ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN

BÁO CÁO ĐỒ ÁN
CƠ SỞ DỮ LIỆU CHO ỨNG DỤNG TÌM KIẾM BÀI NHẠC QUA GIAI ĐIỆU

Sinh viên thực hiện:
Triệu Đức Duy - 23520392
Đỗ Trần Thế Bảo - 23520096
Nguyễn Đình Đại - 23520216

Giảng viên:
TS Nguyễn Gia Tuấn Anh
Th.S Phạm Nguyễn Phúc Toàn

Thành phố Hồ Chí Minh, tháng 05 năm 2025

BÁO CÁO TÓM TẮT

1. Tiêu đề báo cáo: Cơ sở dữ liệu cho ứng dụng tìm kiếm bài nhạc qua giai điệu

2. Danh sách thành viên

MSSV	Họ tên	Ghi chú
23520392	Triệu Đức Duy	Trưởng nhóm
23520096	Đỗ Trần Thế Bảo	
23520216	Nguyễn Đình Đại	

3. Nội dung chi tiết

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN, SO SÁNH THỊ TRƯỜNG, ĐỐI TƯỢNG SỬ DỤNG, LỰA CHỌN CÔNG CỤ

1. Giới thiệu bài toán
2. So sánh thị trường
3. Đối tượng sử dụng
4. Lựa chọn công cụ

CHƯƠNG 2: PHÂN TÍCH BÀI TOÁN

1. Mô hình thực thể mối kết hợp
2. Sơ đồ ERD

CHƯƠNG 3: THIẾT KẾ CSDL

1. Phân tích yêu cầu tìm kiếm bài hát qua giai điệu:
2. Giải thích và lý do lưu trữ một số thuộc tính:
3. Bảng mô tả lược đồ quan hệ:
4. Sơ đồ quan hệ

CHƯƠNG 4: CÀI ĐẶT BẢNG, CỘT, KHÓA CHÍNH, KHÓA NGOẠI, CONSTRAINT

1. Code cài đặt
2. Giải thích một số chi tiết

CHƯƠNG 5: CÀI ĐẶT RÀNG BUỘC ĐỂ BẢO VỆ TÍNH TOÀN VẸN DỮ LIỆU

1. Các ràng buộc cần thiết cho CSDL
2. Cách để tạo các ràng buộc

CHƯƠNG 6: FUNCTION, STORED PROCED, VIEW

1. Function

2. Stored procedure
3. View

CHƯƠNG 7: PHÂN QUYỀN CƠ SỞ DỮ LIỆU

1. Phân tích
2. Code cài đặt

CHƯƠNG 8: BIỂU DIỄN THÔNG TIN

1. Form đăng nhập, đăng ký tài khoản
2. Menu các chức năng của ứng dụng
3. Report thống kê hoạt động của người dùng

CHƯƠNG 9: SAO LƯU, KHÔI PHỤC

1. Sao lưu
2. Khôi phục

CHƯƠNG 10: CÁC VẤN ĐỀ KHÁC

1. Đề xuất một cách thu thập dữ liệu cho cột đặc trưng giai điệu
2. Hướng phát triển

CHƯƠNG 11: KẾT LUẬN

1. Tự đánh giá ưu điểm
2. Tự đánh giá khuyết điểm

4. Phân công công việc

MSSV	**Họ tên**	**Nội dung được phân công**
23520392	Triệu Đức Duy	Chương 1, 2, 3, 5, 6, 8, 10, 11, Code web demo
23520096	Đỗ Trần Thế Bảo	Chương 2, 3, 4, 8
23520216	Nguyễn Đình Đại	Chương 2, 3, 7, 9

MỤC LỤC

DANH MỤC BẢNG (NẾU CÓ)

DANH MỤC HÌNH VẼ (NẾU CÓ)

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN, MỤC TIÊU, ĐỐI TƯỢNG SỬ DỤNG, LỰA CHỌN CÔNG CỤ

1. Giới thiệu bài toán:

- Xây dựng cơ sở dữ liệu cho ứng dụng tìm kiếm bài nhạc qua giai điệu và quản lý danh sách phát nhạc của người dùng
- Quy trình hoạt động của ứng dụng: Người dùng đăng ký tài khoản, đăng nhập vào ứng dụng, sau đó người dùng sẽ được thực hiện các chức năng như tra cứu bài nhạc, tạo danh sách phát…
- Các chức năng cần có:
    - Đăng ký tài khoản, đăng nhập vào ứng dụng
    - Tra cứu bài nhạc thông qua giai điệu (tiếng hát, tiếng ngân nga, âm thanh từ nhạc cụ, thiết bị,…)
    - Phát bài nhạc
    - Xem thông tin các bài nhạc mình tra cứu (tên bài, tác giả, ngày phát hành)
    - Xem thông tin tác giả (tên, tuổi, ảnh tác giả)
    - Tạo các danh sách phát nhạc
    - Thêm, xóa bài nhạc trong danh sách phát
    - Thay đổi thứ tự bài nhạc trong danh sách phát
    - Xem lại lịch sử nhạc đã nghe

2. So sánh thị trường:

- So sánh thị trường:
    - Ứng dụng tìm kiếm bài nhạc qua giai điệu: Shazam
    - Ứng dụng quản lý danh sách phát: Spotify
    - Đặc điểm:
        - Shazam: tìm kiếm rất tốt bài nhạc phát ra từ bản audio, không tìm kiếm được nếu người dùng ngân nga giai điệu bằng giọng hát

- Spotify: quản lý danh sách phát với nhiều tính năng, các tính năng nâng cao cần trả phí để được sử dụng, tìm kiếm bài nhạc theo tên nhưng không thể tìm theo giai điệu

-

3. **Đối tượng sử dụng:**

- Người muốn tìm kiếm bài nhạc nhưng không biết tên, lời bài nhạc
- Người muốn tìm kiếm bài nhạc không lời
- Người muốn tạo, tùy chỉnh các danh sách phát nhạc của mình

