

Pembaruan ModulArtikel ini diperbarui pada tanggal **20 Desember 2021**. Pembaruan terakhir adalah:

Pembaruan kriteria submission

[Lihat riwayat ↗](#)

Kriteria OpenMusic API versi 1

Terdapat 5 kriteria utama yang harus Anda penuhi dalam membuat proyek OpenMusic API.

Kriteria 1 : Pengelolaan Data Album

API harus menyediakan endpoint untuk pengelolaan album dengan spesifikasi berikut:

Endpoint	Body Request	Response	Keterangan
POST /albums	<ul style="list-style-type: none"> • name : string • year : number 	status code: 201 (created) body: <ul style="list-style-type: none"> • status: "success" • data: <ul style="list-style-type: none"> ◦ albumId: "album_id" 	Menambahkan album.
GET /albums/{id}	-	status code: 200 (ok) body: <ul style="list-style-type: none"> • status: "success" • data: <ul style="list-style-type: none"> ◦ album: album 	Mendapatkan album berdasarkan id.
PUT /albums/{id}	<ul style="list-style-type: none"> • name : string • year : number 	status code: 200 (ok) body: <ul style="list-style-type: none"> • status: "success" • message: *any 	Mengubah album berdasarkan id album.
DELETE /albums/{id}	-	status code: 200 (ok) body: <ul style="list-style-type: none"> • status: "success" • message: *any 	Menghapus album berdasarkan id.

*any: merupakan nilai string apa pun selama tidak kosong.

Untuk lebih jelasnya, berikut adalah struktur response body yang harus ditampilkan pada endpoint:

- GET /albums{id}

```

1. {
2.   "status": "success",
3.   "data": {
4.     "album": {
5.       "id": "album-Mk8AnmCp210PwT6B",
6.       "name": "Viva la Vida",
7.       "year": 2008
8.     }
9.   }
10. }
```

Kriteria 2 : Pengelolaan Data Song

API harus menyediakan endpoint untuk pengelolaan song (lagu) dengan spesifikasi berikut:

Endpoint	Body Request	Response	Keterangan
POST /songs	<ul style="list-style-type: none"> • title : string • year : number • genre: string • performer: string • duration: number? • albumId: string? 	status code: 201 (created) body: <ul style="list-style-type: none"> • status: "success" • data: <ul style="list-style-type: none"> ◦ songId: "song_id" 	Menambahkan lagu
GET /songs	-	status code: 200 (ok) body: <ul style="list-style-type: none"> • status: "success" • data: <ul style="list-style-type: none"> ◦ songs: songs 	Mendapatkan seluruh lagu
GET /songs/{id}	-	status code: 200 (ok) body: <ul style="list-style-type: none"> • status: "success" • data: <ul style="list-style-type: none"> ◦ song: song 	Mendapatkan lagu berdasarkan id
PUT /songs/{id}	<ul style="list-style-type: none"> • title : string • year : number • genre: string • performer: string • duration: number? • albumId: string? 	status code: 200 (ok) body: <ul style="list-style-type: none"> • status: "success" • message: *any 	Mengubah lagu berdasarkan id lagu
DELETE /songs/{id}	-	status code: 200 (ok) body: <ul style="list-style-type: none"> • status: "success" • message: *any 	Menghapus lagu berdasarkan id

*?: Boleh null atau undefined.

*any: merupakan nilai string apa pun selama nilainya tidak kosong.

Untuk lebih jelasnya, berikut adalah struktur response body yang harus ditampilkan pada endpoint:

- GET /songs

```

1. {
2.   "status": "success",
3.   "data": {
4.     "songs": [
5.       {
6.         "id": "song-Qbax5Oy7L8WKf741",
7.         "title": "Life in Technicolor",
8.         "performer": "Coldplay"
9.       },
10.      {
11.        "id": "song-poax5Oy7L8WKllqw",
12.        "title": "Centimetres of London",
13.        "performer": "Coldplay"
14.      },
15.      {

```

- GET /songs/{id}

```

1. {
2.   "status": "success",
3.   "data": {
4.     "song": {
5.       "id": "song-Qbax5Oy7L8WKf741",
6.       "title": "Life in Technicolor",
7.       "year": 2008

