

**UNIVERSIDADE DO VALE DO ITAJAÍ
CIÊNCIAS DA COMPUTAÇÃO**

LAURA ERIKA KLUGE SCHETINGER

TRABALHO M3 DE BANCO DE DADOS

SISTEMA DE STREAMING: UNIFY

PROFESSOR: MAURÍCIO PASETTO DE FREITAS

ITAJAÍ / SC

2025

SUMÁRIO

| | |
|-------------------------------------------|-------------------------------|
| 1. DEFINIÇÃO | 3 1.1. |
| FORMULAÇÃO DO PROBLEMA | 3 |
| 1.2. SOLUÇÃO PROPOSTA | 3 |
| | |
| 2. PROJETO CONCEITUAL | 5 |
| | |
| 3. PROJETO LÓGICO | 6 3.1. PROJETO |
| LÓGICO TEXTUAL NORMALIZADO | 6 3.2. |
| DIAGRAMA RELACIONAL NORMALIZADO | 6 |
| | |
| 4. PROJETO FÍSICO | 7 4.1. |
| DIAGRAMA RELACIONAL (WORKBENCH) | 7 |
| 4.2. CÓDIGO SQL (DDL) | 8 4.3. INSERÇÃO DE |
| DADOS (DML) | 9 |
| | |
| 5. API EM PYTHON (DESENVOLVIMENTO) | 10 5.1. TABELA ARTISTA (CRUD) |
| | 10 5.2. TABELA GÊNERO |
| (CRUD) | 12 5.3. TABELA |
| ÁLBUM (CRUD) | 14 5.4. |
| TABELA MÚSICA (CRUD) | 16 |
| | |
| 6. REPOSITÓRIO GITHUB | 19 |

1. DEFINIÇÃO

1.1. FORMULAÇÃO DO PROBLEMA

Atualmente, os ouvintes de músicas enfrentam dificuldades em organizar e centralizar suas bibliotecas de áudio, no qual com a transição das mídias físicas para o formato digital, muitos usuários acabaram com coleções fragmentadas, dispersas entre arquivos locais desorganizados, dispositivos com armazenamento limitado e mídias obsoletas. Essa descentralização impede o acesso rápido e onipresente às faixas desejadas, criando uma experiência de usuário frustrante onde a busca por um álbum ou artista específico torna-se uma tarefa manual e ineficiente, sem a possibilidade de criar listas de reprodução personalizadas ou obter informações detalhadas sobre as obras.

Além disso, existe outro problema no qual desvincula o artista de sua obra pela falta de integridade de dados. Neste caso, sem um sistema relacional estruturado, o cadastro de artistas, lançamentos de álbuns e faixas individuais sofre com redundância de dados e inconsistências, dificultando a catalogação correta de discografias. Esse cenário prejudica tanto o usuário que consome informações erradas, quanto os artistas, que necessitam de uma plataforma confiável para que suas produções sejam armazenadas, catalogadas e recuperadas de forma lógica e hierárquica.

