



Research Project Toegepaste Informatica



Voordelen van PWA's voor administratiesoftware

AON1

Emir Ozdemir

Fabio Ruffolo

Jonathan Vuurstaek

Maxim De Cuyper

Victor Tuerlinckx

Projectomschrijving

De opdracht van het researchproject is het bouwen van een applicatie om het proces van de stage-administratie te stroomlijnen. De administratie wordt hoofdzakelijk gedaan met Google Forms en daarnaast doet Gmail dienst als communicatiemiddel. Momenteel bestaat het takenpakket uit heel wat manueel werk door stageverantwoordelijke Marijke Willems. Het doel is een webapplicatie te maken die beide diensten integreert en bijgevolg op één plaats samenbrengt om het administratieproces te versnellen.

Het onderzoek van dit project behandelt PWA's of *Progressive Web Applications*. Hierbij wordt verduidelijkt wat PWA's zijn, hoe deze werken, waarin ze verschillen van gewone webapplicaties en welke voordelen ze bieden. De antwoorden op deze deelvragen worden verkregen aan de hand van een literatuurstudie. Vervolgens wordt de link met de projectcase gelegd in de hoofdvraag: "Op welke manier biedt een *Progressive Web Application* voordelen voor de stage-administratiesoftware?"

Voor de uitwerking van deze case wordt er gebruikgemaakt van Google Sheets en Google Mail aan de hand van API's. De frontend wordt gebouwd met HTML, CSS, JavaScript en Vue. Java en meer bepaald Springboot wordt gebruikt in de backend.

Inhoudsopgave

Projectomschrijving.....	ii
Inhoudsopgave.....	iii
Lijst van gebruikte afkortingen.....	iv
1 Onderzoeksvraag en hypothese.....	1
2 Onderzoeksmethode.....	2
3 Literatuurstudie.....	3
3.1 Wat is een PWA?	3
3.2 Hoe werkt een PWA?	4
3.3 Wat zijn de verschillen tussen PWA's en gewone applicaties?	4
3.4 Wat zijn de voor- en nadelen van PWA's?	5
3.5 Wat zijn de voordelen van PWA's voor de stage-applicatie?	6
4 Uitvoering.....	7
4.1 Bouw van een PWA	7
5 Conclusie	8
Bibliografie	9
Bijlagen.....	10

Lijst van gebruikte afkortingen

PWA	<i>Progressive Web Application</i>
PDF	<i>Portable Document Format</i>
HTML	<i>HyperText Markup Language</i>
CSS	<i>Cascading Style Sheets</i>
JSON	<i>JavaScript Object Notation</i>

1 Onderzoeksvraag en hypothese

Als onderwerp voor het researchproject is er gekozen voor de ontwikkeling van een stagebeheerapplicatie. Dit programma heeft als doel de verwerking van stageopdrachten te vereenvoudigen en te centraliseren.

In het oude systeem dient een bedrijf een stagevoorstel in aan de hand van een lijvig Google-formulier. Zodra deze info correct in de achterliggende lijst terechtkomt, stuurt de stageverantwoordelijke een bevestigingsmail naar het corresponderende bedrijf en wijst vervolgens een of meerdere reviewers toe aan de overeenkomstige opdracht. Ook de toewijzing gebeurt via een mail met daarin de opgave als PDF. Na de beoordeling van de opdracht wordt deze teruggestuurd naar de stageverantwoordelijke die op basis van de bijgevoegde feedback een van de volgende acties onderneemt: de opdracht dient bijgewerkt te worden op bepaalde aspecten en het bedrijf wordt hierover via mail ingelicht, de opdracht is in orde en het bedrijf ontvangt hierover een goedkeuringsmail of de opdracht voldoet niet en het bedrijf wordt op de hoogte gesteld dat deze werd afgekeurd. Indien het bedrijf de opgave aanpast, kan deze door de stageverantwoordelijke worden aangepast in de lijst en belandt deze, al dan niet, opnieuw in de reviewfase.