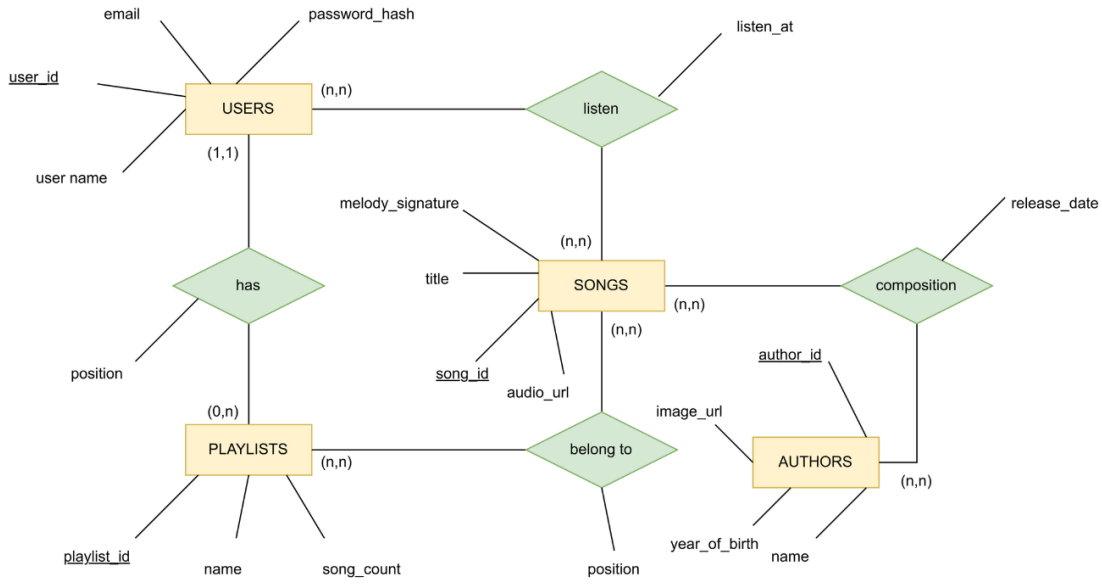
4. **Lựa chọn công cụ:** PostgreSQL, pgAdmin 4 vì tận dụng kiểu dữ liệu mảng sẵn có của PostgreSQL

CHƯƠNG 2: PHÂN TÍCH BÀI TOÁN

1. Mô hình thực thể mối kết hợp:

- Các đối tượng cần quản lý:
    - Người dùng
    - Bài nhạc
    - Tác giả
    - Danh sách phát
- Các mối quan hệ:
    - Người dùng ***nghe*** nhạc: n-n
    - Tác giả ***sáng tác*** nhạc: n-n
    - Người dùng ***có*** danh sách phát: 1-n
    - Bài nhạc ***thuộc*** danh sách phát: n-n

2. Sơ đồ ERD:



Hình 2.1. Sơ đồ ERD

CHƯƠNG 3: THIẾT KẾ CSDL

1. **Phân tích yêu cầu tìm kiếm bài hát qua giai điệu:**

- **Yêu cầu**: Tìm kiếm bài hát qua một đoạn giai điệu (không phải toàn bộ giai điệu của bài) ở mọi tone khác nhau vì người dùng có nam, nữ và có thể ngân nga giai điệu ở các tone khác nhau
- **Vấn đề đặt ra**: Tìm cách số hóa giai điệu bài nhạc thành dạng dữ liệu dễ lưu trữ, dễ truy vấn mà vẫn đáp ứng đủ yêu cầu ở trên
- **Kiến thức nhạc lý cần thiết để giải quyết vấn đề**: Mỗi nốt nhạc có một cao độ khác nhau, thường đo bằng Hz hoặc cung. Trong trường hợp này, lựa chọn đo bằng cung vì cung là thang đo đơn giản chỉ gồm cung và nửa cung. Ví dụ như nốt Đô thấp hơn Rê (ở cùng quãng tám) một cung, Đô thấp hơn đô thăng nửa cung, và cứ thế nốt Đô tiếp theo cũng cách nốt Rê tương tự, hai nốt Đô liền nhau cách nhau 6 cung.
- **Giải pháp cho vấn đề:** Với một đoạn nhạc, ta sẽ số hóa giai điệu của nó như sau

  ví dụ đoạn nhạc là Đô Rê Mi Fa Sol (các nốt cùng quãng tám) thì mảng đặc trưng cho giai điệu đó sẽ là {1, 1, 0.5, 1} vì Rê cách Đô một cung, Mi cách Rê một cung, Fa cách Mi nửa cung, Sol cách Fa một cung.
- **Giải pháp**: ta cài đặt bảng songs có một cột lưu melody_signature dưới dạng mảng các số thực (real[])
- **Vấn đề phát sinh**: việc truy vấn trên mảng khó => ta lưu thêm một cột là melody_signature_str = array_to_string(melody_signature, ',') để dùng truy vấn LIKE ‘%...%’ để truy vấn qua một đoạn giai điệu chứ không phải toàn bộ giai điệu của bài, việc lưu melody_signature_str có hiệu suất tốt hơn so với việc mỗi lần truy vấn ta lại gọi hàm để tạo melody_signature_str từ melody_signature

2. **Giải thích và lý do lưu trữ một số thuộc tính:**

- Password_hash của bảng users: lưu mật khẩu băm, thuật toán băm và thực hiện băm do Backend đảm nhiệm, CSDL chỉ có nhiệm vụ lưu mật khẩu sau khi băm, nhằm đảm bảo an toàn nếu mật khẩu băm ở CSDL bị lộ thì mật khẩu gốc vẫn an toàn

- audio_url của bảng songs: đường dẫn tới bài nhạc, giúp Backend truy cập đến file bài nhạc để thực hiện chức năng phát nhạc
- image_url của bảng authors: đường dẫn tới ảnh tác giả, giúp xuất ảnh ra Frontend
- song_count của bảng playlists: lưu số lượng bài nhạc trong danh sách phát, dù có thể truy vấn đếm số hàng ở playlist_details có playlist_id thỏa mãn để biết số lượng bài nhạc trong danh sách phát nào đó, nhưng vì không muốn mỗi lần như vậy đều phải chạy lại câu truy vấn, kết bảng, so sánh… nên lưu song_count để tang hiệu suất, đổi lại, ta chấp nhận viết ràng buộc để đảm bảo song_count không bị sai khi bảng playlist_details có thêm hoặc xóa
- position của bảng playlist_details: thứ tự trong danh sách phát của bài nhạc

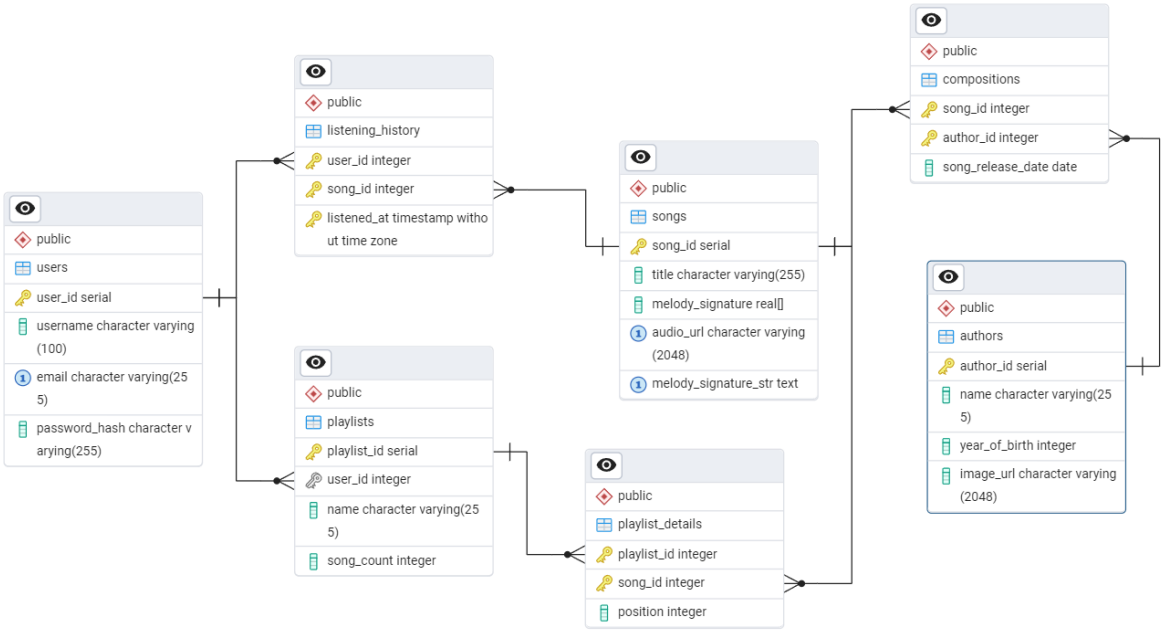
3. Bảng mô tả lược đồ quan hệ:

Quan hệ	Thuộc tính	Kiểu dữ liệu	Diễn giải
users	user_id	serial	Khóa chính
	username	character varying(100)	Not null
	email	character varying(255)	unique
	password_hash	character varying(255)	
songs	song_id	serial	Khóa chính
	title	character varying(255)	Not null
	melody_signature	real[]	
	audio_url	character varying(2048)	unique
	melody_signature_str	text	unique
authors	author_id	serial	Khóa chính
	name	character varying(255)	Not null

	year_of_birth	integer	
	image_url	character varying(2048)	
playlists	playlist_id	serial	Khóa chính
	user_id	integer	Tham chiếu user_id
	name	character varying(255)	Not null
	song_count	integer	
playlist_details	playlist_id	integer	Khóa chính Tham chiếu playlist_id
	song_id	integer	Khóa chính Tham chiếu song_id
	position	integer	
compositions	song_id	integer	Khóa chính Tham chiếu song_id
	author_id	integer	Khóa chính Tham chiếu author_id
	song_release_date	date	
listening_history	user_id	integer	Khóa chính Tham chiếu user_id
	song_id	integer	Khóa chính Tham chiếu song_id
	listened_at	timestamp	Khóa chính

Bảng 3.1. Bảng mô tả lược đồ quan hệ

4. Sơ đồ quan hệ



Hình 3.1. Sơ đồ quan hệ

CHƯƠNG 4: CÀI ĐẶT BẢNG, CỘT, KHÓA CHÍNH, KHÓA NGOẠI, CONSTRAINT

1. Code cài đặt

```
BEGIN;


CREATE TABLE IF NOT EXISTS public.authors
(
    author_id serial NOT NULL,
    name character varying(255) COLLATE pg_catalog."default" NOT NULL,
    year_of_birth integer,
    image_url character varying(2048) COLLATE pg_catalog."default",
    CONSTRAINT authors_pkey PRIMARY KEY (author_id)
);

CREATE TABLE IF NOT EXISTS public.compositions
(
    song_id integer NOT NULL,
    author_id integer NOT NULL,
    song_release_date date,
    CONSTRAINT compositions_pkey PRIMARY KEY (song_id, author_id)
);
```

```
CREATE TABLE IF NOT EXISTS public.listening_history
(
    user_id integer NOT NULL,
    song_id integer NOT NULL,
    listened_at timestamp without time zone NOT NULL DEFAULT
CURRENT_TIMESTAMP,
    CONSTRAINT listening_history_pkey PRIMARY KEY (user_id, song_id,
listened_at)
);

CREATE TABLE IF NOT EXISTS public.playlist_details
(
    playlist_id integer NOT NULL,
    song_id integer NOT NULL,
    "position" integer,
    CONSTRAINT playlist_details_pkey PRIMARY KEY (playlist_id, song_id)
);

CREATE TABLE IF NOT EXISTS public.playlists
(
    playlist_id serial NOT NULL,
    user_id integer NOT NULL,
    name character varying(255) COLLATE pg_catalog."default" NOT NULL,
    song_count integer DEFAULT 0,
    CONSTRAINT playlists_pkey PRIMARY KEY (playlist_id)
);

CREATE TABLE IF NOT EXISTS public.songs
(
    song_id serial NOT NULL,
    title character varying(255) COLLATE pg_catalog."default" NOT NULL,
    melody_signature real[] NOT NULL,
    audio_url character varying(2048) COLLATE pg_catalog."default",
    melody_signature_str text COLLATE pg_catalog."default",
    CONSTRAINT songs_pkey PRIMARY KEY (song_id),
    CONSTRAINT unique_audio_url UNIQUE (audio_url),
    CONSTRAINT unique_melody_signature_str UNIQUE (melody_signature_str)
);

CREATE TABLE IF NOT EXISTS public.users
(
    user_id serial NOT NULL,
    username character varying(100) COLLATE pg_catalog."default" NOT NULL,
    email character varying(255) COLLATE pg_catalog."default" NOT NULL,
```

```
    password_hash character varying(255) COLLATE pg_catalog."default" NOT
NULL,
    CONSTRAINT users_pkey PRIMARY KEY (user_id),
    CONSTRAINT users_email_key UNIQUE (email)
);

ALTER TABLE IF EXISTS public.compositions
    ADD CONSTRAINT compositions_author_id_fkey FOREIGN KEY (author_id)
    REFERENCES public.authors (author_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE RESTRICT;


ALTER TABLE IF EXISTS public.compositions
    ADD CONSTRAINT compositions_song_id_fkey FOREIGN KEY (song_id)
    REFERENCES public.songs (song_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE RESTRICT;


ALTER TABLE IF EXISTS public.listening_history
    ADD CONSTRAINT listening_history_song_id_fkey FOREIGN KEY (song_id)
    REFERENCES public.songs (song_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE RESTRICT;


ALTER TABLE IF EXISTS public.listening_history
    ADD CONSTRAINT listening_history_user_id_fkey FOREIGN KEY (user_id)
    REFERENCES public.users (user_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE RESTRICT;


ALTER TABLE IF EXISTS public.playlist_details
    ADD CONSTRAINT playlist_details_playlist_id_fkey FOREIGN KEY
(playlist_id)
    REFERENCES public.playlists (playlist_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE RESTRICT;


ALTER TABLE IF EXISTS public.playlist_details
    ADD CONSTRAINT playlist_details_song_id_fkey FOREIGN KEY (song_id)
    REFERENCES public.songs (song_id) MATCH SIMPLE
    ON UPDATE NO ACTION
```

```
    ON DELETE RESTRICT;


ALTER TABLE IF EXISTS public.playlists
    ADD CONSTRAINT playlists_user_id_fkey FOREIGN KEY (user_id)
    REFERENCES public.users (user_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE RESTRICT;

END;
```

