**Onderzoeksdocument**

***Internationalisation***

*Fontys*

***Eindhoven***

|  |
| --- |
| **Datum:6-12-21** |
| **Versie:0.1** |
| **Status:WIP** |
| **Auteur:Tim Gooren, Luc van der Zandt** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Versie** | **Datum** | **Auteur(s)** | **Wijzigingen** | **Status** |
| 0.1 | 6-12-21 | Tim Gooren, Luc van der Zandt | Eerste opzet | Opzet |
| 0.2 | 7-12-21 | Tim Gooren, Luc van der Zandt | Invullen | WIP |
|  |  |  |  |  |

**Inhoudsopgave**

[2. Onderzoeksvragen 5](#_Toc1495337670)

[2.1 Internationalisatie 5](#_Toc308017846)

[3. Wat zijn mogelijkheden voor translatie service? 5](#_Toc1319479706)

[3.1 Google Translate 6](#_Toc198130666)

[3.2 DeepL 6](#_Toc1869084910)

[3.3 Microsoft Translator 6](#_Toc849677123)

[3.4 Community Based 6](#_Toc783834315)

[3.5 Conclusie 6](#_Toc2119706291)

[4. Hoe ga je met meertaligheid om in een database? 6](#_Toc399250602)

[4.1 Een tabel maken per taal 7](#_Toc1544756309)

[4.2 Éen tabel met extra language vertaling 7](#_Toc1828175653)

[4.3 Het gebruiken van een koppeltabel. 7](#_Toc1778560350)

[4. Conclusie 8](#_Toc1539990403)

[5. Wat zijn mogelijke design patterns voor backend internationalisatie in een microservice architectuur? 8](#_Toc616929879)

[5.1 Content en vertalingen door dezelfde microservice 8](#_Toc1519116830)

[5.2 Vertalingen voor elke microservice in een aparte microservice 8](#_Toc1667620260)

[5.3 Vertalingen voor elke microservice in één gespecialiseerde microservice 9](#_Toc1986511481)

[5.5 Geen vertalingen in backend 9](#_Toc832598895)

[5.5 Conclusie 9](#_Toc479609178)

[6. Bronnen 9](#_Toc1451592529)

**1. Inleiding**

In dit document wordt onderzoek gedocumenteerd voor het

Deze onderzoeksvragen worden beantwoord door de loop van de sprint heen en worden ook mogelijk uitgebreid.

# **2.** **Onderzoeksvragen**

## 2.1 **Internationalisatie**

Hoofdvraag:

- Hoe implementeer je internationalisatie in de backend van een C# applicatie?

Deelvragen:

* Wat zijn mogelijkheden voor translatie services?
* Hoe ga je met meertaligheid om in een database?
* Wat zijn mogelijke design patterns voor backend internationalisatie in een microservice architectuur?

Aanpak:

<https://ictresearchmethods.nl/Methods>

|  |  |  |
| --- | --- | --- |
| **Tactiek** | **Categorie** | **Beschrijving** |
| Literature Study/Available Product Analysis | Library | Zoek en vergelijk populaire manieren/diensten voor vertaling. |
| Literature Study | Library | Bekijk de documentatie van de gevonden translatie services. |
| Literature Study/Best good and bad practices | Library | Zoek naar best practices om meerdere talen op te slaan. |
| Workshop | Prototyping | Implementeren van bovenstaande concepten |

# **3. Wat zijn mogelijkheden voor translatie service?**

Vaak bezoek je een website, en wordt deze automatisch naar je taal van keuze vertaald. Terwijl deze website niet uit oorsprong in die taal is. Er zijn hiervoor veel verschillende manieren om je de teksten te vertalen naar de gekozen taal, dit kan van handmatig alles vertalen naar alle informatie door een translatie AI halen. Er wordt daarom gekeken naar deze manieren en vergelijken deze met elkaar om een stap dichter bij het antwoord van de hoofdvraag te komen.

## **3.1 Google Translate**

Is bekend voor de gebruikersvriendelijke translator op de website. Vereist een google service account. Ondersteund 109 talen.

Eerste 500,000 karakters zijn gratis. Daarna zijn elke 1,000,000 karakters 20$. De gratis en de betaalde versie gebruiken dezelfde endpoint.

## **3.2 DeepL**

Is een gerichte service zonder front-end facing versie. Ondersteund 26 talen. Mist enkele EU talen, o.a. Kroatisch en Iers. Is een aparte service met een aparte API key.

Eerste 500,000 karakters zijn gratis. Daarna zijn elke 1,000,000 karakters 20$. Aparte endpoint voor gratis en betaalde versie

## **3.3 Microsoft Translator**

Eerste 2m characters gratis, goedkoper dan de andere 2 grote translation services.

Keuze voor pay-as-you-go of een subscriptie model.