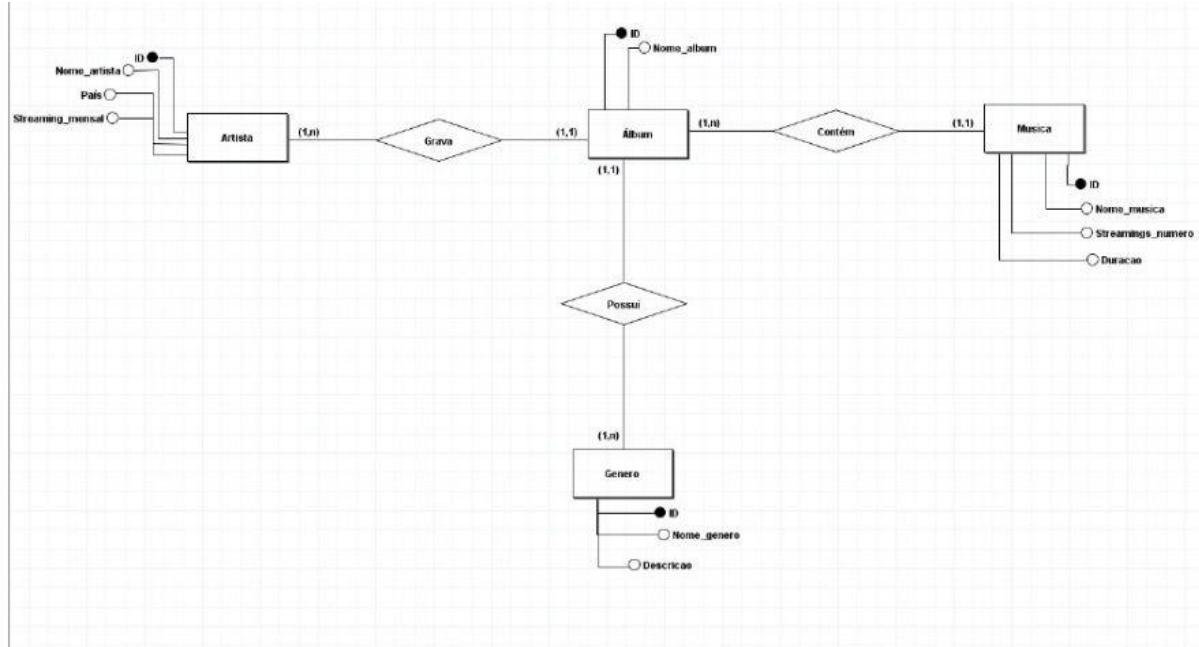
1.2. SOLUÇÃO PROPOSTA

Para mitigar os problemas de gestão e acesso ao conteúdo musical, propõe-se o desenvolvimento do **Unify**, um sistema de streaming de áudio suportado por um banco de dados relacional. A solução consiste numa plataforma centralizada que permite o registo estruturado e hierárquico das três entidades fundamentais do domínio: Artistas, Álbuns, Músicas e Gênero. Através desta arquitetura, o sistema garante a integridade referencial dos dados, assegurando que cada faixa musical esteja inequivocamente ligada ao seu álbum de origem e, consequentemente, ao seu artista criador, eliminando redundâncias e inconsistências comuns em armazenamentos amadores.

Além da organização lógica, o Unify implementará funcionalidades de manipulação de dados (CRUD) que permitirão aos administradores e usuários gerir o catálogo de forma dinâmica. A aplicação oferecerá interfaces para cadastrar

novos lançamentos, consultar discografias completas, atualizar metadados de faixas e remover conteúdos obsoletos. Desta forma, a solução não apenas resolve o problema do armazenamento disperso, mas também agrega valor ao permitir a recuperação rápida e precisa de informações, servindo como base sólida para futuras expansões, como a criação de playlists personalizadas e sistemas de recomendação.

2. PROJETO CONCEITUAL



3. PROJETO LÓGICO

3.1. PROJETO LÓGICO TEXTUAL NORMALIZADO

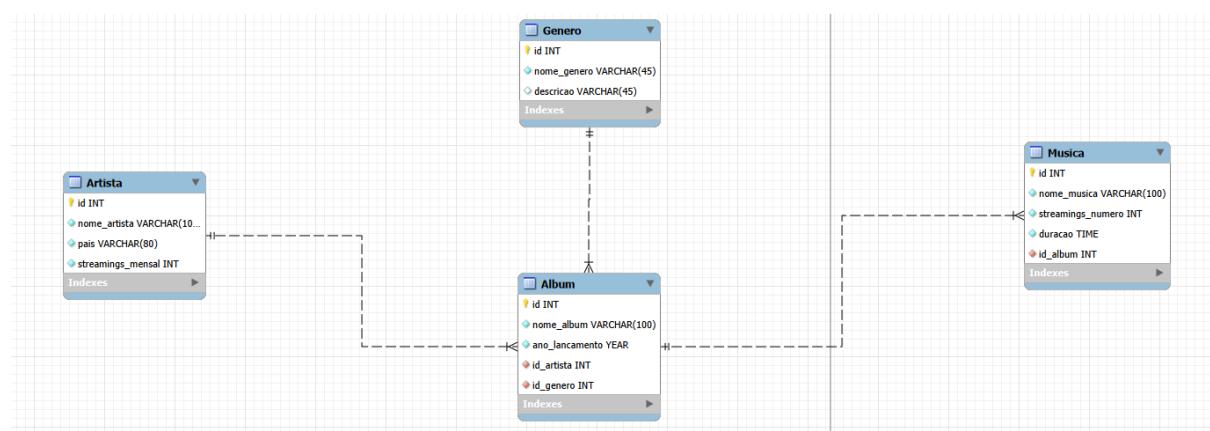
GENERO (id, nome_genero, descricao)

ARTISTA (id, nome_artista, pais, streaming_mensal)

ALBUM (id, nome_album, ano_lancamento, id_artista, id_genero)

