GraphQL vs REST-API – En jämförelse

Robin Blixter
r.blixter89@gmail.com

Introduktion

Bakgrunden till denna studie är ett uppdrag för Intersport Sverige som driver en stor e-handel: https://www.intersport.se/. De vill öppna upp nya möjligheter för sina utvecklare genom att implementera ett nytt API (Application Programming Interface)[1]. Deras webbplats består av dynamisk data som hämtas direkt från deras nuvarande API som är ett REST-API (Representation State Transfer)[2]. Min roll blir att bygga ett nytt API med dem moderna teknikerna GraphQL, Apollo[3], Node[4] och Express[5].

GraphQL är ett frågespråk (query-language), som kan hämta data från olika källor (REST-API, databaser, text-filer eller andra tjänster som ElasticSearch)[6]. Det skapades av Facebook 2012 och släpptes som öppen källkod 2015. Enligt Facebook så skapades GraphQL på grund av att de såg brister i sitt nuvarande REST-API främst för de mobila applikationerna där de inte behövde lika mycket data som de fick från deras end-points, antingen var de tvungna att skapa ett nytt API för de mobila enheterna eller hitta en bättre lösning – den lösningen blev GraphQL.[7] En stor fördel med GraphQL jämfört med REST-API är att klienten bara får den data som efterfrågas.

Fokusområden

I mitt arbete skall jag implementera ett GraphQL-API med teknikerna: Node.js, Typescript, Express.js och ApolloServer. All data skall hämtas från tjänsten ElasticSearch som Intersport använder sig av för att hämta sin data från databasen. ElasticSearch har en egen klient som går att installera får Node.js, det är via den klienten jag kommer hämta alla data som används i mitt GraphQL-API. Jag kommer sedan att göra jämförelser kring GraphQL och REST.

Områden att fokusera på:

- Over-fetching – Hämtar för mycket data, får data som inte används.

- Under-fetching – Hämtar för lite data, får inte all data från en end-point och blir tvungen till att göra ytterligare API-förfrågningar.

- Användarvänlighet/Dokumentation – Hur ser utvecklarna som jobbar på klient-sidan på att arbeta med API:er på detta vis istället? Är det lätt att förstå GraphQL's automatiska dokumentation? Jämför med REST-API.

- Implementering av GraphQL – tekniker som används och svårigheter. Vad krävs det för förkunskaper?

Frågeställningar att besvara i studien:

1. Vid användning av GraphQL, hur många färre API-förfrågningar skickas iväg från klient-sidan? Arbetet skall ta reda på hur många färre end-points som behövs.

2. Vid användning av GraphQL, hur många rader färre JSON-data får klienten tillbaka vid en API-förfrågan? Arbetet skall ta reda på hur många rader JSON-data som kan plockas bort från klient-sidan.

3. Vilka erfarenheter kan göras av att implementera GraphQL? Arbetet skall ta reda på om det finns rekommenderade tekniker och om det krävs speciella förkunskaper. Samt om det är komplicerat att koppla ihop GraphQL med ElasticSearch.

Mål med studien:

Få fram tydliga skillnader mellan dessa tekniker och lista fördelar/nackdelar med respektive teknik. I vilka lägen passar GraphQL bättre och när passar REST bättre?

Avgränsa:

I min studie kommer jag att fokusera på att skapa ett API som hämtar data som kan visas upp. Jag kommer bara att fokusera på att skapa ett API för Intersports produkter och inte de övriga tabellerna från databasen. Det blir bara R:et i CRUD (Create **Read** Update Delete), att läsa in innehållet. Jag kommer inte skapa ny data, uppdatera befintlig data eller radera. Jag kommer bara bygga ett API och inte koppla samman mitt API med en front-end klient.

Samla, analysera, värdera

Min analys kommer handla om att främst jämföra hur mycket data som webbplatsens klienter hämtar från sitt nuvarande API vid varje förfrågan och hur mycket av den data som faktiskt används. Samt hur många API-förfrågningar det krävs för att få all den data som behövs, görs det flera förfrågningar? I båda av dessa fall så använder nätverket mer bandbredd än vad som egentligen behövs. Dessa problem går att optimera med GraphQL.

För att analysera och hämta data från GraphQL kommer jag använda den inbyggda ”GraphQL Playground”[8], där jag även hittar den automatiserade API-dokumentationen.