Het gehele proces uit de voorgaande alinea omvat veel werk met wisselen tussen verschillende applicaties en manuele arbeid om info aan te passen. De taak van dit project bestaat erin het proces samen te brengen in één applicatie die alle functionaliteiten omvat. Hierbij wordt er ook nog gevraagd om onderzoek te doen naar PWA's of voluit Progressive Web Applications. Tijdens dit onderzoek wordt er een antwoord gezocht op de hoofdvraag van dit project: welke voordelen kunnen PWA's bieden voor de stagebeheerssoftware? De deelvragen die hierbij gesteld en beantwoord worden, zijn de volgende:

- Wat is een PWA?
- Hoe werkt een PWA?
- Wat zijn de verschillen tussen PWA's en gewone applicaties?
- Wat zijn de voor- en nadelen van PWA's?
- Wat zijn de voordelen van PWA's voor de stage-applicatie?

2 Onderzoeksmethode

Voor de uitwerking van deze paper is er gekozen voor een combinatie van literatuurstudie en praktische demo's.

Op deze manier worden de PWA's eerst nabij bestudeerd om ze nadien in de praktijk toe te passen. Het doel van de literatuurstudie bestaat erin de deelvragen van dit onderzoek te beantwoorden en een globale kennis op te doen over PWA's.

Met deze verworven inzichten worden dan enkele praktische demo's opgezet om te testen hoe bruikbaar *Progressive Web Applications* zijn voor de stagesoftware. Het is echter niet de bedoeling dat deze demo's nadien ook effectief worden opgenomen in de applicatie. Zij dienen louter als *proof of concept* en bevestiging van de vooraf gestelde hypothese.

3 Literatuurstudie

3.1 Wat is een PWA?

Een PWA of een *Progressive Web Application* wordt door Wikipedia omschreven als een type van applicatie dat beschikbaar is via het web en dat gebruikmaakt van standaardtechnologieën als HTML, CSS, JavaScript en WebAssembly. Het hogere doel van PWA's is de intentie dat deze applicatie werkt op elk platform dat standaardbrowsers benut. [8]

Pete LePage en Sam Richard omschrijven PWA's als volgt: "An sich zijn *Progressive Web Apps* gewone webapplicaties. Ze maken gebruik van *progressive enhancement*, oftewel nieuwe mogelijkheden die mogelijk zijn in moderne browsers. Dankzij *service workers* en een webappmanifest wordt de applicatie betrouwbaar en installeerbaar. Indien deze nieuwe functionaliteiten niet beschikbaar zijn, kunnen gebruikers nog altijd genieten van de basiservaring." [3]

Uit de bovenstaande definities kan de essentie van PWA's worden afgeleid. *Progressive Web Apps* zijn webapplicaties die een verbeterde en meer universele ervaring willen voorzien op alle apparaten die beschikken over een moderne browser. Het doet dat door gebruik te maken van *service workers* en manifestbestand. Wat deze precies zijn en doen, wordt in het volgende hoofdstuk uitgelegd.

3.2 Hoe werkt een PWA?

Progressive Web Apps (PWA's) maken gebruik van verschillende technologieën om de eindgebruiker het gevoel van een *native* applicatie aan te bieden. Een van de eerste stappen om een webapplicatie om te vormen naar een *Progressive Web App*, is de aanwezigheid van een manifestbestand.

Een manifest-bestand is een JSON-bestand dat de browser inlicht hoe de webapplicatie zich moet gedragen na installatie op het apparaat van een gebruiker. Hoewel dit bestand enkel algemene informatie van een webapplicatie bevat, speelt het echter een belangrijke rol voor de imitatie als een *native* applicatie.

Een typisch manifestbestand bevat informatie over de naam van de app, de pictogrammen die deze moet gebruiken, de URL waarnaar gesurft moet worden wanneer de app opstart en nog veel meer. Zodra aan een bepaald criterium is voldaan, zullen de meeste browsers de bezoekers automatisch voorstellen om de applicatie te installeren.

Het belangrijkste element voor de werking van PWA's zijn de *service workers*. In essentie is een *service worker* een JavaScript-bestand dat wordt toegevoegd aan de code-base. Het bevindt zich tussen de server en de browser en voegt een nieuwe laag toe om applicatiefuncties na te bootsen. Een *service worker* voert zijn functies uit zonder nood te hebben aan een geopende webpagina of interactie met de gebruiker. Dit maakt applicatiewaardige functies mogelijk zoals mobiele meldingen, registratie van gebruikersacties terwijl ze offline zijn en ze afleveren terwijl ze online zijn, toegang tot de interne schijf, contactlijst, camera van het apparaat, enzovoort...

