Verantwoordingsdocument

Eindopdracht Frontend

Rick van Campen

NOVI Hogeschool

Inleverdatum: 5 mei

Inleiding

Mijn verantwoordingsdocument geeft inzicht in de technische ontwerpbeslissingen die ik heb gemaakt tijdens het maken van mijn applicatie. Dit verantwoordingsdocument richt zich vooral op de keuzes die ik heb gemaakt met betrekking tot JavaScript en React. Hierbij ligt de nadruk dus op de technische aspecten en dus niet op de visuele aspecten. Het document laat dus duidelijk zien waarom ik bepaalde componenten heb gebruikt voor mijn applicatie, welke alternatieven ik heb gebruikt en wat in de toekomst de mogelijke doorontwikkelingen zijn.

Inhoudsopgave

Technische programmeerkeuzes betreffende JavaScript en React

Gebruik van React Context voor State Management:

Ik heb voor in mijn applicatie gekozen om React Context te gebruiken voor de authenticatie- en gebruikersgegevens te beheren. Dit is heel erg handig, omdat de informatie hierbij wordt doorgegeven aan de, als het ware, diepere verstopte componenten van mijn website, zonder dat de informatie steeds door elke laag van mijn code moet gaan.

Reflectie:

Het gebruik van React Context voor state management heeft het voor mij makkelijk gemaakt om de gegevens door te geven aan de diepere componenten van mijn applicatie, zonder dat de informatie steeds door elke laag van mijn code moet gaan. Dit heet ook wel prop drilling. Hierdoor is mijn code clean en wordt goed leesbaar gehouden voor mezelf.

Gebruik van Fetch API voor het maken van HTTP-verzoeken:

IK heb ervoor gekozen om niet standaard bibliotheken te gebruiken zoals Axios. In plaats daarvan heb ik gebruik gemaakt van de Fetch API voor het maken van HTTP-verzoeken. De API biedt een makkelijke manier om gegevens van een server op te halen en is daarbij ingebouwd in de moderne browsers wat dus ook handig is.

Reflectie:

Het gebruik van de Fetch API voor de HTTP-verzoeken was uiteindelijk erg handig, omdat dit dus in moderne browsers ingebouwde ondersteuning geeft. Hierdoor wordt waarschijnlijk de laadtijd versneld van mijn applicatie en dit heeft ervoor gezorgd dat er een aantal externe afhankelijkheden zijn verminderd.

Gebruik van React Router voor het implementeren van routing:

Ik heb ervoor gekozen om React Router te gebruiken voor de implementatie voor de routing binnen de applicatie. Dit is dus een handige manier om ervoor te zorgen dat wanneer de gebruikers op verschillende knoppen klikken en dus door mijn website navigeren, ze naar de juiste pagina worden gestuurd. Het maakt dus als het ware de reis van de ene pagina naar de andere pagina makkelijker voor de gebruikers die mijn applicatie gebruiken.

Reflectie:

Het gebruik van React Router in mijn applicatie heeft ervoor gezorgd dat de ervaring voor gebruikers van de ene naar de andere pagina erg makkelijk en fijn om te gebruiken is. Je kan dus wel zeggen dat er een goed lopend navigatiesysteem is gecreëerd voor de gebruikers. Ook is React Router handig om verschillende delen van mijn applicatie met bepaalde webadressen te verbinden. Het helpt me uiteindelijk om verschillende stukjes van mijn applicatie uit elkaar te houden en om ervoor te zorgen dat elke link op de juiste plaats terechtkomt wanneer de gebruikers hierop klikken.

Gebruik van useState en useEffect hooks voor component state en side effects:

Ik heb ervoor gekozen om functionele componenten te gebruiken voor mijn applicatie in plats van de klassecomponenten. Ik heb dus voor mijn functionele componenten useState en useEffect hooks gebruikt om zo de component state te beheren en voor het uitvoeren van side effects. Denk hierbij aan het laden van gegevens van mijn externe API.