```

```
..          "year": 2008,
8.        "performer": "Coldplay",
9.        "genre": "Indie",
10.       "duration": 120,
11.       "albumId": "album-Mk8AnmCp210PwT6B"
12.     }
13.   }
14. }
```

Objek song yang disimpan harus memiliki struktur seperti contoh di bawah ini:

```
1. {
2.   "id": "song-Qbax5Oy7L8WKf741",
3.   "title": "Life in Technicolor",
4.   "year": 2008,
5.   "performer": "Coldplay",
6.   "genre": "Indie",
7.   "duration": 120,
8.   "albumId": "album-Mk8AnmCp210PwT6B"
9. }
```

Kriteria 3 : Menerapkan Data Validation

Wajib menerapkan proses Data Validation pada Request Payload sesuai spesifikasi berikut:

- POST /albums
 - **name** : string, required.
 - **year** : number, required.
- PUT /albums
 - **name** : string, required.
 - **year** : number, required.
- POST /songs
 - **title** : string, required.
 - **year** : number, required.
 - **genre** : string, required.
 - **performer** : string, required.
 - **duration** : number.
 - **albumId**: string.
- PUT /songs
 - **title** : string, required.
 - **year** : number, required.
 - **genre** : string, required.
 - **performer** : string, required.
 - **duration** : number.
 - **albumId**: string.

Kriteria 4 : Penanganan Eror (Error Handling)

- Ketika proses validasi data pada request payload tidak sesuai (gagal), server harus mengembalikan response:
 - status code: **400 (Bad Request)**
 - response body:

- status: **fail**
 - message: <apa pun selama tidak kosong>
- Ketika pengguna mengakses resource yang tidak ditemukan, server harus mengembalikan response:
 - status code: 404 (Not Found)
 - response body:
 - status: **fail**
 - message: <apa pun selama tidak kosong>
- Ketika terjadi server eror, server harus mengembalikan response:
 - status code: 500 (Internal Server Error)
 - response body:
 - status: **error**
 - message: <apa pun selama tidak kosong>

Kriteria 5 : Menggunakan Database dalam Menyimpan Data album dan lagu

- Data lagu harus disimpan di dalam database menggunakan PostgreSQL agar ketika di-restart data tidak akan hilang.
- Wajib menggunakan teknik migrations dalam mengelola struktur tabel pada database.
- Wajib menyimpan nilai host, port, maupun kredensial dalam mengakses database pada environment variable dengan ketentuan:
 - **PGUSER** : menyimpan nilai user untuk mengakses database.
 - **PGPASSWORD** : menyimpan nilai password dari user database.
 - **PGDATABASE** : menyimpan nilai nama database yang digunakan.
 - **PGHOST** : menyimpan nilai host yang digunakan oleh database.
 - **PGPORT** : menyimpan nilai port yang digunakan oleh database.
- Wajib menggunakan package [dotenv](#) serta berkas .env dalam mengelola environment variable.

Kriteria Opsional OpenMusic API versi 1

Selain kriteria utama, terdapat kriteria opsional yang dapat Anda penuhi agar mendapat nilai yang baik.

Kriteria 1: Memunculkan daftar lagu di dalam detail album

API harus memunculkan daftar lagu di dalam album pada endpoint **GET /albums/{albumId}**. Berikut contoh response yang harus dihasilkan:

```

1. {
2.   "status": "success",
3.   "data": {
4.     "album": {
5.       "id": "album-Mk8AnmCp210PwT6B",
6.       "name": "Viva la Vida",
7.       "year": 2008,
8.       "songs": [
9.         {
10.           "id": "song-Qbax5Oy7L8WKf741",
11.           "title": "Life in Technicolor",
12.           "performer": "Coldplay"
13.         },
14.         {

```

```
15. "id": "song-poax50y7L8WK11qw",
```

Kriteria 2: Query Parameter untuk Pencarian Lagu

Menerapkan query parameter pada endpoint GET /songs untuk fitur pencarian lagu. Berikut ketentuan parameternya:

- **?title**: mencari lagu berdasarkan judul lagu.
- **?performer**: mencari lagu berdasarkan performer.

Catatan: Penggunaan kedua parameter tersebut dapat dikombinasikan.

