**BACHELORPROEF**

\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| **Single vs. Multi page applications** |
| Wanneer kies je wat? |



|  |  |
| --- | --- |
| Bachelor | Toegepaste Informatica |
| Keuzetraject | Software Engineer |
| Academiejaar | 2022 - 2023 |
| Student | Thibaut Wittevrongel |
| Interne begeleiders | Jill VandenDriessche, Dieter Mourisse (Howest) |

**BACHELORPROEF**

\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| **Single vs. Multi Page applications** |
| Wanneer kies je wat? |



|  |  |
| --- | --- |
| Bachelor | Toegepaste Informatica |
| Keuzetraject | Software Engineer |
| Academiejaar | 2022 - 2023 |
| Student | Thibaut Wittevrongel |
| Interne begeleiders | Jill VandenDriessche, Dieter Mourisse (Howest) |

Toelating tot bruikleen

De auteur(s) geeft (geven) de toelating deze bachelorproef voor consultatie beschikbaar te stellen en delen van de bachelorproef te kopiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de bepalingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze bachelorproef.

The author(s) gives (give) permission to make this bachelor dissertation available for consultation and to copy parts of this bachelor dissertation for personal use. In all cases of other use, the copyright terms have to be respected, in particular with regard to the obligation to state explicitly the source when quoting results from this bachelor dissertation.

4/06/2023

Woord vooraf

In het kader van mijn opleiding bachelor Toegepaste Informatica met keuzetraject Software Engineering heb ik de kans gekregen om een eigen scriptie te mogen schrijven. Dit als finale opdracht van mijn opleiding. De vraag waarop antwoord gegeven wordt gaat als volgt: ‘Single vs. Multi Page applications: wanneer kies je wat?’.

Deze onderzoeksvraag kwam uit de onderzoeksgroep van TI. Om deze vraag op te lossen heb ik na een uitvoerig kwalitatief en kwantitatief onderzoek een antwoord kunnen vinden op de onderzoeksvraag.

Graag zou ik als eerste mijn bachelorproef begeleiders, Dieter Mourisse en Jill VandenDriessche, lectoren aan Howest Brugge, van harte bedanken. Beiden stonden altijd klaar om mij feedback te geven en om me nieuwe inzichten te geven als ik op bepaalde punten even vastliep. Ook Heidi Terryn, lector aan Howest Brugge, wil ik graag in de schijnwerpers zetten. Zonder de lessen over het schrijven van een scriptie die ik kreeg van haar zou dit niet gelukt zijn.

Verder wil ook nog een dankwoord richten aan alle personen die hebben meegewerkt aan het testen van de applicatie. Zonder hen zou ik geen goed onderzoek hebben kunnen uitvoeren.

Ik wens u veel leesplezier toe.

Thibaut Wittevrongel

Sint-Joris, juni 2023

Samenvatting

Bij de samenvatting moet je je bachelorproef in een aantal lijnen samenvatten. Maak de kandidaatlezer van de bachelorproef nieuwsgierig zodat hij zin heeft om je bachelorproef grondig door te nemen. Hier kan ook een overzichtsfiguur geplaatst worden.

Sleutelwoorden: Onderaan de samenvatting - in een aparte paragraaf, voorafgegaan door ‘Sleutelwoorden:’ - plaats je de sleutelwoorden.

Abstract

Dit is de samenvatting in het Engels. Vergeet de sleutelwoorden niet.

Keywords:

Inhoudsopgave

**Woord vooraf**

**Samenvatting**

**Abstract**

**Inhoudsopgave**

**Lijst met figuren**

**Verklarende woordenlijst**

[1 Inleiding 10](#_Toc136799803)

[1.1 Algemeen 10](#_Toc136799804)

[1.1.1 Wat is een Single page application? 10](#_Toc136799805)

[1.1.2 Wat is een Multi page application? 11](#_Toc136799806)

[1.2 Probleemstelling 11](#_Toc136799807)

[1.3 Onderzoeksvraag 11](#_Toc136799808)

[1.4 Experiment 12](#_Toc136799809)

[1.5 Structuur van deze bachelorproef 12](#_Toc136799810)

[2 Theoretisch kader/Literatuuronderzoek 13](#_Toc136799811)

[3 Experiment 14](#_Toc136799812)

[3.1 Methodologie 14](#_Toc136799813)

[3.1.1 Uit welke elementen bestaat een typische webapplicatie? 14](#_Toc136799814)

[3.1.2 Welke applicatie wordt er ontwikkeld? 14](#_Toc136799815)

[3.1.3 De ontwikkelde API 15](#_Toc136799816)

[3.1.4 Ontwikkelen van de MPA 16](#_Toc136799817)

[3.1.5 Ontwikkelen van de SPA 19](#_Toc136799818)

[3.2 Onderzoeksverloop en data-analyse methode(n) 24](#_Toc136799819)

[3.2.1 User experience testing 24](#_Toc136799820)

[3.2.2 Verschillende soorten UX testen 24](#_Toc136799821)

[3.2.3 UX testen binnen dit onderzoek 25](#_Toc136799822)

[3.2.4 Performance testen 27](#_Toc136799823)

[3.2.5 Performance indicators 28](#_Toc136799824)

[3.2.6 Performance testing tools 28](#_Toc136799825)

[3.3 Resultaten 29](#_Toc136799826)

[3.3.1 Hoe verschilt de gebruikerservaring tussen een SPA en MPA? 29](#_Toc136799827)

[3.3.2 Hoe verschilt de SPA en MPA qua performance? 29](#_Toc136799828)

[3.3.3 Is er een groot verschil in development ervaring? 33](#_Toc136799829)

[3.3.4 Hoe verloopt het deployen van een SPA en een MPA? 33](#_Toc136799830)

[4 Conclusie 35](#_Toc136799831)

[5 Discussie 36](#_Toc136799832)

[Referentielijst 37](#_Toc136799833)

[Bijlagen 39](#_Toc136799834)

**Referentielijst**

**Bijlagen**

Lijst met figuren

[Figuur 1 MPA vs. SPA beginpagina vergelijken in KB 28](#_Toc136508666)

[Figuur 2 MPA vs. SPA totaal gedownloade data in KB 29](#_Toc136508667)

[Figuur 3 MPA vs. SPA laadtijd beginpagina in ms 29](#_Toc136508668)

Lijst met afkortingen

MPA – Multi Page Application

SPA – Single Page Application

VPS – Virtual Private Server

Verklarende woordenlijst

De verklarende woordenlijst is een opsomming van alle begrippen in je BAP die een beknopte uitleg nodig hebben.

In deze lijst som je de begrippen alfabetisch op en leg je de begrippen uit door een beknopte uitleg of definitie te geven. Dit ziet er vaak zo uit:

Begrip: Uitleg wat dit begrip betekent of definitie van het begrip.

Volgend begrip: Volgende uitleg.

# Inleiding

## Algemeen

Deze bachelorproef is tot stand gekomen vanuit een onderwerp dat is aangeboden uit het opleidingsteam Toegepaste Informatie van de Hogeschool West-Vlaanderen. Dit om een duidelijker antwoord te kunnen geven op de onderzoeksvraag ‘Single vs. Multi page applications: wanneer kies je wat?’.

Het onderstaande experiment gaat over het opzetten van desbetreffende applicaties om hierop een antwoord te formuleren.