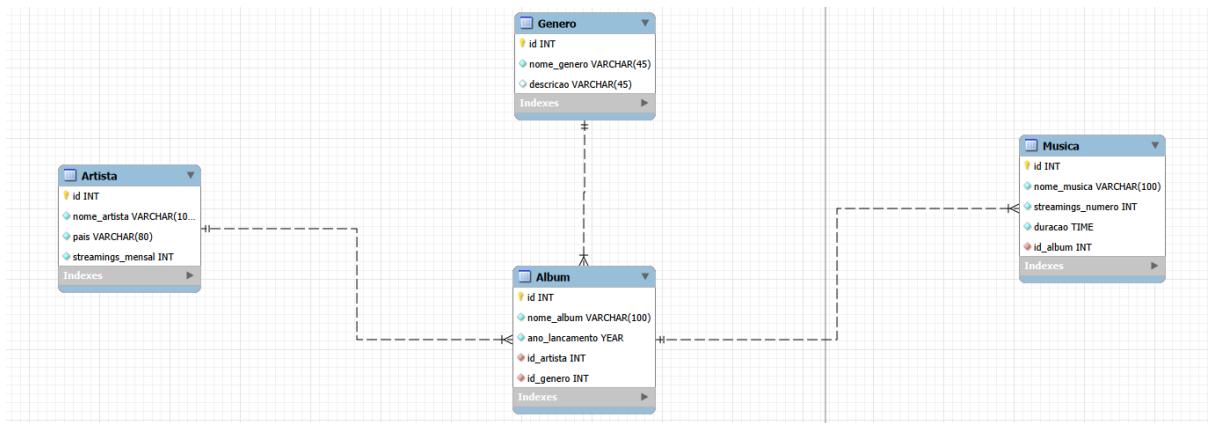
MUSICA (id, nome_musica, streamings_numero, duracao, id_album)

