Documentació Projecte E.D Fase II	
Autors	Laia Alcalde - 1642329 Elena Gutiérrez - 1703828 Laia Cámara - 1710505
Qualificació "grade"	Hem anat polint el codi amb el test de proves proporcionat a caronte. 203
Canvis introduïts respecte a la primera part	+ Classe VideoData:  - Ara utilitza la classe
	- <b>self.current_files</b> conté els

apths dels fitxers MP4
Classe **VideoID**:

- self.file\_uuid\_map és un diccionari que mapeja el camí absolut d'un fitxer MP4 al seu UUID corresponent.
- self.used\_uuids és un conjunt que emmagatzema tots els UUIDs que ja han estat generats i assignats a fitxers.
- + Classe VideoPlayer:
  - self.video\_data conté informació dels vídeos, com els seus paths, metadades (títol, artista, duració, etc.) i UUIDs.
- + Classe PlayList:
  - Ara no permet que un vídeo aparegui més d'una vegada dins d'una llista, evitant així bucles dins el graf.
  - Ara pot llegir llistes de reproducció no només des d'un arxiu M3U sinó també des de llistes Python de UUIDs mitjançant el mètode read list.
- + Classe **SearchMetadata**:
  - S'ha implementat
    get\_similar, que calcula
    la semblança entre vídeos
    basant-se en un algorisme
    específic que té en compte
    distàncies dins el graf i el
    ranking dels vídeos.
  - La funció get\_auto\_play genera llistes de reproducció automatitzades basant-se en les connexions entre vídeos i els seus rankings.

Anàlisis del rendiment

Durant la implementació i prova de la generació automàtica de reproduccions de vídeos, s'han identificat els següents problemes de rendiment:

## Càlcul de semblances entre vídeos:

 L'algorisme per calcular la semblança entre vídeos requereix múltiples recorreguts del graf per obtenir distàncies i rankings. Això

pot ser costós computacionalment en col·leccions grans, especialment si el graf conté molts nodes i arestes.

## **Iteracions repetides**:

 A l'hora de calcular recomanacions per a múltiples vídeos, es produeixen càlculs redundants de distàncies i rankings.

A més del temps d'escriptura i lectura dels fitxers.

## **DIAGRAMA PROJECTE** VideoPlayer VideoData Laia Camara | Laia Alcalde | Elena Gutierrez + video data: viedo data metadata: dict graph: GrafHash + play\_video(self, uuid: str, mode: int) + get\_file\_path(self, uuid: str): str + get\_video\_rank(self, uuid: str): int + print\_video(self, uuid: str): None + get next videos(self, uuid: str): iter + play\_file(file: str): + get previous videos(self, uuid: str): iter + get video distance(self, uuid: str, uuid: str): int, int + play\_all\_videos(self): + iter (self): Iteration + existeix file(self, self, filename): bool + add video(self, uuid, filename) VideoFiles + remove video(self, uuid) + load metadata(self, uuid) + Ilista: list + len (self): int + files added: set + existeix uuid(self,uuid): str + files removed: set + existeix meta(self,uuid): int **PlayList** + get filename(self, uuid): str + get path(self, uuid): str + reload fs(self, root: str): + video id: dict + get duration(self, uuid): int + files added(self): list + \_videoplayer: dict + get title(self, uuid): str + files removed(self): list + playlist: list + get album(self, uuid): str + len (self): int + str (self): str + get\_artist(self, uuid): str + get\_composer(self, uuid): str + iter (self): Iterator + load file(self, file; str); list + get\_genre(self, uuid): str + repr (self): str + read list(self, p llista: list + get\_date(self, uuid): str + play(self, mode: int) + get comment(self, uuid): str + add video at end(self, uuid: str) + metadata(self): dict + remove first video(self) + read playlist(self, obj playlist: Playlist) + remove last video(self) **ElementData** + str (self): str + len (self): int + \_\_repr\_\_(self): str + \_\_str\_\_(self): str + eq (self, other: elementData): bool + title: str + \_\_iter\_\_(self): Iterator + ne (self, other: elementData): bool + artist: str + \_\_repr\_\_(self): str GrafHash + It (self, other: elementData): bool + album: str + hash (self): Hash + composer: str + iter (self): Iteration + nodes: dict + genre: str + out: dict + date: str + in: dict + comment: str + duration: str + filename: str VideoID + \_\_len\_\_(self): int SearchMetadata + \_\_str\_\_(self): str + file uuid map: dict + \_\_iter\_\_(self): Iterator + str (self): str + video data: viedo data + \_\_repr\_\_(self): str + \_\_repr\_\_(self): str + \_\_ne\_\_(self, other: elementData): bool + eq (self, other: elementData): bool + generate uuid(self, file: str): str + It (self, other: elementData): bool + ne (self, other: elementData): bool + filter by attribute(selt, attribute getter: dict, sub: str): list + get\_uuid(self, file: str): str + It (self, other: elementData): bool + es\_digraf(self): bool + duration(self, min duration, max duration): int + remove uuid(self, uuid: str) + getOut(self): dict + hash (self): Hash + title(self, sub): str + len (self): int + title(self): str + album(self, sub): str + insert\_vertex(self, key: str, e: ElementData): Vertex + str (self): str + title(self, title: str) + insert\_edge(self, key1: str, key2: str, weight: int) + artist(self, sub): str + iter (self): Iterator + get(self, key: str): ElementData + artist(self): str + def composer(self, sub): str + repr (self): str + genre(self, sub): str + \_\_containts\_\_(self, key: str): bool + artist(self, artist: str) + \_\_getitem\_\_(self, key: str): str + album(self): str + date(self, sub): str + delitem (self, key: str) + album(self, album: str) + comment(self, sub): str + existeix edge(self, n1: Node, n2: Node): bool + composer(self): str + and operator(self, list1, list2): list + edges out(self, x: str): iter + composer(self, composer: str) + or operator(self, list1, list2): list + edges in(self, x: str): iter + genre(self): str + len (self): int + genre(self, genre; str) + grauPesIn(self, x; str); iter + str (self): str Vertex + grauPesOut(self, x: str): iter + date(self): str + iter (self): Iterator + edges(self, key: str): iter + date(self, date: str) + repr (self): str + value: int + existeix arestes(self, key1: str, key2: str): bool + comment(self): str + eq (self, other: elementData): bool + get weight(self, key1: str, key2: str): int + comment(self, comment: str) + ne (self, other: elementData): bool + upadte edge weight(self, key1: str, key2: str, weight) + duration(self): str + It (self, other: elementData): bool + str (self): str + minDistante(self, dist: dict, visitats: dict): Node + duration(self, duration: str) + repr (self): str + get similar(self, uuid: str, max list: int): list + \_\_eq\_\_(self, other: ElementData): bool + get\_auto\_plau(self, length: int): list + dijkstra(self, n: Node): dict, dict + filename(self): str + hash (self): Hash + dijstraModif(self, n1: Node, n2: Node): dict, dict + filename(self, filename: str) + get similarity score(self, uuid1: str, uuid2: str): float

+ get metadata(uuid: str, attribute: str)

+ search complex(self, criteria: dict): list

+ camiMesCurt(self, n1: Node, n2: Node): dict, dict

+ get element data(self):