

Pokémon API

IPBDAMH groep 2.3

|  |  |
| --- | --- |
| Versienummer | 0.1 |
| Versiedatum | 03-03-2020 |
| Versie type | Concept |
| Auteurs | Kevin Coppens & Jason Kloor |

# Voorwoord

Het doel van dit onderzoeksrapport is het delen van onze bevindingen rondom de Pokémon API. In dit rapport zal worden beschreven wat we onderzoeken, waarom we het onderzoeken, welke onderzoeksmethoden we gebruiken en wat de resultaten zijn van het onderzoek. Verder zullen wij de hoofd - en deelvragen van ons onderzoek beschrijven en beantwoorden.

# Inleiding

# Inhoudsopgave

# Onderzoeksdoel

Het doel van dit onderzoek is het uitpluizen van de Pokémon API. Zo willen wij ons erin verdiepen en erachter komen om wat voor data het gaat. Verder willen wij gaan kijken wat voor technieken wij kunnen toepassen met de API en wat wij er verder mee kunnen.

Wat wij heel graag willen met het onderzoek is om uiteindelijk een voorspellende analyse te kunnen doen op de data van de API. Dit doen wij door te starten met een inhoudelijke analyse over de data. Na de inhoudelijke analyse willen wij een voorspellende analyse uitvoeren. Deze voorspellende analyse zal weergeven wat er in de volgende generatie van Pokémon zou kunnen voorkomen. Denk hierbij aan hoeveel Pokémon er per type zullen worden toegevoegd, maar ook aan hoeveel nieuwe moves er zullen komen. De voorspellende analyse zal plaatsvinden op basis van de data van de huidige 8 generaties. Wij zouden bijvoorbeeld een aanname kunnen doen op het aantal nieuwe Pokémon met het type water, wanneer er blijkt dat er een bijna constant aantal water type Pokémon is geweest in voorgaande generaties. Zo kunnen wij dit doen voor meerdere onderwerpen.

# 

# Vraagstelling

Voor het onderzoek hebben wij een hoofdvraag en een aantal deelvragen opgesteld. De hoofdvraag luidt als volgt:

**“*Kan er aan de hand van de data uit de Pokémon API een voorspellende analyse worden gedaan?”***

Deelvragen:

1. Wat is de betrouwbaarheid van de data uit de Pokémon API?
2. Is de data uit de Pokémon API volledig?
3. Kan de data uit de Pokémon API verwerkt worden in een database?
4. Bevat de data genoeg informatie om analyses uit te voeren?
5. Welke verbanden zijn er te vinden in de data van de Pokémon API?

# Onderzoeksmethode

**MongoDB versus MySQL**

Als we het hebben over databases kunnen we een onderscheid maken tussen MySQL en NoSQL. MongoDB is een NoSQL database en dit houdt in dat de data wordt opgeslagen als JSON-achtige documenten in tegenstelling tot een MySQL database waarin de data wordt opgeslagen in tables. Daarnaast als er informatie uit de MongoDB moet worden gehaald moet er gebruik worden gemaakt van MQL dit staat voor MongoDB query Language. Dit is anders dan bij een MySQL database want dan moet er gebruik worden gemaakt van de MySQL taal dit staat voor structured query language. Verder als er gebruik wordt gemaakt van MongoDB hoef je geen database schema te maken omdat de documenten zelf uitleggend zijn waardoor er geen nood is voor een database schema. Terwijl bij een MySQL database moet er een database schema zijn voor de data. Er kan wel een schema worden gemaakt voor een MongoDB als het nodig is maar het is niet verplicht.

Waarom zou er gekozen moeten worden voor een MongoDB in plaats van een MySQL database. Organisaties kiezen voor MongoDB omdat het de organisaties de mogelijkheid geeft om snel applicaties te ontwikkelen. Die diverse data type kunnen verwerken daarnaast kan MongoDB goed omgaan met schallende applicaties.

MongoDB maakt het ontwikkelen ook makkelijker. Omdat MongoDB documenten natuurlijk passen bij object-georiënteerde programmeertalen. In tegenstelling tot MySQL omdat voor MySQL zit er een extra laag tussen die objecten moet vertalen in code zodat het gebruikt kan worden in relationele tables. Verder heeft MongoDB flexibele data modellen dit betekent dat MongoDB kan meegroeien met de vereisten vanuit de business. Waarin een MySQL database zou moeten worden aangepast als er een verandering is.

