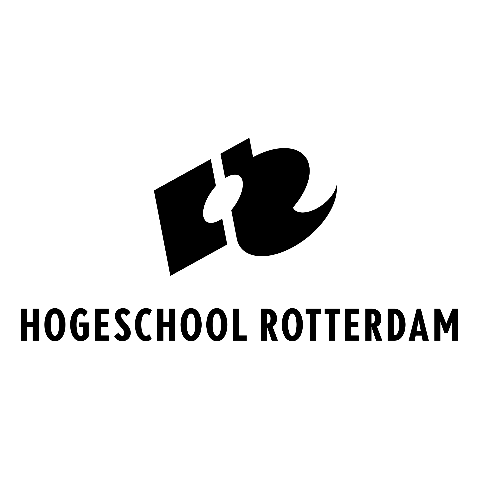
# **Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek**

# Minor IT-Innovatie voor Defensie en veiligheid

## Cyber Security – Web Scraper

## Ruben van Gemeren | 17-1-2022



# Inleiding:

Voor u ligt het verslag “Cyber Security – Web Scraper”. Dit verslag sluit aan op het project dat ik heb gevolgd tijdens de minor “IT-Innovatie voor Defensie en veiligheid” van de opleiding Technische Informatica van de Hogeschool Rotterdam (HR).

Ik ben Ruben van Gemeren. Ik ben een 4de jaar Informatica student bij de HR. Ik heb de eerste helft van mijn afstudeerjaar besteed aan onder andere deze opdracht. Samen met mijn bedrijfsbegeleider Erik Vullings heb ik dit project uitgewerkt en afgerond.

Het doel van dit verslag is om het project toe te lichten en om de volgende studenten die aan dit project werken te assisteren. Het verslag zal het concept web scraper uitleggen om vervolgens het ontwikkelde programma toe te lichten en aanbevelingen te geven voor verdere aanpassingen en werkzaamheden.

Ik wil mijn bedrijfsbegeleider meneer Vullings bedanken voor alle hulp tijdens deze periode. Ook wil ik mijn leeraren mevrouw Schavemaker-Piva en meneer Volders bedanken voor deze minor.

Inhoudsopgave

[**Nederlandse Organisatie voor toegepast-natuurwetenschappelijk onderzoek** 1](#_Toc93349825)

[Minor IT-Innovatie voor Defensie en veiligheid 1](#_Toc93349826)

[Cyber Security – Web Scraper 1](#_Toc93349827)

[Inleiding: 2](#_Toc93349828)

[Wat is een Web Scraper? 4](#_Toc93349829)

[Is Web scraping illegaal? 4](#_Toc93349830)

[Hoe werkt een Web Scraper 4](#_Toc93349831)

[Bestaande Web scrapers 4](#_Toc93349832)

[Calibre 4](#_Toc93349833)

[Twitterscraper van Taspinar 4](#_Toc93349834)

[Scrapedin van linkedtales 4](#_Toc93349835)

[Webscraper van webscraper.io 5](#_Toc93349836)

[Web scraper van TNO: 6](#_Toc93349837)

[Het doel: 6](#_Toc93349838)

[Hoe wordt de Web scraper gemaakt? 6](#_Toc93349839)

[Wat is TypeScript? 6](#_Toc93349840)

[Wat is Node.js 7](#_Toc93349841)

[Wat is Puppeteer? 7](#_Toc93349842)

[Het product: 8](#_Toc93349843)

[Opbouw: 8](#_Toc93349844)

[Het proces: 11](#_Toc93349845)

[Conclusie en volgende werkzaamheden: 13](#_Toc93349846)

[Aanbeveling voor vervolgproject: 13](#_Toc93349847)

# Wat is een Web Scraper?

Web scraping is een techniek waarbij data wordt verzameld van websites. Deze data wordt dan geëxporteerd in een bepaald formaat dat beter gebruikt kan worden. Hoewel het mogelijk is om de scraping handmatig te doen wordt er meestal gebruik gemaakt van een automatisch systeem. Een groot probleem binnen web scraping is het feit dat websites in veel verschillende smaken komen. Dit betekent dat er veel verschillende scrapers zijn die gespecialiseerd zijn voor bepaalde doeleinden.

## Is web scraping illegaal?

Web scraping zelf is niet illegaal. Er zijn wel een aantal regels waar rekening mee moet worden gehouden. Scrapen wordt illegaal als het gaat om data die niet openbaar is. Een nieuwswebsite mag wel gescraped worden, maar wachtwoorden niet.

