



美術館記録アプリ

B4-15-26 小林紹子

January 14, 2026

目次

- 1 題材の説明
- 2 CRUD 操作
- 3 GraphQL Schema の設計
- 4 REST と GraphQL の比較・検討・考察
- 5 REST と GraphQL の適性

目次

- 1 題材の説明
- 2 CRUD 操作
- 3 GraphQL Schema の設計
- 4 REST と GraphQL の比較・検討・考察
- 5 REST と GraphQL の適性

題材の説明

- 美術館情報を管理・閲覧できる Web アプリ
- 主な機能：
 - 美術館・博物館の一覧表示
 - 展覧会の一覧表示
 - 詳細ページで情報確認
 - 登録・編集・削除
- データベース：SQLite
- 使用技術：
 - フレームワーク：Next.js
 - ORM：Prisma

目次

1 題材の説明

2 CRUD 操作

3 GraphQL Schema の設計

4 REST と GraphQL の比較・検討・考察

5 REST と GraphQL の適性

REST API の設計 I

Next.js の App Router の規則よりファイル構造がそのままパスになるため `/api/rest/` ができてしまった.

- GET リクエスト

- `/api/rest/museums` \implies 美術館一覧取得
- `/api/rest/museums/:id` \implies 該当する美術館の情報取得
- `/api/rest/exhibitions` \implies 展覧会一覧取得
- `/api/rest/exhibitions/:id` \implies 該当する展覧会の情報取得

- POST リクエスト

- `/api/rest/museums` \implies 美術館を登録
- `/api/rest/exhibitions` \implies 展覧会を登録

REST API の設計 II

- PUT リクエスト

- `/api/rest/museums/:id` \implies 該当する美術館の情報更新
- `/api/rest/exhibitions/:id` \implies 該当する展覧会の情報更新

- DELETE リクエスト

- `/api/rest/museums/:id` \implies 該当する美術館の情報削除
- `/api/rest/exhibitions/:id` \implies 該当する展覧会の情報削除

目次

- 1 題材の説明
- 2 CRUD 操作
- 3 GraphQL Schema の設計
- 4 REST と GraphQL の比較・検討・考察
- 5 REST と GraphQL の適性


```
model Museum {  
  id          Int          @id @default(autoincrement())  
  name        String  
  address     String  
  officialUrl String  
  description String?  
  exhibitions Exhibition[]  
}
```

Figure: Museum テーブル

```
model Exhibition {  
  id          Int          @id @default(autoincrement())  
  title        String  
  startDate    DateTime  
  endDate      DateTime  
  officialUrl  String  
  description  String?  
  museumId     Int  
  museum       Museum     @relation(fields: [museumId], reference  
}
```

Figure: Exhibition テーブル

Museum (1) \iff Exhibition(多)

GraphQL 設計 (Schema)

```
type Museum {  
  id: Int!  
  name: String!  
  address: String!  
  officialUrl: String!  
  description: String  
  exhibitions: [Exhibition!]!  
}  
  
type Exhibition {  
  id: Int!  
  title: String!  
  startDate: String!  
  endDate: String!  
  officialUrl: String!  
  description: String  
  museum: Museum!  
}
```

Figure: Schema 設計

各クエリ (一部)

エンドポイント: '/api/graphql'

```
query {  
  exhibitions(limit:5) {  
    id  
    title  
    startDate  
    endDate  
    museum {  
      name  
    }  
  }  
}
```

Figure: 最新登録展覧会

```
query {  
  exhibitions {  
    id  
    title  
  }  
}
```

Figure: 展覧会一覧

```
query {  
  exhibition(id: ${id}) {  
    id  
    title  
    startDate  
    endDate  
    officialUrl  
    description  
    museum {  
      id  
      name  
    }  
  }  
}
```

目次

- 1 題材の説明
- 2 CRUD 操作
- 3 GraphQL Schema の設計
- 4 REST と GraphQL の比較・検討・考察
- 5 REST と GraphQL の適性

最新登録展覧会（5件）

- 綾錦 近代西陣が認めた染織の美

開催期間：2025/12/20 ～2026/2/1

開催美術館：取得されていません

展覧会の開催場所がほしい。

アンダーフェッチ (TOP ページにおいて) II

```
▼ 0: Object { id: 14, title: "綾錦 近代西陣が認めた染織の美", startDate: 1766188800000, ... }  
  description: ""  
  endDate: 1769904000000  
  id: 14  
  museumId: 5  
  officialUrl: "https://www.nezu-muse.or.jp/jp/exhibition/index.html"  
  startDate: 1766188800000  
  title: "綾錦 近代西陣が認めた染織の美"
```

Figure: REST の場合

- ⇒ museumId までは得られるが name (美術館名) までは取得できない.
- ⇒ name を得るには GET (/api/rest/museums/4) などする必要あり.
- ⇒ **N+1 問題の発生**

アンダーフェッチ (TOP ページにおいて) III

```
query {  
  exhibitions(limit:5) {  
    id  
    title  
    startDate  
    endDate  
    museum {  
      name  
    }  
  }  
}
```