Daarnaast kan MongoDB ook geschaald worden in en meerdere verdeelde data centers. Dit draagt bij beschikbaarheid en schaalbaarheid wat onmogelijk is met een MySQL database. Verder als je data volume groeit, groeit MongoDB mee zonder enige vorm van downtime. Wat in tegenstelling to MySQL veel malen simpelere is omdat bij MySQL moet er veel werken worden gedaan om mee te schalen.

MongoDB maakt gebruik van replicatie dit houdt in dat er een groep MongoDB nodes zijn die dezelfde data vast houden. Dit betekent als er een node wegvalt zijn er nog andere nodes die de data bevatten. Dit zorgt ervoor dat er een hoge graad is van beschikbaarheid.

MongoDB kan vanuit zichzelf herkennen of een node defect is. Als MongoDB het dan vaststelt kan het zelf een nieuwe primaire node aanstellen. En dit kan dan binnen een paar seconden, hierdoor kunnen applicatie hun functies voortzetten terwijl de kapotte node wordt vervangen. Maar stel dat het een MySQL database is dan is het een manual proces. Dit betekent dat er gebruik moet worden gemaakt van mensuren en het kan een tijd duren totdat het probleem is opgelost.

MongoDB maakt ook gebruik van afstembare consistentie garanties. Dit houdt in als een applicatie met een hoge prioriteit data aanvraagt aan een database met veel schade. Dan kan deze applicatie nog steeds functioneren in ruil voor het terugkrijgen van niet recente data. een MySQL database heeft dit niet waardoor ontwikkelaars weinig opties hebben om zekerheid te geven dat applicaties blijven draaien als database nodes kapot zijn.

Het werken met JSON documenten in plaats van rijen en kolommen helpt het ontwikkelen van applicaties. Dit heeft de volgende redenen:

Documenten zijn natuurlijker. Dit is omdat documenten data representeren net zoals applicaties. En dit geldt niet voor de rijen en kolommen.

Documenten zijn flexibel. Dit houdt in dat een document andere soort attributen van een ander documenten kan bevatten. Een voorbeeld is een product catalog waar een document product details bevat van mannenkleding. Maar deze product catalog heeft ook een product zoals een televisie. En dit product heeft weer andere details maar toch worden de documenten hetzelfde gebruikt dit heet polymorphisme. Er kan altijd attributen worden toegevoegd aan een JSON documenten zonder dat er een database schema moet worden aangepast. En stel dat het een MySQL database was dat moet het schema worden aangepast en dit resulteert in dat de database offline is en niet gebruikt kan worden.

Documenten maken applicaties sneller. Dit is omdat de data nu op een plaats is opgeslagen en niet op meerdere locaties. Waardoor de database maar van een locatie hoeft te lezen.

**Schaal oneindig en goedkoop**

MongoDB geeft support binnen de database voor het verdelen van data over meerdere nodes.

Schaal applicaties goedkoop. Het verdelen van de data is kosteneffectief. Het kopen van een aantal goedkope machines is vaak goedkoper dan het kopen van een klein aantal machines met significante betere specificaties. Dat laatste is nodig als we praten over een MySQL database.

Er hoeven geen veranderingen plaats te vinden binnen de applicatie om te schalen. Bij de meeste MySQL databases als de database wordt uitgebreidt moet de applicatie ook mee veranderen. Maar met MongoDB een nieuw deel kan worden toegevoegd op ieder moment en het zal automatisch de data migreren. En er hoeven ook geen aanpassingen aan de applicatie worden gemaakt. Daarnaast kunnen deze delen ook nog over de hele wereld verdeeld worden door middel van Atlas Global Clusters. Daarnaast geeft Atlas Global Clusters kleine latency toegang aan mensen over de hele wereld.

Welke taal gebruikt iedere database

MySQL maak gebruikt van SQL voor toegang.

MongoDB maakt gebruik van MQL. Hiernaast een link naar de documentatie: https://docs.mongodb.com/manual/reference/sql-comparison/

Is MongoDB sneller dan MySQL

Database performance kan verschillen gebaseerd op een aantal factoren: database ontwerp, Applicatie query patronen en gebruik van de database om er een paar te noemen. Omdat MongoDB gebruik maakt van het documenten model om data op te slaan. Is dit vaak sneller om een document te verkrijgen. Dan een JOIN te gebruiken om door de database de benodigde data te verzamelen.

**Heeft MySQL support voor JSON?**

Ja MySQL heeft support voor JSON het probleem is echter dat:

SQL heeft een eigen extensie. Query's en het aanpassen van de inhoud van een JSON document heeft aparte MySQL-specifieke SQL functies. Dit is alleen niet heel bekend bij de meeste developers. Daarnaast wordt dit ook niet ondersteund door derde partijen SQL tools. Voorbeelden zijn BI platforms, Data warehouse verbinders, ETL en ESB pipelines en meer.