## Hoe werkt een web scraper

Een geautomatiseerde web scraper zit eigenlijk vrij simpel in elkaar. Ten eerste worden er URL’s aan de scraper gegeven waar de informatie uit gehaald kan worden. De scraper laadt de hele HTML-code van de bron in, daarna zal de scraper de nodige data uit deze HTML-code halen en deze opslaan en exporteren naar een ander datatype dat beter gebruikt kan worden. Deze data kan gebruikt worden door een entity extractor, dit is een stuk software dat bepaalde elementen uit een dataset kan halen.

## Bestaande web scrapers

## Calibre

Calibre is een software tool die onder andere e-boeken kan managen. Niet alleen kan Calibre boeken managen, het is ook instaat om nieuws artikelen van verschillende bronnen automatisch te verzamelen. Dit is een functie die nuttig kan zijn voor ons. Calibre heeft een erg modulair ontwerp, dit betekent dat de meeste functionaliteit van plugins komt. Deze plugins zorgen ervoor dat er verschillende operaties uit kunnen worden gevoerd.

Het nadeel is dat de plugins gebouwd worden in Python. En wij gebruiken TypeScript voor onze web scraper.

## Twitterscraper van Taspinar

Dit is een stuk code dat gebruik maakt van de web Library “Beautifulsoup4” voor Python. Deze code kan de meeste informatie van een tweets ophalen, zoals de tekst in de tweet met de gebruiker en de datum waarop de tweet is gemaakt.

Helaas is deze code ook geschreven in Python en is daarom niet bruikbaar voor ons.

## Scrapedin van linkedtales

Scrapedin is een Linkedin scraper die gegevens van bepaalde gebruikers verzameld. Hoewel Linkedin het duidelijk heeft gemaakt niet blij te zijn met bedrijven die de profieldata verzamelen, is het nog niet gelukt om dit daadwerkelijk wettelijk vast te leggen. Sinds Linkedin profielen openbaar zijn is het toegestaan om data van deze pagina’s te verzamelen.

Deze scraper is gebouwd in JavaScript. Dit betekent dat het een goede basis kan zijn voor ons project.

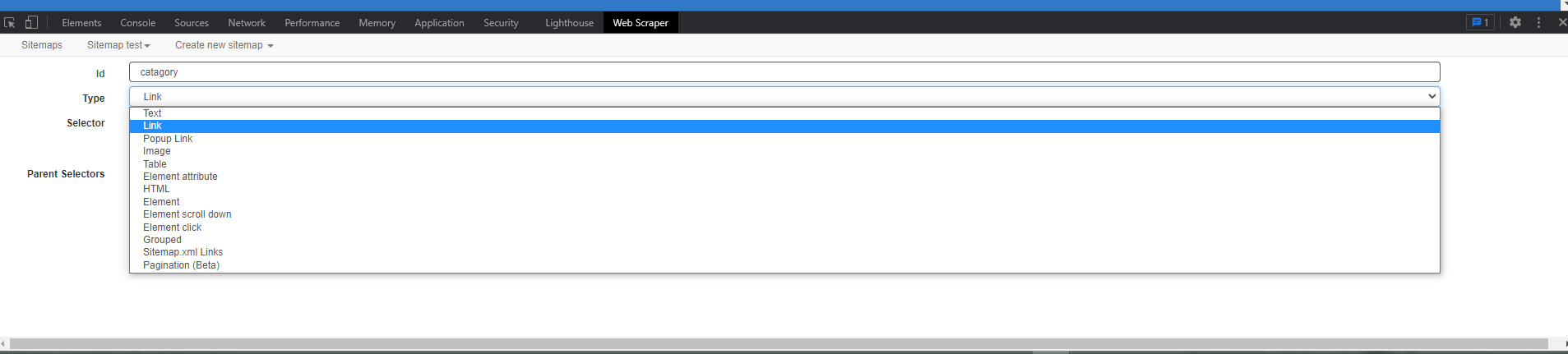
## 

## Webscraper van webscraper.io

Webscraper.io heeft een extensie gemaakt voor de Google Chrome browser waarmee je snel en eenvoudig grote hoeveelheden data kan verzamelen en exporteren. Deze gratis scraper is makkelijk te installeren in Chrome. Na het installeren kan je in de “Developer tools” (F12) aan de slag gaan met je nieuwe web scraper.

