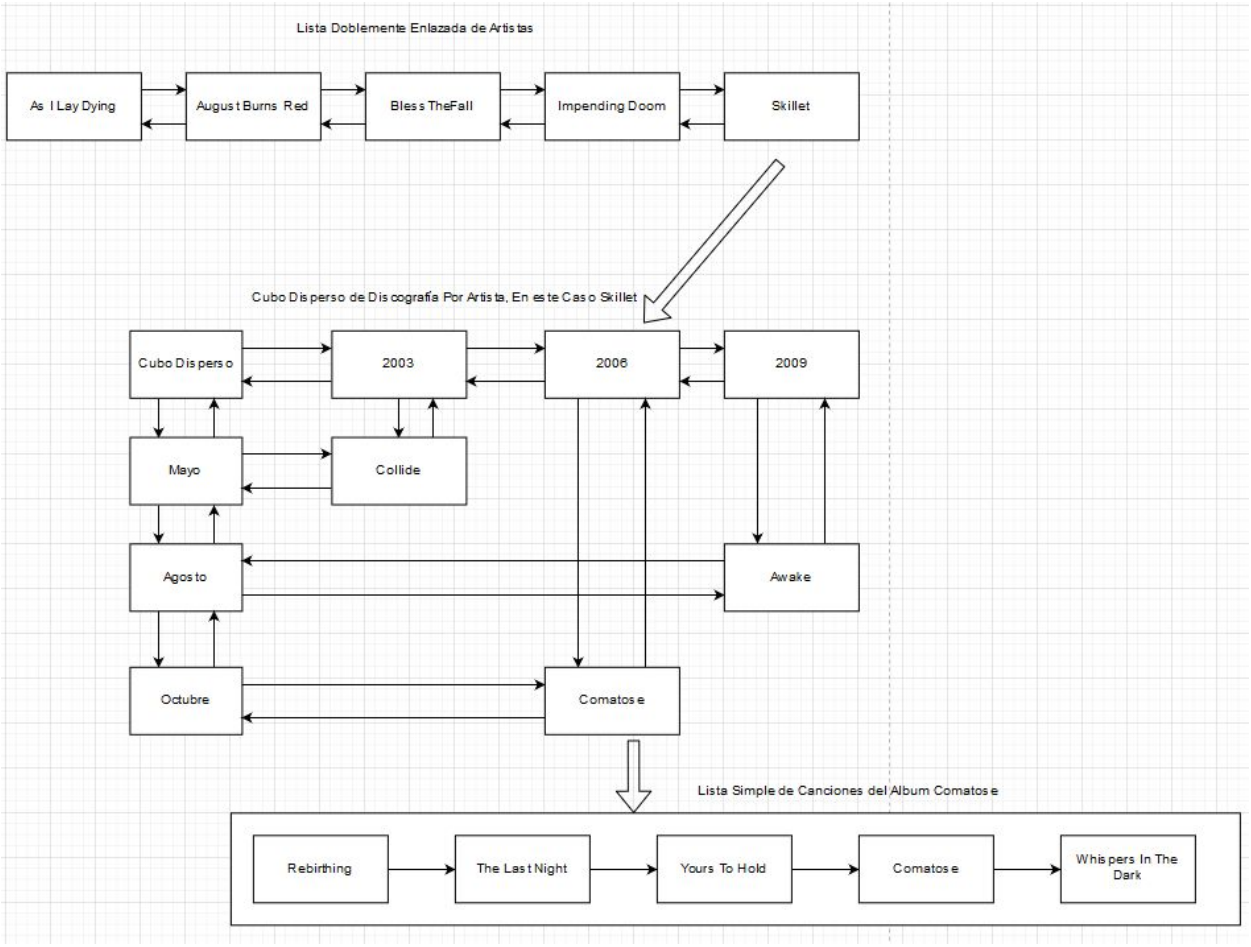


Figura 3. Ejemplo de Estructuras de Datos en Memoria para Almacenar Biblioteca

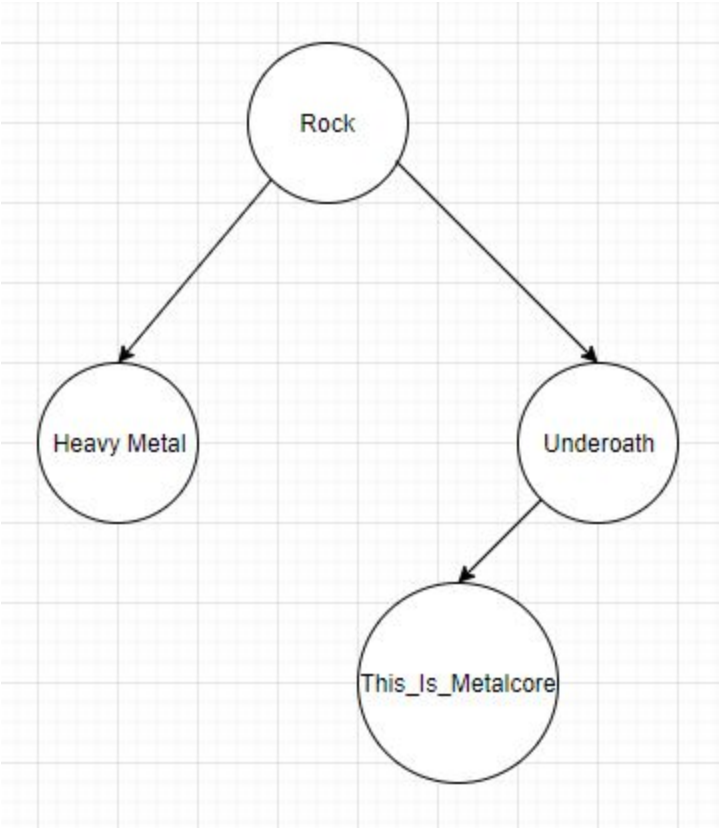


Figura 4. ABB que almacena las playlists

Navegación de Biblioteca

La biblioteca de música debe de permitir la navegación de la misma de las siguientes maneras:

- By Artist: Se debe de poder navegar sobre los diferentes artistas que están cargados en la aplicación. Los artistas deben de estar ordenados de forma alfabéticamente. Al momento de seleccionar uno de los artistas, la aplicación debe de desplegar la discografía del artista, es decir los discos ordenados por año de lanzamiento. Al seleccionar un disco la aplicación debe de mostrar las canciones del disco, en este punto el usuario puede seleccionar cualquier canción y reproducirla.
- By Songs: Se debe de poder navegar por todas las canciones que existen en la biblioteca. Las canciones deben de estar ordenadas de acuerdo al nombre, el usuario puede seleccionar la canción que desee y reproducirla.

Para la reproducción de la canción en las navegaciones, solamente se debe mostrar en pantalla los datos de la canción que se está reproduciendo, es decir que se hace simulado y no se reproducirá audio real.

Navegación de Playlists

