

|  |
| --- |
| Werkstuk webapplicaties SignalR  Ouassim Abdellaoui &  Graduaat Informatica Programmeren  Webapplicaties |

Inhoudstafel

[Inhoudstafel 3](#_Toc133152408)

[1 Inleiding 4](#_Toc133152409)

[2 Aanpak 4](#_Toc133152410)

[2.1 Verwerking van behandelde research 4](#_Toc133152411)

[2.2 Manier van werken 4](#_Toc133152412)

[2.3 Bronnen 4](#_Toc133152413)

[3 Wat is SignalR? 5](#_Toc133152414)

[3.1 Intro 5](#_Toc133152415)

[3.2 Ontstaan 6](#_Toc133152416)

[3.3 Gebruik 7](#_Toc133152417)

[3.4 Belangrijkste features SignalR 8](#_Toc133152418)

[3.4.1 Verbindingsbeheer gebeurt automatisch 8](#_Toc133152419)

[3.4.2 Berichten naar alle, specifieke of groep actieve gebruikers 8](#_Toc133152420)

[3.4.3 Schaalt naar trafiek 8](#_Toc133152421)

[3.4.4 SignalR Hub Protocol 10](#_Toc133152422)

[3.5 Waarom SignalR gebruiken? 10](#_Toc133152423)

[3.6 Voor- en nadelen 10](#_Toc133152424)

[3.6.1 Voordelen 10](#_Toc133152425)

[3.6.2 Nadelen 10](#_Toc133152426)

[3.7 WebSocket vs SignalR 10](#_Toc133152427)

[3.8 Toekomst 10](#_Toc133152428)

[4 Reflectie 10](#_Toc133152429)

# Inleiding

In deze opdracht gaan wij het hebben over SignalR.

We beginnen eerst met onze aanpak. Hoe zijn we tot dit werkstuk gekomen? Welke research hebben we allemaal gedaan en hoe hebben we deze informatie behandeld?

Daarna gaan we dieper op in wat is SignalR juist is. Waarvoor wordt het gebruikt en hoe wordt dit geïmplementeerd binnen ASP.NET? We proberen de materie zo goed mogelijk over te brengen zodat onze medestudenten dit ook kunnen gebruiken in hun toekomstige projecten. Aan de hand van beeldmateriaal hopen wij dat SignalR net iets duidelijker wordt.

Eindigen doen we met een reflectie. Hoe verliep de samenwerking in onze groep?

Wat we allemaal hebben bijgeleerd en of er eventueel ook dingen zijn die we beter anders hadden aangepakt.

# Aanpak

## Verwerking van behandelde research

## Manier van werken

## Bronnen

<https://learn.microsoft.com/nl-nl/aspnet/core/tutorials/signalr?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0&tabs=visual-studio>

<https://learn.microsoft.com/nl-nl/aspnet/core/signalr/introduction?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0>

<https://learn.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr>

<https://learn.microsoft.com/en-us/aspnet/core/signalr/scale?view=aspnetcore-7.0>

<https://www.tutorialspoint.com/what-is-signalr-and-how-to-use-it>

<https://www.c-sharpcorner.com/UploadFile/abhijmk/what-why-and-how-about-signalr/>

<https://kb.2pintsoftware.com/help/signalr-and-connection-management>

<https://en.wikipedia.org/wiki/WebSocket#:~:text=WebSocket%20is%20a%20computer%20communications,as%20RFC%206455%20in%202011>.

<https://scientificprogrammer.net/2022/09/24/sending-messages-to-individual-signalr-clients-or-groups-of-clients/>

<https://scientificprogrammer.net/2022/05/07/signalr-on-net-6-the-complete-guide/>

<https://infinum.com/blog/scaling-out-your-own-signalr-chat-application/>

<https://medium.com/yemeksepeti-teknoloji/what-is-server-sent-events-sse-and-how-to-implement-it-904938bffd73>

<https://www.codemag.com/article/1807061/Build-Real-time-Applications-with-ASP.NET-Core-SignalR>

<https://ably.com/topic/signalr-vs-websocket>

<https://ably.com/topic/signalr-vs-websocket#:~:text=WebSocket%20is%20an%20event%2Ddriven,realtime%20web%20functionality%20to%20apps>.