### Wat is een Single page application?

Een Single Page Application, afgekort een SPA, is een website of webapplicatie die gemaakt is uit één webpagina. Wanneer een gebruiker surft naar een SPA krijgt de browser van de gebruiker alle broncode (HTML, JavaScript, CSS) binnen die nodig is om heel de website te laten werken tijdens het laden van de website. Wanneer deze SPA zeer groot of complex is kan dit ervoor zorgen dat de laadtijd die nodig is om deze code in te laden iets groter is dan een traditionele website. Maar na deze langere initiële laadtijd verloopt het navigeren op de site zelf wel veel sneller omdat er geen andere requests meer moeten gestuurd worden om verschillende pagina’s op te halen.

De manier waarop de navigatie werkt op een SPA is door middel van JavaScript. Met JavaScript kan de inhoud van de website worden bijgewerkt en lijkt het alsof de gebruiker toch aan het werken is op verschillende pagina’s terwijl de gebruiker zich nog altijd bevindt op de originele ingeladen pagina.

Als de SPA nood heeft aan data of inhoud die afkomstig is van een externe server kan er gebruik gemaakt worden van JavaScript om met bijvoorbeeld een fetch request een API of verschillende API’s aan te spreken. De data die daarna ontvangen wordt kan dan verwerkt worden door JavaScript om de inhoud van de SPA aan te passen naar het formaat dat nodig is op de website. Dit allemaal zonder dat de website een reload hoeft uit te voeren.

Bij het maken van een SPA zijn er verschillende mogelijkheden om de website te renderen in de browser. Twee van deze technieken zijn: Client-side rendering (CSR) en Server-side rendering (SSR)

**Client-side rendering (CSR)**

Wanneer een SPA gebruikt maakt van client-side rendering wordt er via de browser een request verstuurd naar de server om een HTML-bestand terug te krijgen samen met de scripts en stijlbestanden die gelinkt staan in het HTML-bestand. Wanneer de browser al deze bestonden heeft ontvangen begint de browser zelf met het uitvoeren van de JavaScript dat in de scripts staat. Dit zorgt ervoor dat de client eigenlijk al het werk zelf lokaal doet en wanneer dit af is, wordt alle inhoud van de website via JavaScript toegevoegd in het Document Object Model (DOM). Tijdens dit proces ziet de gebruiker meestal een laadanimatie of een witte pagina.

CSR duurt over het algemeen het langst om in te laden van de twee methoden. Dit omdat CSR veel resources van het toestel vergt om alles in te laden en om de pagina te genereren. Maar dit zorgt er wel voor dat de server zelf minder belast wordt waardoor er meerdere gebruikers tegelijkertijd de server kunnen aanspreken en zeer snel de benodigde bestanden verkrijgen van de server.

**Server-side rendering (SSR)**

Wanneer een SPA gebruikt maakt van server-side rendering wordt er via de browser een request verstuurd naar de server om een HTML-bestand te verkrijgen. Wanneer de server dit request binnenkrijgt, maakt de server zelf al heel het HTML-bestand op met de verwachte data en genereert al de volledige inhoud van de pagina. Hierdoor hoeft de SPA enkel nog maar de events te linken aan de pagina waardoor dat de browser van de gebruiker minder belast wordt.

Gebruik maken van SSR geeft als voordeel dat dit zeer snel werkt omdat er gebruikt kan gemaakt worden van de snelheid van de server gecombineerd met de snelheid van de browser van de gebruiker.

TODO ADD IMAGE SHOWING REQUEST LIFECYCLE

### Wat is een Multi page application?

Een Multi Page Application, afgekort een MPA, is een website die uit meerdere webpagina’s bestaat. Wanneer een gebruiker surft naar een MPA krijgt de gebruiker de gevraagde pagina te zien en worden alle bestanden die nodig zijn om deze pagina te laten werken gedownload. Wanneer de gebruiker hierna naar een volgende pagina klikt tijdens het navigeren op de MPA downloadt de browser terug de volledige volgende pagina en alle bestanden die hiervoor nodig zijn. Dit zorgt ervoor dat er veel meer data tussen de gebruiker en de server verstuurd wordt.

TODO ADD IMAGE SHOWING REQUEST LIFECYCLE

## Probleemstelling

Het probleem dat zich voordoet, is dat het lastig is om te kiezen tussen single page applicaties of multi page applicaties als structuur voor een website. Dit is een vraag die vaak wordt gesteld door personen of organisaties die een website willen bouwen en bezig zijn met het maken van een beslissing over de structuur van de site. Als het probleem niet wordt opgelost, kan dit leiden tot ontevredenheid met de gebruikerservaring van de website, een lage prestatie van de site, hogere ontwikkelings- en onderhoudskosten en een afname van het aantal bezoekers op de site. Het is daarom belangrijk om een weloverwogen beslissing te maken over de structuur van de site om deze problemen te voorkomen.

## Onderzoeksvraag

In deze bachelorproef wordt volgende onderzoeksvraag behandeld: “Single vs. Multi page applications: wanneer kies je wat?”. Om op deze vraag een goed antwoord te kunnen geven wordt deze opgedeeld in verschillende deelvragen:

1. Wat is een Multi page application?
2. Wat is een Single page application?
3. Uit welke elementen bestaat een typische webapplicatie?
4. Hoe verschilt de gebruikerservaring tussen SPA en MPA?
5. Hoe verschilt de SPA en MPA qua performance?
6. Is er een groot verschil in development ervaring?
7. Hoe verloopt het deployen van een SPA en een MPA?

## Experiment

Om een antwoord te formuleren op de onderzoeksvraag wordt er een experiment opgezet. In dit experiment worden er twee identieke applicaties ontwikkeld. Het enige verschil tussen de twee applicaties is dat de ene in de structuur van een MPA wordt gemaakt. De andere wordt gemaakt in de structuur van een SPA. Deze twee applicaties worden daarna vergeleken met elkaar om te zien welke voor- en nadelen beide structuren hebben. Dit om dan op het einde een conclusie te kunnen hebben over beide structuren.

## Structuur van deze bachelorproef

Eerst zal het kort gaan over de geraadpleegde literatuur dat binnen dit onderzoek gebruikt is. Daarna volgt een hoofdstuk over hoe het experiment is opgezet. Vervolgens worden alle resultaten overlopen. Als laatste volgt de conclusie die een antwoord geeft op de hoofdvraag en een kort overzicht over de volgende stappen van dit onderzoek.

# Theoretisch kader/Literatuuronderzoek

Om op de hoogte te zijn over de al uitgevoerde onderzoeken over MPA’s vs. SPA’s zijn er tijdens dit onderzoek artikels doorgenomen.[1]–[8]

In de meeste artikels worden vaak dezelfde zaken vergeleken met elkaar. Dit gaat dan vooral over de gebruikservaring, performance, development ervaring en de positieve en negatieve kanten van een MPA of SPA.

Een veel voorkomend onderwerp dat niet wordt aangehaald binnen dit onderzoek is search engine optimization (SEO). Een SPA zou namelijk veel minder goed scoren op vlak van SEO dan een MPA. Dit komt doordat de inhoud van een MPA statisch is waardoor het makkelijker is voor webcrawlers om de website een score te geven. Ook kunnen verschillende pagina’s van een MPA betere scores krijgen waardoor deze hoger in de search engine komen te staan dan de enkele SPA pagina.

