

## Documentation for `routes/milvus.py`

This module contains the endpoints for the Milvus service. It provides routes for performing queries such as similarity searches on the Milvus vector database.

### `get_entity_by_id(id, user=Depends(login_manager))`

Retrieves the embedding vector of a specific entity by its ID.

- **id:** str - The unique identifier of the entity.
- **user:** User - The authenticated user making the request.
- **return:** EmbeddingResponse - The embedding vector of the entity.

#### Source code in `routes/milvus.py`

```

28 @router.get("/entity/{id}", response_model=EmbeddingResponse, tags=
29 ["milvus"])
30 def get_entity_by_id(id: str, user=Depends(login_manager)):
31     """
32     Retrieves the embedding vector of a specific entity by its ID.
33
34     - **id**: str - The unique identifier of the entity.
35     - **user**: User - The authenticated user making the request.
36     - **return**: EmbeddingResponse - The embedding vector of the entity.
37     """
38     collection_512 = get_milvus_512_collection()
39     entities = collection_512.query(expr=f"id in [{id}]", output_fields=
40 ["embedding"])
41     if not entities:
42         raise HTTPException(status_code=404, detail="Entity not found")
43
44     embedding = [float(x) for x in entities[0]["embedding"]]
45     return EmbeddingResponse(id=id, embedding=embedding)

```

### `get_genres_plot(query, user=Depends(login_manager))`

`async`

Generates a plot of the top 5 genres for a given entity based on its file path.

- **query:** SongPath - The query containing the file path of the entity.
- **user:** User - The authenticated user making the request.

- **return:** A base64 encoded string of the plot image.

Source code in `routes/milvus.py`

```

130 @router.post("/plot_genres", tags=["milvus"])
131 async def get_genres_plot(query: SongPath, user=Depends(login_manager)):
132     """
133     Generates a plot of the top 5 genres for a given entity based on its
134     file path.
135
136     - **query**: SongPath - The query containing the file path of the
137     entity.
138     - **user**: User - The authenticated user making the request.
139     - **return**: A base64 encoded string of the plot image.
140     """
141     collection_87 = get_milvus_87_collection()
142     entity = collection_87.query(
143         expr=f"path == '{query.file_path}'",
144         output_fields=["predictions", "title", "artist"],
145         limit=1
146     )
147     if not entity:
148         raise HTTPException(status_code=404, detail="Entity not found")
149
150     class_names, top_5_activations, title, artist = await
151     extract_plot_data(entity)
152     fig = await create_plot(class_names, top_5_activations, title, artist)
153     image_base64 = await convert_plot_to_base64(fig)
154
155     return Response(content=image_base64, media_type="text/plain")

```

`get_similar_9_entities_by_path(query,`  
`user=Depends(login_manager))`

Retrieves the 9 most similar entities (by title, artist, album) based on the file path of an entity.

- **query:** FilePathsQuery - The query containing the file path(s) of the entity.
- **user:** User - The authenticated user making the request.
- **return:** A list of the 9 most similar entities with short details.

Source code in `routes/milvus.py`

```

102 @router.post("/similar_short_entity", tags=["milvus"],
103 response_model=SimilarShortEntitiesResponse)
104 def get_similar_9_entities_by_path(query: FilePathsQuery,
105 user=Depends(login_manager)):
106     """
107     Retrieves the 9 most similar entities (by title, artist, album) based
108     on the file path of an entity.
109
110     - **query**: FilePathsQuery - The query containing the file path(s) of
111     the entity.
112     - **user**: User - The authenticated user making the request.
113     - **return**: A list of the 9 most similar entities with short
114     details.
115     """
116     collection_512 = get_milvus_512_collection()
117     entities = collection_512.query(expr=f"path in {query.path}",
118 output_fields=["embedding"])
119     if not entities:
120         raise HTTPException(status_code=404, detail="Entity not found")
121
122     embeddings = [[float(x) for x in entity["embedding"]] for entity in
123 entities]
124     entities = collection_512.search(
125         data=embeddings,
126         anns_field="embedding",
127         param={"nprobe": 16},
128         limit=30,
129         offset=1,
130         output_fields=["title", "album", "artist", "path"],
131     )
132
133     sorted_entities = sort_entities(entities)
134     return {"entities": sorted_entities}

```

`get_similar_entities(id, user=Depends(login_manager))`

Retrieves the top 3 most similar entities to a given entity ID.

