

Een Snellere Datastroom:

WebSockets Onder de Loep

Orens Jasper

Indestege Michelle

Lenaerts Roald

Level27, Hasselt

Woord vooraf

*Het woord vooraf is geen inleiding, maar geeft je de ruimte om een* ***persoonlijke*** *situering te geven van de manier waarop je Graduaatsproef tot stand gekomen is. Je kan in het voorwoord dan ook de gelegenheid nemen om iedereen die van ver of dichtbij betrokken was bij het proces te bedanken. Je ondertekent het woord vooraf normaal gezien ook met je naam. Maximum één A4 lang.*

Inhoudsopgave

Inhoud

[Woord vooraf 1](#_Toc855155699)

[Inhoudsopgave 2](#_Toc209793881)

[1 Bedrijfsvoorstelling 3](#_Toc623627605)

[2 Projectvraag, onderzoeksacties en resultaten 4](#_Toc583653913)

[2.1 Situering probleemstelling 4](#_Toc2057527)

[2.2 Projectvraag en deelvragen 4](#_Toc1481574191)

[2.3 (Onderzoeks-)acties 4](#_Toc651300736)

[2.4 Verzamelde resultaten 4](#_Toc1677972080)

[3 Conclusies en aanbevelingen 4](#_Toc664290631)

[3.1 Conclusies 4](#_Toc370677648)

[3.2 Aanbevelingen 4](#_Toc71535563)

[4 Persoonlijke reflecties en kritische kanttekeningen 4](#_Toc925617323)

[5 Referentielijst 4](#_Toc284178590)

[6 Bijlagen 5](#_Toc2143395565)

# Bedrijfsvoorstelling

## Situering van het bedrijf

Level27 is gevestigd in Hasselt, België en opereert in de wereld van Hosting op internationaal niveau. Het bedrijf werd opgericht in 2007. De afdeling waarin de graduaatsproef situeert is *Platform*. Deze afdeling is verantwoordelijk voor de interface en datastroom waar de klant constant mee in aanmerking komt. Zij zorgen voor verbeteringen aan de servers op de *Application Layer* van het *TCP/IP.*

## Werkomgeving

Level27 is gespecialiseerd in diverse onlineactiviteiten. Deze variëren van webhosting, agency hosting, Managed services en Cloud-services. Het gebruik van een bekende methodologie wordt niet toegepast op het bedrijf. Echter wordt er iedere week een zelfreflectie ingevuld in een template. Deze zelfreflectie wordt ook wekelijks besproken met het afdelingshoofd. Iedere vrijdag presenteert één van de werknemers van Level27 een pitch voor om de loop van deze werknemer hun afdeling te tonen. Overal is het een zeer rustgevende werkplaats waar veel aandacht word besteed aan een goede sfeer tussen de collega’s.

# Projectvraag, onderzoeksacties en resultaten

## Situering probleemstelling

De huidige websocket implementatie bij Level27 werkt niet volledig naar behoeven. Er moet een andere implementatie van WebSockets worden geprogrammeerd om data snelle van server naar client te versturen. Er is geen noodzaak om de implementatie te verbeteren van client naar server.

## Projectvraag en deelvragen

* Wat zijn de meest voorkomende Web Socket implementaties in de werled om data te tonen van de server?
* Zijn er op dit moment alternatieven van WebSockets die eventueel beter kunnen werken?
* Welke rol spelen caching-mechanismen bij het verbeteren van Full-Duplex communicatieprestaties via WebSockets en welke caching-strategieën zijn het meest effectief?
* Hoe beïnvloed de keuze van het gekozen framework voor de WebSocket de prestaties van Full-Duplex communictie en welke framework is het meest geschikt voor optimalisatie tegen eind mei 2024?
* Is de nieuwe implementatie een verbetering op de huidige WebSocket die gebruikt wordt bij Level27?

## (Onderzoeks-)acties

### Wat zijn de meest voorkomende Web Socket implementaties in de wereld om data te tonen van de server?

Om beter begrip te verkrijgen van de werking van WebSockets heb ik drie diverse cursussen gedaan die dieper in gaan op WebSockets. Er is een Node.Js server/applicatie opgesteld om de noodzakelijke data mee teversturen voor alle socket implementaties en libraries op te laten werken. *(8)*. In de eerste fase is de node.js server zelf opgesteld en als tweede stap is de database hierin opgemaakt. De mockata in de databank maakt gebruik van de faker library (9). Deze genereerd valse data die realistisch lijkt.

## Verzamelde resultaten

### Web Socket

**Algemene werking van WebSockets**  
Web Sockets maken gebruik van WS of WSS(secure vorm) in hun HTTP requests. De connectie blijft open tot de client een disconnect request verstuurd.