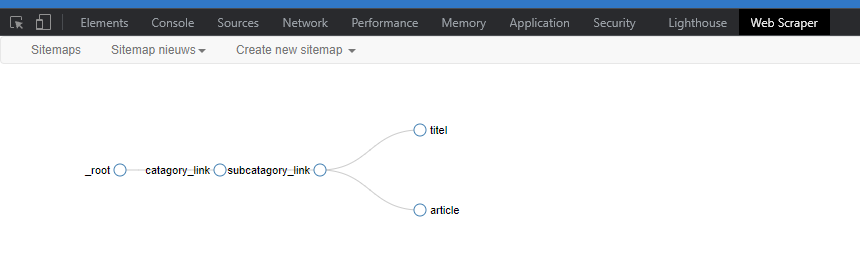
Deze web scraper werkt op basis van sitemaps. Een sitemap is een file waarin de alle pagina’s van een website staan en wat de relatie is tussen deze pagina’s. Google gebruikt zelf sitemaps om beter door websites te kunnen zoeken (ook wel Crawling genoemd).

In deze tool maakt je een sitemap van alle onderdelen waarvan je de data wilt verzamelen. De scraper zal alle onderdelen langs gaan en de aangegeven data verzamelen.



Figuur - Soorten items

In figuur 1 zijn alle soorten selectie items te zien. Deze selectie types worden gebruikt om verschillende soorten elementen te selecteren, maar ook om de website te kunnen navigeren. Zo wordt de “Link” selector gebruikt om naar een andere pagina op de website te kunnen linken.



Figuur - Een sitemap

In figuur 2is te zien hoe een uiteindelijke sitemap van nu.nl in elkaar zit. Deze sitemap begint op de hoofdpagina van nu.nl en gaat naar de “Economie” sectie, hier haalt de scraper van alle artikelen op die pagina de titel en het artikel. Het proces is zeer eenvoudig om op te zetten en er is weinig kennis van web scrapers nodig. Met de betaalde versie kan je zelfs een schema maken waardoor het scrapen automatisch gebeurt.

# Web scraper van TNO:

Vanuit TNO is er vraag naar een web scraper. Hiervoor heeft meneer Vullings een opdracht bedacht voor eventuele studenten die dit interessant zouden vinden. De opdracht heeft geen vaste scope en kan nog aangepast worden.

## Het doel:

Het doel van dit project is om een web scraper te ontwikkelen die gebruikt kan worden voor verschillende nieuwssites. Uit deze sites zouden wij graag alleen de nuttige informatie willen hebben, denk aan de titel en de datum waarop het artikel is geschreven. Er staat veel informatie op een nieuwswebsite die wij niet willen, zoals reclame of meldingen die geen extra waarde toevoegen.

Ook willen wij kijken naar een manier waarbij we verschillende bronnen kunnen scrapen met dezelfde code en waar we met de hulp van een centraal bestand de parameters van de scraper kunnen aanpassen.

De scraper wordt ontwikkeld met Typescript en maakt gebruik van de Chrome Dev tools API genaamd Puppeteer.

## Hoe wordt de Web scraper gemaakt?

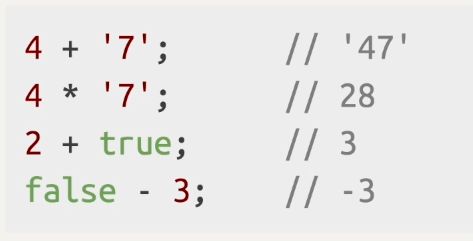
Dit zijn de gebruikte onderdelen:

* Typescript
* Nodejs
* Puppeteer

In een vervolgproject moet er ook gewerkt worden met Docker en Kafka, dit wordt later behandeld.

### Wat is TypeScript?

TypeScript is een “Strongly typed” versie van JavaScript. Maar wat betekent dat? Hoewel er geen concrete definitie is van een “Strongly typed” programmeertaal, is het verschil wel goed weer te geven. In het voorbeeld in figuur 3 is te zien wat een “Weakly typed” JavaScript doet met deze opdrachten.

Als je een getal (4) en een string (‘7’) bij elkaar wilt optellen maakt JavaScript er een string van. Maar als je dezelfde waardes wilt vermenigvuldigen krijg je een getal. Ook wordt de Boolean (true) gezien als het getal 1 en false als het getal 0.

Figuur 3 - "Weak" Typed JavaScript

Andere programmeertalen zullen bij deze opdrachten een error geven, omdat je niet waardes van verschillende types kunt optellen. Dit is het verschil tussen een “Strong” en een “Weak” taal.

JavaScript is van nature vrij “Weak”. TypeScript maakt het gebruik van JavaScript makkelijker. Door “Strong” typing toe te voegen wordt de code makkelijker te begrijpen en te debuggen.