La aplicación debe de mostrar todas las playlists que fueron cargadas, ordenadas alfabéticamente (recorrido inorden del árbol binario) de acuerdo al nombre de la playlist. El usuario al seleccionar la playlist, la aplicación debe de mostrar las canciones que contiene la playlist, debe mostrar una opción para iniciar o detener la reproducción de la playlist, la reproducción debe mostrarse mediante el reporte de la estructura que utiliza la playlist (Reporte en graphviz de el estado de la playlist, este reporte debe de actualizarse de acuerdo al estado de la estructura, debe de pintar el nodo que representa la canción). La forma de reproducción de la playlist depende de el atributo “Type” que se definió al momento de cargarla por medio del archivo. A continuación se explica la forma de reproducción de acuerdo al atributo “Type”:

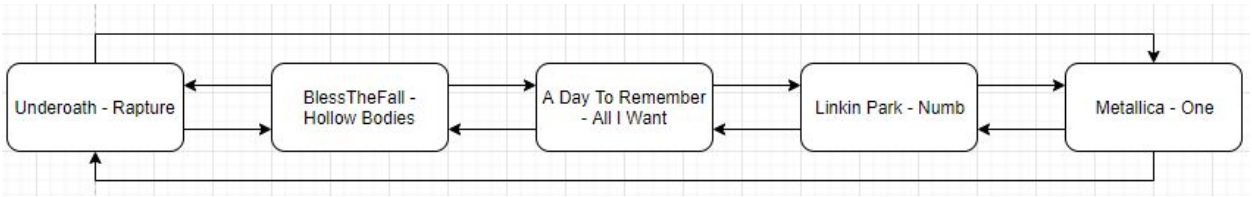
- Stack: Este tipo de playlist debe de reproducirse utilizando **una pila**, es decir que se utiliza el método “Pop” de la pila para ir sacando las canciones de la estructura y reproducirlas, utilizar LIFO. La estructura debe de quedar vacía al final de la reproducción completa de la playlist. La playlist solamente tiene un sentido de reproducción y no puede regresar a una canción anterior. La reproducción se hace automáticamente, dando un intervalo de 3 segundos por canción (Reporte Graphviz).