Om de scope van dit onderzoek te beperken is er in binnen dit onderzoek niet dieper op SEO ingegaan.

Vervolgens werd tijdens dit onderzoek ook veel de MDN-documentatie geraadpleegd. Deze website geeft veel informatie rond allerlei zaken binnen web development.[9]

# Experiment

## Methodologie

### Uit welke elementen bestaat een typische webapplicatie?

Om te kunnen beginnen aan dit experiment moet er eerst gedacht worden aan de elementen waaruit een typische webapplicatie bestaat. Dit zodat de applicaties die ontwikkeld worden tijdens dit onderzoek zo dicht mogelijk tegen applicaties liggen die te vinden zijn op het web.

Het eerste element waaruit een webapplicatie bestaat is de effectieve content die een bedrijf of persoon wil tonen. Dit zijn bijvoorbeeld pagina’s met informatie over een product of een bepaald onderwerp. Deze pagina’s zijn dan vooral opgebouwd uit tekstelementen en foto’s.

Een volgend element dat veel voorkomt op verschillende websites is een vorm van navigatie. Dit kan bijvoorbeeld een navigatiemenu zijn of door een knop die een user navigeert naar een ander deel van de website.

Een vorm van user input is een volgend element dat op zeer veel sites terugkomt. Dit kan bijvoorbeeld tekst input, een lijst checkboxen of een formulier zijn.

Ten vierde is een vorm van authenticatie. Heel veel websites hebben een manier om hun users te kunnen herkennen en om persoonlijke content van gebruikers bij te houden. Hier hoort dan ook natuurlijk een manier bij om als gebruiker een eigen account aan te maken.

Heel veel websites bestaan dus uit bovenstaande elementen. Daarom zal er in dit onderzoek zeker voor gezorgd worden dat alle elementen die vernoemd zijn aanwezig zullen zijn in de applicaties.

### Welke applicatie wordt er ontwikkeld?

Een website wordt natuurlijk niet ontwikkeld zonder een bepaalde use case. Daarom is er in dit onderzoek gedacht aan een onderwerp dat in beide een SPA en MPA gegoten kan worden. Belangrijk is dat het idee voldoende breed genoeg is om ervoor te zorgen dat alle elementen van een typische webapplicatie toepasbaar zijn.

Het concept van de webapplicatie is dat de gebruiker een lijst van maaltijden kan opvragen die gebruikt kan worden als weekmenu. Na dat een weekmenu is opgevraagd kan de gebruiker de recepten bekijken van elke maaltijd. Ook een boodschappenlijst met alle ingrediënten van het weekmenu kan worden opgevraagd in de applicatie.

Wanneer de applicatie geopend wordt is het eerste element dat een gebruiker zal zien een formulier. In dit formulier kan de gebruiker aangeven van welke maaltijdperiodes de gebruiker maaltijden wil ontvangen. Daarna kan ook het aantal maaltijden per periode ingegeven worden. Na het verzenden van het formulier vindt ook de allereerste navigatie plaats. De gebruiker wordt doorgestuurd naar een pagina met gegenereerde maaltijden.

Op deze pagina kan de gebruiker aan de hand van knoppen maaltijden opnieuw genereren en navigeren naar het recept van de maaltijd. Wanneer de gebruiker blij is met het volledige menu kan deze opgeslagen worden. Aan een menu kan er een naam meegegeven worden, hier bevindt zich een tekstinput.

Wanneer een menu wordt opgeslagen wordt er gecontroleerd of de gebruiker ingelogd is. Als dit niet het geval is wordt de gebruiker doorgestuurd naar de login pagina. Hier kan dan ingelogd of een nieuw account aangemaakt worden. Authenticatie wordt hier gebruikt om opgeslagen menu’s te linken aan de gebruiker.

Na dat een menu aangemaakt is kan een gebruiker deze opnieuw opvragen via de profielpagina. Op deze pagina is er ook de mogelijkheid om uit te loggen. Wanneer een opgeslagen menu geopend wordt kunnen alle recepten terug geraadpleegd worden. Ook is er een knop die wanneer erop geklikt wordt de user navigeert naar de ingrediëntenpagina. Deze pagina bevat alle ingrediënten die nodig zijn om heel het menu te maken. Deze lijst gedownload worden.

### De ontwikkelde API

Om authenticatie en externe data beschikbaar te maken in een webapplicatie wordt er vaak een Application Programming Interface (API) gebruikt. Een API is eigenlijk een stukje technologie die ervoor zorgt dat applicaties met elkaar kunnen communiceren. In dit onderzoek zorgt de opgezette API er dus voor dat er communicatie tussen de webapplicatie en de server mogelijk is.

Er zijn tal van manieren om een API op te zetten. Zo is er de keuze om een Simple Object Acces Protocol (SOAP), Remote Procedure Call (RPC) of een Representational State Transfer (REST) API te maken. In dit onderzoek is er gebruik gemaakt van de REST-structuur.[10]

De gekozen programmeertaal om deze API mee te ontwikkelen is PHP. Meer specifiek met het Laravel framework. De reden om Laravel te kiezen was omdat deze uitgebreid tijdens de opleiding is gezien en dit framework alle mogelijkheden aanbiedt die nodig is voor dit onderzoek.

Een API bestaat uit verschillende endpoints. Deze endpoints worden allemaal opgenomen in een API-specificatie. Aan de hand van dit document kunnen developers weten welke endpoints er toegankelijk zijn en welke parameters eventueel meegeven moeten worden.

Een end-point ziet er als volgt uit:

Afbeelding met tekst, Lettertype, schermopname, lijn

Automatisch gegenereerde beschrijving

Zoals te zien op de afbeelding bestaat een end-point uit volgende elementen:

* Base-URL: Dit is de URL van de server waarop de API gehost staat.
* Endpoint: Dit is het pad dat gedefinieerd staat binnen de API-specificatie om een bepaalde actie uit te voeren.
* Query’s: Aan een endpoint kunnen vaak query’s meegegeven worden. De eerst query wordt altijd gestart met een vraagteken. Daarna als er nog query’s toegevoegd moeten worden kan dit gedaan worden met een ampersand als scheidingsteken.

Hieronder een kort overzicht van alle endpoints die ontwikkeld zijn in de Laravel API:

* POST /login

Deze endpoint wordt gebruikt op de gebruiker te laten inloggen. De developer kan via dit request de gebruikersnaam en het wachtwoord meesturen. Daarna geeft de API een response terug met een JWT-token als het request succesvol was of met een error code als het mislukt was.

* POST /logout

Met het /logout endpoint kan de developer de gebruiker laten uitloggen. Wanneer dit request binnenkomt in de API verwijdert deze de JWT-token.

* GET /recipes/{ID}

Wanneer de developer deze endpoint oproept, geeftde server het recept terug dat het meegegeven ID bevat.

* GET /recipes/random/{params}

Met dit endpoint kan er door middel van parameters een lijst van willekeurige recepten als response terug gestuurd worden.

* GET /menus

Dit endpoint kan gebruikt worden om alle opgeslagen menu’s van een gebruiker op te halen.

* POST /menus

Met de POST-versie van het /menus endpoint kan er een nieuw menu aangemaakt worden voor een bepaalde gebruiker.

* GET /recipes/in-menu/{menu-ID}

Dit endpoint kan gebruikt worden om alle recepten van een menu op te halen.