Motiv och Värde

Studien kommer kunna hjälpa utvecklare som vill se jämförelser mellan REST-API och GraphQL-API samt hjälpa till att visa hur man kan göra för att implementera GraphQL och koppla samman med en annan datakälla, ElasticSearch i mitt fall.

Min kund vill i framtiden bygga sin webbplats mer statisk med tekniker som Gatsby.JS, vilket kan bli problematiskt med deras nuvarande API som innehåller mycket logik och där det inte går att styra hur mycket eller lite data som skall hämtas. Med ett nytt API där deras utvecklare själva kan justera den data som hämtas, blir det enklare att skapa statiska webbsidor, som i sin tur skapar en snabbare webbplats.

Ifall Intersport Sverige vill skapa en ny webbplats för mobila enheter eller en applikation med ett eget API, som inte hämtar samma data som deras nuvarande API. Då kan de istället använda sig av GraphQL där front-end utvecklarna själva styr över vilken data som skall hämtas och inte behöver oroa sig för ”overfetching”. Det är lätt att lägga till ny data i GraphQL utan att förstöra eller tvinga utvecklarna att skapa en ny API-version, som är problematiskt med REST-API.

Litteratursökning

För att hitta litteratur kommer jag främst använda mig av BTHs egna biblioteksverktyg som kallas för Summon, där det går att hitta litteratur från hela världen. Jag kommer även använda mig av Googles version som heter Google Scholar. Mina sökfraser består främst av `GraphQL' och `REST API`.

Ur en ren akademisk synvinkel så finns det inte mycket att hitta på GraphQL, förmodligen eftersom det fortfarande är en relativt ny teknik. Jag hittar en konferenshandling `Generating GraphQL-Wrappers for REST(-like) APIs`[9], som går igenom hur man kan "wrappa" sina nuvarande REST-API:er i GraphQL. Fokus ligger att använda och jämföra olika tekniker för att "wrappa" dessa befintliga API:er i GraphQL. I denna handling finns det mycket fakta kring GraphQL som jag kommer använda mig av.

Jag hittade ytterligare en konferenshandling som studerade ett liknande ämne: `Experiences on Migrating RESTful Web Services to GraphQL`[10]. I detta arbete går de igenom de utmaningar som finns för att migrera från REST-API till GraphQL.

Konferenshandlingen `Migrating to GraphQL: A Practical Assessment`[11] är ett arbete utfört av `Federal University of Minas Gerais`. De hade intressanta slutsatser om litteraturmaterial inom detta ämne, och hade samma slutsats som mig, att det inte finns många vetenskapliga källor, därför beslöt dig sig för att gå igenom grå litteratur (bloggar, tutorials och liknande webb-artiklar) och filtrerade sedan ut sådant som inte var trovärdigt. De gick igenom över 1200 artiklar och hamnade till slut på 28 trovärdiga artiklar som analyserades. Dessa 28 artiklar fick sedan svara på frågan vad som var fördelarna och nackdelarna med GraphQL, resultatet av detta användes sedan i deras forskningsfrågor. De frågorna som de fick fram är liknande frågor som jag har valt att svara på i mitt arbete:

`*When using GraphQL, what is the reduction in the number of API calls performed by clients?*`

och

`*When using GraphQL, what is the reduction in the number of fields of the JSON documents returned by servers?*`.

Referenser

1 https://en.wikipedia.org/wiki/Application_programming_interface
2 https://en.wikipedia.org/wiki/Representational_state_transfer
3 https://www.apollographql.com/docs/apollo-server/
4 https://nodejs.org/en/
5 https://expressjs.com/
6 https://graphql.org/
7 https://engineering.fb.com/core-data/graphql-a-data-query-language/
8 https://www.apollographql.com/docs/apollo-server/testing/graphql-playground/
9 Wittern, E., Cha, A. & Laredo, J.A. 2018, "Generating GraphQL-wrappers for REST(-like) APIs", , pp. 65.
10 Vogel, M., Weber, S. & Zirpins, C. 2018, "Experiences on Migrating RESTful Web Services to GraphQL", , pp. 283.
11 Brito, G., Mombach, T. & Valente, M.T. 2019, "Migrating to GraphQL: A Practical Assessment", IEEE, , pp. 140.