- Queue: Este tipo de playlist debe de reproducirse utilizando **una cola**, es decir que se utiliza el método “Dequeue” de la cola para ir sacando las canciones de la estructura y reproducirlas, utilizar FIFO. La estructura debe de quedar vacía al final de la reproducción completa de la playlist. La playlist solamente tiene un sentido de reproducción y no puede regresar a una canción anterior. La reproducción se hace automáticamente, dando un intervalo de 3 segundos por canción (Reporte Graphviz).
- Shuffle: Este tipo de playlist debe de reproducirse de acuerdo al orden en que fueron insertadas en la **lista doblemente enlazada** (Aleatoria). Para la reproducción de este tipo de lista debe de hacerse recorriendo la estructura de lista, es decir que las canciones no se eliminan de la estructura. La playlist tiene dos sentidos de reproducción, es decir que puede continuar con la siguiente canción y regresar a la anterior. La reproducción se hace mediante las opciones de siguiente y anterior (Reporte Graphviz).
- Circular: Este tipo de playlist se debe de reproducir de acuerdo al orden en que fueron insertadas a la **lista doblemente enlazada circular** (De acuerdo al archivo). Para la reproducción de este tipo de lista debe de hacerse recorriendo la estructura, es decir que las canciones no se eliminan de la estructura. La playlist tiene dos sentidos de reproducción, es decir que puede continuar con la siguiente canción y regresar a la anterior, al llegar al final de la playlist y darle “siguiente” esta debe de volver al inicio por ser una estructura circular, de igual forma al estar al inicio y darle “anterior” esta debe de ir al final de la playlist ya que es una estructura circular. La reproducción se hace mediante las opciones de siguiente y anterior (Reporte Graphviz).

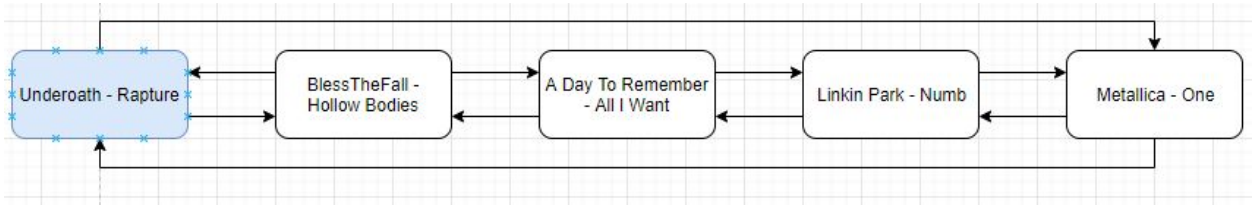
Para la reproducción de las playlist se hace uso graphviz, actualizando la imagen de la estructura en cada canción que se reproduce. Para realizar esto, en debe de pintar el nodo que representa la canción que se está reproduciendo en ese instante ya que será la forma que se hará la comprobación de los recorridos de las estructuras.

Ejemplo:

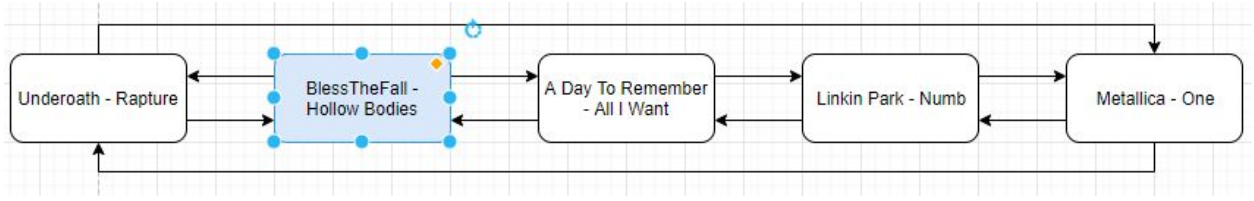
Se tiene la siguiente playlist importada de tipo Circular:



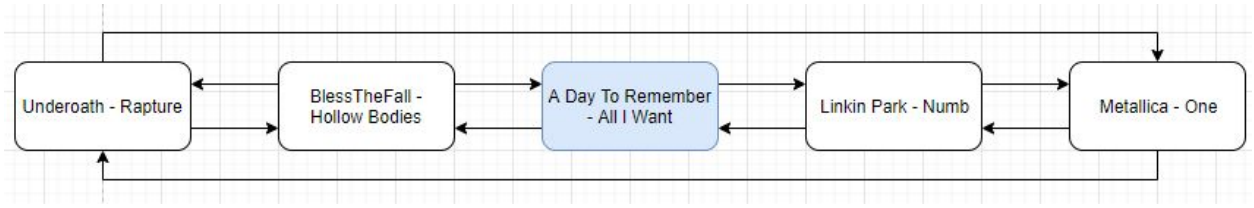
Se inicia la reproducción entonces el reporte debe de actualizarse indicando la canción que se está reproduciendo, si iniciamos de la primera:



Si se continúa con la siguiente canción:



La siguiente canción:



Así debe representarse la reproducción de una playlist, siempre depende del tipo de playlist que se va a reproducir, así será el recorrido.