* GET /recipes/in-menu/{menu-ID}/ingredients

Met dit endpoint is het mogelijk om alle ingrediënten die nodig zijn om een bepaald menu klaar te maken op te halen.

Deze API wordt ook gehost op dezelfde server als waar de SPA en MPA staan.

### Ontwikkelen van de MPA

De eerste webapplicatie die ontwikkeld wordt tijdens dit onderzoek is de MPA. Deze applicatie wordt enkel ontwikkeld met HTML, SCSS en JavaScript. De reden hiervoor is om te zorgen dat frameworks of externe libraries de applicatie niet zouden kunnen beïnvloeden.

De MPA is opgebouwd uit 9 verschillende HTML-pagina’s. De eerste pagina die wordt ingeladen is de index pagina. Deze pagina bestaat uit een navigatie-element naar de login pagina en een formulier om een menu te genereren.

Afbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijving

Na dat een gebruiker een menu genereert op de index pagina wordt de gebruiker doorgestuurd naar de tweede pagina. De tweede pagina is de menuCreator pagina. Hier kan een gebruiker een menu aanpassen naar eigen voorkeuren en kunnen de recepten van het menu geraadpleegd worden.

Afbeelding met schermopname, tekst, lijn

Automatisch gegenereerde beschrijving

Wanneer een recept geraadpleegd wordt opent de derde pagina. Dit is de receptpagina van het aangeklikte gerechtje.

Afbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijving

De gebruiker kan een menu opslaan op de menuCreator pagina. Hiervoor moet een gebruiker wel ingelogd zijn. Als dit niet geval is wordt de gebruiker automatisch doorgestuurd naar de login of registerpagina.

Wanneer de gebruiker hierna succesvol een account aanmaakt of inlogt, komt de savedMenus pagina tevoorschijn. Hier vindt een gebruiker alle opgeslagen menu’s terug.

Afbeelding met schermopname, tekst, lijn, Lettertype

Automatisch gegenereerde beschrijving

Een menu kan hier dan open geklikt worden en alle recepten kunnen terug geraadpleegd worden. Deze pagina heet de menupagina. Hier heeft de gebruiker ook de mogelijkheid om een ingrediëntenlijst te openen en deze eventueel te downloaden.

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving

Als een gebruiker de opgeslagen recepten wil raadplegen kan dit ook via de profielpagina. Hier heeft de gebruiker de mogelijkheid om uit te loggen en om de opgeslagen recepten op te halen.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Bij de MPA van dit onderzoek is gekozen om geen verschillende pagina’s te maken per recept. Dit om de scope van het experiment te groot te maken. Hierdoor is er gekozen om op de receptpagina een generale structuur op te bouwen die daarna ingevuld wordt met de data die uit de API verkregen wordt.

### Ontwikkelen van de SPA

De tweede webapplicatie die ontwikkeld wordt tijdens dit onderzoek is de SPA. Net zoals de MPA is deze volledig ontwikkeld met HTML, SCSS en JavaScript.

De SPA bestaat uit een enkele HTML-pagina. De manier waarop dit werkt, is dat door middel van JavaScript de inhoud van de pagina veranderd kan worden. Dit zorgt ervoor dat de gebruiker het gevoel heeft dat er verschillende pagina’s zijn terwijl de gebruiker zich nog altijd op dezelfde pagina bevindt.

Bij een SPA komt er qua development wel meer bij kijken. Zo moet er bijvoorbeeld veel meer rekening gehouden worden met de flow van de applicatie. Dit omdat de gebruiker niet meer kan navigeren naar een andere pagina. Dit moet allemaal gebeuren door middel van JavaScript. Ook moet er rekening gehouden worden dat de back- en forwardknoppen van de browser niet meer werken zoals bij een MPA. Nu moet dit door de developer volledig zelf bijgehouden worden.

Bij een SPA zijn er drie vragen die gesteld kunnen worden:

1. Hoe kan de HTML aangepast worden?
2. Hoe kan de navigatie behouden worden net zoals een MPA?
3. Hoe kan er data van de server gebruikt worden binnen de SPA?

Om de HTML aan te passen in een webapplicatie zijn er verschillende methoden. Zo is een methode om met queryselectors in JavaScript bepaalde elementen te verwijderen of aan te passen. Ook is het mogelijk om met JavaScript nieuwe HTML-elementen te creëren en deze dan in het HTML-document te plaatsen.

Maar er bestaat een nieuwe manier om een document aan te passen. Deze manier maakt gebruik van HTML <template> elementen. Een template-element kan worden gebruikt om HTML die nog niet direct nodig is op de webpagina op te slaan. De MDN-documentatie[11] definieert de template tag als volgt:

“Het <template> HTML-element is een mechanisme voor het opslaan van HTML die niet onmiddellijk moet worden weergegeven wanneer een pagina wordt geladen, maar die later tijdens runtime met behulp van JavaScript kan worden geconcretiseerd.

Zie een <template> als een inhoudsfragment dat wordt opgeslagen voor later gebruik in het document. Hoewel de parser de inhoud van het <template>-element verwerkt tijdens het laden van de pagina, doet hij dat alleen om ervoor te zorgen dat die inhoud geldig is; de inhoud van het element wordt echter niet weergegeven.”

Met deze template-elementen kan er op een propere manier HTML bijgehouden worden. Een voorbeeld hoe dit gebruikt is tijdens dit onderzoek gaat als volgt.

In de applicatie wil de gebruiker inloggen. Deze HTML staat momenteel niet zichtbaar in de browser, maar zit mee in de HTML als een template tag.

<template id="loginFormTemplate">

<form action="index.html" method="post" class="auth-form">

<label for="email">Email</label>

<input type="text" name="email" id="email">

<label for="password">Password</label>

<input type="password" name="password" id="password">

<input type="submit" class="button" value="Login">

<p>Don’t have an account? <a href="#register">Register here!</a></p>

</form>

</template>

Wanneer de gebruiker een bepaalde actie uitvoert, is het mogelijk om via JavaScript de HTML in dit template-element zichtbaar te maken. Een voorbeeld hiervan is volgende code:

function showLoginForm(){

clearMain();

const loginFormTemplate = document.querySelector("#loginFormTemplate");

const loginFormHtml = loginFormTemplate.content.firstElementChild.cloneNode(true);

loginFormHtml.querySelector("a").addEventListener("click", showRegisterForm);

loginFormHtml.querySelector('input[type="submit"]').addEventListener('click', login);

document.querySelector("main").appendChild(loginFormHtml);

}

Het eerste wat gebeurt wanneer deze functie wordt opgeroepen is dat de functie ‘clearMain()’ wordt uitgevoerd. In dit project zorgt deze functie ervoor dat het HTML-element <main> volledig leeg gemaakt wordt.

Daarna wordt met behulp van een querySelector het gewenste template-element opgehaald. Hierna wordt de inhoud van het template-element, de HTML die zichtbaar gemaakt moet worden, gekloond in een andere variabele.

Vervolgens worden er aan de verschillende elementen binnen de opgehaalde HTML eventListeners toegevoegd. Als laatste wordt de nieuwe HTML toegevoegd aan het main-element.

