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.
De vill öppna upp nya möjligheter för sina utvecklare genom att implementera ett nytt API (Application Programming Interface)[11]. Deras webbplats består av dynamisk data som hämtas direkt från deras nuvarande API som är ett REST-API (Representation State Transfer)[10]. Min roll blir att bygga ett nytt API med dem moderna teknikerna GraphQL, Apollo[1], Node[7] och Express[5].

GraphQL

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[3]. En stor fördel med GraphQL jämfört med REST-API är att klienten bara får den data som efterfrågas.

ElasticSearch

ElasticSearch är en opensource tjänst som hämtar all typ av data från olika källor. Elasticsearch är byggt på Apache Lucene. Fördelen med ElasticSearch är att det är lätt att skapa REST-apier, som är snabba och framför allt skalbara. ElasticSearch tar emot rådata och denna data indexeras in i ElasticSearch, sedan kan användarna hämta data genom att ställa komplexa frågor. [4]

FORSKNINGSFRÅGOR

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.

Fokusområden

- 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?
- Hållbarhet - Genom att minska datan som skickas från server till klient så minskas strömförbrukningen.

Forskningsfrågor

1. Vid användning av GraphQL, hur många färre API-förfrågningar skickas iväg från klient-sidan? Arbetet skall ta reda på hur många färre endpoints 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. Går det att få ner sitt koldioxidutsläpp genom att byta till GraphQL? Detta skall undersökas i mitt arbete.
4. Hur ser andra utvecklare på GraphQL, en enkätstudie kommer skickas ut till andra utvecklare som kommer få svara på frågor kring GraphQL. Där studien kommer svara på hur populärt GraphQL är bland dessa utvecklare.

I slutänden kommer jag att presentera en rapport över mitt arbete. Där jag går igenom hur jag har lyckats besvara forskningsfrågorna.

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.

METOD

- **Överflödig data (Forskningsfråga 1 och 2)**: Till en början kommer jag gå igenom vilka REST end-points som används på klient-sidan för att få fram all produktdata. Sedan skall jag gå igenom hela JSON-dokumentet och se vilken data som inte används på klienten - vilka rader som är överflödiga och skulle kunna plockas bort i GraphQL.
- **Storlek/Nedladdningstid (Forskningsfråga 2 och 3)**: För att jämföra storlek och nedladdningstid på den data som skickas till klienten från servern vid förfrågan, kommer jag använda programmet Postman. Det finns stöd för GraphQL och vanlig REST i detta program. Jag kan spara svaren från servern och sedan gå igenom all data för att göra jämförelser.
- **Enkätundersökning (Forskningsfråga 4)**: I och med att REST är standardsättet för att skapa API och GraphQL en ny teknik så kommer jag att skicka ut enkäter till andra utvecklare för att få med i min studie hur många av dessa utvecklare som har kännedom om GraphQL. Hur många av utvecklarna som har använt REST respektive GraphQL, samt vilken teknik de hade valt idag.

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‘[12] , 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‘[9]. 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‘[2] 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?”.

Vid undersökning av hållbarhet på nätet hittade jag ett blogginlägg från Danny Van Kooten, där han undersöker möjligheten kring att minska sitt koldioxidfotavtryck genom att reducera storleken på sina webbplatser. Enligt honom så lyckades han med detta genom att ta bort 20kb från sitt Wordpress-plugin: “*Just last week I reduced global emissions by an estimated 59.000 kg CO2 per month by removing a 20 kB JavaScript dependency in Mailchimp for WordPress. There’s no way I can have that kind of effect in other areas of my life.*“[8]

PLANERING

- Projektet på Intersport skall vara klart senast den 31/3.
- Enkäterna för enkätundersökningen skall vara färdiga senast den 1/3 och börja skickas ut till utvecklare på olika sociala medier.
- Enkäterna skall räknas ihop senast den 1/5.
- Den första rapporten skall vara klar senast den 4/5.

REFERENCES

[1] Apollo. Introduction, 2020. URL `https://www.apollographql.com/docs/apollo-server/`. Accessed 2020-02-14.

[2] G. Brito, T. Mombach, and M. T. Valente. Migrating to graphql: A practical assessment. pages 140–150. IEEE, 2019.

[3] L. Byron. Graphql: A data query language, 2020. URL `https://graphql.org/`. Accessed 2020-02-14.

[4] Elasticsearch. What is elasticsearch, 2020. URL `https://www.elastic.co/what-is/elasticsearch`. Accessed 2020-02-21.

[5] Express.js. Express, 2020. URL `https://expressjs.com/`. Accessed 2020-02-14.

[6] GraphQL. Graphql, 2020. URL `https://engineering.fb.com/core-data/graphql-a-data-query-language/`. Accessed 2020-02-14.

[7] Node.js. Introduction, 2020. URL `https://nodejs.org/en/`. Accessed 2020-02-14.

[8] D. van Kooten. Co2 emissions on the web, 2020. URL `https://dannyvankooten.com/website-carbon-emissions/?fbclid=IwAR3bSi6BVR5fhNAoJG5lDrVDMest6IUimvOQf_dtscifZlEAYOXybgLZWuI`. Accessed 2020-02-21.

[9] M. Vogel, S. Weber, and C. Zirpins. Experiences on migrating restful web services to graphql. volume 10797, pages 283–295, 2018. ISBN 0302-9743.

[10] Wikipedia. Representational state transfer, 2020. URL `https://en.wikipedia.org/wiki/Representational_state_transfer`. Accessed 2020-02-14.

[11] Wikipedia. Application programming interface, 2020. URL `https://en.wikipedia.org/wiki/Application_programming_interface`. Accessed 2020-02-14.

[12] E. Wittern, A. Cha, and J. A. Laredo. Generating graphql-wrappers for rest(-like) apis. In T. Mikkonen, R. Klamma, and J. Hernández, editors, *Web Engineering*, pages 65–83, Cham, 2018. Springer International Publishing. ISBN 978-3-319-91662-0.