Annexe 1 - API GraphQL

Pour le laboratoire, nous avons mis en place un endpoint https://mobile.iict.ch/graphql offrant un service web au format *GraphQL*. Il s'agit d'une implémentation relativement basique mais celle-ci vous permettra de comprendre les principes de base de cette technologie.

La Fig. 1 présente la structure des données de tests utilisées pour cet exemple. Il s'agit d'une liste d'auteurs liés à des livres, nous avons environ 9'200 auteurs et 11'100 livres.

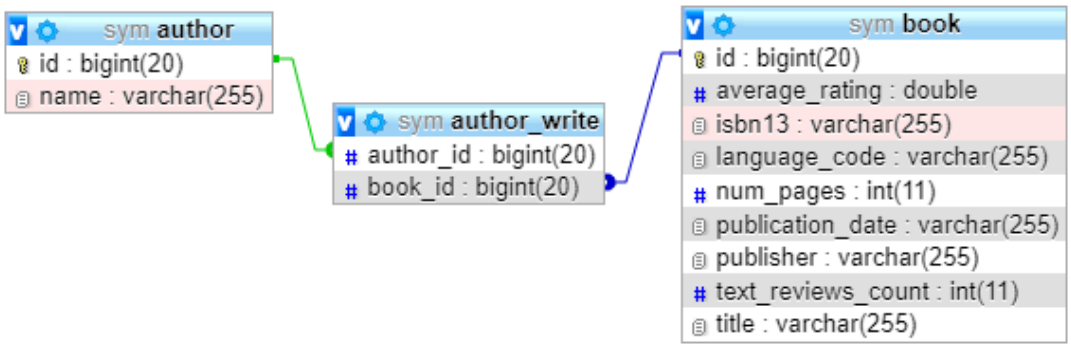

Figure 1 – Schéma de la base de données

Envoi d'une requête

Toutes les interactions avec le service *GraphQL* se réalisent à partir du *endpoint* unique, celui-ci s'attend à recevoir la requête *GraphQL* dans le corps d'une requête http *POST*, au format *json*. Il répondra directement dans le corps de la réponse au format *json* également. La Fig. 2 présente un exemple réalisé avec *POSTMAN*. Vous noterez la possibilité de préciser dans la requête *GraphQL* les champs dont on souhaite recevoir les données, et que le service permet de résoudre les relations entre les 2 tables et de récupérer les livres associés à un auteur, en précisant aussi les champs nécessaires.

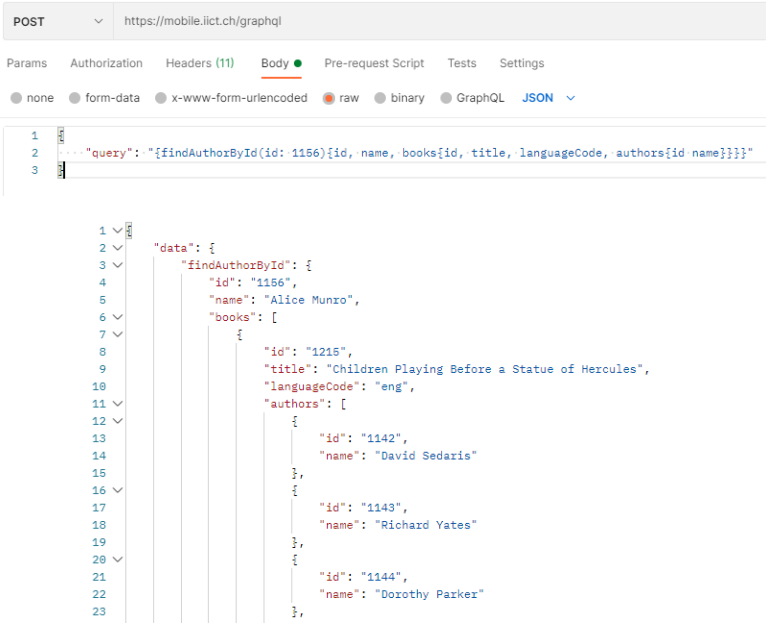

Figure 2 – Exemple d'une requête GraphQL (en haut) et la réponse associée (en bas)

Format des données

```
type Author {
    id: ID!
    name: String!
    books: [Book]
}

type Book {
    id: ID!
    title: String!
    isbn13: String
    languageCode: String
    numPages: Int
    publicationDate: String
    publisher: String
    textReviewsCount: Int
    averageRating: Float
    authors: [Author]
}
```

Méthodes disponibles

```
findAllAuthors: [Author]!
findAuthorById(id: ID!): Author
countAuthors: Int!
findAllBooks: [Book]!
findBookById(id: ID!): Book
countBooks: Int!
```

(cf. exemple sur Fig. 2)