Deel van azure dus uitbeidbaar met andere azure opties, dus wel azure account nodig

Ondersteund 50 talen

## **3.4 Community Based**

Het laten vertalen door een translatie AI bespaard veel tijd ten opzichte van handmatige vertaling. Maar toch zijn er voordelen dat er handmatig een vertaling gedaan zou kunnen worden gedaan. Het grootste voordeel is dat de kwaliteit van de vertaling beter is dan door een translatie AI omdat de persoon die kan vertalen beter de context kan over brengen dan een letterlijke vertaling. Het grote nadeel is dat dit veel tijd in beslag neemt en daardoor, als dit niet vrijwillig gedaan wordt, veel geld. Waardoor dit niet rendabel is in vergelijking met het gebruiken van een translatie AI.

## **3.5 Conclusie**

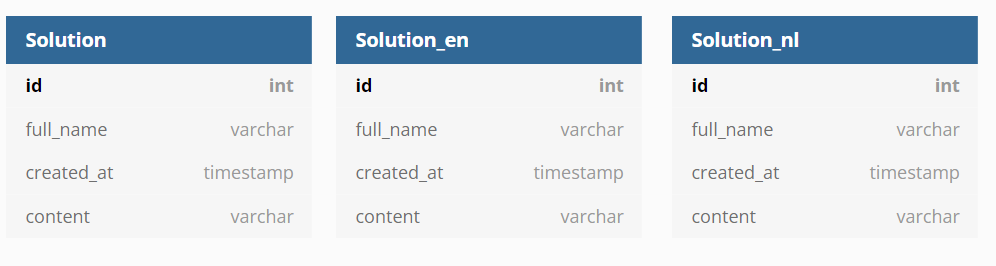
Doordat DeepL niet alle EU talen ondersteund valt die optie snel weg. Community Based translation is geen mogelijkheid vanwege de hoeveelheid artikelen die verwacht worden. Wel kan de optie voor het toesturen van vertalingen een optie zijn. Hierdoor is de keuze tussen Microsoft Translator en Google Translate, welke redelijk vergelijkbaar zijn.. Het grote voordeel aan Microsoft Translator is dat vertalen goedkoper is, ook al is de vertalingskwaliteit mogelijk lager. De keuze is hierdoor voor Microsoft Translator, omdat in de context van een non-profit project kosten minimaliseren het belangrijkste is.

# **4. Hoe ga je met meertaligheid om in een database?**

Er zijn meerdere manier om je database in te richten voor meertaligheid. Hier gaat gekeken worden naar een paar verschillende manier hoe dit kan worden gedaan, en de voor- en nadelen voor elk.

## **4.1 Een tabel maken per taal**

Het is niet aan te raden om op deze manier meertaligheid aan te pakken. Het grootste nadeel is dat elke taal een extra tabel nodig heeft, waardoor het lastig is om extra talen toe te voegen. Ook is hierdoor heel veel duplicatie van data in de database. Dit is heel inefficiënt, ook zorgt dit ervoor dat de backend hier rekening mee moet houden elke keer als er een nieuwe taal bij komt en is niet aan te raden om te doen.



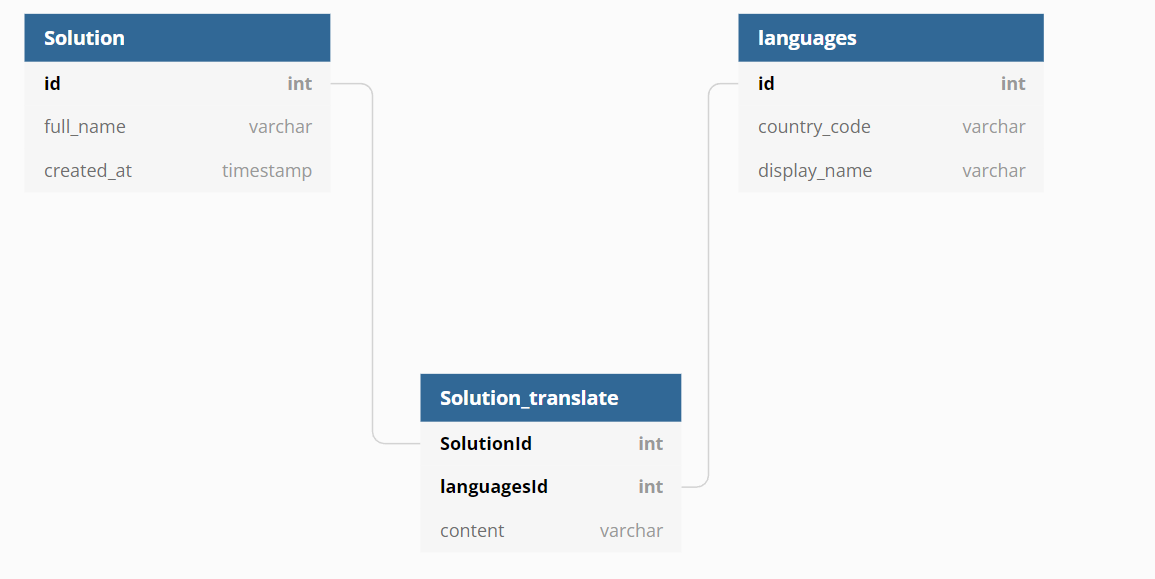
## **4.2 Éen tabel met extra language vertaling**

Dit kan werken als snel een oplossing opgezet wordt en als er maar 1 of 2 extra talen nodig zijn, maar dit is niet handig uitbreidbaar en kan snel het overzicht verloren worden, vooral als er meerdere velden zijn die meertalig zijn ontstaan er grote tabellen. Ook zal je de meertalige content er uit moeten filteren als je deze ophaalt.



