Wat is een microservice?

Een microservice is een architectuurstijl voor het bouwen van software-applicaties, waarbij de applicatie is opgedeeld in kleinere, onafhankelijke services die samenwerken om de functionaliteit van de applicatie te maken. In deze stijl van architectuur is elke microservice een zelfstandig onderdeel van de applicatie die kan worden ontwikkeld, geüpdatet en gedeployed zonder afhankelijk te zijn van de andere delen van de applicatie. Hierdoor wordt de applicatie flexibeler, schaalbaarder en eenvoudiger te onderhouden.

Elke microservice heeft zijn eigen interface en communicatieprotocol, waardoor het mogelijk is om verschillende technologieën en talen te gebruiken voor het bouwen van verschillende microservices in de applicatie. Zo kan de ene microservice bijvoorbeeld geschreven zijn in C#, terwijl een andere microservice bijvoorbeeld gebruik maakt van Java of Python. Door deze segregatie kunnen specifieke problemen binnen een applicatie opgelost worden die bijvoorbeeld beter geschikt zijn bij een bepaalde programmeertaal.

Een ander belangrijk kenmerk van microservices is dat ze goed geen directe afhankelijkheden hebben met andere microservices. Dit betekent dat een storing in één microservice niet de hele applicatie zal laten crashen, en dat elke microservice afzonderlijk kan worden getest en gedeployed kan worden naar bijvoorbeeld een cloud provider. Een microservice-applicatie wordt dan ook vaak gebruikt in combinatie met een kubernetes cluster. Een van de uitzonderingen hierop is de authentication service. Zodra deze een storing heeft kan de rest van de applicatie hier problemen mee ondervinden. Dit zal verder onderzocht gaan worden in dit onderzoek.

In het kort biedt de architectuur van microservices een manier om complexe softwareapplicaties te bouwen door deze op te splitsen in kleine, afzonderlijke diensten die kunnen worden ontwikkeld, getest, gedeployed en onderhouden met een hoge mate van autonomie en onafhankelijkheid.

Lewis, J., & Fowler, M. (2014). Microservices: a definition of this new architectural term. Retrieved from <https://martinfowler.com/articles/microservices.html>

<https://www.true.nl/blog/microservices-waarom-inzetten-voor-je-applicatie/>

<https://azure.microsoft.com/nl-nl/solutions/microservice-applications/>

Wat zijn de voor- en nadelen van een microservice?

Zoals bij elk softwareproject wordt aan het begin van het ontwerp een keuze gemaakt over welke soort architectuur er gebruikt gaat worden. Terwijl veel applicaties nog gebruik maken van een monolieten architectuur, is een steeds voor de hand liggender ontwerp de microservice. In de volgende opsomming wordt kort toegelicht waarom je juist wel voor een microservice architectuur zou kunnen kiezen.

1. Betere Performance

Door een applicatie op te delen in kleine stukjes kan men beter inzien welke delen van het systeem juist wel of niet goed performen. In een grote applicatie kan het dus zijn dat een context van het systeem extreem veel gebruikt wordt, terwijl andere delen van de applicatie juist niet veel worden gebruikt. Door deze contexten te splitsen van elkaar krijgt men de mogelijkheid om beter horizontaal te schalen voor de meest gebruikte services. Zo blijft het systeem snel en draait het op een uitstekend performance level.

1. Beter onderhoudbare software

Dit betekent concreet dat software sneller doorontwikkeld kan worden, en dat aanpassingen en nieuwe features dus sneller bij de eindgebruiker terecht komen. De gedachtegang hier is dat kleine applicaties makkelijker te begrijpen zijn, en daarmee makkelijker aanpasbaar zijn en er minder bugs geïntroduceerd worden.

1. Efficientere werkverdeling

Naarmate een softwareproject groter wordt, en er meer mensen aan werken, groeit de benodigde communicatie tussen mensen in het project. Dit kan zo ernstig worden dat developers en productmanagers alleen nog maar aan het praten zijn, maar daadwerkelijk niets meer opleveren. Bij een microservices-architectuur kun je teams rond een microservice bouwen. Omdat de services per definitie klein zijn vermijd je dit probleem van de verlammende communicatie-overhead, en kan de werkverdeling optimaal ingedeeld worden.

1. Ontwikkeling in meerdere programmeertalen

Doordat alle applicaties hun eigen codebase hebben, en alleen via standaardprotocollen met elkaar praten is het mogelijk dat elke microservice in een andere taal geschreven wordt. Dit biedt de mogelijkheid om een taal uit te kiezen die precies past bij het presente probleem, en biedt je ook de mogelijkheid om uit een grotere pool developers te vissen.