<https://medium.com/geekculture/ajax-polling-vs-long-polling-vs-websockets-vs-server-sent-events-e0d65033c9ba#:~:text=Polling%20is%20a%20standard%20technique,an%20empty%20response%20is%20returned>.

https://www.codemag.com/article/1807061/Build-Real-time-Applications-with-ASP.NET-Core-SignalR

# Wat is SignalR?

## Intro

SignalR is een open-source library in ASP.NET die voor realtime webfunctionaliteit zorgt. Maar wat bedoelen we hier juist mee?

Wanneer je als client gebruikmaakt van een website of webapplicatie, dan wordt er data opgehaald van of verstuurd naar een server. Telkens wanneer je dus data opstuurt of opvraagt bij een webapplicatie maak je gebruik van de server. De communicatie tussen client en server gebeurt met HTTP. Het HTTP verstuurt een request (vaak een POST of GET methode). Dit heet het HTTP request-response model, via een request krijg je een antwoord van de server. Meestal gaat dit via een REST API.

Zoals in de onderstaande afbeelding te zien gebruiken John en Mary een chatapplicatie.

John stuurt een bericht via een POST methode naar de server. Omdat het een POST request is blijft het bericht alleen opgeslagen in de database, er gebeurt voor de rest niks.

Als Mary het bericht wil lezen zou zij eerst een GET-request moeten sturen. Mary weet niet wanneer John een bericht heeft verstuurd. Mocht zij het zo willen dat haar chatapplicatie elke seconde data opvraagt bij de server, dan zou zij elke seconde een GET-request moeten sturen. Dit maakt het realtime maar is alles behalve efficiënt. Dit legt namelijk teveel druk op de performance van zowel de server als de achterliggende database.

Diagram

Description automatically generated

Zoals je dus in het voorbeeld kan zien is in een typische web-applicaties de communicatie in één richting, namelijk van de client naar de server. De client stuurt een verzoek naar de server, de server doet wat taken en stuurt een antwoord terug naar de client. Met SignalR is het mogelijk om server-code te schrijven die meteen met de clienten communiceert. SignalR maakt gebruik van WebSocket API om berichten naar een server te sturen en meteen antwoorden terug te krijgen zonder dat er een pull moet gebeuren naar de server voor een antwoord. Met het gebruik van SignalR is het dus mogelijk om een interactie communicatiesessie in twee richtingen te openen tussen een gebruiker en een server. Hier gaan we zo meteen dieper op in.

Enkele goede voorbeelden hiervan zijn de gamingindustrie, sociale media, veilingsites en GPS. Een bekend voorbeeld waarbij SignalR wordt gebruikt zijn Google documenten. Als 2 mensen dezelfde spreadsheet openen en persoon A begint te typen, kan persoon B dit meteen zien zonder zijn webpagina te verversen.

## Ontstaan

SignalR is in 2011 in het leven geroepen door David Fowler en Damian Edwards. Het werd toegevoegd in het ASP.NET project en uitgebracht als deel van ASP.NET in 2013.

In 2013 was het WebSocket protocol juist gestandaardiseerd maar ondersteunde de meeste browsers dit nog niet. WebSocket maakt het mogelijk om data tegelijkertijd in twee richtingen te sturen, hier zo meteen meer over.

Voor SignalR was er niet echt een goede methode voor real-time communicatie.

Twee manieren die wel gebruikt werden waren AJAX polling en server-sent events, dat laatste werd bovendien niet ondersteund door de meeste browsers. Beide methodes waren redelijk omslachtig en hiervoor is SignalR dan ook in level geroepen.

Om het verschil duidelijk te maken leggen we beide even uit:

AJAX (long) polling is eigenlijk wat er gebeurt in ons voorbeeld van daarjuist. Er worden constant HTTP calls naar de server gestuurd en de client wacht af tot de server iets terugstuurt. Als er geen data beschikbaar is wordt er gewoon een leeg antwoord gestuurd.

Met long-polling wordt er eigenlijk bijna hetzelfde gedaan. Het verschil is hier dat als er geen data beschikbaar is de server de request vasthoudt tot er iets is om terug te sturen. Dit wordt ook wel een “hanging GET” genoemd. Als er data beschikbaar wordt stuurt de server deze terug naar de client. De client stuurt dan meteen een request terug naar de server, waardoor de server bijna altijd wel een wachtende request heeft.