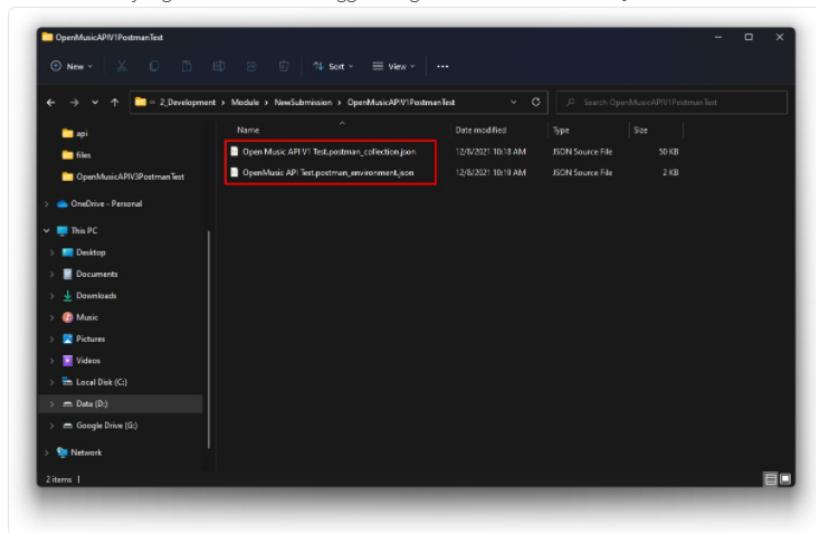
Pengujian API

Ketika membangun OpenMusic API versi 1, tentu Anda perlu menguji untuk memastikan API berjalan sesuai dengan kriteria yang ada. Kami sudah menyediakan berkas Postman Collection dan Environment yang dapat Anda gunakan untuk pengujian. Silakan unduh berkasnya pada tautan berikut:

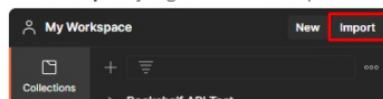
- [Postman OpenMusic API V1 Test Collection dan Environment](#)

Cara Import Collection dan Environment OpenMusic V1

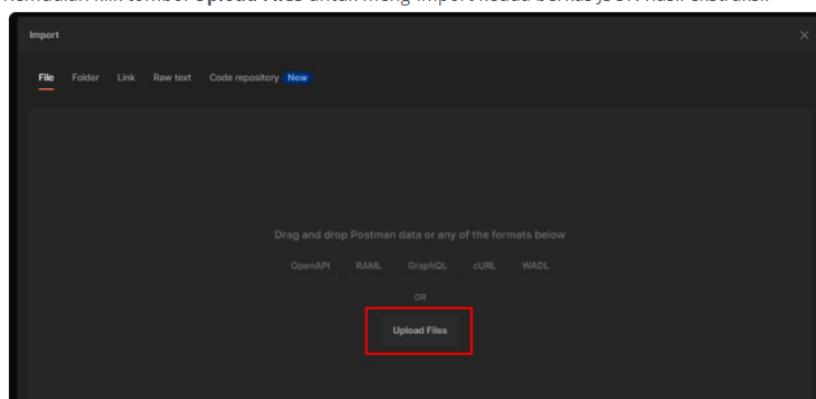
1. Silakan unduh tautan yang diberikan di atas.
2. Ekstrak berkas yang sudah diunduh hingga menghasilkan dua berkas file JSON.