<https://www.sqli.com/nl-nl/insights/blog/monoliet-vs-microservices-de-juiste-architectuur-kiezen-voor-uw-organisatie>

<https://www.emerce.nl/achtergrond/microservices-wel-of-niet-gebruiken>

Fowler, M., & Lewis, J. (2014). Microservices: a definition of this new architectural term. Retrieved from https://martinfowler.com/articles/microservices.html

Na het benoemen van een aantal voordelen zijn er natuurlijk ook nadelen aan het gebruiken van een microservices architectuur

1. Grotere complexiteit

Het Gebruik van microservices vereist meer coördinatie en management binnen een (of meerdere) development teams. Dit kan leiden tot extra complexiteit, waardoor de ontwikkeling en het onderhoud van een applicatie ingewikkelder kan worden. Daarnaast moeten de developers die werken aan een microservice meer kennis hebben dan een gemiddelde monolieten developers, sinds een groot onderdeel van een microservice-applicatie te maken heeft met gedistribueerde systemen en containerization zoals bijvoorbeeld Docker of Kubernetes.

1. Hogere kosten

Het gebruik van microservices kan leiden tot hogere kosten, bijvoorbeeld doordat er meer infrastructuur en resources nodig zijn voor het beheer en onderhoud van de applicatie. Vaak moet een microservice applicatie gedeployed worden naar bijvoorbeeld een Cloud provider, wat zeker niet goedkoop is. Ook kan de ontwikkeling van microservices duurder zijn, omdat er meer coördinatie en planning nodig is om de diensten te ontwikkelen en te integreren.

1. Hogere latency en foutgevoeligheid

Doordat microservices communiceren met elkaar door bijvoorbeeld HTTP of een message broker, kan de latency (vertraging) tussen de microservices hoger zijn dan bijvoorbeeld een monolieten applicatie. Daarnaast zijn er ook meer resources en werknemers nodig om deze microservices te onderhouden en beheren.

Welke build in authenticatie mogelijkheden zijn er voor microservices?

Voor deze deelvraag is er gekeken naar de mogelijkheden voor een build in autorisatie voor microservices. Ondanks dat deze opties ook beschikbaar zijn als externe providers, is het doel van deze deelvraag dat deze technieken binnen de microservices worden geimplementeerd en niet via een externe Identity provider.

1. JWT (Json Web Tokens)

JWT is een populaire manier van authenticatie binnen microservices. Een JWT is een compact, zelf-inhoudelijk en cryptografisch beveiligd token dat informatie kan bevatten over de gebruiker en zijn/haar rol binnen de applicatie. De JWT kan worden gebruikt om de identiteit van de gebruiker te verifiëren bij elke microservice die de gebruiker aanroept. Voor verschillende programmeertalen zijn packages en extensies beschikbaar waardoor men in een microservice een token kan generen en valideren.

1. OAuth (2.0)

OAuth is een autorisatieprotocol dat veel wordt gebruikt voor authenticatie binnen microservices. Met OAuth kan een gebruiker een token krijgen waarmee hij/zij toegang heeft tot bepaalde resources binnen de applicatie. De verschillende microservices kunnen dit token verifiëren om te bepalen of de gebruiker toegang heeft tot de gewenste resources. Deze token wordt gegeven zonder dat een gebruiker zijn/haar gebruikersnaam en wachtwoord uit handen hoeven te geven.

Dit wordt bijvoorbeeld gedaan door in te loggen via google of facebook. Daar wordt gecontroleerd of dit de juiste gebruiker is en zo hoeft een gebruiker zijn/haar inloggegevens dus niet te delen met een bepaalde website.

1. OpenID

OpenID is een open standaard voor authenticatie, waarmee gebruikers één set aanmeldingsgegevens kunnen gebruiken om toegang te krijgen tot verschillende websites en applicaties. In plaats van dat gebruikers voor elke website of applicatie een nieuw account en wachtwoord moeten aanmaken, kunnen ze inloggen met hun bestaande OpenID.

OpenID is ontwikkeld om het proces van inloggen op verschillende websites te vereenvoudigen en te beveiligen. Het maakt gebruik van een open protocol om de authenticatie van gebruikers te verifiëren en biedt een veiligere methode om toegang te krijgen tot meerdere websites en applicaties. Ondanks dat deze manier vooral gebruikt wordt als externe authenticatie, zijn er ook build in mogelijkheden om dit lokaal op te lossen. Bij de deelvraag welke externe tools zijn er zal deze techniek ook nog voorbij komen.

<https://jwt.io/introduction/>

<https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2>

<https://www.digitalocean.com/community/tutorials/an-introduction-to-openid>

<https://openid.net/>