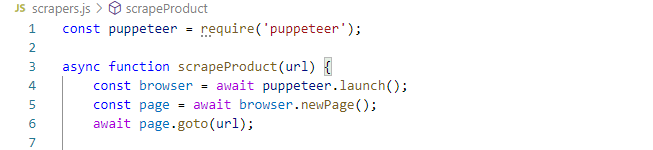
### Wat is Node.js

Node.js is een Framework waarmee je JavaScript kan gebruiken buiten de context van een browser. Het wordt veel gebruikt voor het maken van Backend services zoals API’s. Omdat Node.js JavaScript gebruikt kan iemand die veel ervaring heeft met de Frontend van een applicatie zonder veel problemen werken aan de Backend, dit betekent ook dat de code zelf schoner wordt omdat er maar één programmeertaal gebruikt wordt.

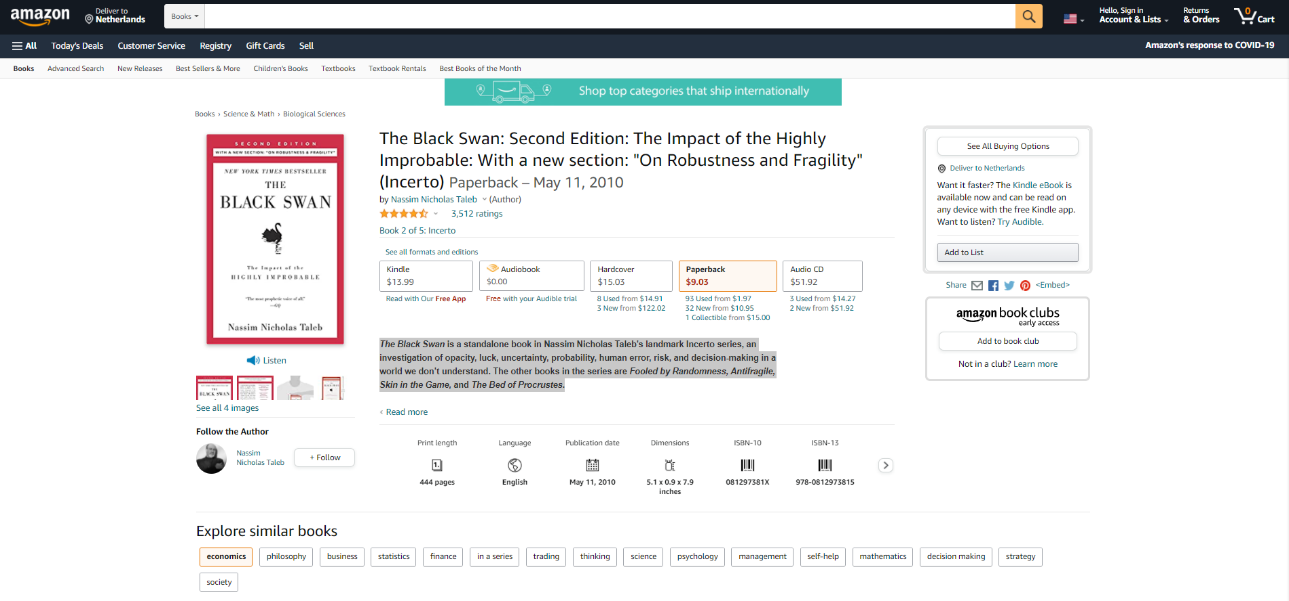
### Wat is Puppeteer?

Puppeteer is een Node.js API gemaakt door Google voor het gebruiken van Chrome. De API maakt gebruik van Headless Chrome. Dit is een manier om Chrome te gebruiken zonder dat je een Chrome webpagina te zien krijgt. Met deze versie van Chrome heb je meer controle over welke data je wilt gebruiken, dit maakt het ideaal voor het testen van applicaties zonder gebruik te moeten maken van de volledige versie van Chrome.

#### Voorbeeld:

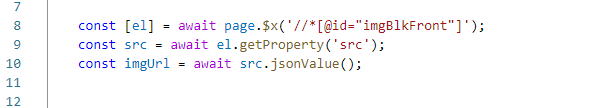
Het gebruik van Puppeteer is relatief eenvoudig. Voordat je Puppeteer kunt gebruiken moet je eerst in de code “require” gebruiken (Zie figuur 4). Dit zorgt ervoor dat je de Puppeteer daadwerkelijk kunt gaan gebruiken. De API werkt door een browser te openen met de hulp van Chromium. Dit is een versimpelde versie van de Google Chrome browser die veel gebruikt wordt om met het web te werken. Zodra de browser is opgestart kunnen we Puppeteer commands gebruiken om informatie van de meegegeven URL (Zie figuur 5) op te halen.