Figure: クエリ

⇒ アンダーフェッチの解消

```
▼ 1: Object { id: 13, title: "オルセー美術館所蔵 いまを生きる歓び", startDate: "1794614400000",  
  endDate: "1806192000000"  
  id: 13  
  ► museum: Object { name: "東京都美術館" }  
  startDate: "1794614400000"  
  title: "オルセー美術館所蔵 いまを生きる歓び"
```

Figure: GraphQL の場合

オーバーフェッチ（展覧会の一覧表示）！

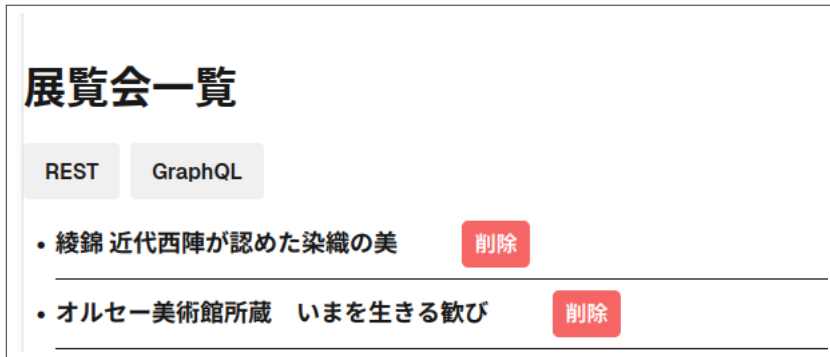


Figure: 展覧会一覧表示

必要なもの：Exhibition.id と Exhibition.title のみ

オーバーフェッチ（展覧会の一覧表示）II

```
▼ 0: Object { id: 14, title: "綾錦 近代西陣が認めた染織の美", startDate: 1766188800000, ... }  
  description: ""  
  endDate: 1769904000000  
  id: 14  
  museumId: 5  
  officialUrl: "https://www.nezu-muse.or.jp/jp/exhibition/index.html"  
  startDate: 1766188800000  
  title: "綾錦 近代西陣が認めた染織の美"  
  ▶ <prototype>: Object { ... }
```

Figure: REST の場合

⇒ オーバフェッチの発生

オーバーフェッチ（展覧会の一覧表示）III

```
▶ 0: Object { id: 14, title: "綾錦 近代西陣が認めた染織の美" }  
▶ 1: Object { id: 13, title: "オルセー美術館所蔵 いまを生きる歓び" }  
▶ 2: Object { id: 12, title: "スウェーデン絵画 北欧の光、日常のかがやき" }  
▶ 3: Object { id: 11, title: "ゴッホ展 家族がつないだ画家の夢" }  
▶ 4: Object { id: 10, title: "テート美術館 ターナー展——崇高の絵画、現代美術との対話" }  
▶ 5: Object { id: 9, title: "版画家レンブラント 挑戦、継承、インパクト" }  
▶ 6: Object { id: 8, title: "北斎 富嶽三十六景 井内コレクションより" }  
▶ 7: Object { id: 7, title: "チュルリョーニス展 内なる星図" }  
▶ 8: Object { id: 6, title: "オルセー美術館所蔵 印象派一室内をめぐる物語" }
```

Figure: GraphQL の場合

⇒ オーバフェッチの解消

REST と GraphQL の比較（展覧会一覧表示画面）

比較項目	REST	GraphQL
呼び出し回数	3	1
レスポンス量	7.97KB	1.10KB
サイズ	12.01KB	739Byte

Table: システム変更前後のパフォーマンス比較

⇒ 他のページでも一覧取得では同じような差がでた.

考察：REST と GraphQL の比較

- 小規模構成における有意差

- わずか 2 テーブルの構成でも、回数・通信量ともに顕著な差を確認.
- これは、GraphQL のクエリ最適化が最小構成から有効であることを示す.

- 実運用の場合

- テーブル数やリレーションが増加する実環境では、この差はさらに顕著になると推測される.

- 結論

- 特定フィールドの抽出や複数リソースの統合において,GraphQL は極めて高い通信効率を持つ.

目次

- 1 題材の説明
- 2 CRUD 操作
- 3 GraphQL Schema の設計
- 4 REST と GraphQL の比較・検討・考察
- 5 REST と GraphQL の適性

本アプリにおける技術選定

結論：GraphQL の採用が適している

美術館アプリのような複雑なデータ関連性を持つシステムでは、**GraphQL の採用が通信効率および開発効率の両面で優位**.

- **リソースの統合:**

将来的に作品詳細画面で「作者」や「関連作品」を一度に取得できるため、REST 特有の「何度もリクエストを送る手間」を排除.

- **通信量の最適化:**

必要なフィールドを絞り込める GraphQL は、ユーザーの通信制限や端末負荷を軽減.

- **将来性:**

今後テーブル数や機能が増加しても、今回の実験結果が示す通り、REST に比べてパフォーマンスの悪化を最小限に抑制.