3.2. DIAGRAMA RELACIONAL NORMALIZADO



4. PROJETO FÍSICO

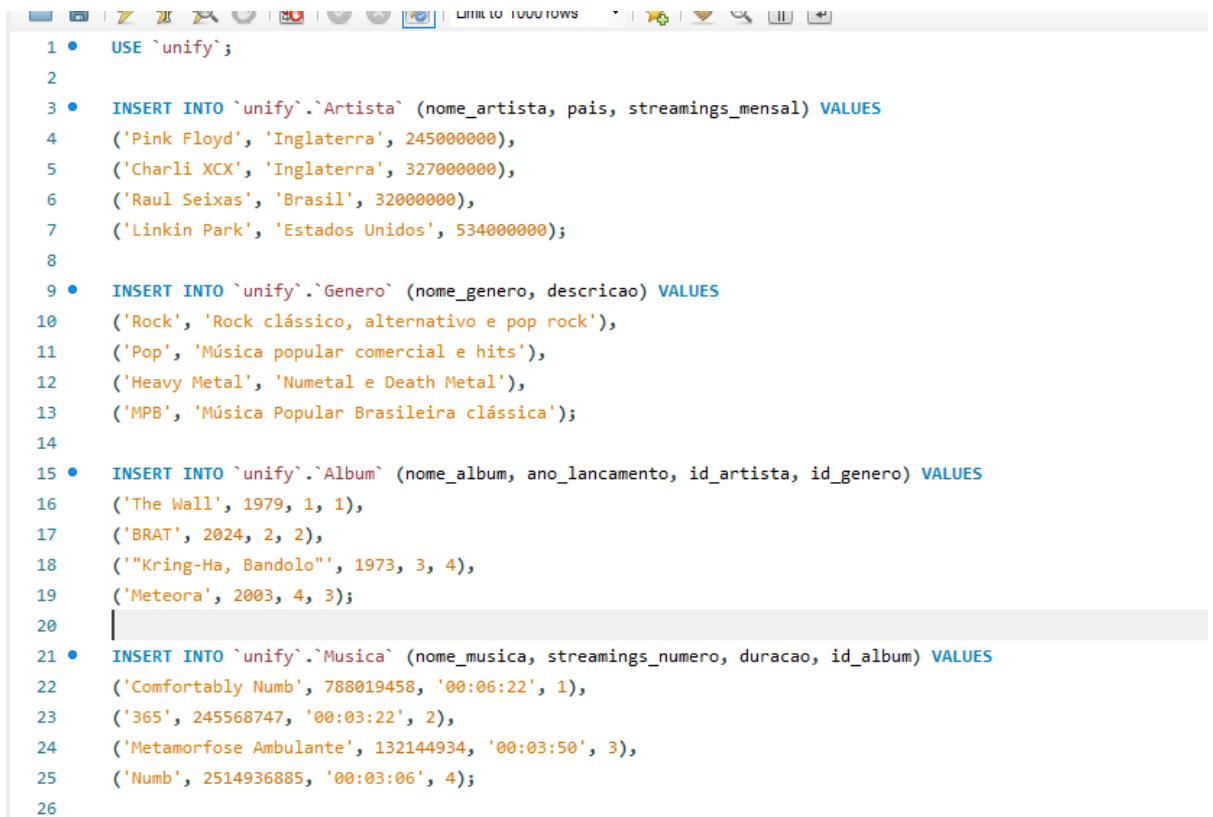
4.1. DIAGRAMA RELACIONAL



4.2. CÓDIGO SQL

```
1 • SET @OLD_UNIQUE_CHECKS:=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
2 • SET @OLD_FOREIGN_KEY_CHECKS:=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
3 • SET @OLD_SQL_MODE:=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
4
5 CREATE SCHEMA IF NOT EXISTS `unify` DEFAULT CHARACTER SET utf8 ;
6 USE `unify`;
7
8 • CREATE TABLE IF NOT EXISTS `unify`.`Artista` (
9   `id` INT NOT NULL AUTO_INCREMENT,
10  `nome_artista` VARCHAR(100) NOT NULL,
11  `pais` VARCHAR(80) NOT NULL,
12  `streamings_mensal` INT NOT NULL,
13  PRIMARY KEY (`id`),
14  ENGINE = InnoDB;
15
16 • CREATE TABLE IF NOT EXISTS `unify`.`Genero` (
17   `id` INT NOT NULL AUTO_INCREMENT,
18   `nome_genero` VARCHAR(45) NOT NULL,
19   `descricao` VARCHAR(45) NULL,
20   PRIMARY KEY (`id`),
21  ENGINE = InnoDB;
22
23 • CREATE TABLE IF NOT EXISTS `unify`.`Album` (
24   `id` INT NOT NULL AUTO_INCREMENT,
25   `ano_lancamento` YEAR NOT NULL,
26   `ano_lancamento` YEAR NOT NULL,
27   `id_artista` INT NOT NULL,
28   `id_genero` INT NOT NULL,
29   PRIMARY KEY (`id`),
30   INDEX `album_artista_idx` (`id_artista` ASC) VISIBLE,
31   INDEX `album_genero_idx` (`id_genero` ASC) VISIBLE,
32   CONSTRAINT `album_artista`
33     FOREIGN KEY (`id_artista`)
34       REFERENCES `unify`.`Artista` (`id`)
35       ON DELETE NO ACTION
36       ON UPDATE NO ACTION,
37   CONSTRAINT `album_genero`
38     FOREIGN KEY (`id_genero`)
39       REFERENCES `unify`.`Genero` (`id`)
40       ON DELETE NO ACTION
41       ON UPDATE NO ACTION,
42  ENGINE = InnoDB;
43
44 • CREATE TABLE IF NOT EXISTS `unify`.`Musica` (
45   `id` INT NOT NULL AUTO_INCREMENT,
46   `nome_musica` VARCHAR(100) NOT NULL,
47   `streamings_numero` INT NOT NULL,
48   `duracao` TIME NOT NULL,
49   `id_album` INT NOT NULL,
50   PRIMARY KEY (`id`),
51   INDEX `musica_album_idx` (`id_album` ASC) VISIBLE,
52   CONSTRAINT `musica_album`
53     FOREIGN KEY (`id_album`)
54       REFERENCES `unify`.`Album` (`id`)
55       ON DELETE NO ACTION
56       ON UPDATE NO ACTION,
57  ENGINE = InnoDB;
58
59 • ALTER TABLE `unify`.`Musica`
60   MODIFY COLUMN `streamings_numero` BIGINT;
61
62 • SET SQL_MODE=@OLD_SQL_MODE;
63 • SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
64 • SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

4.3. INSERÇÃO DE DADOS



The screenshot shows a MySQL Workbench interface with a query editor window. The code is a script for inserting data into three tables: Artista, Genero, and Musica. The Artista table has columns nome_artista, pais, and streamings_mensal. The Genero table has columns nome_genero and descricao. The Musica table has columns nome_musica, streamings_numero, duracao, and id_album. The script uses the 'unify' database and includes comments indicating the start of each table's data insertion.