Tot slot is de compatibiliteit met de browsers een ander belangrijk onderdeel voor de werking van *Progressive Web Apps*. *Service workers* zijn de spil van PWA's, maar niet alle browsers ondersteunen deze. Safari bijvoorbeeld blijft vaak achter en wordt dikwijls gezien als de *Internet Explorer* van PWA's. Een gebrek aan ondersteuning voor specifieke functies belemmert het gebruik van *Progressive Web Applications* echter niet omdat het websites zijn. Bijgevolg zullen ze nog steeds werken in alle browsers, hoewel dit soms zonder de volledige set van functies zal zijn. [1], [2]

3.3 Wat zijn de verschillen tussen PWA's en gewone applicaties?

Naast *Progressive Web Apps* zijn er nog andere soorten applicaties, zoals bijvoorbeeld *native app*, *hybrid app* en een *responsive website*.

Een *native* applicatie wordt ontworpen voor een specifiek besturingssysteem en is bijgevolg geschreven in een specifieke programmeertaal voor dat besturingssysteem.

Een concreet verschil tussen een *progressive* en een *native* applicatie is dat bij de laatstgenoemde verschillende, compatibele versies voor elk besturingssysteem moeten worden gemaakt. Een *progressive* applicatie wordt één keer geschreven aan de hand van gemakkelijk toegankelijke frameworks en is beschikbaar voor alle verschillende besturingssystemen.

In tegenstelling tot een *native* applicatie is het voor de gebruiker mogelijk om *Progressive Web Apps* te installeren op bijvoorbeeld hun bureaublad. Deze mogelijkheid wordt in het algemeen positief ervaren door de eindgebruiker omdat het enkele voordelen met zich meebrengt zoals toegankelijkheid en snelheid. [3]

3.4 Wat zijn de voor- en nadelen van PWA's?

Progressive Web Apps hebben verschillende voordelen voor zowel de developers als de eindgebruiker. Zo ervaart de eindgebruiker een snellere service die gerealiseerd wordt door het feit dat *Progressive Web Apps* data kunnen opslaan in het cachegeheugen. Dit geheugen is een plaats waar data tijdelijk wordt bewaard en die veel sneller toegankelijker is in vergelijking met andere opslagplaatsen.

Een tweede voordeel van *Progressive Web Apps* is hun toegankelijkheid. Ze zijn verkrijgbaar op bijna alle apparaten en na een simpele installatie blijft de applicatie ten alle tijden beschikbaar op het bureaublad. Door middel van een simpele dubbelklik op het logo opent de applicatie zich en kan de gebruiker ermee aan de slag gaan.

Het onderhoud van *Progressive Web Apps* gebeurt automatisch. Er wordt geen verzoek naar de eindgebruiker verstuurd. Indien er een nieuwe versie van de applicatie beschikbaar is, kan de developer de vernieuwde versie live zetten op de server. De applicatie staat hiermee in contact en gaat automatisch de update installeren op een manier waarop de eindgebruiker er zo min mogelijk last van ondervindt. Dankzij deze technologie maakt de eindgebruiker te allen tijde gebruik van de beste functionaliteiten en de nieuwste services.

PWA's bevatten niet enkel voordelen voor de eindgebruiker, maar ook voor de developer. Over het algemeen zijn de ontwikkelingskosten van *Progressive Web Apps* lager dan die van een *native* webapplicatie. Voor niet-*Progressive Web Apps* dienen er echter drie verschillende applicaties ontwikkeld te worden, namelijk: een website, een iOS-app en een Android-App. Hiervoor worden dan drie keer de ontwikkelingskosten en de hostkosten betaald.

Daarnaast zijn *Progressive Web Apps* ook gemakkelijker te distribueren. Hoewel reguliere applicaties nood hebben aan Google Play of de App Store, kunnen *Progressive Web Apps* eenvoudig online gevonden worden en eenvoudig worden toegevoegd aan het bureaublad zonder tussenkomst van een medium.

Er zijn ook echter enkele nadelen. Zo zijn verschillende functionaliteiten hier niet toegankelijk, maar wel bij een *native* applicatie. Enkele voorbeelden hiervan zijn bluetooth, gps en camera's. Bovendien ondersteunen niet alle mobiele browsers *Progressive Web Apps*. De functionaliteiten die door PWA's niet gebruikt kunnen worden, zijn inmiddels standaard ingeburgerd bij de eindgebruiker. Hierdoor dient een weloverwogen keuze gemaakt te worden afhankelijk van het doel van de applicatie. [3]

3.5 Wat zijn de voordelen van PWA's voor de stage-applicatie?

Zoals eerder besproken zijn er veel voordelen die een *Progressive Web App* kan bieden. Maar welke voordelen zijn nu echt van toepassing op de stagesoftware?