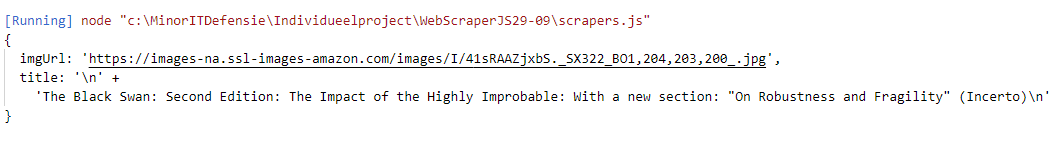
Figuur 4 - Require

Bij figuur 6is te zien hoe wij het plaatje van het product ophalen. Ten eerste hebben wij de XPath van het plaatje opgezocht op de webpagina, dit kunnen we doen door de pagina te inspecteren en het juiste element te kopiëren. De XPath is de syntax die wordt gebruik om elementen van onder andere webpagina’s af te leiden. Dit wordt gebruikt op elke website en kan worden gevonden door de pagina te inspecteren.

Figuur 5 - Amazon.com

De volgende stap is om de informatie uit het element te halen wat we zojuist hebben gescraped. Dit doen we door de “getProperty” te gebruiken. Omdat we een plaatje willen, gebruiken wij de ‘”src”, dit geeft ons de url van het plaatje. Als laatste gebruiken we “jsonValue” om de waarde om te zetten naar het JSON format. In figuur 7is te zijn welke data we krijgen. Deze data kunnen we opslaan en gebruiken voor verschillende doeleinden.



Figuur 6 - Voorbeeld code Puppeteer

Figuur 7 - Het resultaat van de Puppeteer code

## Het product:

Dit is het project aan het eind van de minor “IT-Innovatie voor Defensie en veiligheid”. Het is gebouwd met Visual Studio Code. De code is beschikbaar op de Github Repository van TNO. Waar ook verdere aanpassingen in moeten worden gedaan.

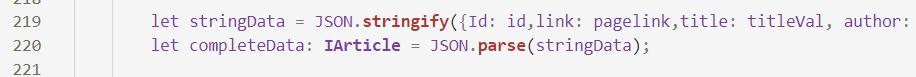
### Opbouw:

Het programma bestaat in totaal uit vijf verschillende Typescript bestanden die met elkaar verbonden staan via “Import statements”. De bestanden zijn:

* **index.ts**
  + Hier wordt de code mee aangestuurd. Het bevat de “calls” naar andere “classes” en wordt gebruikt als de index pagina, vandaar de benaming. In de index pagina kunnen de verschillende taken van het programma aangepast worden. Dit bestand bevat twee “classes” en twee functies.
    - *Timer()*
      * Deze class bevat code waardoor het programma wacht met scrapen. En gebruikt één variabel genaamd “counter” om voor een bepaalde tijd te wachten, de waarde van counter wordt geïnterpreteerd als x aantal minuten.
    - *Main()*
      * Deze asynchrone class bevat code die alle functionaliteiten van het programma aanstuurt. Zo wordt de browser opgestart en de belangrijke variabelen geïnstantieerd. Binnen deze class wordt alle data die is verzameld geëxporteerd naar een JSON-file.
    - *Delay()*
      * Deze asynchrone functie is eigenlijk niks anders dan een timer. Het gebruikt één variabel genaamd “ms” die gebruikt wordt als het aantal milliseconden dat het programma wacht. Deze class wordt gebruikt door de *Timer()* class.
    - Run()
      * Deze asynchrone functie wordt gebruikt om het hele programma te runnen. Het instantieert de *Main()* class en start het programma.
* **page.ts**
  + Hier worden meeste functionaliteiten van het programma uitgevoerd. Het bevat één class genaamd *Page()* die in het index.ts bestand wordt geinstantieerd als “scraper”. Onder deze naam worden er zeven functies aangeroepen.
    - *LoadPage()*
      * Deze asynchrone functie wordt gebruikt om de juiste webpagina in te laden. Deze waarde heet VarObj.WebPage en wordt ingeladen vanuit een config.json bestand. Ook wordt er een pop-up venster afgesloten die verschijnt op het scherm.
    - *LoadArticles()*
      * Deze asynchrone functie wordt gebruikt om de artikelen op de webpagina te halen. Er wordt op een knop gedrukt waardoor er meer artikelen worden ingeladen. Van deze artikelen wordt later de data verzameld. Ook hier worden variabelen gebruikt die terug te vinden zijn in de config.json.
    - *Collect()*
      * Bij deze asynchrone functie worden alle links van de ingeladen artikelen verzameld. Er wordt een lijst van links terug gegeven die later wordt gebruikt om van elke webpagina data te verzamelen.
    - *GetId()*
      * Deze asynchrone functie haalt van elke link twee belangrijke dataonderdelen, de titel en de datum waarop het artikel voor het laatst is geüpdatet. Deze data is belangrijk om later te kunnen checken of een artikel opnieuw gescraped moet worden of niet.
    - *GetData()*
      * Deze asynchrone functie gaat daadwerkelijk scrapen. In het index.ts bestand wordt deze functie in een loop opgeroepen waarbij de link wordt meegegeven. In deze functie wordt er naar de webpagina verwezen en worden de titel, de datum, de auteur en de content gehaald. Dit wordt omgezet naar een JSON-object. Dit object wordt later opgeslagen in een JSON-bestand.
    - *Update()*
      * Deze functie update een bepaald artikel. De functie krijgt alle data en het artikel dat geüpdatet moet worden. Het geeft de data terug met daarin het nieuwe artikel. Eerder in het proces wordt aangegeven of een artikel een update nodig heeft of niet.
    - *DeleteOldData()*
      * Deze functie kijkt aan het begin van het proces of er artikelen ouder zijn dan 15 dagen. Deze worden uit het JSON-bestand verwijdert. Op deze manier blijven oude artikelen niet hangen.
    - *TimeConverter()*
      * Dit is een interne functie die gebruikt wordt om een uitgeschreven tekst te veranderen in een daadwerkelijke datum. Op de webpagina staat wanneer er voor het laatst een update plaats heeft gevonden. Deze functie neemt die tekst en vertaalt dit naar een datum met hulp van Regular Expressions.
* **Compare.ts**
  + Hier worden twee functies uitgevoerd die te maken hebben met het vergelijken van data. In het index.ts bestand worden deze functies gebruikt om aan te geven of een bepaald artikel een update nodig heeft.
    - *CompareId()*
      * Deze functie krijgt een Id mee en geeft twee waardes terug. Of het artikel dat aan het Id vastzit kan worden gescraped en of het geüpdatet moet worden. Deze Boolean’s worden later gebruikt om de juiste functie aan te roepen.
    - *Updated()*
      * Dit is een interne functie die gebruikt wordt door *CompareId()* om aan te geven of een artikel geüpdatet moet worden.