Deze manier van werken kan worden toegepast op elk stuk HTML die pas later zichtbaar mag zijn in de flow van de applicatie. Ook wordt de template tag volgens de MDN-documentatie op elke browsers ondersteund.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Een tweede topic die zeker aangehaald moet worden bij het ontwikkelen van een SPA is de navigatie. Sinds er maar één HTML-pagina is zal de standaard navigatie die werkt binnen een MPA niet werken in dit geval.

Een oplossing voor dit probleem is om gebruik te maken van de History API.[12] Deze API geeft toegang tot de browsers session history. Dit betekent dat een developer met behulp van deze API zelf volledig de session history kan bepalen en de navigatie regelen binnen een SPA. Wanneer er gebruik gemaakt wordt van de History API en er worden sessions opgeslagen in de history werken de back- en forwardknoppen zoals ze in een MPA werken.

Aan de hand van een voorbeeld uit de code van de SPA zal uitgelegd worden hoe de History API gebruikt kan worden.

Het allereerste wat moet gedaan worden om te werken met de History API is het maken van een functie. Deze functie zal als functionaliteit hebben om de pagina waarvan weg genavigeerd wordt op te slaan. Dit kan met volgende code:

function savePreviousPage(nameOfPage, url){

const currentState = {page: nameOfPage}

window.history.pushState(currentState, "", url);

}

Deze code neemt als parameters de naam van de pagina en de URL van deze pagina. Daarna wordt de naam van de pagina opgeslagen in een object binnen een variabele. Hierna wordt dan de History API aangesproken. De functie pushState zorgt ervoor dat de pagina toegevoegd wordt aan de browsers history session.[13] Deze functie neemt drie parameters:

1. State, dit is een willekeurig JavaScript object dat de developer kan opslaan in de browsers history stack.
2. Unused, de tweede parameter die de functie verwacht, wordt eigenlijk niet meer gebruikt. Deze is er wel nog door historische redenen. De veiligste manier om hier mee om te gaan is door een lege string mee te geven.
3. URL, de laatste parameter die kan meegegeven worden is een URL. Deze URL hoeft niet absoluut te zijn. Wanneer het een relatieve URL zou zijn, wordt deze ten opzichte van de huidige URL toegepast.

Na dat deze functie uitgevoerd is staat de data opgeslagen binnen de browsers history stack. Deze functie wordt nu gebruikt bij elke navigatie tussen verschillende delen van de website.

Nu moeten er nog twee functies voorzien worden om de volledige flow van de navigatie te laten werken.

Het eerste wat moet gebeuren is een evenListener toevoegen aan de browser window. Dit kan als volgt:

window.addEventListener('popstate', (event) => {

replaceMain(event.state.page);

});

Wat in bovenstaande code gebeurt, is dat aan de browser window een eventListener gehangen wordt. Deze luistert naar het event ‘popstate’. Het ‘popstate’ event wordt getriggerd elke keer dat er een verandering is met de history stack van de browser. Wanneer dit gebeurt, wordt de replaceMain functie uitgevoerd.

De replaceMain functie ziet er als volgt uit:

function replaceMain(page){

if(page === "menuCreator"){

initialiseMenuCreator();

} else if (page === "profile"){

showProfileScreen();

} else if (page === "savedMenus"){

initialiseSavedMenus();

}

else if (page === "viewSavedMenu"){

initialiseViewMenu();

}

}

Uit de vorige functie wordt aan bovenstaande functie de paginanaam meegegeven. De replaceMain functie loopt dan alle mogelijkheden af om een bepaalde actie uit te voeren. Dit alles samen zorgt ervoor dat de navigatie hetzelfde werkt als bij een MPA.

Een derde topic binnen een SPA is het verkrijgen van de data. Dit kan door middel van de Fetch API.[14] Met de methode fetch kan binnen een webapplicatie data opgehaald worden van een externe server. Aan dit request kunnen dan ook headers meegegeven worden. Hieronder een voorbeeld over hoe een fetch request eruitziet.

async function APIgetMenuRecipes(id){

return fetch(`${API\_URL}/recipes/in-menu/${id}`, {

method: "GET",

headers: {

"Content-Type": "application/json",

"Authorization": "Bearer " + "JWT Token"

}

})

.then(response => response.json())

.then(data => { return data; });

}

Binnen een fetch request wordt een URL meegegeven. Ook een object met aangepaste instellingen kunnen meegeven worden zoals bijvoorbeeld: method, headers, body…

Vervolgens krijgt het fetch request een response terug en kan daarna deze response verwerkt worden.

Al deze bovenstaande technieken zijn in dit onderzoek gebruikt. Dit heeft ervoor gezorgd dat de SPA een zeer gelijkaardige functionaliteit heeft aan de MPA.

## Onderzoeksverloop en data-analyse methode(n)

### User experience testing

Tijdens dit onderzoek worden er user experience (UX) testen afgenomen. UX testen of usability testen zijn eigenlijk processen waarin een product of een deel van een product getest wordt op echte personen.

UX testen evalueren nu eenmaal hoe gebruikers omgaan met de service of het product. Tijdens een UX test wordt er gevraagd aan een tester om een aantal taken uit te voeren. Terwijl een tester deze taken uitvoert kan door een onderzoeker bekeken worden hoe de tester reageert op bepaalde zaken. Dit kan bijvoorbeeld zijn op het design van een website of op hoe bepaalde componenten van een website reageren op elkaar.[15]

Nu wanneer zijn UX testen het best van toepassing? Dit verschilt van project tot project. Zo zijn er projecten waar er al testen gebeuren tijdens het designen van een nieuw product of feature. Dit om direct al verder te kunnen werken met het inzicht van echte personen.

Tijdens het ontwikkelen van een product kunnen UX testen ook afgenomen worden. Dit kan ervoor zorgen dat tijdens het ontwikkelen nog aanpassingen gemaakt kunnen worden voor een product uitgebracht wordt.

Als laatste kunnen testen ook altijd na een product release uitgevoerd worden. Dit kan om eventueel pijnpunten uit de applicatie te kunnen aanpakken die pas later naar boven komen.

### Verschillende soorten UX testen

Er zijn verschillende soorten UX testen die kunnen uitgevoerd worden. Deze kunnen opgesplitst worden in volgende types.[16]

* Gemodereerd vs. Niet gemodereerd
* Op afstand vs. In persoon
* Exploratief vs. Vergelijkend

**Gemodereerd + in persoon**

Bij een gemodereerde test die persoonlijk afgenomen wordt is er altijd een onderzoeker aanwezig. De onderzoeker helpt niet met het beantwoorden van vragen van de gebruiker, maar heeft enkel de rol als observator. Een voordeel hiervan is dat op het einde van de test de onderzoeker iets specifieker kan doorvragen op de ervaring van de gebruiker.

Een voorbeeld van dit soort test is de ‘Lab usability test’. In deze test wordt een toestel klaargezet en altijd op dezelfde manier ingesteld. Hierna kan een tester de test uitvoeren onder toezicht van een onderzoeker. Dit zorgt ervoor dat de testen op en meer gestandaardiseerde manier verlopen.

Een ander voorbeeld van dit soort test is de ‘Guerrilla test’. Hierbij gaat een onderzoeker op een publieke plaats zitten. Dit kan bijvoorbeeld een shoppingcenter, een treinstation of een luchthaven zijn. Daar probeert de onderzoeker willekeurige mensen te overtuigen om een user test uit te voeren. Dit is een goede manier om veel testdata te verkrijgen van verschillende profielen. Maar er moet wel goed gelet worden op de tijd. De meeste mensen spenderen graag niet liever meer dan 5 tot 10 minuten van hun tijd hieraan.

