**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 ma bachelor ster 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.

29/05/2023

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 voor mij om 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](#_Toc135646177)

[1.1 Algemeen 10](#_Toc135646178)

[1.1.1 Wat is een Single page application? 10](#_Toc135646179)

[1.1.2 Wat is een Multi page application? 11](#_Toc135646180)

[1.2 Probleemstelling 11](#_Toc135646181)

[1.3 Onderzoeksvraag 12](#_Toc135646182)

[1.4 Experiment 12](#_Toc135646183)

[1.5 Structuur van deze bachelorproef 12](#_Toc135646184)

[2 Experiment 13](#_Toc135646185)

[2.1 Methodologie 13](#_Toc135646186)

[2.1.1 Uit welke elementen bestaat een typische webapplicatie? 13](#_Toc135646187)

[2.1.2 Welke applicatie wordt er ontwikkeld? 14](#_Toc135646188)

[2.1.3 De ontwikkelde API 14](#_Toc135646189)

[2.1.4 Ontwikkelen van de MPA 16](#_Toc135646190)

[2.1.5 Ontwikkelen van de SPA 16](#_Toc135646191)

[2.2 Onderzoeksverloop en data-analyse methode(n) 16](#_Toc135646192)

[2.2.1 Subsubtopic 16](#_Toc135646193)

[2.2.2 Subsubtopic 16](#_Toc135646194)

[2.3 Resultaten 16](#_Toc135646195)

[2.3.1 Hoe verschilt de gebruikerservaring tussen een SPA en MPA? 17](#_Toc135646196)

[2.3.2 Hoe verschilt de SPA en MPA qua performance? 18](#_Toc135646197)

[2.3.3 Is er een groot verschil in development ervaring? 20](#_Toc135646198)

[2.3.4 Hoe verloopt het deployen van een SPA en een MPA? 20](#_Toc135646199)

[3 Theoretisch kader/Literatuuronderzoek 22](#_Toc135646200)

[4 Conclusie en discussie 23](#_Toc135646201)

[Referentielijst 24](#_Toc135646202)

[Bijlagen 25](#_Toc135646203)

**Referentielijst**

**Bijlagen**

Lijst met figuren

In de lijst met figuren vermeld je alle figuren en die in de bijlage(n), samen met de paginanummers waar de figuren terug te vinden zijn.

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

*Let op:*

* *Verbeter je inleiding een laatste keer nadat je klaar bent met je conclusie, discussie en aanbevelingen. Voeg je structuur van je BAP als laatste toe.*
* *Als je antwoord geeft op de volgende vragen en per vraag een alinea schrijft, schrijf je een goed gestructureerde inleiding. Gebruik hierbij paragrafen en voeg hieraan eventueel kopjes toe ter verduidelijking waar iedere paragraaf over gaat.*

## Algemeen

Deze bachelorproef is tot stand gekomen vanuit een onderwerp die 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 in één request alle broncode (HTML, Javascript, CSS) binnen die nodig is om heel de website te laten werken. 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 doormiddel van Javascript. Met Javascript kan de inhoud van de website worden geüpdatet 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-call 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 langs om in te laden van de drie methoden. Dit omdat CSR veel resources van de lokale computer 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 ontwikkeling- 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

Ruwweg beschrijven wat de lezer in volgende hoofdstukken kan verwachten vanuit het standpunt van de auteur.

# Experiment

## Methodologie

Hier ga je in op je methoden. Je gaat in op het soort onderzoek, de dataverzameling, datakenmerken, het onderzoeksverloop en de data-analysemethode(n). Je grijpt hierbij ook terug op de onderzoeksopzet.

### 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 checkboxes 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 loginpagina. 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.

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 uitgebereid 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-spec. 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-spec 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 end-points 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 errorcode 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 geeft de server het recept terug dat het meegegeven ID bevat.