```
1 • USE `unify`;
2
3 • INSERT INTO `unify`.`Artista` (nome_artista, pais, streamings_mensal) VALUES
4     ('Pink Floyd', 'Inglaterra', 245000000),
5     ('Charli XCX', 'Inglaterra', 327000000),
6     ('Raul Seixas', 'Brasil', 32000000),
7     ('Linkin Park', 'Estados Unidos', 534000000);
8
9 • INSERT INTO `unify`.`Genero` (nome_genero, descricao) VALUES
10    ('Rock', 'Rock clássico, alternativo e pop rock'),
11    ('Pop', 'Música popular comercial e hits'),
12    ('Heavy Metal', 'Numetal e Death Metal'),
13    ('MPB', 'Música Popular Brasileira clássica');
14
15 • INSERT INTO `unify`.`Album` (nome_album, ano_lancamento, id_artista, id_genero) VALUES
16    ('The Wall', 1979, 1, 1),
17    ('BRAT', 2024, 2, 2),
18    ('"Kring-Ha, Bandolo"', 1973, 3, 4),
19    ('Meteora', 2003, 4, 3);
20
21 • INSERT INTO `unify`.`Musica` (nome_musica, streamings_numero, duracao, id_album) VALUES
22    ('Comfortably Numb', 788019458, '00:06:22', 1),
23    ('365', 245568747, '00:03:22', 2),
24    ('Metamorfose Ambulante', 132144934, '00:03:50', 3),
25    ('Numb', 2514936885, '00:03:06', 4);
26
```

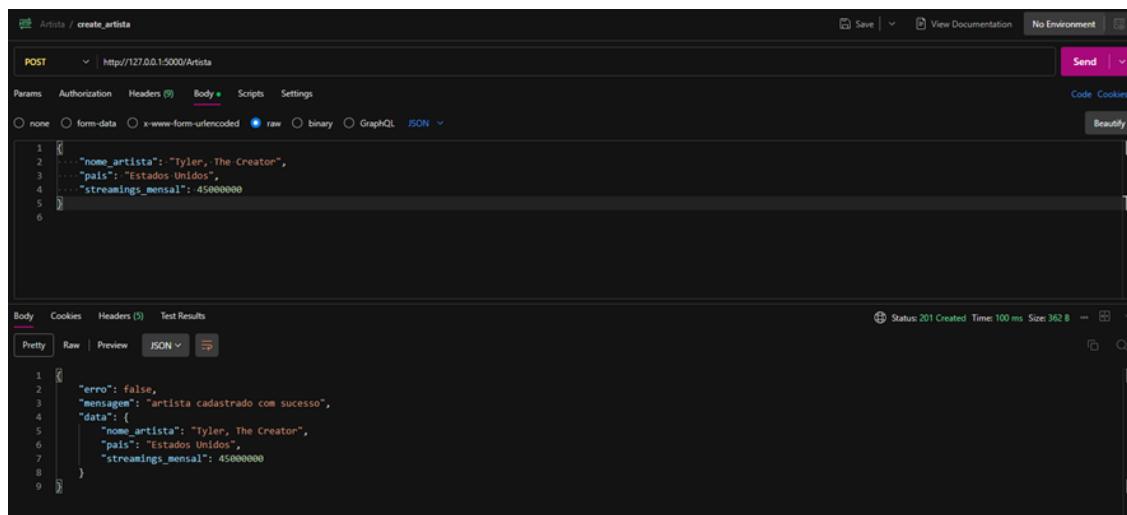
6. API EM PYTHON

Na segunda parte que foi solicitado, foi utilizado a linguagem Python para produzir uma API utilizando as operações CRUD (Create, Read, Update, Delete) com o serviço de streaming produzido no mySQL, foi utilizado o Postman para ser realizado as requisições.

A seguir, prints tirados para visualizar os comandos feitos juntamente com as tabelas utilizadas.

6.1. TABELA 'ARTISTA'

6.1.1. Create



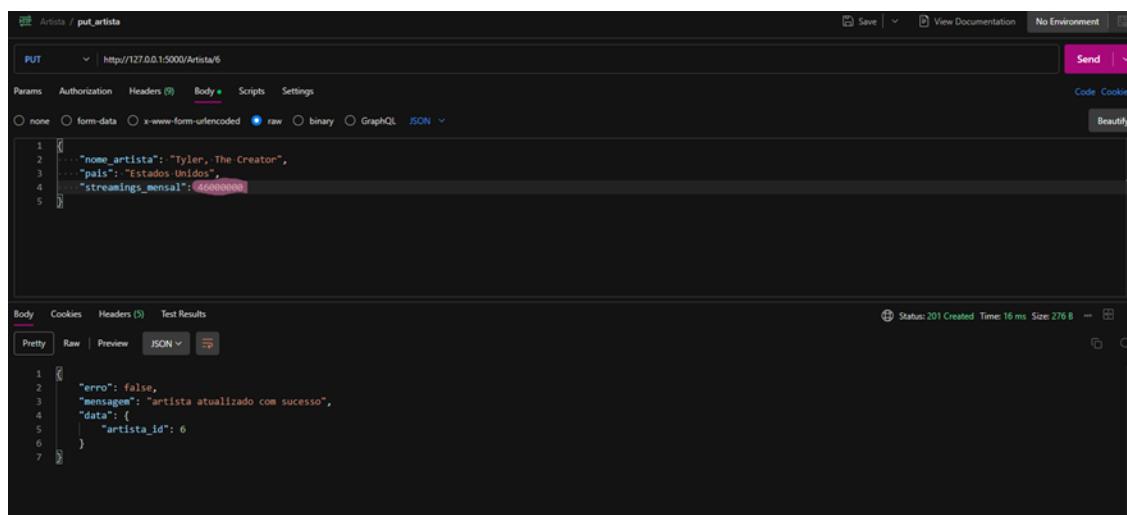
Postman screenshot showing a POST request to create an artist. The URL is `http://127.0.0.1:5000/Artista`. The request body is:

```
1 {
2     "nome_artista": "Tyler, The Creator",
3     "pais": "Estados Unidos",
4     "streamings_mensal": 45000000
5 }
```

The response status is 201 Created, with a response body:

```
1 {
2     "erro": false,
3     "mensagem": "artista cadastrado com sucesso",
4     "data": {
5         "nome_artista": "Tyler, The Creator",
6         "pais": "Estados Unidos",
7         "streamings_mensal": 45000000
8     }
9 }
```

6.1.2. Update



Postman screenshot showing a PUT request to update an artist. The URL is `http://127.0.0.1:5000/Artista/6`. The request body is:

```
1 {
2     "nome_artista": "Tyler, The Creator",
3     "pais": "Estados Unidos",
4     "streamings_mensal": 46000000
5 }
```

The response status is 201 Created, with a response body:

```
1 {
2     "erro": false,
3     "mensagem": "artista atualizado com sucesso",
4     "data": {
5         "artista_id": 6
6     }
7 }
```

6.1.3. Read

The screenshot shows the Postman interface with a successful response from the endpoint `/Artista`. The response body is a JSON object containing an array of artist data:

```
1 {
2     "erro": false,
3     "mensagem": "Artista listados com sucesso",
4     "data": [
5         {
6             "id": 1,
7             "nome_artista": "Pink Floyd",
8             "pais": "Inglaterra",
9             "streamings_mensal": 245000000
10        },
11        {
12            "id": 2,
13            "nome_artista": "Charli XCX",
14            "pais": "Inglaterra",
15            "streamings_mensal": 327000000
16        },
17        {
18            "id": 3,
19            "nome_artista": "Raul Seixas",
20            "pais": "Brasil",
21            "streamings_mensal": 32000000
22        },
23        {
24            "id": 4,
25            "nome_artista": "Linkin Park",
26            "pais": "Estados Unidos",
27            "streamings_mensal": 534000000
28        },
29        {
30            "id": 5,
31            "nome_artista": "Lata Mangeshkar",
32            "pais": "Índia",
33            "streamings_mensal": 460000000
34        },
35        {
36            "id": 6,
37            "nome_artista": "Tyler, The Creator",
38            "pais": "Estados Unidos",
39            "streamings_mensal": 460000000
40        },
41        {
42            "id": 7,
43            "nome_artista": "Deftones",
44            "pais": "Estados Unidos",
45            "streamings_mensal": 16000000
46        }
47    ]
48 }
```

Below the main interface, there is a warning message: `WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.`

The screenshot shows the Postman interface with a successful response from the endpoint `/Artista`. The response body is a JSON object containing an array of artist data, identical to the one in the previous screenshot:

```
1 {
2     "erro": false,
3     "mensagem": "Artista listados com sucesso",
4     "data": [
5         {
6             "id": 1,
7             "nome_artista": "Pink Floyd",
8             "pais": "Inglaterra",
9             "streamings_mensal": 245000000
10        },
11        {
12            "id": 2,
13            "nome_artista": "Charli XCX",
14            "pais": "Inglaterra",
15            "streamings_mensal": 327000000
16        },
17        {
18            "id": 3,
19            "nome_artista": "Raul Seixas",
20            "pais": "Brasil",
21            "streamings_mensal": 32000000
22        },
23        {
24            "id": 4,
25            "nome_artista": "Linkin Park",
26            "pais": "Estados Unidos",
27            "streamings_mensal": 534000000
28        },
29        {
30            "id": 5,
31            "nome_artista": "Lata Mangeshkar",
32            "pais": "Índia",
33            "streamings_mensal": 460000000
34        },
35        {
36            "id": 6,
37            "nome_artista": "Tyler, The Creator",
38            "pais": "Estados Unidos",
39            "streamings_mensal": 460000000
40        },
41        {
42            "id": 7,
43            "nome_artista": "Deftones",
44            "pais": "Estados Unidos",
45            "streamings_mensal": 16000000
46        }
47    ]
48 }
```

6.1.4. Delete

The screenshot shows the Postman interface with a successful response from the endpoint `/Artista/7` using a `DELETE` method. The response body is a JSON object indicating the success of the deletion:

```
1 {
2     "erro": false,
3     "mensagem": "Artista excluido com sucesso",
4     "data": {
5         "artista_id": 7
6     }
7 }
```

```

26     "pais": "Estados Unidos",
27   },
28   {
29     "id": 6,
30     "nome_artista": "Tyler, The Creator",
31     "pais": "Estados Unidos",
32     "streamings_mensal": 46000000
33   }
34 ]
35

```

6.2. TABELA 'GÊNERO'

6.2.1. Create

```

1 {
2   "error": false,
3   "mensagem": "genero cadastrado com sucesso",
4   "data": {
5     "nome_genero": "Hip hop",
6     "descricao": "Ritmo ritmico e rimas faladas."
7   }
8 }
9

```

6.2.2. Update

Genero / put_genero

PUT | http://127.0.0.1:5000/Genero/5

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "nome_genero": "Hip Hop",
3   "descricao": "Ritmo rítmico e rimas faladas."
4 }

```

Send | Code Cookies Beautify

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

```

1 {
2   "erro": false,
3   "mensagem": "genero atualizado com sucesso",
4   "data": {
5     "genero_id": 5
6   }
7 }