**Niet gemodereerd + op afstand**

Bij een niet gemodereerde test op afstand wordt er meestal gebruik gemaakt van software om de test af te nemen. Bij deze soort testen hoeft geen onderzoeker aanwezig te zijn. Deze testen kunnen zeer grootschalig ondernomen worden.

Een voorbeeld van dit soort test is met behulp van opname software. Wanneer een tester een test start wordt er opname software opgestart. Deze software maakt dan een opname van alle acties de gebruiker uitvoert. Vanaf dat de tester dan klaar is, wordt deze opname doorgestuurd naar de onderzoekers.

**Niet gemodereerd + in persoon**

Niet gemodereerde persoonlijke tests worden uitgevoerd in een gecontroleerde omgeving. Hier hoeft in principe geen persoon aanwezig te zijn om de test af te nemen. Dit biedt als voordeel dat onderzoeker zelf minder invloed heeft op de test.

Een voorbeeld van dit soort test is een observatie test. Hierbij krijgt een tester een document met vooropgestelde scenario’s die uitgevoerd moeten worden. Hierbij kijkt de observator naar hoe de tester de scenario’s uitvoert en wat de lichaamstaal van de tester is. Ook kan de observator niet helpen tenzij de tester volledig de verkeerde kant opgaat.

Een ander voorbeeld van dit soort test is een eye-tracking test. Deze test gebruikt software om de ogen van de tester te volgen doorheen het gebruik van de website. Daarna wordt er een volledig rapport opgemaakt waar de tester het meeste gekeken heeft en wat de eerste elementen zijn die in het zicht springen.

### UX testen binnen dit onderzoek

Binnen dit onderzoek is er gebruik gemaakt van de observatiemethode beschreven in het vorige hoofdstuk. De keuze om deze methode te gebruiken is gebaseerd op de mogelijkheid die het biedt om een gelijke testomgeving te creëren voor alle testen. Daarnaast was er al een gevarieerd testpubliek beschikbaar om de observaties uit te voeren.

In dit onderzoek werden er zeven personen bevraagd om deze usertesten uit te voeren. Een belangrijk aspect van usertesten is om ervoor te zorgen dat er binnen het testpubliek een grote variatie van leeftijden en beroepen is. Dit om ervoor te zorgen dat een groot deel van de bevolking gemakkelijk gebruik kan maken van de applicatie.

Binnen dit onderzoek bestaat de testersgroep uit volgende leeftijden:

* 3 personen tussen 17 en 20 jaar oud.
* 2 personen tussen 40 en 50 jaar oud.
* 2 personen boven de 60 jaar oud.

Het testpubliek bevat ook volgende beroepen:

* Student informatica;
* Student humane wetenschappen;
* Student lager onderwijs;
* Strijkhulp;
* Metser;
* Verpakkingsmedewerker;
* Magazijnier;

Binnen dit onderzoek zijn er drie scenario’s opgesteld die de testpersonen moesten uitvoeren. Deze scenario’s zijn de volgende:

|  |  |
| --- | --- |
| Test scenario’s | |
| Scenario 1 | Je wilt een nieuw menu samenstellen om dit weekend klaar te maken. Graag zou je 2 receptjes hebben voor het ontbijt en 2 receptjes voor de lunch.  Na het maken van het menu wil je naar de winkel en zou je een boodschappenlijstje willen raadplegen. |
| Scenario 2 | Je hebt al een account op de website met volgende gegevens:  Gebruikersnaam: Tester  Wachtwoord: test123  Je zou graag een menu aanmaken voor de komende 3 dagen die voor alle maaltijdperiodes een maaltijd bevat. In de lijst staat er een receptje waarvan je niet goed weet hoe je het zou moeten klaarmaken. Uiteindelijk vind je het receptje, maar niks en kies je ervoor om deze te veranderen. Je beslist om dit menu de naam ‘HalfWeekMenu’ te geven. Graag had je ook een lijstje van de ingrediënten gedownload om mee te nemen naar de winkel. |
| Scenario 3 | Je wilt graag voor vandaag een menu aanmaken. Je hebt zin om een als ontbijt pannenkoeken te eten, als lunch zou je graag het recept hebben om een ‘Creamy Tomato Soup’ te maken en als avondmaal wil je graag ‘Black Bean Tacos’ maken. Dit menu geef je de naam ‘DagMenu’. Hiervan zou je ook graag de ingrediëntenlijst downloaden. |

Alle testers hadden uitgebreid de tijd om deze drie scenario’s uit te voeren. Dit onder toezicht van een onderzoekers. De onderzoeker zorgde enkel voor eventuele hulp als een tester de weg kwijt was in de applicatie en niet meer verder kon in het scenario.

Om ervoor te zorgen dat er geen bevooroordeling ontstond voor een bepaalde structuur startte niet alle testers met dezelfde applicatie. De helft van de testers startte met de MPA en daarna de SPA en dan andere helft startte met de SPA.

Op het einde van de scenario’s kregen de testers ook een finale vraag voorgeschoteld. Deze vraag gaat als volgt: ‘Welke website van de twee die je nu getest hebt, krijgt uw voorkeur?’

### Performance testen

Een belangrijk aspect binnen het maken van een applicatie is het kiezen van een architectuur dat efficiënt werkt. De reden ligt hier dicht bij de gebruikerservaring. Volgens een artikel van Hostinger[17] verlaat een gebruiker een website als deze langer dan 3 seconden duurt om in te laden. [18]

Om een duidelijk beeld te scheppen over hoe de performance verschilt tussen de MPA en SPA worden er performancetesten afgenomen. De data uit deze testen kan dan naast elkaar gelegd worden om te zien op welke vlakken de MPA en SPA beter scoren in opzicht van elkaar.

### Performance indicators

Om een website te kunnen testen op performance is het belangrijk om te weten welke indicatoren er belangrijk zijn. In dit hoofdstuk wordt ingegaan op een paar van deze indicatoren om een duidelijker beeld te scheppen over wat deze inhouden.[19]

**Load time + grootte van de website**

De allereerste indicatoren hangen erg nauw samen en dit zijn de load time en de grootte van de website. Hoe groter de bestanden zijn van de website, hoe langer het zal duren om als gebruiker de website te kunnen openen. Deze indicator moet zeker in de gaten gehouden worden. Zoals eerder vermeld in het vorige hoofdstuk is de aanvaarde load time maar maximum 3 seconden.

**First contentful paint**

Een tweede belangrijke indicator is de First contentful paint (FPC). Wat FPC concreet inhoud is de tijd die de browsers nodig heeft om de allereerste stukken inhoud in te laden. Dit zijn dan bijvoorbeeld stukken tekst of foto’s.

**Speed index**

Een volgende indicator is de speed index. De speed index toont aan hoe snel de inhoud visueel beschikbaar is tijdens het laden van de webpagina.

### Performance testing tools

Om performance testen uit te voeren op de beide applicaties worden er verschillende tools gebruikt.