Het eerste voordeel is dat de klant de software niet moet installeren op het apparaat waarop hij het wil gebruiken. De gebruiker kan simpelweg met de gegeven URL de Web App overal installeren en gebruiken wanneer hij maar wil.

Het tweede grote voordeel is de gebruiksvriendelijkheid van een PWA. Deze kan immers als snelkoppeling worden toegevoegd op het startscherm van het apparaat van de gebruiker waardoor deze makkelijk en snel opgestart kan worden.

Het derde voordeel dat zeer belangrijk is voor de stagesoftware is dat een PWA offline gebruikt kan worden. De gebruiker kan hierdoor altijd gebruik maken van de applicatie met de data die opgeslagen is. Wanneer het toestel weer online is, zal de data up-to-date gebracht worden. In het geval van de stagesoftware betekent dit dat de wijzigingen die aangebracht werden, doorgevoerd zullen worden naar de database. Nieuwe aanvragen en mails zullen opnieuw binnenkomen en mails die in de wachtrij gezet werden, zullen worden verzonden. De klant heeft aangegeven dat hij dit belangrijk vindt.

Ten slotte wordt bij een PWA de pagina van een applicatie in één keer volledig ingeladen en is de grootte van *requests* kleiner. Dit zorgt voor een enorme performancewinst. Hierdoor moet de gebruiker veel minder lang wachten en dat vergroot de gebruiksvriendelijkheid. [4], [5]

4 Uitvoering

4.1 Bouw van een PWA

Als developer een volledig nieuwe techniek implementeren is de eerste keer altijd spannend. Om het risico op onnodig werk uit de weg te gaan, werd er besloten een tutorial te volgen. [8], [9] Hierdoor is de manier van aanpak gestructureerd van bij het begin en is er een duidelijker einddoel waar naartoe gewerkt wordt.

De eerste benodigdheden voor de voorbeeldapplicatie zijn vanzelfsprekend de verschillende pagina's. Deze worden simpelweg in HTML (Figuur 1) aangemaakt en vervolgens wat opgesmukt met behulp van CSS. Als tweede stap wordt er een JavaScript-bestand geschreven om sommige HTML-elementen van inhoud te voorzien. In dit bestand zitten twee verschillende *fetch*functies waarvan de ene data opvraagt aan de hand van een URL en de andere data opvraagt die in de cache is opgeslagen. (Figuur 3) Welke functie wordt aangeroepen, wordt bepaald door het feit of er een internetverbinding is of niet. (Figuur 4)

Vervolgens wordt het manifestbestand gecreëerd wat van deze traditionele applicatie een PWA maakt. Dit bestand bevat info over de applicatie zoals de opstartnaam, displaynaam, startURL, achtergrondkleur en nog vele andere items die worden geraadpleegd bij het opstarten. Dankzij dit bestand is de applicatie op dit moment al een halve PWA. De andere helft heeft een *service worker* (Figuur 5).

Service workers zijn ook JavaScript-bestanden en werken als verbeteringen van een traditionele webapp. Het is echter de bedoeling dat de applicatie nog blijft draaien zodra de connectie van bepaalde onderdelen verloren gaat. Bij de implementatie in de voorbeeldapplicatie gaat de *service worker* controleren of de *fetch* over het correcte type en adres beschikt. Indien dat het geval is, gaat het de teruggegeven data opslaan in de cache waardoor dit bij toekomstige aanvragen zeer snel kan worden aangeboden. Dankzij deze *service worker* kan er dus nog steeds met de data gewerkt worden indien de verbinding met het internet onderbroken wordt. (Figuur 2) De enige taak die nog rest, is de registratie van de *service worker* in de *main.js*. Wanneer dat is gebeurd, is de applicatie een simpele, maar volwaardige PWA.

Dankzij het volgen van de tutorial werd duidelijk dat de bouw van een PWA heel eenvoudig kan zijn. Op gebied van moeilijkheid of kennisvereisten is een PWA dus zeker geen obstakel. Het voorgaande bewijst ook dat een PWA een doeltreffende oplossing is om een applicatie te bouwen die geen gegevensverlies ervaart vanaf het moment dat er geen verbinding meer is.