2. Giải thích một số chi tiết

- listened_at timestamp without time zone NOT NULL DEFAULT CURRENT_TIMESTAMP: thuộc tính listened_at là mốc thời gian nhưng bỏ múi giờ và được mặc định là thời gian lúc thêm hàng vào bảng
- ON UPDATE NO ACTION: để không cho phép sửa đổi thuộc tính nào có khóa ngoại tham chiếu tới
- ON DELETE RESTRICT: để không cho phép xóa hàng nào mà có khóa ngoại tham chiếu tới thuộc tính của nó

CHƯƠNG 5: CÀI ĐẶT RÀNG BUỘC ĐỂ BẢO VỆ TÍNH TOÀN VẸN DỮ LIỆU

1. **Các ràng buộc cần thiết cho CSDL.** (Đánh số các ràng buộc ở cuối dòng để dễ gọi tên)

   - Email của người dùng là duy nhất: đã cài đặt ở trên khi tạo bảng (1)

   - Url của bài hát là duy nhất: đã cài ở trên khi tạo bảng (2)

   - Giai điệu của bài nhạc là duy nhất (3)

   - Ở mỗi hàng trong bảng songs, thuộc tính melody_signature_str luôn bằng array_to_string(melody_signature, ‘,’) (4)

   - Danh sách phát mới tạo (chưa được thêm bài nhạc) sẽ có song_count = 0, song_count sẽ cập nhật lại khi bài nhạc được thêm vào danh sách hoặc xóa khỏi danh sách (5)

   - Đảm bảo position của bài nhạc là đúng mỗi khi thêm, xóa, sửa(6)

   - Không được sửa khóa ngoại playlist_id và song_id trong bảng playlist_details (7)

   - Không được sửa thuộc tính có khóa ngoại tham chiếu tới: đã cài đặt ở trên khi tạo bảng (8)

- Không được xóa hàng có khóa ngoại tham chiếu tới thuộc tính của nó: đã cài đặt ở trên khi tạo bảng (9)

2. Cách để tạo các ràng buộc

- Các ràng buộc (1), (2), (8), (9) đã được cài đặt ở trên khi tạo bảng

- Cài đặt ràng buộc (3), (4):

  - Nếu cài đặt unique ở thuộc tính melody_signature có cấu trúc là array thì sẽ gây khó khăn trong quá trình so sánh để xác định tính unique từ đó làm chậm các quá trình thêm dữ liệu và giảm hiệu suất nên ta sẽ cài đặt unique cho melody_signature_str (đã cài ở trên), đồng thời không cho phép sửa trực tiếp melody_signature_str, chỉ cho thêm, sửa melody_signature và sau đó melody_signature_str sẽ được tự động cập nhật theo
  - Cài đặt không cho sửa trực tiếp melody_signature_str:

```
CREATE OR REPLACE FUNCTION prevent_direct_update_melody_signature_str()
RETURNS TRIGGER AS $$

BEGIN
   IF current_setting('pg_temp.trigger_update_melody', true) = 'true' THEN
      RETURN NEW;
   END IF;
   -- Nếu giá trị melody_signature_str bị thay đổi, chặn hành động
   IF NEW.melody_signature_str IS DISTINCT FROM OLD.melody_signature_str
THEN
      RAISE EXCEPTION 'Không được phép chỉnh sửa trực tiếp cột
melody_signature_str. Hãy chỉnh sửa cột melody_signature.';
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_prevent_direct_update_melody_signature_str
BEFORE UPDATE OF melody_signature_str
ON songs
FOR EACH ROW
EXECUTE FUNCTION prevent_direct_update_melody_signature_str();
```

  - Cài đặt tự động cập nhật melody_signature_str khi melody_signature thay đổi hoặc được thêm:

```
CREATE OR REPLACE FUNCTION update_melody_signature_str()
RETURNS TRIGGER AS $$
```

```
BEGIN
   -- Chỉ cập nhật melody_signature_str nếu melody_signature thay đổi
   IF NEW.melody_signature IS DISTINCT FROM OLD.melody_signature THEN
      PERFORM set_config('pg_temp.trigger_update_melody', 'true', true);
      NEW.melody_signature_str = array_to_string(NEW.melody_signature, ',');
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_update_melody_signature_str
BEFORE INSERT OR UPDATE
ON songs
FOR EACH ROW
EXECUTE FUNCTION update_melody_signature_str();
```

- Cài đặt ràng buộc (5):
- Cài đặt không cho trực tiếp sửa song_count, khi tạo danh sách phát mới, song_count tự động bằng 0:

```
CREATE OR REPLACE FUNCTION
prevent_direct_update_song_count()
RETURNS TRIGGER AS $$
BEGIN
     -- Bỏ qua nếu thao tác được thực hiện bởi trigger
(được đánh dấu qua biến session pg_temp)
    IF current_setting('pg_temp.trigger_update', true) =
'true' THEN
        RETURN NEW;
    END IF;

    -- Khi thêm mới (INSERT)
    IF TG_OP = 'INSERT' THEN
        -- Nếu giá trị song_count được cung cấp và khác 0
-> Báo lỗi
        IF NEW.song_count IS NOT NULL AND NEW.song_count
<> 0 THEN
            RAISE EXCEPTION 'Không được phép thêm trực
tiếp số lượng bài nhạc khác 0. Giá trị mặc định là 0.';
        END IF;

        -- Đảm bảo song_count luôn được khởi tạo bằng 0
        NEW.song_count = 0;

    -- Khi cập nhật (UPDATE)
    ELSIF TG_OP = 'UPDATE' THEN
        -- Nếu giá trị song_count bị thay đổi -> Báo lỗi
        IF NEW.song_count <> OLD.song_count THEN
```

```
            RAISE EXCEPTION 'Không được phép chỉnh sửa
trực tiếp số lượng bài nhạc.';
        END IF;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_prevent_direct_update_song_count
BEFORE INSERT OR UPDATE
ON playlists
FOR EACH ROW
EXECUTE FUNCTION prevent_direct_update_song_count();
```

- Cài đặt để mỗi khi thêm bài nhạc vào danh sách phát, song_count tự động tăng thêm 1

```
CREATE OR REPLACE FUNCTION increment_song_count()
RETURNS TRIGGER AS $$
BEGIN
  -- Đánh dấu rằng thao tác cập nhật được thực hiện bởi trigger
  PERFORM set_config('pg_temp.trigger_update', 'true', true);

  -- Tăng số lượng bài nhạc trong playlists
  UPDATE playlists
  SET song_count = song_count + 1
  WHERE playlist_id = NEW.playlist_id;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_increment_song_count
BEFORE INSERT OR UPDATE ON playlist_details
FOR EACH ROW
EXECUTE FUNCTION increment_song_count();
```

- Cài đặt để khi xóa bài nhạc khỏi danh sách phát, song_count tự động giảm 1

```
CREATE OR REPLACE FUNCTION update_song_count_on_delete()
RETURNS TRIGGER AS $$
BEGIN
  PERFORM set_config('pg_temp.trigger_update', 'true', true);
  -- Giảm giá trị song_count trong bảng playlists tương ứng
  UPDATE playlists
  SET song_count = song_count - 1
  WHERE playlist_id = OLD.playlist_id;
```

```
    RETURN OLD;

END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_update_song_count_on_delete
AFTER DELETE ON playlist_details
FOR EACH ROW
EXECUTE FUNCTION update_song_count_on_delete();
```