- **id**: str - The unique identifier of the entity to compare.
- **user**: User - The authenticated user making the request.
- **return**: SimilarFullEntitiesResponse - A list of the most similar entities.

Source code in `routes/milvus.py`

```

46 @router.get("/similar/{id}", tags=["milvus"],
47 response_model=SimilarFullEntitiesResponse)
48 def get_similar_entities(id: str, user=Depends(login_manager)):
49     """
50     Retrieves the top 3 most similar entities to a given entity ID.
51
52     - **id**: str - The unique identifier of the entity to compare.
53     - **user**: User - The authenticated user making the request.
54     - **return**: SimilarFullEntitiesResponse - A list of the most similar
55     entities.
56     """
57     collection_512 = get_milvus_512_collection()
58     entities = collection_512.query(expr=f"id in [{id}]", output_fields=
59     ["embedding"])
60     if not entities:
61         raise HTTPException(status_code=404, detail="Entity not found")
62
63     embedding = [float(x) for x in entities[0]["embedding"]]
64     entities = collection_512.search(
65         data=[embedding],
66         anns_field="embedding",
67         param={"nprobe": 16},
68         limit=3,
69         offset=1,
70         output_fields=["*"],
71     )
72
73     response_list = [full_hit_to_dict(hit) for hit in entities[0]]
74     return SimilarFullEntitiesResponse(hits=response_list)

```

`get_similar_entities_by_path(query,`  
`user=Depends(login_manager))`

Retrieves the top 3 most similar entities based on the file path of an entity.

- **query:** FilePathsQuery - The query containing the file path(s) of the entity.
- **user:** User - The authenticated user making the request.
- **return:** SimilarFullEntitiesResponse - A list of the most similar entities with full details.

Source code in `routes/milvus.py`

```

74 @router.post("/similar_full_entity", tags=["milvus"],
75 response_model=SimilarFullEntitiesResponse)
76 def get_similar_entities_by_path(query: FilePathsQuery,
77 user=Depends(login_manager)):
78     """
79     Retrieves the top 3 most similar entities based on the file path of an
80     entity.
81
82     - **query**: FilePathsQuery - The query containing the file path(s) of
83     the entity.
84     - **user**: User - The authenticated user making the request.
85     - **return**: SimilarFullEntitiesResponse - A list of the most similar
86     entities with full details.
87     """
88     collection_512 = get_milvus_512_collection()
89     entities = collection_512.query(expr=f"path in {query.path}",
90 output_fields=["embedding"])
91     if not entities:
92         raise HTTPException(status_code=404, detail="Entity not found")
93
94     embeddings = [[float(x) for x in entity["embedding"]] for entity in
95 entities]
96     entities = collection_512.search(
97         data=embeddings,
98         anns_field="embedding",
99         param={"nprobe": 16},
100         limit=3,
101         offset=1,
102         output_fields=["*"],
103     )
104
105     response_list = [short_hit_to_dict(hit) for hit in entities[0]]
106     return SimilarFullEntitiesResponse(hits=response_list)

```

## ping\_milvus\_collection()

Checks the connectivity with the Milvus vector database. Mostly used to make prometheus ping milvus everyday, so milvus doesn't get idle for 7 days and shutdown.

- **return:** The status of the Milvus service.

**” Source code in** routes/milvus.py

```
155 @router.get("/ping", tags=["milvus"])
156 def ping_milvus_collection():
157     """
158     Checks the connectivity with the Milvus vector database. Mostly used
159     to make prometheus ping milvus everyday, so milvus doesn't get idle for 7
160     days and shutdown.
161
162     - **return**: The status of the Milvus service.
163     """
164     milvus_status = ping_milvus()
165     return milvus_status
```