```

Status 201 Created Time: 11 ms Size: 274 B

6.2.3. Read

Genero / get_genero

GET | http://127.0.0.1:5000/Genero

Params Authorization Headers (7) Body Scripts Settings

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

```

1 {
2   "erro": false,
3   "mensagem": "generos listados com sucesso",
4   "data": [
5     {
6       "id": 1,
7       "nome_genero": "Rock",
8       "descricao": "Rock clássico, alternativo e pop rock"
9     },
10    {
11      "id": 2,
12      "nome_genero": "Pop",
13      "descricao": "Música popular comercial e hits"
14    },
15    {
16      "id": 3,
17      "nome_genero": "Heavy Metal",
18      "descricao": "Numetal e Death Metal"
19    },
20    {
21      "id": 4,
22      "nome_genero": "MPB",
23      "descricao": "Música Popular Brasileira clássica"
24    },
25    {
26      ...
27    }
28  ]
29 }

```

Status 200 OK Time: 8 ms Size: 925 B

6.2.4. Delete

DELETE | http://127.0.0.1:5000/Genero/6

Params Authorization Headers (7) Body Scripts Settings

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

```
1 [ { "erro": false, "mensagem": "genero excluido com sucesso", "data": { "genero_id": 6 } } ]
```

Status: 201 Created Time: 14 ms Size: 272 B

6.3. TABELA 'ÁLBUM'

6.3.1. Create

POST | http://127.0.0.1:5000/Album

Params Authorization Headers (9) Body * Scripts Settings

None form-data x-www-form-urlencoded raw binary GraphQL JSON

```
[ { "nome_album": "Chromakopia", "ano_lancamento": 2024, "id_artista": 6, "id_genero": 5 } ]
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

```
1 [ { "erro": false, "mensagem": "album lancado com sucesso", "data": { "nome_album": "Chromakopia", "ano_lancamento": 2024, "id_artista": 6, "id_genero": 5 } } ]
```

Status: 201 Created Time: 6 ms Size: 352 B

6.3.2. Update

The screenshot shows a POSTMAN interface with the following details:

- Method:** PUT
- URL:** <http://127.0.0.1:5000/Album/5>
- Body:** Raw JSON (Pretty)

```
1 {
2     "nome_album": "True Romance",
3     "ano_lancamento": 2011,
4     "id_artista": 2,
5     "id_genero": 2
6 }
```

- Response Status:** 201 Created
- Response Time:** 34 ms
- Response Size:** 272 B

The response body is:

```
1 {
2     "erro": false,
3     "mensagem": "album atualizado com sucesso",
4     "data": [
5         {
6             "album_id": 5
7         }
8     ]
9 }
```

6.3.3. Read

The screenshot shows a GET request in POSTMAN to list albums.

- Method:** GET
- URL:** <http://127.0.0.1:5000/Album>
- Body:** Raw JSON (Pretty)

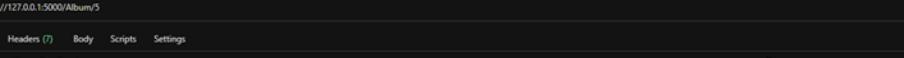
```
1 {
2     "erro": false,
3     "mensagem": "album listados com sucesso",
4     "data": [
5         {
6             "id": 1,
7             "nome_album": "The Dark Side Of The Moon",
8             "ano_lancamento": 1973,
9             "id_artista": 1,
10            "id_genero": 1
11        },
12        {
13            "id": 2,
14            "nome_album": "BRAT",
15            "ano_lancamento": 2024,
16            "id_artista": 2,
17            "id_genero": 2
18        },
19        {
20            "id": 3,
21            "nome_album": "\\'Kring-Ha, Bandolo\\",
22            "ano_lancamento": 1973,
23            "id_artista": 3,
24            "id_genero": 4
25        }
26    ]
27 }
```

- Response Status:** 200 OK
- Response Time:** 11 ms
- Response Size:** 1.06 KB

The screenshot shows the Postman application interface. At the top, there's a header bar with 'GET' selected, the URL 'http://127.0.0.1:5000/Album', and a 'Send' button. Below the header are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Scripts', and 'Settings'. The 'Body' tab is currently active, showing a 'Pretty' representation of the JSON response. The response body contains data for three albums, each with fields: id, nome_album, ano_lancamento, id_artista, and id_genero. The albums are: 1. Meteora (id: 4), 2. True Romance (id: 5), and 3. Chromakopia (id: 6). Each album has a specific combination of artist and genre IDs. The bottom of the screen shows navigation links like 'PROBLEMAS', 'SAÍDA', 'CONSOLE DE DEPURAÇÃO', 'TERMINAL', 'PORTAS', and 'POSTMAN CONSOLE'. On the right side, there are status indicators for 'Status 200 OK', 'Time: 11 ms', 'Size: 1.06 KB', and a copy icon.

