

Web Yazılım Geliştirme

Ders 12 - GraphQL

Erciyes Üniversitesi
Bilgisayar Mühendisliği Bölümü

Eğitmen: Ömür ŞAHİN

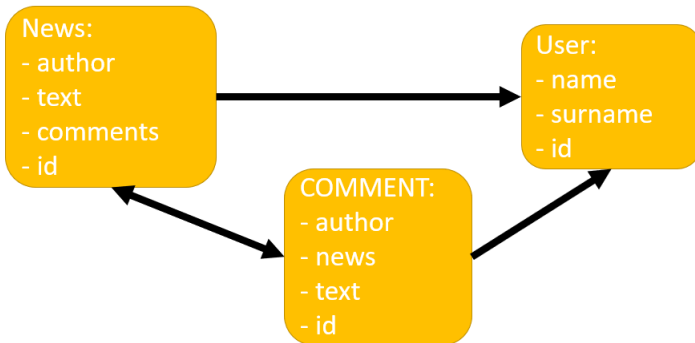
- GraphQL web servis geliştirilmesi ve kullanılmasını anlamak
- REST ve GraphQL arasındaki farklılıkları anlamak

Graph Query Language (GraphQL)

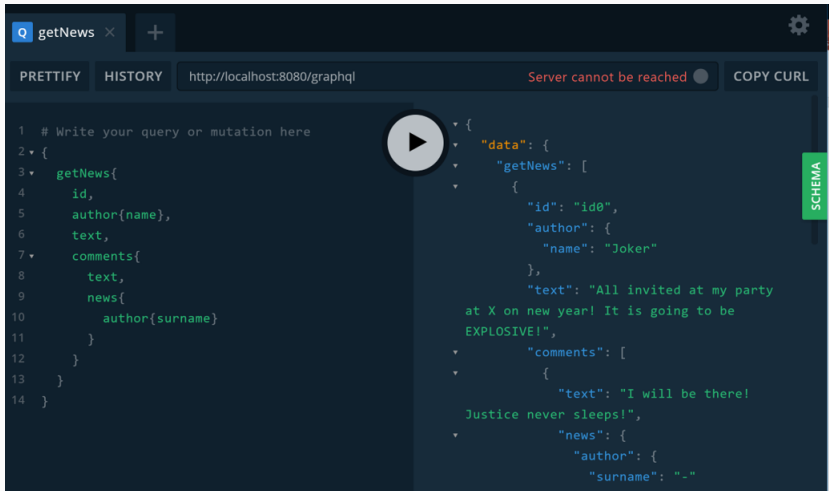
- Facebook tarafından 2012 yılında geliştirilmiş 2015 yılında yayınlanmıştır.
- Bir API'nin belli bir sorgu diliyle sorgulanabileceğini tanımlamak için kullanılan protokoldür.
- GraphQL web servisler HTTP üzerinde çalışmaktadır. Sorgular HTTP mesajının bir parçası olarak gönderilir.
- GraphQL, HTTP dışında da kullanılabilir.

API verisi yönlü bir graftır

- Örnek: Yorumlarla birlikte bir forum girdisi ve kullanıcı bilgileri
- Backend'de veriler SQL veritabanında saklanabilir.



GraphQL Sorguları



The screenshot shows the GraphQL Playground interface. At the top, there's a search bar with 'getNews' and a plus icon. Below it are tabs for 'PRETTIFY', 'HISTORY', and a URL bar showing 'http://localhost:8080/graphql'. To the right of the URL bar is a red error message 'Server cannot be reached' and a 'COPY CURL' button. The main area is split into two panes. The left pane contains a query template with line numbers 1 to 14. The right pane shows the JSON response for the query. A play button icon is in the center of the split view. On the far right, there's a green 'SCHEMA' button.

```
1 # Write your query or mutation here
2 {
3   getNews{
4     id,
5     author{name},
6     text,
7     comments{
8       text,
9       news{
10         author{surname}
11       }
12     }
13   }
14 }
```

```
{
  "data": {
    "getNews": [
      {
        "id": "id0",
        "author": {
          "name": "Joker"
        },
        "text": "All invited at my party
at X on new year! It is going to be
EXPLOSIVE!",
        "comments": [
          {
            "text": "I will be there!
Justice never sleeps!",
            "news": {
              "author": {
                "surname": "-"
              }
            }
          }
        ]
      }
    ]
  }
}
```

- Bir başlangıç noktasına ihtiyaç vardır.
 - Örnek: `getNews`
- Hangi alanlara ihtiyaç duyulduğu belirtilmelidir.
 - Örnek: `id`, `author`, `text`, `comments`
- Eğer alan graftaki bir başka türü belirtiyorsa onun da alanlarını belirtmek gerekmektedir.
 - Örnek: `author` tipinin `name` alanı

```
{  
  getNews{  
    id,  
    author{name},  
    text,  
    comments{  
      text,  
      news{ author{surname}}  
    }  
  }  
}
```