- Cài đặt ràng buộc (6):
- Cài đặt không cho sửa trực tiếp position của bài nhạc:

```
CREATE OR REPLACE FUNCTION prevent_update_position()
RETURNS TRIGGER AS $$

BEGIN
    -- Bỏ qua nếu biến pg_temp.ignore_position_update được thiết lập
    IF current_setting('pg_temp.ignore_position_update', true) = 'true' THEN
        RETURN NEW;
    END IF;

    -- Kiểm tra nếu cột position bị thay đổi
    IF NEW.position <> OLD.position THEN
        RAISE EXCEPTION 'Không được phép chỉnh sửa trực tiếp cột position trong
bảng playlist_details.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_prevent_update_position
BEFORE UPDATE OF position
ON playlist_details
FOR EACH ROW
EXECUTE FUNCTION prevent_update_position();
```

- Cài đặt mỗi khi thêm bài nhạc vào danh sách phát, thứ tự của bài nhạc đó ở cuối danh sách phát

```
CREATE OR REPLACE FUNCTION set_position_in_playlist_details()
RETURNS TRIGGER AS $$
DECLARE
    current_song_count INTEGER;
BEGIN
    -- Lấy giá trị song_count từ bảng playlists với playlist_id tương ứng
    SELECT song_count
```

```
    INTO current_song_count
    FROM playlists
    WHERE playlist_id = NEW.playlist_id;

    -- Gán giá trị position là song_count + 1
    NEW.position := current_song_count + 1;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_set_position_in_playlist_details
BEFORE INSERT
ON playlist_details
FOR EACH ROW
EXECUTE FUNCTION set_position_in_playlist_details();
```

- Cài đặt để khi xóa một bài nhạc khỏi danh sách phát, thứ tự các bài nhạc còn lại trong danh sách phát vẫn đúng logic

```
CREATE OR REPLACE FUNCTION update_position_on_delete()
RETURNS TRIGGER AS $$
BEGIN
        PERFORM set_config('pg_temp.ignore_position_update', 'true', true);
    -- Giảm giá trị position của các hàng có cùng playlist_id và position lớn hơn
    UPDATE playlist_details
    SET position = position - 1
    WHERE playlist_id = OLD.playlist_id
      AND position > OLD.position;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_update_position_on_delete
AFTER DELETE
ON playlist_details
FOR EACH ROW
EXECUTE FUNCTION update_position_on_delete();
```

- Cài đặt procedure để đổi thứ tự của hai bài nhạc, không cho đổi trực tiếp mà chỉ có thể dùng procedure này để đổi

```
CREATE OR REPLACE PROCEDURE swap_song_positions(playlist_id_input
INTEGER, song_id_1 INTEGER, song_id_2 INTEGER)
LANGUAGE plpgsql AS $$
DECLARE
    position_1 INTEGER;
    position_2 INTEGER;
BEGIN
    -- Thiết lập biến pg_temp.ignore_position_update để cho phép cập nhật
    PERFORM set_config('pg_temp.ignore_position_update', 'true', true);
```

```
    -- Lấy giá trị position của song_id_1
    SELECT position INTO position_1
    FROM playlist_details
    WHERE playlist_id = playlist_id_input AND song_id = song_id_1;

    -- Lấy giá trị position của song_id_2
    SELECT position INTO position_2
    FROM playlist_details
    WHERE playlist_id = playlist_id_input AND song_id = song_id_2;

    -- Kiểm tra nếu một trong hai bài nhạc không tồn tại
    IF position_1 IS NULL OR position_2 IS NULL THEN
        RAISE EXCEPTION 'Một hoặc cả hai bài nhạc không tồn tại trong playlist_id
= %', playlist_id_input;
    END IF;

    -- Hoán đổi giá trị position
    UPDATE playlist_details
    SET position = position_2
    WHERE playlist_id = playlist_id_input AND song_id = song_id_1;

    UPDATE playlist_details
    SET position = position_1
    WHERE playlist_id = playlist_id_input AND song_id = song_id_2;

    -- Hoàn thành hoán đổi
    RAISE NOTICE 'Đã hoán đổi vị trí của song_id_1 = % và song_id_2 = % trong
playlist_id = %',
                 song_id_1, song_id_2, playlist_id_input;
END;
$$;
```

- Cài đặt ràng buộc (7)
- Không được sửa playlist_id

```
CREATE OR REPLACE FUNCTION prevent_update_playlist_id()
RETURNS TRIGGER AS $$
BEGIN
    RAISE EXCEPTION 'Không được phép thay đổi playlist_id trong bảng
playlist_details.';
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_prevent_update_playlist_id
BEFORE UPDATE OF playlist_id
ON playlist_details
```

```
FOR EACH ROW
EXECUTE FUNCTION prevent_update_playlist_id();
```

- Không được sửa song_id

```
CREATE OR REPLACE FUNCTION prevent_update_song_id()
RETURNS TRIGGER AS $$
BEGIN
   RAISE EXCEPTION 'Không được phép thay đổi song_id trong bảng
playlist_details.';
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_prevent_update_song_id
BEFORE UPDATE OF song
ON playlist_details
FOR EACH ROW
EXECUTE FUNCTION prevent_update_song_id();
```

CHƯƠNG 6: FUNCTION, STORED PROCEDURE, VIEW

1. Function

a) Xác minh đăng nhập

```
CREATE OR REPLACE FUNCTION verify_login(
   email_input TEXT,
   password_hash_input TEXT
)
RETURNS TABLE(user_id INTEGER, username CHARACTER VARYING(100))
AS $$
BEGIN
   -- Truy vấn để xác minh thông tin đăng nhập
   RETURN QUERY
   SELECT u.user_id, u.username
   FROM users u
   WHERE u.email = email_input AND u.password_hash = password_hash_input;
END;
$$ LANGUAGE plpgsql;
```

b) Tìm kiếm bài nhạc qua giai điệu

```
CREATE OR REPLACE FUNCTION
search_by_melody(melody_signature_str_input TEXT)
RETURNS TABLE(song_title TEXT, song_audio_url TEXT) AS $$
BEGIN
   RETURN QUERY
   SELECT s.title::TEXT AS song_title, s.audio_url::TEXT AS song_audio_url
   FROM songs s
```

```
    WHERE s.melody_signature_str LIKE '%,' || melody_signature_str_input || ',%'
       OR s.melody_signature_str LIKE melody_signature_str_input || ',%'
       OR s.melody_signature_str LIKE '%,' || melody_signature_str_input
       OR s.melody_signature_str = melody_signature_str_input;
END;
$$ LANGUAGE plpgsql;
```

c) Truy vấn lịch sử nghe nhạc của người dùng

```
CREATE OR REPLACE FUNCTION get_listening_history(user_id_input
INTEGER)
RETURNS TABLE(
    user_id INTEGER,
    song_id INTEGER,
    listened_at TIMESTAMP
) AS $$
BEGIN
    -- Truy vấn dữ liệu từ bảng listening_history
    RETURN QUERY
    SELECT l.user_id, l.song_id, l.listened_at
    FROM listening_history l
    WHERE l.user_id = user_id_input;
END;
$$ LANGUAGE plpgsql;
```