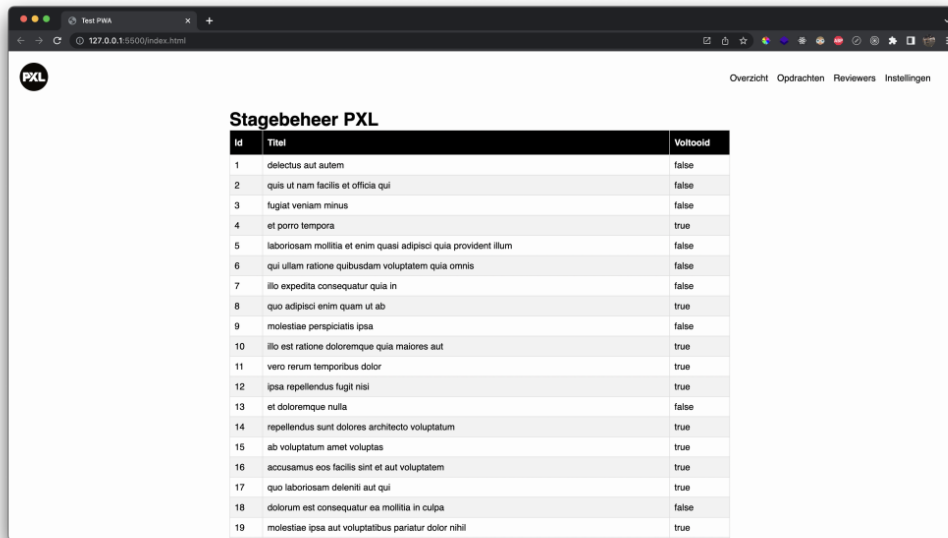
5 Conclusie

Ter conclusie kan gesteld worden dat een PWA een nieuw soort webapplicatie is waarbij *service workers* zorgen voor een verbeterde ervaring in moderne browsers. Het is eenvoudiger en goedkoper in ontwikkeling en onderhoud, wat zowel in het voordeel van de klant als de ontwikkelaar speelt. De offline interactie die nadien gesynchroniseerd wordt, is de grootste troef van een PWA voor de stagesoftware. Deze maakt het echter mogelijk om te allen tijde aanpassingen uit te voeren in het programma. Kortom, de toekomst van webapplicaties lijkt verzekerd en het ziet ernaar uit dat PWA's over enkele jaren alom tegenwoordig zullen zijn.

Bibliografie

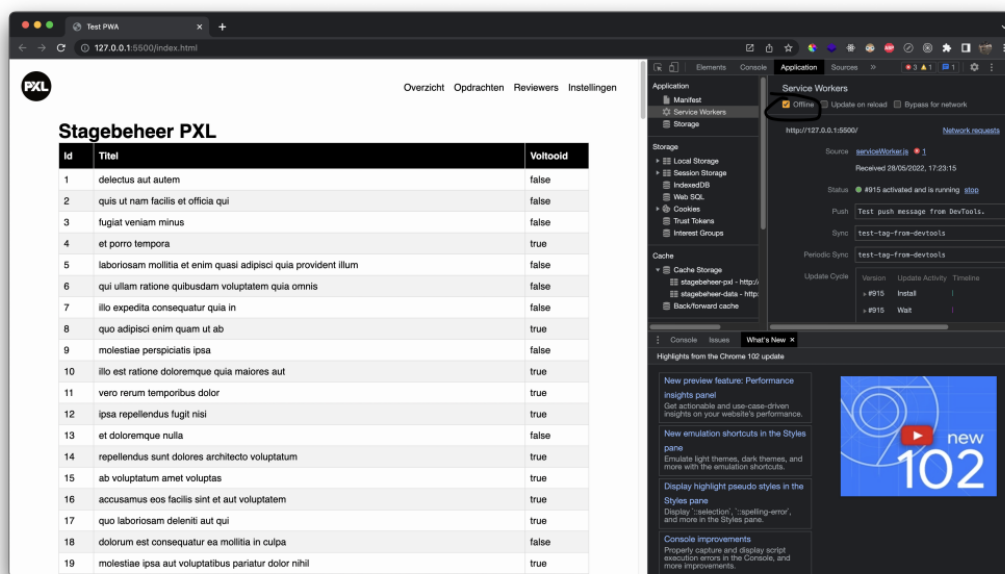
- [1] „Progressive web apps (PWAs) | MDN,” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction.
- [2] „Introduction to Progressive Web App Architectures | Google Developers,” [Online]. Available: <https://developers.google.com/web/ilt/pwa/introduction-to-progressive-web-app-architectures>.
- [3] „Progressive Web Apps,” [Online]. Available: <https://web.dev/progressive-web-apps/>.
- [4] „PWA (progressive web apps): de voordelen en nadelen op een rij,” [Online]. Available: <https://www.d-tt.nl/artikelen/pwa-progressive-web-apps-voordelen-nadelen>.
- [5] „Wat is een PWA en is dit iets voor mijn organisatie?,” 01 10 2021. [Online]. Available: <https://www.frankwatching.com/archive/2020/03/10/pwa-progressive-web-app-voordelen-nadelen/>.
- [6] „What are Progressive Web Apps?,” [Online]. Available: <https://web.dev/what-are-pwas/>.
- [7] Wikipedia, „Progressive web application,” [Online]. Available: https://en.wikipedia.org/wiki/Progressive_web_application. [Geopend 29 mei 2022].
- [8] MDN Contributors, „Introduction to progressive web apps,” 1 mei 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction.
- [9] MDN Contributors, „How to make PWAs installable,” 18 februari 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Installable_PWAs.