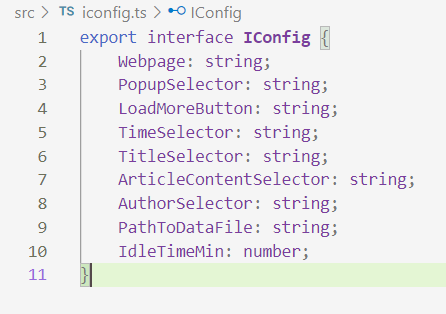
##### iarticle.ts en iconfig.ts:

Dit zijn allebei Typescript interfaces. Een interface is een soort blauwdruk van een “class”. In een interface geef je aan welke variabelen je wilt hebben en welk type ze zullen hebben, maar er wordt geen waarde gegeven. Als je dit importeert kan je deze gebruiken om een nieuwe instantie te maken van deze interface. Dit kan zo vaak als je wilt.

Tijdens het proces wordt er gewerkt met data die uit webpagina’s wordt gehaald. Hoewel wij zelf weten dat er een relatie is tussen al deze data weet de computer dat niet, daarom gieten we de data in een interface.

Figuur 8 - Een JSON object met type IArticle

De iarticle.ts interface wordt geïmporteerd en alle data krijgt het type “IArticle”. In figuur 8 is dit proces te zien. De variabel “completeData” heeft nu het type “IArticle” en krijgt de data van het artikel. Later in het proces hebben we dus een variabel die een vast type heeft dat altijd bestaat uit dezelfde waardes, hierdoor is het makkelijker om te werken met deze data. Ook is er een iconfig.ts interface. Deze wordt gebruikt om op een centrale plek bepaalde variabelen te hebben. Dit is handig omdat deze waardes dus maar op één plek aangepast hoeven te worden, in plaats van overal in de code. In figuur 9is deze interface (links) en de waardes die gebruikt worden (rechts) te zien.

