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
        embedding = [float(x) for x in entities[0]["embedding"]]
        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`

```python
@router.post("/plot_genres", tags=["milvus"])
async def get_genres_plot(query: SongPath, user=Depends(login_manager)):
    """
    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.
    """
    collection_87 = get_milvus_87_collection()
    entity = collection_87.query(
        expr=f"path == '{query.file_path}'",
        output_fields=["predictions", "title", "artist"],
        limit=1
    )
    if not entity:
        raise HTTPException(status_code=404, detail="Entity not found")

    class_names, top_5_activations, title, artist = await
extract_plot_data(entity)
    fig = await create_plot(class_names, top_5_activations, title, artist)
    image_base64 = await convert_plot_to_base64(fig)

    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`

```python
@router.post("/similar_short_entity", tags=["milvus"],
response_model=SimilarShortEntitiesResponse)
def get_similar_9_entities_by_path(query: FilePathsQuery,
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.
    """
    collection_512 = get_milvus_512_collection()
    entities = collection_512.query(expr=f"path in {query.path}",
output_fields=["embedding"])
    if not entities:
        raise HTTPException(status_code=404, detail="Entity not found")

    embeddings = [[float(x) for x in entity["embedding"]] for entity in
entities]
    entities = collection_512.search(
        data=embeddings,
        anns_field="embedding",
        param={"nprobe": 16},
        limit=30,
        offset=1,
        output_fields=["title", "album", "artist", "path"],
    )

    sorted_entities = sort_entities(entities)
    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`

```python
@router.get("/similar/{id}", tags=["milvus"],
response_model=SimilarFullEntitiesResponse)
def get_similar_entities(id: str, 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.
    """
    collection_512 = get_milvus_512_collection()
    entities = collection_512.query(expr=f"id in [{id}]", output_fields=
["embedding"])
    if not entities:
        raise HTTPException(status_code=404, detail="Entity not found")

    embedding = [float(x) for x in entities[0]["embedding"]]
    entities = collection_512.search(
        data=[embedding],
        anns_field="embedding",
        param={"nprobe": 16},
        limit=3,
        offset=1,
        output_fields=["*"],
    )

    response_list = [full_hit_to_dict(hit) for hit in entities[0]]
    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`

```python
@router.post("/similar_full_entity", tags=["milvus"],
response_model=SimilarFullEntitiesResponse)
def get_similar_entities_by_path(query: FilePathsQuery,
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.
    """
    collection_512 = get_milvus_512_collection()
    entities = collection_512.query(expr=f"path in {query.path}",
output_fields=["embedding"])
    if not entities:
        raise HTTPException(status_code=404, detail="Entity not found")

    embeddings = [[float(x) for x in entity["embedding"]] for entity in
entities]
    entities = collection_512.search(
        data=embeddings,
        anns_field="embedding",
        param={"nprobe": 16},
        limit=3,
        offset=1,
        output_fields=["*"],
    )

    response_list = [short_hit_to_dict(hit) for hit in entities[0]]
    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`

```python
@router.get("/ping", tags=["milvus"])
def 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.
    """
    milvus_status = ping_milvus()
    return milvus_status
```