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.

Endpoints

Endpoint: `/milvus/entity/{id}`

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`:

```
@router.get("/entity/{id}", response_model=EmbeddingResponse, tags=["milvus"])
def get_entity_by_id(id: str, user=Depends(login_manager)):
    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"]]
    return EmbeddingResponse(id=id, embedding=embedding)
```

Endpoint: `/milvus/plot_genres`

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`:

```
@router.post("/plot_genres", tags=["milvus"])
async def get_genres_plot(query: SongPath, user=Depends(login_manager)):
    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")
```

Endpoint: `/milvus/similar_short_entity`

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` :

```
@router.post("/similar_short_entity", tags=["milvus"], response_model=SimilarShortEr
def get_similar_9_entities_by_path(query: FilePathsQuery, user=Depends(login_manager
    collection_512 = get_milvus_512_collection()
    entities = collection_512.query(expr=f"path in {query.path}", output_fields=["em
    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}
```

Endpoint: `/milvus/similar/{id}`

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` :

```
@router.get("/similar/{id}", tags=["milvus"], response_model=SimilarFullEntitiesResp
def get_similar_entities(id: str, user=Depends(login_manager)):
    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)
```

Endpoint: `/milvus/similar_full_entity`

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`:

```
@router.post("/similar_full_entity", tags=["milvus"], response_model=SimilarFullEnti
def get_similar_entities_by_path(query: FilePathsQuery, user=Depends(login_manager))
    collection_512 = get_milvus_512_collection()
    entities = collection_512.query(expr=f"path in {query.path}", output_fields=["em
    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)
```

Endpoint: `/milvus/ping`

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

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

Source code in `routes/milvus.py`:

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

For more details, visit the documentation.