**HandShake**  
De client verstuurd een Request met ‘upgrade’ als onderdeel van de header. Deze heeft als waarde ‘WebSocket’ en een veld ‘connection’ met als waarde ‘Upgrade’.



*Screenshot van console in Firefox van een WebSocket implementatie*

De client verstuurd dan een BASE64 encoded Key. Wanneer de Server dan een reactie geeft. Als deze bekwaam is voor WebSockets te aanvaarden. Gebruikt de Server het HTTP 101 response. (Switching Protocols). Nadien verstuurd de server een header met dezelfde waardes als de client initieel verstuurde. Namelijk ‘Upgrade: websocket’ en ‘Connection: Upgrade’. De server neemt dan de Key van de client, voegt een GUID hier aan toe en berekend hier een SHA-Hash op. Op deze manier weet de client dat de server het verzoek heeft aanvaard. Dit een noodzakelijke authenticatie voor ‘Man in the Middle’ attacks te voorkomen, wanneer een hacker zich kan voordoen als de server. Als deze acties uitgevoerd zijn is er Bidirectionele communicatie tussen de Server en Client.

A blue and black rectangle with triangles

Description automatically generated

*Afbeelding door Jasper Orens*

**Use case**Chat applicatie:   
Facebook Messenger maakt bijvoorbeeld gebruik van WebSockets om constant data te verzenden van client 1 naar server en naar client 2.

Social Media updates:   
Push notificaties kunnen van social media kunnen gebruikt worden voor de gebruiker berichten te laten ontvangen wanneer er updates binnen komen (bv.: “Iemand heeft je getagged in een foto”, van Facebook).

Gok websites:   
Om scores aan te tonen van Live wedstrijden moeten deze altijd gelijktijdig naar alle gebruikers worden ge-update. WebSockets zijn hier zeer geschikt voor.

IoT-toepassingen:   
WebSockets kunnen worden gebruikt om communicatie met Internet of Things (IoT)-apparaten mogelijk te maken, waardoor apparaten zoals sensoren en slimme lampen real-time gegevens naar een server kunnen sturen.

### Socket.IO

**Wat is het?**  
Socket.IO is een library dat low-latency, bidirectioneel en event-based communicatie tussen client en server voorziet. De connectie kan opgesteld worden tussen verschillende low-level transports. Deze zijn:

* HTTP long-polling
* WebSocket
* WebTransport

Socket.IO kiest zelf het beste pad afhankelijk van

* De mogelijkheden van de browser
* Het netwerk (sommige netwerken blokeren WebSockets en/of WebTransport connecites

Deze library is geen implementatie van een WebSocket. Deze gebruikt wel WebSockets voor transport wanneer mogelijk, maar het voegt extra metadata toe in elk packet. Hierdoor kan een traditionele WebSocket geen connectie maken met Socket.IO omdat de WebSocket geen weg weet met deze extra data van Socket.IO. (7.)

**Gebruiksgemak en documentatie**text

**Ondersteunende functies en mogelijkheden**text

**Performance en schaalbaarheid**text

**Ondersteuning en community**text

**Licentie en kosten**Er zijn geen kosten verbonden aan het gebruik van Socket.IO, Het is een gratis open-source toolkit. Alsvolgt is er geen licentie nodig voor Socket.IO te implementeren.

**Compatibiliteit en integratiemogelijkheden**text

**Voordelen en Nadelen**text

### WebSocket API (Native)

**Wat is het?**  
text

|  |  |
| --- | --- |
| Var socket = new WebSocket(url); socket.onopen = () => {}; socket.onmessage = (e) => {}; socket.onclose = (e) => {} socket.onerror = (e) => {} | * We maken een connectie met de server * De connectie is open * Wij ontvangen een bericht * De connectie is gesloten * Fout afhandeling |

(1)