Sin el reporte de la estructura que representa la playlist en reproducción LA FUNCIONALIDAD QUEDA ANULADA DE LA CALIFICACIÓN. Este reporte debe de realizarse con graphviz.

My Music++

Esta sección de la aplicación es un área que permite mostrar los reportes de la información almacenada en la biblioteca. **ES INDISPENSABLE CONTAR CON ESTA SECCIÓN DE**

REPORTES PARA CALIFICAR EL PROYECTO. TODOS LOS REPORTES SE REALIZAN CON GRAPHVIZ.

- **Artists Report:** Este reporte permite mostrar la lista doblemente enlazada de los artistas que tiene la biblioteca.
- **Discography Report:** Este reporte muestra el cubo disperso que representa la discografía de un artista. Para generar el reporte se le debe de indicar un artista como parámetro y la aplicación debe generar el reporte del cubo de acuerdo al artista que se indicó.
- **Album Report:** Este reporte muestra la lista de canciones de un disco. Para generar el reporte se le debe de indicar el artista, año, mes y disco como parámetros de búsqueda para que luego se genere el reporte de la lista de canciones.
- **Playlists Report:** Este reporte debe de mostrar el árbol binario que almacena las playlists importadas en la aplicación.
- **Top 5 Albums By Artist Report:** Este reporte debe de generar una lista que muestre los top 5 discos mejor valorados (del más valorado al menos valorado) de un artista (Se indica como parámetro para el reporte). La forma de calcular la valoración de un disco es realizando un promedio de el atributo “Rating” que tienen las canciones de ese disco. Al reporte se debe de indicar el artista como parámetro.
- **Top 5 Artists:** Este reporte debe de generar una lista que muestre los 5 artistas mejor valorados (del más valorado al menos valorado). La forma de calcular la valoración de un artista es realizando un promedio de la valoración de los discos de ese artista, para realizar la valoración de los discos se debe de realizar un promedio del atributo “Rating” de las canciones de cada disco.

IMPORTANTE: para tener derecho a la calificación de las funcionalidades de la aplicación es indispensable contar con los reportes de las estructuras, de no tener uno de los reportes se anula la funcionalidad asociada a la estructura que representa el reporte faltante.

Restricciones

Las estructuras deben de ser desarrolladas por los estudiantes sin el uso de ninguna librería o estructura predefinida en el lenguaje a utilizar. **Los reportes son esenciales para verificar si se trabajaron correctamente las estructuras solicitadas, Si no se tiene el reporte de alguna estructura se anularán los puntos que tengan relación tanto al reporte como a la estructura en cuestión.**

Penalizaciones

1. Falta de seguimiento de desarrollo continuo por medio Github tendrá una penalización del 10% (mínimo 2 commits por semana).

2. Falta de seguimiento de instrucciones conforme al método de entrega (nombre del repositorio) tendrá una penalización del 5%.
3. Falta de puntualidad conforme a la entrega tendrá una penalización de la siguiente manera:
    a. 0-6 horas – 20%.
    b. 6-12 horas – 40%
    c. 12-18 horas 60%
    d. 18-24 horas – 80%.
    e. Pasados 24 horas tendrá una nota de 0 y no se calificara.

NOTA: Las estructuras utilizadas deben ser implementadas por el estudiante y no se permite el uso librerías para realizar las estructuras, en caso de que fuese así el proyecto tendrá nota de 0.

Observaciones

- Lenguaje a utilizar: C++
- IDE: Libre
- Sistema Operativo: Libre
- Librería para lectura de json: Libre
- Herramienta para desarrollo de reportes gráficos: Graphviz.
- La aplicación debe de ser capaz de generar y abrir con un visor de imágenes predeterminado las imágenes generadas con Graphviz.
- La entrega se realizará por medio de: Github, cada estudiante deberá crear un repositorio con el nombre: EDD_DIC_2019_PY1_#carnet, y agregar a su auxiliar correspondiente como colaborador del mismo, para poder analizar su progreso y finalmente a partir del mismo repositorio realizar la calificación correspondiente. Usuario de Github: ricardcutzh
- Además de tener a su auxiliar como colaborador del repositorio para tener un control y orden de las personas que entreguen deberán de colocar el Link de su repositorio en la Tarea que cada auxiliar asignará en su classroom correspondiente.
- Fecha y hora de entrega: 18 de Diciembre, a las 23:59 horas.
- Copias serán penalizadas con una nota de 0 y castigadas según lo indique el reglamento.