2. Stored procedure

a) Đổi thứ tự bài nhạc trong danh sách phát
Đã làm ở phần ràng buộc (6)

b) Xóa danh sách phát
Phải xóa tất cả bài nhạc thuộc danh sách phát đó rồi mới xóa danh sách phát (xóa playlist_detail rồi mới xóa playlist)

```
CREATE OR REPLACE PROCEDURE delete_playlist(playlist_id_input INTEGER)
LANGUAGE plpgsql AS $$
BEGIN
    -- Xóa tất cả các mục trong playlist_details có playlist_id
    DELETE FROM playlist_details
    WHERE playlist_id = playlist_id_input;

    -- Xóa playlist trong bảng playlists
    DELETE FROM playlists
    WHERE playlist_id = playlist_id_input;

    -- Thông báo hoàn thành
    RAISE NOTICE 'Playlist với playlist_id = % đã được xóa thành công cùng với các
chi tiết liên quan.', playlist_id_input;
END;
$$;
```

c) Đăng ký tài khoản

```
CREATE OR REPLACE PROCEDURE register_user(
    email_input CHARACTER VARYING(255),
    username_input CHARACTER VARYING(100),
    password_hash_input CHARACTER VARYING(255),
    OUT result INTEGER
)
LANGUAGE plpgsql AS $$
BEGIN
    -- Thêm thông tin người dùng vào bảng users
    BEGIN
        INSERT INTO users (email, username, password_hash)
        VALUES (email_input, username_input, password_hash_input);

        -- Nếu thêm thành công, trả về 1
        result := 1;
    EXCEPTION
        WHEN unique_violation THEN
            -- Nếu vi phạm UNIQUE constraint, in thông báo và trả về 0
            RAISE NOTICE 'Đăng ký thất bại: Email "%s" đã tồn tại.', email_input;
            result := 0;
    END;
END;
$$;
```

3. VIEW

a) Top 10 bài nhạc được phát nhiều nhất

```
SELECT s.song_id,
    s.title,
    count(l.song_id) AS listen_count
   FROM songs s,
    listening_history l
  WHERE l.song_id = s.song_id
  GROUP BY s.song_id, s.title
  ORDER BY (count(l.song_id)) DESC
 LIMIT 10;
```

b) Các danh sách phát của một người dùng

```
CREATE OR REPLACE VIEW playlists_of AS
SELECT p.playlist_id,
            p.user_id,
            p.name,
            p.song_count
FROM playlists p
```

- Ví dụ cách dùng:

```
SELECT * from playlists_of p WHERE p.user_id =  105004
```

c) Top 10 tác giả được nghe nhiều nhất

```
CREATE OR REPLACE VIEW top_10_most_listened_authors AS
SELECT
   a.author_id,
   a.name AS author_name,
   COUNT(lh.song_id) AS total_listens
FROM
   authors a
JOIN
   compositions c ON a.author_id = c.author_id
JOIN
   songs s ON c.song_id = s.song_id
JOIN
   listening_history lh ON s.song_id = lh.song_id
GROUP BY
   a.author_id, a.name
ORDER BY
   total_listens DESC
LIMIT 10;
```

d) Các bài nhạc trong một danh sách phát

```
CREATE OR REPLACE VIEW songs_of_a_playlist AS
SELECT pd.song_id,
                    pd.playlist_id,
                    s.title,
                    pd.position
FROM playlist_details pd, songs s
WHERE pd.song_id = s.song_id
```

- Ví dụ cách dùng:

```
SELECT * FROM songs_of_a_playlist WHERE playlist_id = 525433
```

e) Các bài nhạc của một tác giả

```
CREATE OR REPLACE VIEW songs_of_an_author AS
SELECT s.song_id,
                    a.author_id,
                    s.title
FROM songs s, compositions c, authors a
WHERE s.song_id = c.song_id AND c.author_id = a.author_id
```

- Ví dụ cách dùng:

```
SELECT * FROM songs_of_an_author WHERE author_id = 22502
```

CHƯƠNG 7: PHÂN QUYỀN CƠ SỞ DỮ LIỆU

1. Phân tích

- Cơ sở dữ liệu không có nhiều nhóm đối tượng người dùng khác nhau, mỗi người dùng đều có quyền trên các bảng giống nhau
- Kết luận: tạo 2 role là admin_user và app_user
  admin_user: Quản trị viên, có toàn quyền trên cơ sở dữ liệu

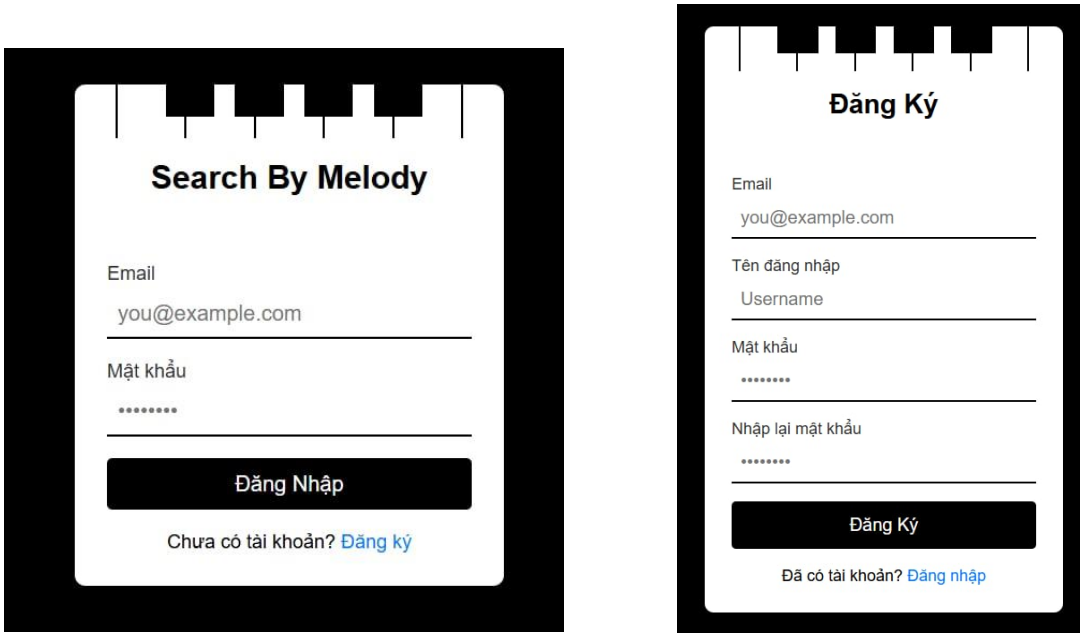
app_user: Tài khoản để những người dùng ứng dụng kết nối tới, có quyền xem, thêm, sửa, xóa ở các bảng users, playlists, playlist_details, có quyền xem và thêm ở bảng listening_history, và chỉ được quyền xem ở bảng songs, compositions và authors