Server-sent events zijn events die door een browser gebruikt kunnen worden. Een browser “abonneert” zogezegd op een aantal events. Wanneer er zo’n event voorvalt stuurt de browser een notificatie naar de server. Zo kan er bepaalde code verstuurd worden telkens een specifiek event gebeurt. Denk hier aan alarmen, meldingen, nieuws of aan bijvoorbeeld sportscores. Stel dat een goal een event is. Elke keer er dus een goal gemaakt wordt stuurt de server data door naar de webbrowser waarna deze wordt geüpdate.

Diagram

Description automatically generated

## Gebruik

Er zijn twee versies van SignalR: ASP.NET SignalR en ASP.NET Core SignalR.

Het verschil hier is dat ASP.NET de oude versie is. Toen SignalR in het begin uitkwam maakte bijna elke webapplicatie gebruikt van jQuery. Het was dus logisch dat SignalR ook gebruikt maakte van de functionaliteiten van jQuery. Het bevatte ook complexe logica om de protocollen van het realtime webfunctionaliteit te kunnen bekomen.

Omdat in 2016 ASP.NET helemaal opnieuw geschreven is heeft SignalR ook een update gekregen. De grootste lijnen zijn wel hetzelfde gebleven. Hubs blijven het voornaamste verbindingspunt tussen servers en clienten. Clienten roepen methodes op op deze hubs en de hubs hebben controle over de verbindingen.

Het grootste verschil in de browser is dat jQuery niet wordt ondersteund. Dit maakt het gebruik van JavaScript/TypeScript veel makkelijker. Hierdoor kunnen ook modernere frameworks zoals Angular, React, Vue,.. gebruikt worden.

Nog iets dat nieuw was tegenover de standaard ASP.NET versie is het gebruik van dependency injections. Met depen

Als we dan terug gaan naar het voorbeeld van John en Mary kunnen we het verschil aantonen van het gebruik van SignalR. John en Mary maken beide een connectie met de server en database. Telkens wanneer een gebruiker een bericht naar de server stuurt herkent de server de gebruiker en stuurt dit door naar de andere gebruiker. Deze verbinding blijft open tot John of Mary de verbinding sluit.

Diagram

Description automatically generated

## Belangrijkste features SignalR

We weten nu dat SignalR instaat voor een bidirectionele open connectie tussen een webgebruiker en een server. Maar welke voordelen zijn er nog aan het gebruik van SignalR? We hebben hier de belangrijkste features opgelijst:

### Verbindingsbeheer gebeurt automatisch

SignalR gebruikt een API die instaat voor het verbindingsbeheer. Hiervoor wordt WebSocket gebruikt. WebSocket is communicatieprotocol dat het mogelijk maakt om in realtime data te sturen van en naar de server. Communicatie is consistent, bidirectioneel en de data kan tegelijk in twee richtingen gestuurd worden.

Bij het gebruik van SignalR zijn er veel functionaliteiten die je normaal extra moet installeren (denk hier bijvoorbeeld aan oudere systemen die niet compatibel zijn met nieuwere technologieën). Door het gebruik van WebSocket worden deze automatisch geïmplementeerd.

### Berichten naar alle, specifieke of groep actieve gebruikers

SignalR gebruikt hubs om te communiceren tussen gebruikers en servers. Een hub is een pijplijn die toelaat dat een gebruiker en server methodes kunnen oproepen bij elkaar. Dit wordt automatisch gedaan door SignalR, het zorgt ervoor dat de machines met elkaar verbinden en er zo communicatie mogelijk is tussen server en gebruiker of vice versa. Dit gebeurt in JSON of MessagePack, wat een compactere vorm van JSON is.

Deze hubs worden gebruikt op de serverzijde voor het versturen van berichten naar gebruikers. Het volstaat om een klasse aan te maken die over erft van Hub. Hierin kunnen verschillende methodes, tasks, interfaces,… aangemaakt worden.

Voor berichten te sturen zijn er 3 methoden: SendMessage gebruikt de property Clients.All om naar alle gebruikers een bericht te sturen. SendMessageToCaller heeft Clients.Caller om te antwoorden op de gebruiker die juist een bericht heeft gestuurd. Als laatste is er nog SendMessageToGroup die SignalR Users groups gebruikt.

