# Inleiding

In dit document zal ik mijn onderzoek naar het gebruik van microservices toelichten. Aangezien de applicatie die ik wil maken uitbreid baar moet zijn, is het belangrijk om dit ook zo te maken. Daarom doe ik onderzoek naar microservices om dit probleem op te lossen.

# Onderzoeksvraag

Op welke manier kan ik de microservices het beste met elkaar laten communiceren binnen mijn applicatie?

# Inleiding

Mijn applicatie gaat bestaan uit een front-end waar de back-end mee bestuurd kan worden. Er zullen een aantal services zijn welke die verschillende verantwoordelijkheden hebben. In dit onderzoek ga ik uitzoeken welke manier van communiceren tussen deze services goed zal werken voor de applicatie die ik ga maken.

## Wat zijn microservices?

Microservices, ook wel microservice architecture genoemd, is een aanpak om software te ontwikkelen. Wat hiermee bereikt wordt is dat een grote back-end applicatie wordt opgedeeld in kleine componenten welke dienen als services. Hierdoor is elke service onafhankelijk van elkaar, maar kunnen wel samenwerken. Elke service heeft een taak of verantwoordelijkheid en wordt gezien samen gezien als grote applicatie. Het voordeel hiervan is dat als er een fout in een service ontstaat, de rest van de services verder normaal kunnen functioneren en dus niet heel de applicatie niet meer werkt.

## Waarom gebruikt men microservices?

Aangezien bij het gebruik van een monolithische architectuur (zie [bron](https://www.techtarget.com/whatis/definition/monolithic-architecture)) de applicatie enorm groot kan worden. Hierdoor is een applicatie moeilijker schaalbaar te maken, omdat voor één aanpassing, de hele applicatie opnieuw gecompileerd en getest worden. Door gebruik te maken van microservices is dit probleem aangepakt, doordat het eigenlijk uit meerdere applicaties bestaat. Deze applicaties worden ook wel services genoemd. Als er in een service een aanpassing nodig is, hoeft alleen de betreffende service aangepast te worden, en niet de gehele applicatie zoals bij een monolithische architectuur. Een ander voordeel is dat service A gebruik kan maken van een andere techniek dan service B. Hierdoor kan er voor een service specifiek een ontwikkelingstaal worden gekozen, wat beter past bij de betreffende toepassing.

## Welke manieren zijn er om te communiceren tussen microservices?

Het is op meerdere manieren mogelijk microservices met elkaar te laten communiceren. Hieronder heb ik 2 mogelijkheden uitgewerkt.

Het is mogelijk om met http-protocol te communiceren tussen microservices. Hier wordt dan gebruik gemaakt van het REST principe. Hierdoor kan er gecommuniceerd worden door bijvoorbeeld een URL aan te spreken waar een REST service achter zit, welke bijvoorbeeld een JSON-formaat terugstuurt.

Het is ook mogelijk om gebruik te maken van real-time communicatie. Hier kan dan gebruik worden gemaakt van WebSocket. Dit is een ander communicatieprotocol dan het http-protocol welke gebruikt wordt om data door te sturen dat real-time moet zijn.

Zie [bron](https://learn.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/communication-in-microservice-architecture).

## API gateway

De client kan rechtstreeks verbinding maken met de microservices. Het is echter ook mogelijk om gebruik te maken van een API gateway. Deze dient dan als een reverse proxy. De client stuurt een API verzoek naar de gateway en de gateway zorgt er dan voor dat deze bij de juiste microservice terecht komt. Het voordeel hiervan is dat gedecentraliseerde microservice van een centraal punt aangesproken worden. Een ander voordeel is dat protocol dat de microservice verwacht niet uitmaakt voor de client. De client stuurt namelijk een verzoek welke de gateway omzet naar de juiste protocol die de microservice verwacht.

Zie [bron](https://www.techtarget.com/whatis/definition/API-gateway-application-programming-interface-gateway).

## Wat zijn de voor- en nadelen REST vs WebSocket

REST:

Voordelen:

* Schaalbaar, doordat microservices niet afhankelijk van elkaar zijn is het makkelijk om deze uit te breiden of er een bij te maken voor extra functionaliteit.
* Gebruik maken van verschillende talen of technologieën, gezien een microservice een applicatie op zich is, is het mogelijk om voor verschillende services andere talen of technologieën te gebruiken.

Nadelen:

* Infrastructuur complexer, onderhoud moet op verschillende plaatsen gebeuren omdat alles los van elkaar draait
* Moeilijker te monitoren omdat het een uitgebreid en complex systeem is

WebSocket:

Voordelen:

* Twee-weg communicatie is mogelijk
* Veel sneller dan data verzenden over HTTP

Nadelen:

* Werkt alleen met browsers die HTML-5 ondersteunen
* Er zijn geen succes mechanismes mogelijk, zoals responsecodes.

# Conclusie

In dit onderzoek ben ik tot de conclusie gekomen om gebruik te maken van microservices omdat mijn applicatie in verschillende fases opgebouwd zal worden. Dit omdat ik dan per functionaliteit een microservice kan bouwen. Zodra deze klaar is en volledig werkt kan ik door naar de volgende microservice. Het gebruiken van microservices is eigenlijk niet de beste oplossing voor mijn applicatie maar aangezien ik graag wil leren hoe microservices werken ga ik deze gebruiken in mijn individueel project. In dit onderzoek heb ik het ook gehad over een API gateway, hier ga ik geen gebruik van maken aangezien dat niet relevant is voor de applicatie die ik aan het maken ben.

Bronnen:

* Microservice communicatie bron: <https://medium.com/the-sixt-india-blog/microservice-communication-53cbc93cf4de>
* <https://www.kabisa.nl/blog/waarom-zou-je-voor-microservices-kiezen/>
* <https://raygun.com/blog/what-are-microservices/>
* <https://konghq.com/learning-center/api-gateway/why-microservices-need-api-gateway>
* <https://www.techtarget.com/whatis/definition/API-gateway-application-programming-interface-gateway>