2. **Code cài đặt**

```
CREATE ROLE admin_user WITH LOGIN PASSWORD 'adminpassword';
GRANT ALL PRIVILEGES ON DATABASE "SearchByMelody" TO admin_user;
CREATE ROLE app_user WITH LOGIN PASSWORD 'userpassword';
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE users, playlists,
playlist_details TO app_user;
GRANT SELECT, INSERT ON TABLE listening_history TO app_user;
GRANT SELECT ON TABLE songs, compositions, authors TO app_user;
```

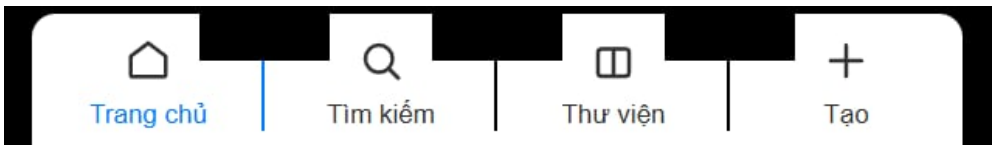
CHƯƠNG 8: BIỂU DIỄN THÔNG TIN

1. **Form đăng nhập, đăng ký tài khoản**



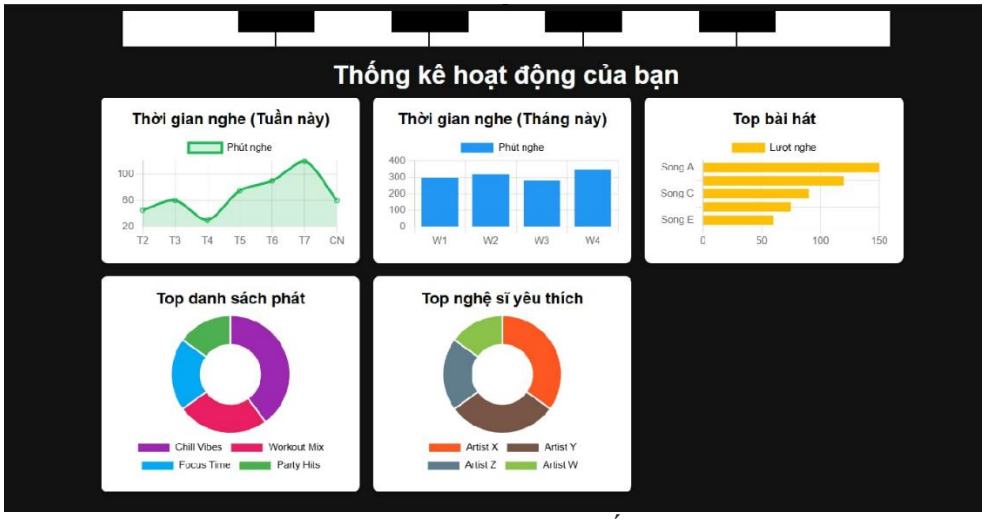
Hình 8.1: Form đăng nhập

2. **Menu các chức năng của ứng dụng**



Form 8.2: Menu chức năng

3. **Report thống kê hoạt động của người dùng**

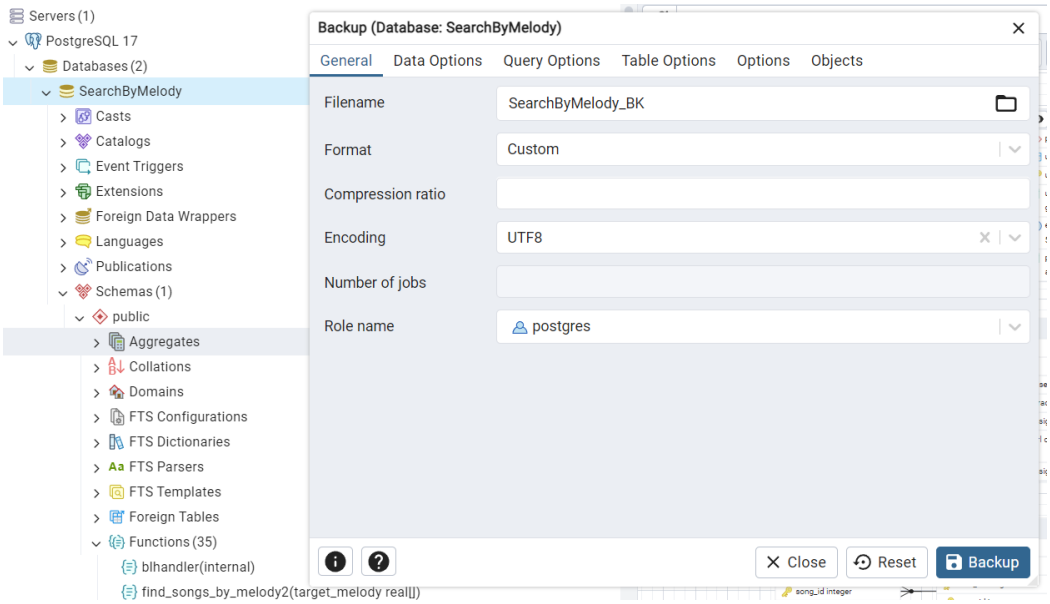


Hình 8.3: Report thống kê hoạt động

CHƯƠNG 9: SAO LƯU, KHÔI PHỤC

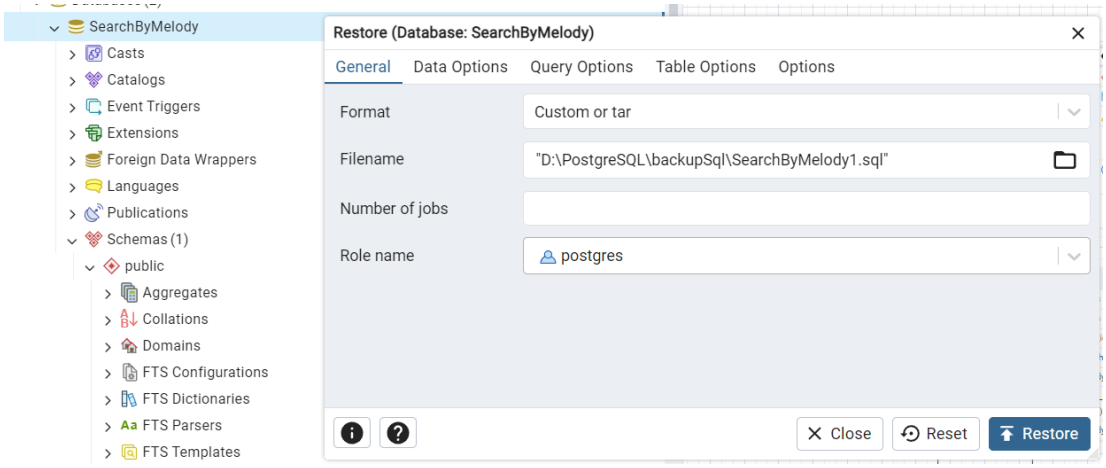
- Dùng cách thức đơn giản, kết nối với cơ sở dữ liệu qua pgadmin4 và export file backup
- Import file backup để khôi phục dữ liệu thông qua pgadmin4