**Gebruiksgemak en documentatie**text

**Ondersteunende functies en mogelijkheden**text

**Performance en schaalbaarheid**text

**Ondersteuning en community**text

**Licentie en kosten**text

**Compatibiliteit en integratiemogelijkheden**text

**Voordelen en Nadelen**text

### Pusher

**Wat is het?**  
text

**Gebruiksgemak en documentatie**text

**Ondersteunende functies en mogelijkheden**text

**Performance en schaalbaarheid**text

**Ondersteuning en community**text

**Licentie en kosten**text

**Compatibiliteit en integratiemogelijkheden**text

**Voordelen en Nadelen**text

### RxJS WebSocket

**Wat is het?**  
text

**Gebruiksgemak en documentatie**text

**Ondersteunende functies en mogelijkheden**text

**Performance en schaalbaarheid**text

**Ondersteuning en community**text

**Licentie en kosten**text

**Compatibiliteit en integratiemogelijkheden**text

**Voordelen en Nadelen**text

### GraphQL Subscriptions

**Wat is het?**  
text

**Gebruiksgemak en documentatie**text

**Ondersteunende functies en mogelijkheden**text

**Performance en schaalbaarheid**text

**Ondersteuning en community**text

**Licentie en kosten**text

**Compatibiliteit en integratiemogelijkheden**text

**Voordelen en Nadelen**text

### SockJS

**Wat is het?**  
text

**Gebruiksgemak en documentatie**text

**Ondersteunende functies en mogelijkheden**text

**Performance en schaalbaarheid**text

**Ondersteuning en community**text

**Licentie en kosten**text

**Compatibiliteit en integratiemogelijkheden**text

**Voordelen en Nadelen**text

### WS / WSS

**Wat is het?**  
text

**Gebruiksgemak en documentatie**text

**Ondersteunende functies en mogelijkheden**text

**Performance en schaalbaarheid**text

**Ondersteuning en community**text

**Licentie en kosten**text

**Compatibiliteit en integratiemogelijkheden**text

**Voordelen en Nadelen**text

### Alternatieven voor WebSockets

#### HTTP/2 Push

Er werd onderzocht naar HTTP/2 Push. Maar Google heeft besloten om deze functie te verwijderen. Verdere implementatie van deze functie is dus niet uitgewerkt binnen het project. Google heeft het aantal requests op Chrome Browsers bestudeerd dat push requests verzonden en ontvangen. Uit hun onderzoek bleek dat 1.25% van HTTP/2 websites gebruik maakten van deze functie. Bij verdere opvolging bleek dat dit aantal gezakt was naar 0.7%. Op basis van deze analyse besloot Google om deze functie niet verder uit te werken. *(2)*

##### Werking van HTTP2/Push

De client verstuurt een verzoek voor een bepaalde webpagina of bron, zoals HTML, De Server ontvangt het verzoek van de client en identificeert alle vereiste bronnen die nodig zijn om de gevraagde webpagina correct te renderen. In plaats van te wachten tot de client afzonderlijke verzoeken voor elke bron verzendt, kan de server proactief beslissen om bepaalde bronnen naar de client te duwen. Dit wordt gedaan door extra bronnen te ‘pushen’ naar de client over dezelfde verbinding zonder te wachten op een afzonderlijke verzoek van de client. Zodra de client de gepushte bronnen ontvang, kan hij deze lokaal opslaan in de cache. Wanneer de client later die bron nodig heeft, hoeft hij geen nieuwe verzoek naar de server te sturen, omdat de bron al beschikbaar is in de cache van de client. *(3)*

A black background with green arrows

Description automatically generated

*Afbeelding door Jasper Orens*

#### Long Polling

##### Werking van Long Polling

Deze methode stuurt een verzoek naar de resource, als er geen antwoord komt op het verzoek van de client dan blijft de connectie met de resource open tot er wijzigingen gebeuren en data wordt overgemaakt van de resource naar de client. Via deze methode worden het aantal requests naar de server dus beperkt omdat de connectie langer openblijft en de client dus niet voortdurend requests moet sturen tot er een positief antwoord komt. Long Polling maakt gebruik van HTTP request in de formaten applicatoin/json, tekst/plain en de open connectie blijft 100 tot 300 seconden geldig.

A screenshot of a video game

Description automatically generated

*Afbeelding door Jasper Orens*

##### Use case

De beste benutting voor Long polling is voor chat applicaties waarin realtime updates noodzakelijk zijn. Specifiek chat communicatie tussen twee partijen. (Voor meerdere partijen zijn websockets de betere keuze.) Dit omdat het de connectie met de server openhoudt tot er nieuwe data (een bericht bv.) beschikbaar is. Zodra er nieuwe data aanwezig is op de server verstuurd deze de inhoud naar de client. De long polling open connectie wordt dan gesloten tot er van uit de client een nieuw polling request wordt verstuurd. *(4)*

#### Server-Sent Events (SSE)

##### Werking van SSE

Bij een SSE request wordt de aanvraag gedaan via een ‘event-stream’ in plaats van een .JSON-formaat of tekstformaat. Via deze weg weet de server dat er gewerkt wordt met een SSE-connectie. De server geeft als volgt meerdere antwoorden in de vorm van ‘event-stream’ in chunks. Dit zijn grootte blokken code die werden opgesplitst. Er wordt één request gestuurd voor een connectie te openen en de server beslist nadien welke data het wilt sturen naar de client. SSE maakt gebruik van HTTP requests enkel i het tekst/event-stream formaat. Data wordt verstuurd als Prefix. De connectie blijft open tot de client een disconnect request verstuurd.