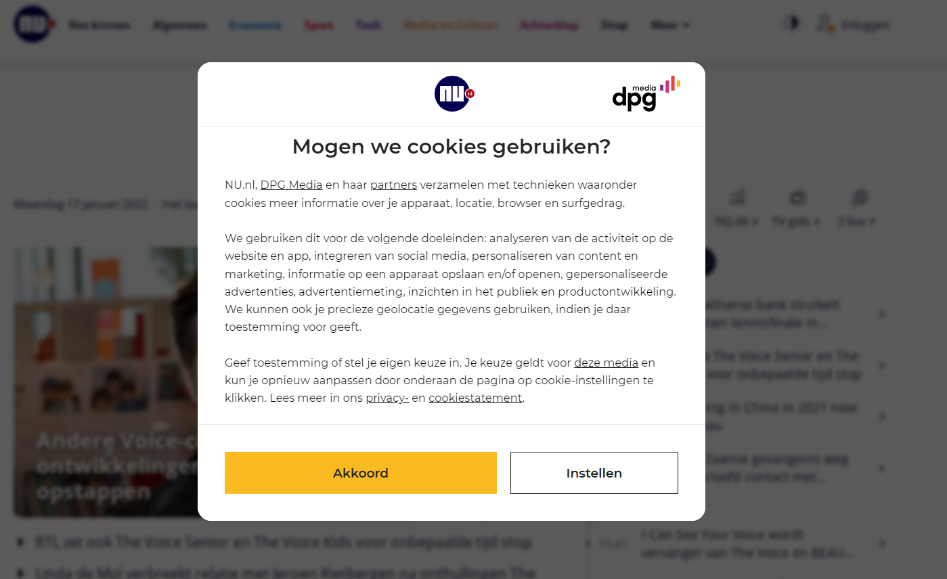


Figuur 9 - Het config bestand

### Het proces:

Om het programma te runnen heb je een terminal nodig. Dit kan de terminal zijn van Windows of Mac Os of een terminal in een editor zoals Visual Studio Code. Voordat je het programma kan starten moet je eerst “npm i” typen. Dit installeert npm en alle “packages” die gebruikt worden door het programma.

Hierna kan je het programma starten met “npm run node”.

Het programma zal een webpagina openen met een cookie melding, voordat het kan continueren moet deze melding geaccepteerd worden door de gebruiker. Het programma zal gelijk hierna starten. In figuur 10is te zien hoe deze melding weer wordt gegeven. Het proces zal doorgaan totdat de gebruiker het programma stopt met “ctrl + c” in de terminal.

Figuur 10 - Cookie melding

#### Het programma:

Het programma bestaat uit vier stappen, bij elke stap gebeuren en verschillende dingen om het proces goed te laten verlopen.

**Laden van de data:**

De eerste stap is het laden van de webpagina en de elementen die uiteindelijk worden verzameld. De manier waarop dit gebeurd is voor elke website anders, maar het moet gebeuren om data te kunnen verzamelen. In het programma wordt eerst de website geladen waarnaar er wordt genavigeerd naar de juiste pagina. Deze pagina is uitgekozen omdat deze een lijst met artikelen bevat die steeds wordt geüpdatet. Zonder deze pagina zou het programma op een andere manier data moeten verzamelen.

**Verzamelen van de artikelen:**

Na het laden van de juiste pagina worden de artikelen die we gaan verzamelen ingeladen en wordt er al wat data verzamelt. Dit gebeurt omdat we informatie nodig hebben over deze artikelen voordat we ze kunnen scrapen. We moeten kunnen bepalen of een artikel nieuw is of dat er een update plaats moet vinden. Zodra we dat weten kunnen we daadwerkelijk scrapen.

**Scrapen van de data:**

Per artikel wordt de juiste data verzameld, deze stap is vrij eenvoudig sinds we alle voorbereiding al hebben gedaan. We hoeven alleen nog maar de elementen te vinden op de pagina, dit kan met de hulp van de Chrome Dev Tools. Door de pagina te inspecteren kan je vinden welke “tags” een bepaald element op de pagina hebben. Deze “tags” vormen een “selector”, deze “selector” kan met Puppeteer worden gebruikt om dat element te vinden en verschillende acties uit te voeren.

**Opslaan van de data:**

Na het verzamelen van de data moet het worden opgeslagen. Het oorspronkelijke idee was om gebruik te maken van Docker en Apache Kafka.

Docker probeert een veelvoorkomend probleem op te lossen. Als een applicatie wordt ontwikkeld werkt het prima op de hardware van de Developer. Maar zodra een applicatie verplaatst wordt naar een andere locatie zijn er vaak problemen. Dit is een klassiek probleem binnen Software Development en Docker lost dit op door gebruik te maken van Containers. In een Docker Container zit alles wat de applicatie nodig heeft om te werken. Als iemand deze Container gebruikt zal het resultaat altijd hetzelfde zijn.

Docker containers kunnen ook op servers worden gezet, zodat deze altijd beschikbaar zijn. Docker is zeker een goede optie voor een applicatie zoals deze scraper. Helaas was er voor dit project geen tijd meer over om het allemaal op te zetten.