```
24 |     "id_genero": 4
25 |   },
26 |   {
27 |     "id": 4,
28 |     "nome_album": "Meteora",
29 |     "ano_lancamento": 2003,
30 |     "id_artista": 4,
31 |     "id_genero": 3
32 |   },
33 |   {
34 |     "id": 5,
35 |     "nome_album": "True Romance",
36 |     "ano_lancamento": 2011,
37 |     "id_artista": 2,
38 |     "id_genero": 2
39 |   },
40 |   {
41 |     "id": 6,
42 |     "nome_album": "Chromakopia",
43 |     "ano_lancamento": 2024,
44 |     "id_artista": 6,
45 |     "id_genero": 5
46 |   }
47 | }
```

6.3.4. Delete



Album / del_album

DELETE http://127.0.0.1:5000/Album/5

Params Authorization Headers (7) Body Scripts Settings

Body Cookies Headers (5) Test Results

Pretty Raw | Preview JSON ↻

1 {
2 "erro": false,
3 "message": "album excluido com sucesso",
4 "data": {
5 "album_id": 5
6 }
7 }

Status: 201 Created Time: 16 ms Size: 269 B

The screenshot shows the Postman application interface. At the top, there's a header bar with 'GET' selected, the URL 'http://127.0.0.1:5000/Album', and a 'Send' button. Below the header are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Scripts', and 'Settings'. The 'Body' tab is currently active, showing a JSON response with 41 numbered lines. The response contains three objects representing albums, each with fields like id, nome_album, ano_lancamento, id_artista, and id_genero. To the right of the body, status information is displayed: 'Status: 200 OK Time: 10 ms Size: 944 B'. At the bottom, there are navigation buttons for 'PROBLEMAS', 'SAÍDA', 'CONSOLE DE DEPURAÇÃO', 'TERMINAL', 'PORTAS', and 'POSTMAN CONSOLE'. On the far right, there are icons for Python, Node.js, Java, and Go.

```
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
```

```
[{"id": 1, "nome_album": "Kring-Ha, Bandolo", "ano_lancamento": 1973, "id_artista": 3, "id_genero": 2}, {"id": 4, "nome_album": "Meteora", "ano_lancamento": 2003, "id_artista": 4, "id_genero": 3}, {"id": 6, "nome_album": "Chromakopia", "ano_lancamento": 2024, "id_artista": 6, "id_genero": 5}]
```

6.4. TABELA ‘MÚSICA’

6.4.1. Create

POST | http://127.0.0.1:5000/Musica

Params Authorization Headers (9) Body Scripts Settings

Body

```
1 {
2     "nome_musica": "St. Chroma (feat. Daniel Caesar)",
3     "streamings_numero": 291000000,
4     "duracao": "00:03:17",
5     "id_album": 6
6 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

```
1 [
2     {
3         "erro": false,
4         "mensagem": "musica lancada com sucesso",
5         "data": {
6             "nome_musica": "St. Chroma (feat. Daniel Caesar)",
7             "streamings_numero": 291000000,
8             "duracao": "00:03:17",
9             "id_album": 6
10        }
11    }
12 ]
```

Status: 201 Created Time: 17 ms Size: 368 B

6.4.2. Update

PUT | http://127.0.0.1:5000/Musica/10

Params Authorization Headers (9) Body Scripts Settings

Body

```
1 {
2     "nome_musica": "Chromakopia",
3     "streamings_numero": 291274061,
4     "duracao": "00:03:17",
5     "id_album": 6
6 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

```
1 [
2     {
3         "erro": false,
4         "mensagem": "musica atualizada com sucesso",
5         "data": {
6             "musica_id": 10
7         }
8     }
9 ]
```

6.4.3. Read

GET http://127.0.0.1:5000/Musica

Params Authorization Headers (7) Body Scripts Settings

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON ↻

```
1  [
2      "erro": false,
3      "mensagem": "musicas listadas com sucesso",
4      "data": [
5          {
6              "id": 5,
7              "nome_musica": "Comfortably Numb",
8              "streamings_numero": 788019458,
9              "duracao": "0:06:22",
10             "id_album": 1
11         },
12         {
13             "id": 6,
14             "nome_musica": "365",
15             "streamings_numero": 245568747,
16             "duracao": "0:03:22",
17             "id_album": 2
18         },
19         {
20             "id": 7,
21             "nome_musica": "Metamorfose Ambulante",
22             "streamings_numero": 132144934,
23             "duracao": "0:03:50",
24             "id_album": 3
25         },
26         {
27             "id": 8,
28             "nome_musica": "Numb",
29             "streamings_numero": 2514936885,
30             "duracao": "0:03:06",
31             "id_album": 4
32         },
33         {
34             "id": 10,
35             "nome_musica": "Chromakopia",
36             "streamings_numero": 291274661,
37             "duracao": "0:03:17",
38             "id_album": 6
39     }
```

Status: 200 OK Time: 7 ms Size: 1000 B ⌂ ↻

6.4.4. Delete

HTTP Musica / del_musica

DELETE http://127.0.0.1:5000/Musica/8

Params Authorization Headers (7) Body Scripts Settings

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON ↻

```
1  [
2      "erro": false,
3      "mensagem": "musica excluida com sucesso",
4      "data": [
5          {
6              "musica_id": 8
7      }]
]
```

7. REPOSITÓRIO GITHUB