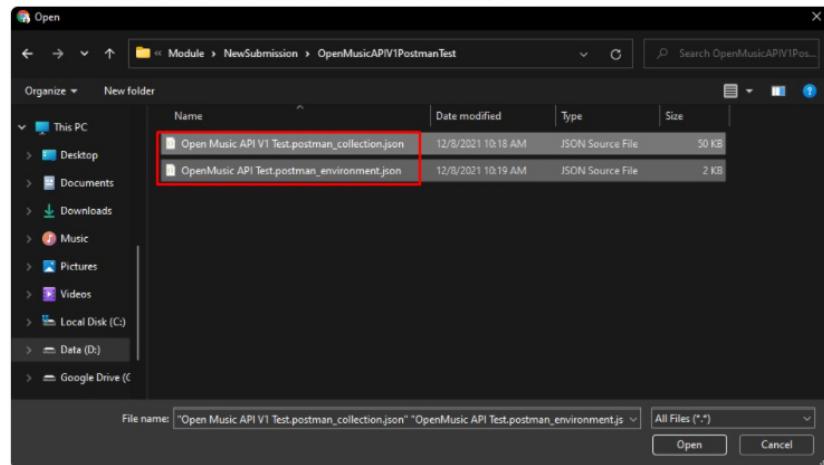
3. Kemudian import kedua berkas tersebut pada Postman. Caranya, buka aplikasi Postman, klik tombol **Import** yang berada di atas panel kiri aplikasi Postman.



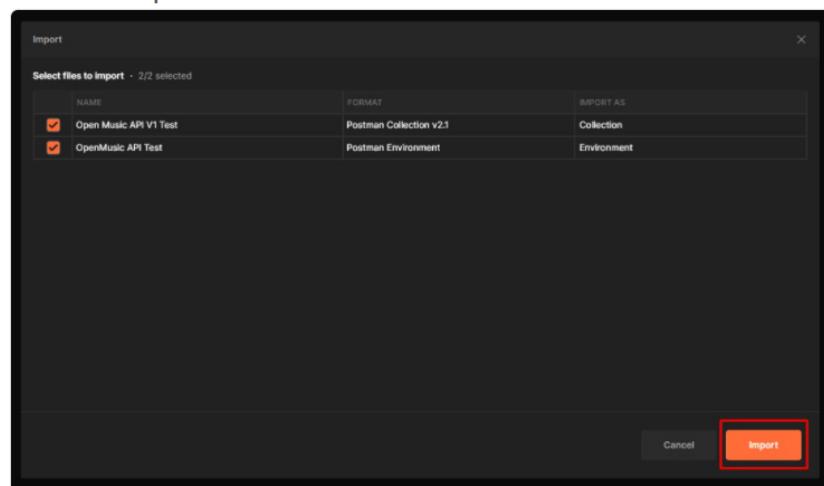
4. Kemudian klik tombol **Upload Files** untuk meng-import kedua berkas JSON hasil ekstraksi.



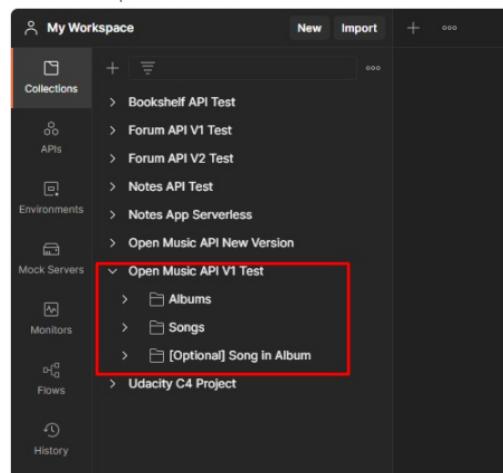
5. Pilih kedua berkas JSON hasil ekstraksi dan klik tombol **Open**.



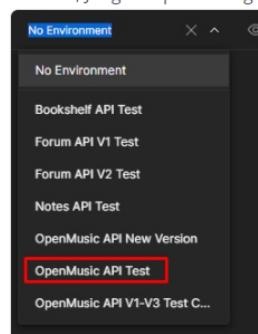
6. Kemudian klik **Import**.