* GET /recipes/random/{params}

Met dit endpoint kan er doormiddel 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 ingredienten 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 loginpagina 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 doormiddel 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 doormiddel 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 definieert de templatetag 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.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 laatst 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 templatetag volgends 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 developen 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. 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. 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 doormiddel van de Fetch API. 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 call wordt een URL meegegeven. Ook een object met aangepaste instellingen kunnen meegeven worden zoals bijvoorbeeld: method, headers, body…

Vervolgens krijgt de fetch 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)

Ga in op het onderzoeksverloop en de data-analysemethode(n).

* Hoe is het onderzoek verlopen? Wat verschilde met je onderzoeksopzet?
* Heb je voldoende zaken kunnen aftoetsen?
* Wat was de rol van de onderzoeker en hoe was de onderzoekssituatie?
* Hoe heb je daarna de data geanalyseerd (geef alleen de methoden en niet de resultaten!)?

### Subsubtopic

Gebruik maximaal drie niveaus!

### Subsubtopic

## Resultaten

Deel dit hoofdstuk in op basis van de deelvragen waarop je antwoord geeft. Op die manier kun je het hoofdstuk een duidelijke structuur geven.

Bespreek per deelvraag de belangrijkste resultaten. Verbind hieraan nog geen conclusies.

Je kunt tabellen, figuren en grafieken gebruiken om je resultaten overzichtelijk te presenteren.

De resultaten die niet relevant zijn om je deelvragen te beantwoorden horen hier niet thuis. Die plaats je in je bijlagen, daar kun je in de tekst naar verwijzen.

Kun je je hoofdstuk niet indelen per deelvraag? Deel het dan in op basis van de onderzoeksmethoden die je gebruikt hebt of op basis van de deelonderwerpen.

Als je veel resultaten presenteert, kun je de belangrijkste resultaten aan het eind van je hoofdstuk kort samenvatten. Je kunt dit ook per paragraaf/beantwoording van je deelvraag doen. Of dit nodig is, hangt af van de richtlijnen van je opleiding.

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

Tijdens dit onderzoek wordt er gebruik gemaakt van usertesten om de gebruikerservaring tussen een MPA en SPA te meten. Een usertest is een test waarin gebruikers met behulp van op voorhand geschreven scenario’s verschillende elementen van een website kunnen testen.

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.

TODO MAAK TABEL MET LEEFTIJDEN

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;

Tijdens de usertesten hadden alle testers uitgebreid de tijd om drie scenario’s uit te voeren. Deze werden uitgevoerd onder begeleiding van een onderzoeker. De onderzoeker zorgde voor eventuele hulp als een tester de weg kwijt was in de applicatie en niet meer verder kon in het scenario.

De helft van de testers startte met de MPA en daarna de SPA en de andere helft startte met de SPA. Dit om ervoor te zorgen dat er geen bevooroordeling voor een bepaalde structuur wanneer iedereen moest starten op dezelfde structuur.

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.

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

Een belangrijk aspect binnen het maken van een applicatie is het kiezen voor een architectuur dat efficiënt werkt. De reden ligt hier dicht bij de gebruikerservaring. Volgends een artikel van Hostinger verlaat een gebruiker een website als deze langer dan 3 seconden duurt om in te laden. Daarom dat tijdens dit onderzoek er performance testen zijn uitgevoerd. Dit om te kunnen zien of er effectief performanceverschillen zijn tussen de twee types websites.

Een eerste parameter om te bekijken is de grote van de website. Wanneer een website groter wordt begint deze vanzelfsprekend een langer laadtijd te krijgen. Als een gebruiker surft naar de beginpagina van de website krijgen we volgend resultaat.

Figuur MPA vs. SPA beginpagina vergelijken in KB

Zoals te zien op bovenstaande grafiek is het verschil ongeveer 6 KB. Nu zou er al gedacht kunnen worden dat de MPA over het algemeen kleiner is dan de SPA. Maar het grote verschil met elkaar hier is dat dit enkel de beginpagina van de MPA is. Wanneer er gesurft wordt naar de SPA downloadt de browser al volledig de sites, sinds de MPA maar 1 pagina groot is.