Verouderde relationele overhead. Zelfs met JSON ondersteuning moeten MySQL gebruikers nog steeds gebruik maken van meerdere lagen van SQL/relationele functionaliteiten die met JSON data communiceren. Voorbeelden zijn op laag niveau JDBC/ODBC drivers en ORM’s. Deze lagen leggen hoge overhead kosten op. Daarnaast zijn ORM’s over het algemeen ook moeilijk om te optimaliseren als er gekeken wordt naar performance en query effectivitijd.

Complexe data omgang. Wanneer er gebruikt wordt gemaakt van JSON data. Hebben de MySQL drivers niet de capaciteiten om goed en nauwkeurig de JSON data te converteren naar nuttige native data die wordt gebruikt door applicaties. Dit betreft verschillende types numerieke waarden zoals: drijvende punten, 64-int integers en decimalen,timestamps en datums. Daarnaast kan het ook niet in java Map en lists of dictionaries of lists in python. Ontwikkelaars moeten handmatig tekst-gebaseerde JSON converteren binnen hun applicatie. Waarmee de ontwikkelaars de mogelijkheid verliezen om velden te hebben met meerdere soorten datatypen in verschillende documenten. Hier wordt dan polymorphisme verloren.

Geen data governance. MySQL heeft geen natuurlijk mechanisme om een JSON schema te valideren als het in de database wordt gezet of wordt geupdate. Ontwikkelaars moeten zelf functionaliteiten hiervoor toevoegen.

Schema stijfheid. MySQL ontwikkelaars moeten nog steeds een schema vaststellen voor de database. Als het schema wordt aangepast omdat de applicatie is veranderd dan moet de table nog steeds veranderd worden.

**MongoDB versus Spark**

MongoDB is beter geplaatst in grote projecten, met grote schaalbaarheid. Hiermee kan je ook heel comfortabel werken met projecten op basis van programmeertalen zoals javascript, angular, typescript en C#. Wij geloven dat de prestaties veel beter zijn met het soort technologieën dat zeer logische, vergelijkbare programmeer voorwaarden hanteert. Als we bijvoorbeeld talen als Java en PHP gebruiken, is het beter om te werken met relationele databases zoals Postgres of MySQL. Aangezien dit soort technologie u in staat stelt om beter te werken met databasebeheer kaders die veel flexibeler zijn voor deze omgevingen, zoals JPA, HIBERNATE, Oracle, denk ik dat ze veel beter zijn met dit type architectuur en programmeertalen. MongoDB support ook meer programmeertalen dan Spark, wat voor dit project meer mogelijkheden biedt.

MongoDB is een van de beroemdste niet-relationele databases ter wereld, er zijn beroemde actieve projecten die deze database gebruiken. Accounts met eersteklas ondersteuning om de juiste implementatie van de database te kunnen realiseren, naast het aanleren van de best practices om uw projecten te optimaliseren, geloof ik dat het met deze beslissing meer dan vanzelfsprekend is wat de beste beslissing is bij de tijdstip van kijken met welke database te werken. Daarnaast kan je binnen MongoDB ook werken met users en rollen. Dit zorgt voor een betere integriteit van je data, doordat je de accessibility per user kan aanpassen. In ons project zal dat niet veel verschil maken, maar voor een database oplossing is dit wel een sterke factor om juist voor MongoDB te kiezen in plaats van voor Spark.

**Conclusie**

Wij hebben ervoor gekozen om binnen het project te werken met MongoDB. Dit doen wij omdat de toegankelijkheid met .JSON files goed is. Dit is belangrijk omdat we de data uit de API ook in .JSON formaat extraheren. Ook kiezen wij voor MongoDB, omdat deze een grote schaalbaarheid biedt, wat belangrijk is als je met Big Data werkt.

# Resultaten

# Conclusie

# 

# Bronnen

Bron: <https://www.mongodb.com/compare/mongodb-mysql>

Geschreven door: MongoDB

Geschreven op:

Gebruikt op 2020-02-13

Document kopje: MongoDB Versus MySQL

Bron: <https://www.freecodecamp.org/news/learn-mongodb-a4ce205e7739/>

Geschreven door: Navindu Jayatilake ,

Geschreven op 27 januari 2019:

Gebruikt op 2020-02-13

Document kopje: MongoDB Versus MySQL

Bron: <https://docs.atlas.mongodb.com/compass-connection/>

Geschreven door: MongoDB

Geschreven op:

Gebruikt op 2020-02-18