7. Setelah itu, OpenMusic API V1 Test Collection dan Environment akan tersedia pada Postman Anda.

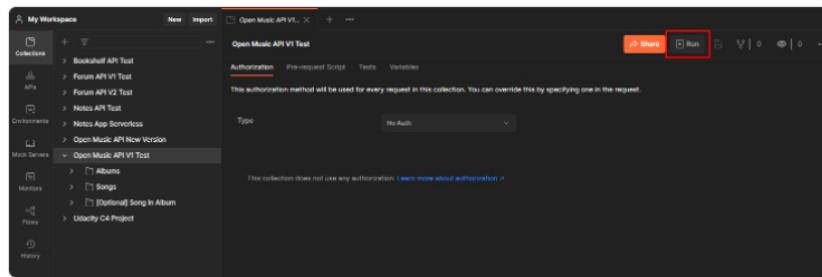


8. Terakhir, Jangan lupa untuk gunakan Environment **OpenMusic API Test**.



Tips Menjalankan Pengujian OpenMusic V1

- Pastikan Anda selalu menjalankan pengujian secara berurutan. Karena beberapa request nilai yang didapat dari request sebelumnya. Contoh, pengujian Get Detail Album with Valid Id membutuhkan Add Album with Valid Payload. Karena untuk melihat detail album, harus ada album yang dibuat/dimasukkan terlebih dahulu.
- Untuk menjalankan request secara berurutan sekaligus, Anda bisa memanfaatkan fitur collection runner.



- Jika merasa seluruh fitur yang dibangun sudah benar namun pengujinya selalu gagal, kemungkinan database Anda kotor dengan data pengujian yang Anda lakukan sebelum-sebelumnya, dan itu bisa menjadi salah satu penyebab pengujian selalu gagal. Solusinya, silakan hapus seluruh data pada tabel melalui psql dengan perintah:

```
1. truncate songs, albums;
```

Kriteria Penilaian Submission

Submission Anda akan dinilai oleh Reviewer guna menentukan kelulusan Anda. Untuk lulus dari kelas ini, proyek OpenMusic API versi 1 harus memenuhi seluruh pengujian otomatis pada seluruh Postman Request. Bila salah satu pengujinya gagal, maka proyek Anda akan kami tolak.

Submission Anda akan dinilai oleh Reviewer dengan skala 1-5. Untuk mendapatkan nilai tinggi, Anda bisa menerapkan beberapa saran berikut:

- Memenuhi kriteria opsional yang diberikan.
- Menggunakan ESLint dan salah satu style guide agar gaya penulisan kode JavaScript lebih konsisten dan menghindari disable linter yang tidak diperlukan.
- Menuliskan kode dengan bersih:
 - Menghapus kode yang redundant (tidak perlu).
 - Menghapus impor kode yang tidak digunakan.
 - Menghapus berkas yang tidak digunakan.

Berikut adalah detail penilaian submission:

- **Bintang 1** : Semua ketentuan wajib terpenuhi, namun terdapat indikasi kekurangan dalam mengerjakan submission.
- **Bintang 2** : Semua ketentuan wajib terpenuhi, namun terdapat kekurangan pada penulisan kode. Seperti tidak menerapkan modularization atau gaya penulisan tidak konsisten.
- **Bintang 3** : Semua ketentuan wajib terpenuhi, namun tidak terdapat improvisasi atau persyaratan opsional yang dipenuhi.
- **Bintang 4** : Semua ketentuan wajib terpenuhi dan menerapkan minimal dua saran di atas.
- **Bintang 5** : Semua ketentuan wajib terpenuhi dan menerapkan seluruh saran di atas.

Catatan:

Jika submission Anda ditolak maka tidak ada penilaian. Kriteria penilaian bintang di atas hanya berlaku jika submission Anda diterima.

Ketentuan Berkas Submission

- Berkas submission yang dikirim merupakan folder proyek dari OpenMusic API versi 1 dalam bentuk **ZIP**.
- Pastikan di dalam folder proyek yang Anda kirim terdapat berkas **package.json**.
- Untuk menjaga kredensial, Anda diperbolehkan untuk tidak melampirkan berkas **.env** selama penamaan variable environment sesuai dengan persyaratan.
- Pastikan Anda hapus dulu berkas **node_modules** pada folder proyek sebelum mengkompresi dalam bentuk ZIP.

Submission Anda akan Ditolak bila

- Kriteria wajib OpenMusic API versi 1 tidak terpenuhi.
- Ketentuan berkas submission tidak terpenuhi.
- Menggunakan bahasa pemrograman dan teknologi lain, selain JavaScript dan Node.js.
- Menggunakan Framework Node.js selain Hapi Framework.
- Melakukan kecurangan seperti tindakan plagiasi.

Forum Diskusi

Jika mengalami kesulitan, Anda bisa menanyakan langsung ke forum diskusi.

<https://www.dicoding.com/academies/271/discussions>.