Wanneer het verschil van alle pagina’s van de MPA met de SPA vergelijkt wordt komt volgend resultaat boven.

Figuur MPA vs. SPA totaal gedownloade data in KB

Wanneer alle pagina’s van de MPA gedownload zijn is er wel daadwerkelijk een groot verschil. De SPA blijft dezelfde grote sinds alle zaken die nodig zijn om de site te laten werken al gedownload zijn. De MPA aan de andere kant is zeer in vergelijking met de SPA erg gestegen in aantal gedownloade KB. De reden hiervoor is omdat elke pagina afzonderlijk alle resources opnieuw downloadt.

Nu er geweten is hoe groot beide applicaties zijn kan er gekeken worden naar hoe lang dit effectief geduurd heeft om in te laden. Dit is terug te vinden op onderstaande grafiek.

Figuur MPA vs. SPA laadtijd beginpagina in ms

Zoals kon verwacht worden heeft de beginpagina van de MPA een lagere laadtijd dan de SPA. Dit komt natuurlijk terug doordat de browser alle resultaten van de SPA ophaalt en maar enkel de beginpagina van de MPA.

Wanneer er gekeken wordt naar het navigeren naar alle pagina’s is er wel een groter verschil en dit ligt dan in het voordeel van 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-files. 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-files 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 doormiddel 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 doormiddel 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 development proces.

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 bestuurtingsysteem Debian 11. 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 gebasseerde websites op een Debian bestuuringsysteem.



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 toegangelijke 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.

# Theoretisch kader/Literatuuronderzoek

Hier bespreek je de bestaande theorieën en onderzoeken die relevant zijn voor je onderwerp.

Geef daarna antwoord op de volgende vragen.

* Wat zijn de belangrijkste wetenschappelijke artikelen die je gebruikt? Ga hier kort op in.
* Welke theorieën en modellen zijn het meest belangrijk voor je onderzoek?

In sommige gevallen is het logischer om dit hoofdstuk voor het vorige te plaatsen. De keuze ligt hier bij jou.

# Conclusie en discussie

In het conclusie-gedeelte formuleer je het antwoord op de onderzoeksvraag.

Hierin komen geen nieuwe resultaten aan bod die je nog niet eerder hebt aangehaald. Gebruik hier geen subsecties.

Ga eerst kort in op de hoofdvraag (en herhaal deze indien nodig), zodat je lezer weet waar je ook alweer antwoord op gaat geven.

Deel dit hoofdstuk in door conclusies te trekken met behulp van de antwoorden op de deelvragen die je hebt gegeven in je theoretisch kader en de resultaten. Je kunt per deelvraag een alinea schrijven.

Geef tot slot op basis hiervan duidelijk antwoord op je hoofdvraag óf bevestig of ontkracht je hoofdhypothese. Dit is de conclusie van jouw onderzoek.

Bij het discussie-gedeelte mag je jouw resultaten interpreteren en kan je vrijer schrijven dan in de conclusie. Schrijf wel formeel en onderbouw je gedachtegang.

* Waarom is jouw onderzoek valide?
* Komen de resultaten overeen met je verwachtingen? Waarom wel/niet?
* Wat is een verklaring voor je resultaten? Hier is ruimte voor interpretatie!
* Wat heeft jouw onderzoek toegevoegd aan de huidige literatuur/kennis over je onderwerp?
* Welk losstaand onderzoek is nuttig om te starten naar aanleiding van/voortbordurend op jouw onderzoek? Geef hier concrete suggesties

Sluit op het einde af met een krachtige slotzin die jouw conclusie in een zin kort samenvat.

Richtlijn is maximaal twee pagina’s.

# Referentielijst

Alle bronnen waarvan je gebruikmaakt zet je in de referentielijst

# Bijlagen

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