1. Sao lưu



Hình 9.1: Giao diện sao lưu

2. Khôi phục



Hình 9.2: Giao diện khôi phục

CHƯƠNG 10: CÁC VẤN ĐỀ KHÁC

1. **Đề xuất một cách thu thập dữ liệu cho cột đặc trưng giai điệu**

- Yêu cầu thu thập dữ liệu: trích xuất được ra khoảng cách cao độ (tính theo cung) giữa các nốt nhạc liền nhau trong bài nhạc. Lưu ý, nốt nhạc phải là giai điệu chính, không có nốt nhạc của nhạc đệm, bè
- Lựa chọn nguồn để trích xuất dữ liệu: các file MusicXML vì chúng là định dạng file XML, dễ dàng truy xuất thông tin, chúng cũng lưu giai điệu chính tách bạch với nhạc đệm
- Giải pháp: viết tool bằng python để tự động trích xuất mảng đặc trưng giai điệu Code cài đặt như sau:

```
import os
import json
from music21 import converter, note, chord, stream
def find_melody_part(score):
    """
    Tìm phần (part) chứa giai điệu chính trong file MusicXML.
    """
    melody_part = None
    max_notes = 0

    for part in score.parts:
        # Kiểm tra nhãn của part (nếu có)
        if part.partName and "melody" in part.partName.lower():
            return part

        # Đếm số lượng nốt nhạc trong part
        num_notes = len([n for n in part.flat.notes if isinstance(n,
note.Note)])
        if num_notes > max_notes:
            max_notes = num_notes
```

```
            melody_part = part

    return melody_part

def extract_intervals_from_melody(file_path):
    """
    Trích xuất khoảng cách độ cao giữa các nốt liền nhau trong giai điệu chính
từ file MusicXML.
    """
    try:
        # Đọc file MusicXML
        score = converter.parse(file_path)

        # Tìm phần giai điệu chính
        melody_part = find_melody_part(score)
        if melody_part is None:
            print(f"Không tìm thấy giai điệu trong file {file_path}.")
            return None

        # Lọc các nốt nhạc (bỏ qua hợp âm, nhịp nghỉ)
        melody_notes = []
        for element in melody_part.flat.notes:
            if isinstance(element, note.Note):
                melody_notes.append(element)
            elif isinstance(element, chord.Chord):
                # Lấy nốt có cao độ lớn nhất trong hợp âm
                highest_note = max(element.notes, key=lambda n: n.pitch.midi)
                melody_notes.append(highest_note)

        # Tính khoảng cách độ cao giữa các nốt liền nhau (theo cung)
        intervals = []
        for i in range(len(melody_notes) - 1):
            pitch1 = melody_notes[i].pitch.midi  # Cao độ nốt hiện tại (MIDI
number)
            pitch2 = melody_notes[i + 1].pitch.midi  # Cao độ nốt tiếp theo
(MIDI number)
            interval = (pitch2 - pitch1) / 2  # Chuyển đổi từ MIDI number sang
cung
            intervals.append(interval)

        return intervals
    except Exception as e:
        print(f"Lỗi khi xử lý file {file_path}: {e}")
        return None

def process_musicxml_folder(folder_path):
    """
```

```
    Xử lý tất cả các file MusicXML (.mxl) trong thư mục và trích xuất đặc
trưng.
    """
    all_intervals = {}
    for file_name in os.listdir(folder_path):
        if file_name.endswith('.mxl'):
            file_path = os.path.join(folder_path, file_name)
            print(f"Đang xử lý file: {file_name}")
            intervals = extract_intervals_from_melody(file_path)
            if intervals is not None:
                all_intervals[file_name] = intervals
    return all_intervals

def save_intervals_to_json(data, output_file):
    """
    Lưu kết quả trích xuất vào file JSON.
    """
    with open(output_file, 'w', encoding='utf-8') as f:
        json.dump(data, f, ensure_ascii=False, indent=4)

def main():
    # Đường dẫn tới thư mục chứa các file MusicXML
    folder_path = r"D:\MusicXMLFolder"

    # Kiểm tra thư mục có tồn tại không
    if not os.path.isdir(folder_path):
        print("Thư mục không tồn tại. Vui lòng kiểm tra lại.")
        return

    # Đường dẫn tới file JSON để lưu kết quả
    output_file = r"D:\MusicXMLFolder\output.json"

    # Xử lý các file MusicXML và lưu kết quả
    print("Bắt đầu xử lý các file .mxl...")
    intervals_data = process_musicxml_folder(folder_path)
    save_intervals_to_json(intervals_data, output_file)
    print(f"Đã lưu kết quả trích xuất vào {output_file}")

if __name__ == "__main__":
    main()
```

2. Hướng phát triển

- Cách trích đặc trưng giai điệu này có thể dùng để tạo bộ dữ liệu training model AI có khả năng sáng tác nhạc
- Thêm nhiều chức năng khác cho việc quản lý danh sách phát như chia sẻ danh sách phát, đồng sở hữu danh sách phát, tạo danh sách phát phù hợp dựa vào gu nghe nhạc của nhiều người,…
- Tối ưu hiệu suất của truy vấn khó là tìm kiếm bài nhạc qua giai điệu

- Tối ưu cách lưu đặc trưng giai điệu bằng cách rút gọn những phần trùng lặp trong giai điệu (ví dụ như nhiều câu hát cùng giai điệu, hoặc là điệp khúc lặp lại hoặc lời một, lời hai nhưng giai điệu giống nhau…)

CHƯƠNG 11: KẾT LUẬN

1. **Tự đánh giá ưu điểm**

Đồ án của nhóm đã giải quyết được bài toán đặt ra:

- Nghĩ ra giải pháp số hóa dữ liệu giai điệu âm nhạc và cách truy vấn trên dữ liệu đó bằng phương pháp đơn giản, dễ triển khai
- Xây dựng một cơ sở dữ liệu phục vụ được các chức năng cần có của ứng dụng, tạo các ràng buộc để đảm bảo tính toàn vẹn dữ liệu, phân quyền để kiểm soát hành vi trong cơ sở dữ liệu
- Tạo được bản web demo chức năng tìm kiếm bài nhạc qua giai điệu

2. **Tự đánh giá khuyết điểm**

Vẫn còn những thứ cần cải thiện:

- Chưa tối ưu được truy vấn khó: tìm kiếm bài nhạc thông qua giai điệu
- Chưa thu thập số lượng lớn dữ liệu thực tế được và phải fake dữ liệu vì chưa thu thập được nhiều file MusicXML