- Burada comment comment listesini çekmektedir.
- author için önemli not: Aynı örneği iki sefer çekmektedir ancak farklı alanlarını.
 - Örnek: name ve surname alanları.
 - `news.author === news.comments[i].news.author`
- Graflarda bir sorgu nodelarla bağlantılıdır ve derinlik istenildiği kadar ileri gidebilir.

```
{  
  getNews{  
    id,  
    author{name},  
    text,  
    comments{  
      text,  
      news{ author{surname}}  
    }  
  }  
}
```

Response

- Bir JSON verisi döner.
- Payload data alanının içindedir.
- Payload ile sorgu yapısı birbirine benzemektedir.
- Eğer bir hata alınırsa data alanı null olacak ve errors alanında hata ile ilgili bilgiler gelecektir.

```
{
  "data": {
    "getNews": [
      {
        "id": "id0",
        "author": {
          "name": "Joker"
        },
        "text": "All invited at my party at X on new
is going to be EXPLOSIVE!",
        "comments": [
          {
            "text": "I will be there! Justice never sleep
            "news": {
              "author": {
                "surname": "-"
              }
            }
          }
        ]
      }
    ]
  }
}
// other posts...
```


- Bir veriyi değiştirmek için, GraphQL'de "mutation" adı verilen operatörler bulunmaktadır.
- Bunlar birer Remote Procedure Calls (RPC)'dir ve pek çok şeyi arkaplanda gerçekleştirir.
- Faydaları: Oldukça esnektir ve istediğiniz her şeyi yapabilirsiniz.
- Dezavantajları: Oldukça esnektir ve her bir API farklı davranış gösterebilir.

- POST veya GET ile erişim sağlanabilir.
- Örnek: POST localhost/graphql
 - JSON payload: `{"query": "{all{id}}"}"`
 - Burada sorgu query değişkeninde string olarak tutulmaktadır.
- Örnek: GET
localhost/graphql?query=%7Ball%7Bname%7D%7D
 - Burada sorgu URL query parametresi olarak gönderilmektedir.
 - `{}` karakterleri escape karakterleri ile değiştirilmiştir.

- GET idempotent operatörken POST değildir.
- Mutation'lar ise sunucunun durumunu değiştirdiği için GET ile gönderilmemelidir.
 - GraphQL HTTP Servis bu durumda muhtemelen bir exception fırlatacaktır.
- Yani "mutation"lar POST ile yapılırken okuma eylemleri hem POST hem de GET ile yapılabilir.

- Facebook yalnızca REST'i kullanmak yerine neden bir başka tipte web servis oluşturmuştur?
- İstemci hangi veriyi çekmek istediği üzerinde tamamen kontrol sahibidir.
 - İhtiyaç duymadığı alanları çekmeyebilir.
 - Bütün ihtiyaç duyduğu verileri tek bir HTTP çağrısı ile çekebilir.
 - Veri ve enerji tüketimini düşürmenin önemli olduğu mobil cihazlar için oldukça önemlidir.
- Aynı şeyleri REST ile de yapabiliriz ancak GraphQL'in manuel olarak yeniden uygulanması ile sonuçlanacaktır.

- Veritabanları ile birlikte daha zor uygulanabilirdir.
 - Var olan kütüphanelerle gerçekleştirilebilir ancak hala sunucu tarafında yüksek performans elde edilememektedir.
 - Veritabanına dayalı optimize edilmiş sorguların nasıl oluşturulacağı problem yaratmaktadır.
 - REST ile yüksek performanslı endpointler oluşturulabilir.
- "mutation" için servisler arasında ortak bir semantik bulunmamaktadır.
 - Böylece, her bir yeni servis için kodun ne yapmaya çalıştığı ile ilgili dokümanların okunması gerekmektedir. REST'deki POST/PUT'un tipik davranışından oldukça farklı davranışlar sergiliyor olabilir.
- Authentication, versiyonlama ve önbellekleme için native bir çözüm bulunmamaktadır.

- Nispeten yeni bir teknolojidir ve araçların geliştirilmesi gerekmektedir.
 - JavaScript ile iyi çalışmaktadır ancak diğer dillerde iyi bir destek bulunmamaktadır.
 - Her geçen gün daha iyiye gitmektedir.
- Rekürsif ilişkilerde bir sınırlama bulunmamaktadır.
 - Örnek: Yorumların içindeki yorumları çekebilirsiniz.

- GraphQL REST'in yerini alabilir.
- Ancak bunu söylemek için oldukça erken.
- İstemci tarafı için daha iyi olsa da sunucu tarafı için kötüdür.
- Mutation'ın dezavantajları bulunmaktadır.

- Apollo isimli kütüphane ile kullanılabilir. Eklendiği zaman `"/graphql"` endpointini ekler.
- `"Accept:*/*"` veya JSON bir istekte bulunulduğunda JSON bir cevap gelecektir.
- Ancak `"Accept:text/html"` bir istek gönderilirse web uygulaması gelecektir.