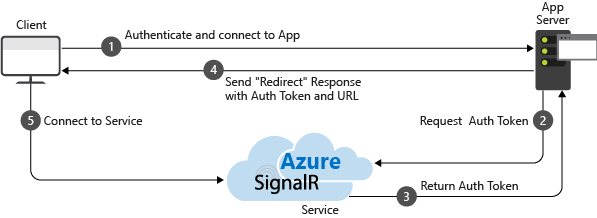
### Schaalt naar trafiek

Een app dat SignalR gebruikt moet steeds alle verbindingen bijhouden, wat problemen geeft bij meerdere servers. Stel één server wil een bericht sturen naar alle gebruikers. Dat bericht wordt alleen verzonden naar de gebruikers die met die ene server zijn verbonden. Er zijn twee opties om dit probleem op te lossen: Azure SignalR Service en Redis backpane.

Diagram

Description automatically generated

**Azure SignalR Service:** Azure verhelpt dit probleem door bij elke connectie van de gebruiker een omleiding te leggen naar de service. De gebruiker verbindt met de service zelf waardoor elke server maar een beperkt aantal connecties moet bijhouden.



**Redis backpane:** Redis werkt werkt met een publiceer/abonneer manier. Wanneer een gebruiker verbindt wordt deze informatie doorgegeven naar de “backplane” van Redis. Wanneer een server dan informatie wil sturen naar alle gebruikers wordt dit via het backplane gedaan.

Timeline

Description automatically generated with medium confidence

### SignalR Hub Protocol

SignalR maakt gebruik van hubs om de communicatie van de server naar cliënten en van cliënten naar server te doen. In de code definieer je methoden die via de server de aangeroepen kunnen worden, SignalR handelt zelf de routing af.

Hieronder is een voorbeel van zo’n hub. Door middel van de methode “SendMessage” kan een gebruiker een bericht sturen. Door de methode ReceiveMessage wordt dit dan verstuurd naar alle cliënten die verbonden zijn.

Text

Description automatically generated

In de code van de webpagina is dan de methode “ReceiveMessage” toegevoegd. Deze heeft dezelfde naam als degene die in de hub is aangemaakt. Het luistert naar berichten waarin de ReceiveMessage wordt opgeroepen.

Text

Description automatically generated

## Waarom SignalR gebruiken?

Zoals daarjuist in het voorbeeld te zien is is de SignalR methode veel efficienter, legt het geen druk op de performance van de server en biedt het realtime functionaliteit aan.

## WebSocket vs SignalR

Daarjuist bij de features haalden we aan dat SignalR WebSocket gebruikt voor het verbindingsbeheer. Als we met WebSocket ook een bidirectionele verbinding kunnen bereiken met WebSocket, waarom dan SignalR gebruiken? Om het verschil uit te begrijpen moeten we eerst WebSocket wat uitleggen.

WebSocket begint als HTTP protocol en doet eerst een handshake. Een handshake is kan je zien als een signaal tussen twee toestellen of programma’s om een link op te zetten tussen beide. Daarna wordt de HTTP Upgrade header gebruikt en wordt de HTTP een WebSocket protocol.

Verschillend met HTTP is dat WebSocket een open connectie houdt. Gebruiker en server kunnen data versturen en ontvangen zonder dat de gebruiker hier eerst een verzoek voor moet sturen. WebSocket is ook stateful

## Voor- en nadelen

### Voordelen

* Meerdere opties om te schalen naargelang trafiek
* Maakt deel uit van het ASP.Net Framework, waardoor het makkelijk te gebruiken is in combinatie met bijvoorbeeld authenticatie, authorizatie en dependency injections

### Nadelen

* Het afleveren van berichten is niet altijd gegarandeerd
* SignalR heeft maar een beperkt aantal software ontwikkelingspakketten: C#, Java, Python and Javascript. Modernere technologieën zoals iOS, Android, PHP worden niet ondersteund
* SignalR is ontworpen voor single-region gebruik

We weten nu dat SignalR gebruikt maakt van WebSocket voor de

## Toekomst

# Reflectie