Een eerste tool die gebruikt wordt tijdens dit onderzoek is WebPageTest. WebPageTest is een online website performance testing tool dat het mogelijk maakt prestaties te analyseren en te meten. Deze tool geeft ook de mogelijkheid om de testen uit te voeren op verschillende locaties, met verschillende internetsnelheden en verschillende browsers.[20]

Met deze tool is er voornamelijk data verzameld omtrent de verschillende pagina’s van de MPA. Ook de SPA is getest met deze tool. Het was enkel niet mogelijk om de volledige SPA te testen met deze tool. Dit komt omdat de tool enkel in staat is om website op te vragen via de URL. Hierdoor was het niet mogelijk om verdere requests binnen de SPA te testen.

Een tweede tool die gebruikt wordt tijdens dit onderzoek zijn developer tools van Mozilla Firefox. Binnen de developer tools van Firefox is er de mogelijkheid om performance testen uit te voeren op de website waar een gebruiker zich op bevind. Deze tool is voornamelijk gebruikt om de SPA te testen. Ook het aantal requests de MPA of SPA uitvoerde tijdens een volledige flow door de applicatie werd opgehaald via de developer tools.[21]

De flow die getest werd tijdens de performance testen ging als volgt:

1. Op de indexpagina wordt een menu opgevraagd voor alle maaltijdperiodes met maar 1 maaltijd per periode.
2. In het gegenereerde menu wordt er één recept geopend en daarna teruggekeerd naar het menu.
3. Het aangemaakte menu wordt opgeslagen en er wordt ingelogd om dit te voltooien
4. Het opgeslagen menu wordt geopend en er wordt genavigeerd naar de ingrediëntenlijstpagina
5. De ingrediëntenlijst wordt gedownload.

## Resultaten

### Hoe verschilt de gebruikerservaring tussen een SPA en MPA?

Uit de resultaten van de usertesten is te zien dat qua gebruiksgemak en flow van de website beide structuren bijna geen verschil aantonen. Ook op de mobiele versie werd er geen verschil opgemerkt in gebruiksgemak en flow.

Op het einde van elke usertest kreeg de tester de vraag welke van de twee websites hun voorkeur kregen. In bijna alle gevallen ging de voorkeur uit naar de SPA en de reden waarom is omdat deze volgens de testers rapper werkte dan de MPA.

Dit verschil was duidelijk te zien wanneer een tester de usertest uitvoerde met een browser die gelimiteerde internetsnelheid had. De tester was duidelijk gefrustreerd tijdens het gebruik van de MPA wanneer een pagina er even over deed om in te laden. Bij de SPA werd al snel verduidelijkt dat deze veel beter aanvoelde om te gebruiken sinds alles veel efficiënter verliep.

Wat ook opviel tijdens de usertesten is dat de gebruikers veel vlugger kiezen om de ingebouwde browser navigatie te gebruiken. Voor de MPA is deze functionaliteit al volledig ingebouwd, maar voor de SPA moet dit zelf ontwikkeld worden door de developer.

### Hoe verschilt de SPA en MPA qua performance?

**Load time + grootte van de website**

Een eerste indicator die eerder in dit onderzoek aangehaald werd is de grootte van de website. Hoe groter een website is, hoe langer deze normaal gezien duurt om in te laden. Wanneer de eerste pagina van beide applicaties worden ingeladen komen volgende resultaten uit de testen.

Wanneer er gesurft wordt naar de beginpagina van de SPA krijgt de browser 36 kB aan files door. Dit is een groot verschil met de beginpagina van de MPA die maar 12 kB aan files bevat. Dit zorgt ervoor dat de laadtijd van de SPA in eerste instantie hoger is dan de MPA.

Als volgt moet er ook bij stil gestaan worden hoeveel de totale grootte van de website bedraagt. Met de totale grote wordt hier bedoeld hoeveel data de browser heeft binnengekregen na één volledige flow door de website.

Bij het vergelijken van de volledige flow door de applicatie is dan uiteindelijk wel te zien dat de MPA meer data verbruikt heeft dan de SPA. Het verschil bedraagt ongeveer 13 kB. Dit is op zich niet echt veel, maar deze applicatie is natuurlijk aan de kleinere kant.

Een andere indicator is ook hoeveel requests de browsers heeft moeten maken in totaal voor beide applicaties.

Afbeelding met tekst, schermopname, diagram, ontwerp

Automatisch gegenereerde beschrijving

Op bovenstaande afbeelding is te zien hoeveel verschillende requests de browser heeft gemaakt om door de volledige flow van de SPA te gaan. In dit geval bij de SPA zijn dat er dus 33 in totaal.

Afbeelding met schermopname, tekst, Multimediasoftware, diagram

Automatisch gegenereerde beschrijving

Bij de MPA kan er gezien worden dat er veel meer requests nodig zijn dan bij de SPA. Maar liefst 24 requests extra zijn nodig om hetzelfde resultaat te verkrijgen als de SPA.

**First contentful paint**

De volgende parameter die vergeleken wordt is de First contentful paint. Hieronder op de grafiek is te zien dat de MPA op dit vlak beter scoort dan de SPA. Dit met een verschil rond de 107 ms. Dit is voornamelijk te linken aan de SPA waarvan alle bestanden worden ingeladen in het begin. De MPA hoeft namelijk enkel maar de bestanden van de beginpagina op te halen.

**Speed index**

Bij de speed index is de MPA terug net dat tikkeltje sneller. Hier zit er een verschil van 136 ms. Dit is opnieuw te wijten aan het feit dat de SPA meer bestanden moet inladen tijdens het eerste bezoek aan de website.

**Laadtijden bij lagere snelheden**

Tegenwoordig worden websites ook veel vaker bezocht via een 3G/4G-netwerk. Dit kunnen bijvoorbeeld gebruikers zijn die onderweg zijn met het openbaar vervoer. Daarom is het belangrijk om eens dieper in te gaan hoe de MPA en SPA presteren op een lagere netwerksnelheid.

Op bovenstaande grafiek is te zien hoeveel tijd de user moet wachten tijdens het gebruik van de applicaties op 3G. Wat opvalt is dat de MPA in tegenstelling tot de andere testen hier niet sneller scoort dan de SPA. Wanneer er op een 3G netwerk doorheen de flow van de applicatie gegaan wordt presteert de MPA 1,30 seconden trager dan de SPA.

### Is er een groot verschil in development ervaring?

Een belangrijk aspect bij het kiezen van een bepaalde structuur van een applicatie is het afwegen of een bepaalde structuur een goede development ervaring heeft. Dit is een zeer persoonlijk iets, maar zeker vermeldingswaardig binnen dit onderzoek.

Als er gekeken wordt naar een MPA die meerdere HTML-pagina’s heeft moet op elke pagina een link gelegd worden naar bijvoorbeeld JavaScript of CSS-bestanden. Dit is een zeer repetitieve taak en één die niet echt aangenaam is. Bij een SPA aan de andere kant is er maar een enkele HTML-pagina. Hier wordt er een verwijzing gemaakt naar de nodige JavaScript of CSS-bestanden en daarna hoeft dit nooit meer opnieuw te gebeuren.

Een voordeel van meerdere HTML-pagina’s te hebben is dat er veel geïsoleerder kan gewerkt worden. Er hoeft geen rekening gehouden worden met eventuele CSS die niet op deze pagina nodig zou zijn of met logica die niet moet uitgevoerd worden op een bepaalde pagina. Wanneer er maar een enkele HTML-pagina is moet er goed uitgekeken worden naar welke logica er eerst moet uitgevoerd worden. Daarna moet er ook rekening gehouden worden met welke CSS op bepaalde elementen moet zichtbaar zijn op een bepaald moment.