## **4.3 Het gebruiken van een koppeltabel.**

Met deze methode zal er, zoals te zien in het model hieronder, een koppeltabel aangemaakt worden waar de solution en language samen komen in solution\_translate tabel, met hierin de vertaling van de benodigde tekst. Dit is de meest efficiënte manier voor het implementeren van meertaligheid, je kan makkelijk nieuwe talen toevoegen en dit is niet rommelig zoals de andere opties.



## **4. Conclusie**

Het gebruiken van de optie besproken in 4.3 is de beste optie, als er een database opgezet wordt waar meertaligheid overzichtelijk en uitbreidbaar geïmplementeerd is.

# **5. Wat zijn mogelijke design patterns voor backend internationalisatie in een microservice architectuur?**

In de context van microservices zijn er meerdere opties voor het opslaan van meertalige informatie:   
De content en de vertalingen van die content kunnen in dezelfde microservice opgeslagen worden.  
Elke microservice die vertalingen nodig heeft kunnen een eigen vertaal-microservice krijgen.

Elke microservice die vertalingen nodig praat met één gespecialiseerde microservice.

Alle meertalige content kan door de frontend afgehandelt worden.

## **5.1 Content en vertalingen door dezelfde microservice**

+ Domein blijft bij 1 microservice  
+ Geen harde koppeling tussen microservices

* Microservice wordt monolitisch
* Microservice krijgt erg veel verantwoordelijkheden

## **5.2 Vertalingen voor elke microservice in een aparte microservice**

+ Elke microservice blijft gespecialiseerd  
+ Vertalings API blijft bij de vertalingsmicroservices

* Hoeveelheid microservices wordt erg groot
* Hoeveelheid communicatie tussen microservices wordt haast verdubbeld
* Mogelijkheid tussen harde koppeling tussen 2 microservices

De harde koppeling van microservices kan gedeeltelijk voorkomen worden door de vertaling een optioneel verzoek te maken, waarbij de originele, niet-vertaalde content intact blijft. Hierdoor wordt de vertalingsmicroservice een optionele stap.

## **5.3 Vertalingen voor elke microservice in één gespecialiseerde microservice**

+ Elke microservice blijft gespecialiseerd

* Er is 1 punt van falen. Als de vertalingsmicroservice er uit ligt, heeft elke microservice daar last van.
* Hoeveelheid communicatie tussen microservices wordt erg groot
* Harde koppeling van microservices

De harde koppeling van microservices kan gedeeltelijk voorkomen worden door de vertaling een optioneel verzoek te maken, waarbij de originele, niet-vertaalde content intact blijft. Hierdoor wordt de vertalingsmicroservice een optionele stap.

## **5.4 Geen vertalingen in backend**

Dit is de meest aangeraden manier, maar niet altijd mogelijk. Alleen statische context kan in de frontend gehouden worden; wanneer user-made dynamische context vertaalt moet worden, is het niet realistisch dit in de frontend te doen: als de front-end een request zou doen naar een machine translatie API, zou dat zeer repetitieve calls opleveren die snel in kosten groeien.

## **5.5 Conclusie**

Indien mogelijk is het altijd wenselijk om vertalingen niet in de backend op te slaan. Vaak is dit echter niet mogelijk, omdat gebruikers-gedefinieerde content niet realistisch in de frontend afgehandelt kan worden.

De content en vertalingen in dezelfde microservice houden is de meest efficiënte oplossing, maar zorgt ervoor dat microservices erg veel verantwoordelijkheden en calls erbij krijgen.

Vertalingen voor elke microservice in een aparte microservice kan tot erg veel meer microservices leiden, maar is de netste oplossing door de verantwoordelijkheid van microservices gescheiden te houden.

# **6. Bronnen**

<https://locize.com/blog/how-does-server-side-internationalization-look-like/>

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/localization?view=aspnetcore-6.0>

<https://www.codemag.com/Article/2009081/A-Deep-Dive-into-ASP.NET-Core-Localization>

<https://www.dutchtrans.co.uk/microsoft-translator-app-the-advantages-and-disadvantages/>

<https://azure.microsoft.com/en-us/pricing/details/cognitive-services/translator/>

<https://www.codeproject.com/Articles/8084/Creating-multilingual-websites-Part-2#databasedesign>

<https://cloud.google.com/translate/pricing>

<https://www.deepl.com/pro>

<https://softwareengineering.stackexchange.com/questions/391498/handle-localization-in-microservices-architecture>

[https://www.dbdiagram.io/](https://www.Dbdiagram.io/)

<https://www.reddit.com/r/microservices/comments/cbhdkn/localization_and_microservices_how_do_you_do_it/>