Bijlagen



Id	Titel	Voltooid
1	delectus aut autem	false
2	quis ut nam facilis et officia qui	false
3	fugiat veniam minus	false
4	et porro tempora	true
5	laboriosam mollitia et enim quasi adipisci quia provident illum	false
6	qui ullam ratione quibusdam voluptatem quia omnis	false
7	illo expedita consequatur quia in	false
8	quo adipisci enim quam ut ab	true
9	molestiae perspiciatis ipsa	false
10	illo est ratione doloremque quia maiores aut	true
11	vero rerum temporibus dolor	true
12	ipsa repellendus fugit nisi	true
13	et doloremque nulla	false
14	repellendus sunt dolores architecto voluptatum	true
15	ab voluptatum amet voluptas	true
16	accusamus eos facilis sint et aut voluptatem	true
17	quo laboriosam deleniti aut qui	true
18	dolorum est consequatur ea mollitia in culpa	false
19	molestiae ipsa aut voluptatibus pariatur dolor nihil	true

Figuur 1: index.html



The screenshot shows the Stagebeheer PXL application running in a web browser. The application is displayed in the main content area, showing a table with 19 rows and 3 columns: Id, Titel, and Voltooid. The table data is identical to the one in Figure 1. The browser's developer tools are open, showing the Application, Service Workers, and Console panels. The Service Workers panel indicates that the application is running offline, with the status 'Offline' and 'Update on reload' and 'Bypass for network' options. The Console panel shows a message 'What's New' with a 'new 102' badge.

Figuur 2: Applicatie zonder internetverbinding

```
1  const BASE_URL = "https://jsonplaceholder.typicode.com/todos";
2
3  const getTodos = async () => {
4    try {
5      const res = await fetch(BASE_URL);
6      return await res.json();
7    } catch (ex) {
8      console.log(ex.message);
9    }
10 };
11
12 const getTodosFromCache = async () => {
13   try {
14     const cache = await caches.open("stagebeheer-data");
15     const res = await cache.match(BASE_URL);
16     const todos = await res.json();
17
18     return todos;
19   } catch (ex) {
20     console.log(ex.message);
21   }
22 };
23
```

Figuur 3: Fetch in main.js

```
38
39  const main = async () => {
40    let todos;
41    if (navigator.onLine) {
42      todos = await getTodos();
43    } else {
44      todos = await getTodosFromCache();
45    }
46    fillTable(todos);
47  };
48
```

Figuur 4: Verbindingscontrole

```
17
18 self.addEventListener("fetch", (event) => {
19   if (
20     event.request.method === "GET" &&
21     event.request.url.indexOf("https://jsonplaceholder.typicode.com/todos") !==
22     -1
23   ) {
24     event.respondWith(
25       caches.open("stagebeheer-data").then((cache) => {
26         return fetch(event.request).then((response) => {
27           cache.put(event.request, response.clone());
28           return response;
29         });
30       });
31     );
32   }
33   event.respondWith(
34     caches.match(event.request).then((res) => {
35       return res || fetch(event.request);
36     })
37   );
38 });
39
```

Figuur 5: serviceworker.js