Apache Kafka is een platform waarmee je beter kan communiceren met andere applicaties, doordat je bepaalde “events” kan streamen. Een applicatie (producers) stuurt een bericht (records) naar Kafka. Andere applicaties kunnen deze berichten gebruiken door met Kafka te communiceren. Door Apache Kafka te gebruiken hoef je niet elke integratie tussen applicaties handmatig te bouwen. Je zorgt ervoor dat een applicatie bij Kafka kan en haalt daar je informatie op.

Voor de web scraper zou dit betekenen dat de applicatie zelf draait in de “cloud” in een Docker Container en de data van de applicatie beschikbaar is met Apache Kafka.

Tijdens dit project wordt de data van de scraper opgeslagen in een JSON-bestand, deze data bestaat uit losse JSON-objecten. Voor een vervolgproject is het opzetten van een Docker Container en een Kafka omgeving aangeraden.

# Conclusie en volgende werkzaamheden:

Voor dit project begon waren er verwachten over de resultaten en welke onderdelen er allemaal geïmplementeerd zouden worden. Door persoonlijke complicaties was er uiteindelijk niet genoeg tijd om alle onderdelen te implementeren.

Wel is er een applicatie die werkend is en doorlopend data verzameld van een nieuwswebsite. Deze applicatie kan gebruikt worden als basis om een systeem te creëren dat in de “cloud” kan draaien en waar via Kafka met meerdere applicaties mee kan worden gecommuniceerd.

## Aanbeveling voor vervolgproject:

Ik raad aan om dit project door te zetten, hierbij zijn er twee onderdelen die onderzocht moeten worden.

Ten eerste moet er worden gekeken naar een manier waarop deze applicatie (of een nieuwe applicatie) in de “cloud” kan worden gezet, hierbij is het gebruik van Docker aangeraden. Het gebruik van Apache Kafka kan ook een goede toevoeging zijn.

Hoewel deze onderdele al een verbetering zouden zijn van het project is het ook belangrijk om te kijken naar de manier waarop de scraper werkt.

In de huidige staat is het aanpassingsniveau gelimiteerd tot nieuwswebsites. Het is mogelijk om andere nieuwswebsites te gebruiken om data te verzamelen zonder de hele applicatie opnieuw te hoeven schrijven. Maar sommige methodes moeten worden vervangen omdat deze uitgaan van een bepaalde formulering van data.

Dit is iets wat erg moeilijk op te lossen is sinds alle websites anders zijn. Om dit op te lossen moet er dus naast het gebruik van Puppeteer gekeken worden of en andere “libraries” zijn die gebruikt kunnen worden om algemener te scrapen waardoor de applicatie bij meerdere webistes gebruikt kan worden.

Ik raad aan om daar de focus op te leggen.

Literatuurlijst

Google. (2021a, november 2). *Puppeteer | Tools for Web Developers |*. Google Developers. Geraadpleegd op 11 september 2021, van <https://developers.google.com/web/tools/puppeteer>

Google. (2021b, november 22). *What Is a Sitemap | Google Search Central |*. Google Developers. Geraadpleegd op 20 oktober 2021, van <https://developers.google.com/search/docs/advanced/sitemaps/overview>

Kulyk, O. (2021, 14 maart). *6 Puppeteer Tricks to Avoid Detection and Make Web Scraping Easier | ScrapingAnt Blog*. Scrapingant. Geraadpleegd op 3 november 2021, van <https://scrapingant.com/blog/puppeteer-tricks-to-avoid-detection-and-make-web-scraping-easier>

Programming with Mosh. (2018, 23 januari). *What is Node js?* [Video]. YouTube. <https://www.youtube.com/watch?v=uVwtVBpw7RQ>

*Puppeteer Not Finding Element iframe*. (2020, 15 juni). Stack Overflow. Geraadpleegd op 13 november 2021, van <https://stackoverflow.com/questions/62380628/puppeteer-not-finding-element-iframe>

Selenium. (2021, 11 december). *The Selenium Browser Automation Project*. Geraadpleegd op 25 september 2021, van <https://www.selenium.dev/documentation/>

T. (2020, 28 juli). *GitHub - taspinar/twitterscraper: Scrape Twitter for Tweets*. GitHub. Geraadpleegd op 17 september 2021, van <https://github.com/taspinar/twitterscraper>

Tech, C. (2018, 29 augustus). *iFrames in Puppeteer*. CherCherTech. Geraadpleegd op 7 november 2021, van <https://chercher.tech/puppeteer/iframes-puppeteer>

Wagner, L. (2021, 21 mei). *GitHub - linkedtales/scrapedin: LinkedIn Scraper (currently working 2020)*. GitHub. Geraadpleegd op 18 september 2021, van <https://github.com/linkedtales/scrapedin>