A screenshot of a computer screen

Description automatically generated

*Afbeelding door Jasper Orens*

##### Use case

Server-Sent Events (SSE) worden gebruikt voor het live streamen van gegevens, vergelijkbaar met platforms zoals YouTube of Twitch voor video streams. Deze technologie maakt het mogelijk om real-time gegevens te verzenden van de server naar de client, waardoor een continue stroom van informatie wordt gecreëerd zonder dat de client herhaaldelijk om updates hoeft te vragen. SSE is vooral nuttig voor applicaties die afhankelijk zijn van live data, zoals sportupdates, financiële marktgegevens en meer. Het bijzondere van SSE is dat het een bidirectioneel communicatiekanaal biedt, hoewel het geen authentieke full-duplex communicatie is. In plaats daarvan stelt SSE servers in staat om gegevens in realtime naar clients te sturen, terwijl clients geen mogelijkheid hebben om direct naar de server te communiceren via hetzelfde SSE-kanaal. Deze eenrichtingsstroom van gegevens maakt het mogelijk voor clients om continu bijgewerkte informatie te ontvangen zonder de noodzaak van polling, waardoor de serverbelasting wordt verminderd en de efficiëntie van de communicatie wordt verbeterd.*(5)*

#### WebRTC (Web Real-Time Communication)

##### Werking van WebRTC

##### Use case

#### Short Polling

##### Werking van API

##### Use case

A black background with green arrows

Description automatically generated

*Afbeelding door Jasper Orens*

# Conclusies en aanbevelingen

*In dit gedeelte geef je je conclusies die je vanuit je uitgevoerde onderzoek kan afleiden, weer. Je beantwoordt hier je onderzoeksvraag: je geeft op een synthetische en goed beargumenteerde manier een antwoord op je onderzoeksvraag en staat stil bij de belangrijkste vaststellingen van je onderzoek. Je geeft tips en aanbevelingen voor je werkplek naar voor schuiven die je vanuit je onderzoek aangereikt krijgt/gekregen hebt. Je kadert de meerwaarde hiervan.*

## Conclusies

Tekst tekst tekst

## Aanbevelingen

Tekst tekst tekst

# Persoonlijke reflecties en kritische kanttekeningen

*Je noteert hier hoe jij het werken aan je Graduaatsproef ervaren hebt a.d.h.v. het STARR reflectiemodel. Belangrijke persoonlijke leerinzichten en kritische kanttekeningen die je meeneemt vanuit het doorgemaakte proces kunnen hier hun plaats krijgen. Je beschrijft wat voor jou persoonlijk de meerwaarde van de uitwerking van je Graduaatsproef is; wat kan/wil je* ***voor jezelf*** *vanuit het doorlopen van je onderzoek als* ***leerinzichten*** *naar voor schuiven, koppel deze leerinzichten naar de* ***OLR*** *en* ***X-factor****. Daarnaast zijn ook jouw persoonlijke kritische kanttekeningen belangrijk om te vermelden: waar zit voor jou de* ***meerwaarde en eventuele beperkingen van je onderzoek zoals jij dit doorlopen hebt****? Belangrijk hierbij is dat je in deze reflecties de link legt met hetgeen bovenstaand werd neergeschreven en dat je in je reflecties dus verwijst naar concrete informatie/acties … vanuit je Graduaatsproef.*

Tekst tekst tekst

# Referentielijst

*De referenties die je raadpleegde en die belangrijk zijn in de uitwerking van je Graduaatsproef dienen hier vermeld te worden. Je noteert dit op een consequente en correcte manier.*

1. ***Cursus network requests in JavaScript***: Network Requests in JavaScript by Christian Wenz(<https://app.pluralsight.com/library/courses/javascript-network-requests/table-of-contents>)
2. <https://developer.chrome.com/blog/removing-push>
3. <https://www.ioriver.io/terms/http-2-server-push>
4. <https://www.pubnub.com/guides/long-polling/>
5. <https://medium.com/deliveryherotechhub/what-is-server-sent-events-sse-and-how-to-implement-it-904938bffd73>
6. <https://webrtc.org/getting-started/firebase-rtc-codelab>
7. <https://socket.io/docs/v4/>
8. <https://www.w3schools.com/nodejs/nodejs_mysql_create_db.asp>
9. <https://www.npmjs.com/package/@faker-js/faker>

# Bijlagen

*Je voegt hier je genummerde bijlagen toe (bijv de enquête die je hanteerde, de uitprint van bepaalde resultaten, ….) die belangrijk zijn om het verslag op een goede manier te kunnen ‘lezen’ en ‘interpreteren’.*

Tekst tekst tekst