Een voordeel bij het werken van een MPA is dat de browser al volledig ingebouwd heeft hoe er gereageerd moet worden op het navigeren tussen verschillende pagina’s. Dit kan door middel van de ingebouwde back- en forward knoppen in de browser. Binnen een SPA werken deze knoppen niet zonder dit zelf volledig te programmeren. Dit kan door middel van de History API die ingebouwd zit in elke moderne browser. Het enig probleem met deze API is dat deze zeer lastig is om te debuggen en waardoor het een hele klus is om deze 100% foutloos te laten werken.

Bij een SPA is het ook zeer vervelend dat wanneer een pagina refreshed alle logica gereset wordt. Dit is lastig tijdens de development wanneer er aan een feature gewerkt wordt die pas na vele stappen binnen de app van toepassing komt. Dit zorgt voor een onnodige vertraging binnen het developmentproces.

Een ander minpuntje binnen het maken van een SPA is dat er rekening moet gehouden worden met welke elementen op de pagina mogen te zien zijn. Binnen een MPA is dit gemakkelijk omdat als er genavigeerd wordt naar een andere pagina alle content van de vorige verdwijnt. Bij een SPA moet de developer zelf code schrijven om ervoor te zorgen dat content die niet zichtbaar moet zijn verwijderd of onzichtbaar wordt en dat daarna de volgende content tevoorschijn komt.

### Hoe verloopt het deployen van een SPA en een MPA?

Beide applicaties zijn gedeployed op een Virtual Private Server (VPS). Op deze VPS draait het besturingsysteem Debian 11.[22] Debian 11 is een Linux distributie die zeer stabiel is en wordt veel gebruikt om websites mee te hosten. Volgens een artikel draaien 16% van alle Linux gebaseerde websites op een Debian besturingsysteem.[23]



Om ervoor te kunnen zorgen dat de website beschikbaar is op het internet is er ook nood aan webserver software. Op de server in dit onderzoek wordt gebruik gemaakt van NGINX om de websites toegankelijk te maken via het internet.

Wanneer NGINX op de server geïnstalleerd staat kan de code van de website op de server geplaatst worden. Hierna hoeft enkel nog de configuratie van NGINX vervolledigt te worden zodat de webserver weet welke bestanden moeten beschikbaar gesteld worden.

Voor zowel de MPA als SPA is deze configuratie volledig hetzelfde. Er moet gezorgd worden dat NGINX weet op welk bestand de website gestart moet worden en verder is er geen extra configuratieverschil.

# Conclusie

Dit onderzoek geeft antwoord op de hoofdvraag “Single vs. Multi page applications: wanneer kies je wat?”. Om hierop een conclusie uit te schrijven wordt er teruggekeken naar de resultaten van het onderzoek.

Een aller eerste

# Discussie

# Referentielijst

[1] ‘Single Page Application (SPA) vs Multi Page Application (MPA) – Two Development Approaches | ASPER BROTHERS’. https://asperbrothers.com/blog/spa-vs-mpa/ (geraadpleegd 4 april 2023).

[2] ‘Single Page vs Multi-Page Application – What are The Main Differences? - Lightrains’. https://lightrains.com/blogs/single-page-application-vs-multi-page-application/ (geraadpleegd 4 april 2023).

[3] ‘Single-page application vs Multi-page application: What Is Better for Your Project? | LinkedIn’. https://www.linkedin.com/pulse/single-page-application-vs-multi-page-what-better-your-/?trk=organization\_guest\_main-feed-card\_feed-article-content (geraadpleegd 4 april 2023).

[4] ‘What is a Single Page Application? Pros and Cons of the SPA Technology - HUSPI’. https://huspi.com/blog-open/definitive-guide-to-spa-why-do-we-need-single-page-applications/#1-client-side-rendering-csr (geraadpleegd 4 april 2023).

[5] ‘(95) What is a single page application? SPA - YouTube’. https://www.youtube.com/watch?v=eIxDHgzGCnY (geraadpleegd 4 april 2023).

[6] ‘Pros and Cons of Single-Page Applications’. https://www.spiceworks.com/tech/devops/articles/what-is-single-page-application/ (geraadpleegd 4 april 2023).

[7] ‘SPA (Single-page application) - MDN Web Docs Glossary: Definitions of Web-related terms | MDN’. https://developer.mozilla.org/en-US/docs/Glossary/SPA (geraadpleegd 4 april 2023).

[8] K. Lawson, ‘What Is a Single Page Application? | Bloomreach’, 2022. https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application (geraadpleegd 4 april 2023).

[9] ‘MDN Web Docs’. https://developer.mozilla.org/en-US/ (geraadpleegd 4 april 2023).

[10] ‘Wat is een API en wat kan je ermee? - Salesforce NL Blog’. https://www.salesforce.com/nl/blog/2019/10/wat-is-een-api.html (geraadpleegd 28 mei 2023).

[11] ‘<template>: The Content Template element - HTML: HyperText Markup Language | MDN’. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/template (geraadpleegd 28 mei 2023).

[12] ‘History API - Web APIs | MDN’. https://developer.mozilla.org/en-US/docs/Web/API/History\_API (geraadpleegd 29 mei 2023).

[13] ‘History: pushState() method - Web APIs | MDN’. https://developer.mozilla.org/en-US/docs/Web/API/History/pushState (geraadpleegd 29 mei 2023).

[14] ‘Fetch API - Web APIs | MDN’. https://developer.mozilla.org/en-US/docs/Web/API/Fetch\_API (geraadpleegd 29 mei 2023).

[15] ‘Why UX testing is so important for your product in 2023 - UX Design Institute’. https://www.uxdesigninstitute.com/blog/why-ux-testing-is-so-important/ (geraadpleegd 29 mei 2023).

[16] ‘8 Usability Testing Methods That Work (Types + Examples)’. https://www.hotjar.com/usability-testing/methods/ (geraadpleegd 29 mei 2023).

[17] ‘Everything You Need to Know About Website Performance’. https://www.hostinger.com/tutorials/website-performance (geraadpleegd 28 mei 2023).

[18] ‘Why Site Speed Is So Important: Conversions, Loyalty, and Google Search Ranking (2023)’. https://www.shopify.com/blog/site-speed-importance (geraadpleegd 14 mei 2023).

[19] ‘Top 10 Key Website Performance Metrics You Should Measure - Sematext’. https://sematext.com/blog/website-performance-metrics/ (geraadpleegd 1 juni 2023).

[20] ‘WebPageTest - Website Performance and Optimization Test’. https://www.webpagetest.org/ (geraadpleegd 7 mei 2023).

[21] ‘Performance Analysis — Firefox Source Docs documentation’. https://firefox-source-docs.mozilla.org/devtools-user/network\_monitor/performance\_analysis/ (geraadpleegd 1 juni 2023).

[22] ‘Debian -- Reasons to use Debian’. https://www.debian.org/intro/why\_debian (geraadpleegd 14 mei 2023).

[23] ‘Linux Statistics 2023 - TrueList’. https://truelist.co/blog/linux-statistics/ (geraadpleegd 14 mei 2023).

# Bijlagen

Alle bijlagen verzamel je hier. Bijlagen zijn bedoeld voor de lezer die interesse heeft in meer gedetailleerde informatie over je onderwerp.