RELATIONAL DATA - BASIC

import { GraphQLServer } from 'graphql-yoga';

*//DummyUserData*

const users = [

  {

    id: '1',

    name: 'Mucahid',

    email: 'mucahidyazar@gmail.com',

    age: 28

  },

  {

    id: '2',

    name: 'Sarah',

    email: 'sarah@gmail.com',

    age: 20

  },

  {

    id: '3',

    name: 'Mike',

    email: 'mike@gmail.com',

    age: 45

  },

]

*//DummyPostData*

const posts = [

  {

    id: '11',

    title: 'Javascript',

    body: 'Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo qui iste a quam eveniet pariatur!',

    published: false,

    author: '1'

  },

  {

    id: '12',

    title: 'HTML',

    body: 'Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo qui iste a quam eveniet pariatur!',

    published: true,

    author: '1'

  },

  {

    id: '13',

    title: 'CSS',

    body: 'Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo qui iste a quam eveniet pariatur!',

    published: true,

    author: '2'

  }

]

*//DummyCommentData*

const comments = [

  {

    id: '102',

    text: 'Lorem ipsum dolor sit amet consectetur adipisicing elit.',

    author: '1',

    post: '11'

  },

  {

    id: '103',

    text: 'Lorem ipsum dolor sit amet consectetur adipisicing elit.',

    author: '2',

    post: '11'

  },

  {

    id: '104',

    text: 'Lorem ipsum dolor sit amet consectetur adipisicing elit.',

    author: '2',

    post: '12'

  },

  {

    id: '105',

    text: 'Lorem ipsum dolor sit amet consectetur adipisicing elit.',

    author: '3',

    post: '13'

  }

]

const typeDefs = `

  type Query {

    users(query: String): [User!]!

    posts(query: String): [Post!]!

    comments: [Comment!]!

    me: User!

    post: Post!

  }

  type User {

    id: ID!

    name: String!

    email: String!

    age: Int

    posts: [Post!]!

    comments: [Comment!]!

  }

  type Post {

    id: ID!

    title: String!

    body: String!

    published: Boolean!

    author: User!

    comments: [Comment!]!

  }

  type Comment {

    id: ID!

    text: String!

    author: User!

    post: Post!

  }

`

const resolvers = {

  Query: {

    users: (parent, args, ctx, info) => {

      if(!args.query) {

        return users;

      }

      return users.filter((user) => {

        return user.name.toLowerCase().includes(args.query.toLowerCase())

      })

    },

    posts: (parent, args, ctx, info) => {

      if(!args.query) {

        return posts;

      }

      return posts.filter(post => {

        const isTitleMatch = post.title.toLowerCase().includes(args.query.toLowerCase());

        const isBodyMatch = post.body.toLowerCase().includes(args.query.toLowerCase());

        return isTitleMatch || isBodyMatch;

      })

    },

    comments: (parent, args, ctx, info) => {

      return comments

    },

    me: () => ({

      id: 'abc123',

      name: 'Mucahid Yazar',

      email: 'mucahidyazar@gmail.com',

      age: 28

    }),

    post: () => ({

      id: '000001',

      title: 'GraphQL 101',

      body: '',

      published: false,

    })

  },

*###Eger alanlarimizdan birisi yukaridaki gibi SCALAR type degilse CUSTOM TYPE icinde bu alanlarimizi belirlememiz lazim.*

*###Yani author mesela bir obje donecegi icin bize asagidaki gibi Custom type tanimlanip icinde donecek ama id title body ise string yani scalar type dondugu icin yukarida tanimlanabilir.*

*###Aslinda author posts FIIELDS inin icinde yukarida fakat orada bulamayinca buraya gelecek ve buraya geldiginde her bir post sirasiyla gelecek.*

*###Ve buradaki parent herbir post'un sirasiyla yerini alacak. ve partent ile asagidaki gibi post.author post.id post.body. post.title post.published gibi yerlerini alabilir olacagiz.*

  Post: {

    author: (parent, args, ctx, info) => {

      return users.find(user => {

        return user.id === parent.author

      })

    },

    comments: (parent, args, ctx, info) => comments.filter(comment => comment.post === parent.id)

  },

  User: {

    posts: (parent, args, ctx, info) => posts.filter(post => post.author === parent.id),

    comments: (parent, args, ctx, info) => comments.filter(comment => comment.author === parent.id)

  },

  Comment: {

    author: (parent, args, ctx, info) => users.find(user => user.id === parent.author),

    post: (parent, args, ctx, info) => posts.find(post => post.id === parent.post)

  }

}

const server = new GraphQLServer({

  typeDefs,

  resolvers

});

server.start(() => {

  console.log('The server is up!');

});

HINTS

! exclamatıon mark devamlı sonuna geldiği şeyi alacağımızı söyler.

String! === Yani devamlı bir string döneceğini söyler bize. Eğer ! işareti olmazsa null geri alabiliriz çağırdıgımızda.

[User!]! === Yanı devamlı olarak User arrayı dondurecegını sart koymus oluyoruz null donmemesı ıcın. Icindeki userdada ! oldugu icin asla null donmeyecektir yani ici bos olmayacaktirda.