Reflectie:

Het gebruik van useState en useEffect hooks heeft het mogelijk gemaakt dat ik functionele componenten kan gebruiken die ook nog eens de code compact houden en het makkelijker maakt om het te begrijpen voor mij. Door klassecomponenten te vermijden is de manier waarop mijn code is geschreven makkelijker geworden en waarschijnlijk zal mijn applicatie hierdoor beter werken.

Gebruik van arrow functions voor componenten en event handlers:

Ik heb ervoor gekozen om arrow functions te gebruiken bij het definiëren van mijn functionele componenten en event handlers. Deze arrow functions geeft een makkelijke en duidelijke structuur, waardoor mijn code uiteindelijk makkelijk te begrijpen is en ik ook minder code hoef te schijven die helemaal niet nodig is. Ook is een voordeel dat arrow functions automatisch het 'this'-bindingsgedrag regelen, waardoor het gebruik van de bind() methode niet meer nodig is bij het doorgeven van event handlers. Uiteindelijk maakt dit mijn code nog makkelijker om te schrijven.

Reflectie:

Het gebruik van arrow functions heeft de basis van mijn code makkelijker gemaakt en mijn ontwikkelingsproces versneld. Dit komt door de duidelijke structuur die het aan mijn code geeft. Ook is mijn code schoner en meer gestructureerd geworden door de bind()-aanroepen weg te laten. De arrow functions zorgen er dus voor dat mijn code efficiënter is en makkelijker te lezen is voor mij.

Limitaties van de applicatie

Beperkte functionaliteit zonder internetverbinding:

Momenteel draait mijn applicatie alleen wanneer er internetverbinding is. Dit komt met name ook door het ophalen van de gegevens van de externe API's. De applicatie werkt dus niet wanneer er geen internetverbinding is. Hierbij zou en doorontwikkeling zijn om ervoor te zorgen dat mijn applicatie ook draait zonder een internetverbinding. Dit zou bijvoorbeeld kunnen met behulp van service workers en caching. Hierdoor zou je de applicatie ook offline kunnen gebruiken.

Gebrek aan gebruikersauthenticatie en autorisatie:

Ik heb in mijn applicatie een vorm van gebruikersauthenticatie geïmplementeerd, maar er ontbreekt eigenlijk een uitgebreid autorisatiesysteem. Dit zou uiteindelijk mijn applicatie nog beter maken. Daarom zou een mogelijke doornontwikkeling ook zijn dat er meer geavanceerde autorisatiemogelijkheden in mijn applicatie komen. Denk bijvoorbeeld aan rollen en machtigingen voor verschillende gebruikersgroepen.

Beperkt assortiment van externe API's:

Op dit moment maakt de applicatie gebruik van een paar API's voor het ophalen van bepaalde gegevens in de applicatie. De applicatie zou meer API's kunnen hebben om de applicatie uiteindelijk groter te maken waarbij er meer gegevens zouden worden opgehaald. Een mogelijke doorontwikkeling is dan ook dat er meer externe API's worden gebruikt in de applicatie, waardoor uiteindelijk meer diversiteit aan de inhoud en functionaliteit wordt gegeven.

Gebrek aan gebruikersfeedback en notificaties:

De applicatie mist een functie waarbij de gebruikers feedback kunnen geven en waarbij er meldingen worden gegeven. Dit zou voor de gebruikers wel handig zijn om uiteindelijk de applicatie nog beter te maken. Daaro zou een mogelijke doorontwikkeling zijn dat er een systeem wordt geïmplementeerd in de applicatie die het mogelijk maakt dat gebruikers feedback kunnen invullen en dat er meldingen worden getoond aan de gebruikers. Dit zal uiteindelijk dus de applicatie verbeteren en de gebruikerservaring zal ook worden verbeterd.

Gebrek aan geavanceerde zoek- en filtermogelijkheden:

Op dit moment kunnen gebruikers alleen door beperkte filters navigeren of zoeken op basis van specifieke criteria die de applicatie zelf al geeft. De gebruikers zijn dus niet echt in staat om dingen zelf op te zoeken of te filteren. Daarom is een mogelijke doorontwikkeling hierbij het implementeren van een uitgebreide zoekfunctionaliteit. Deze zoekfunctionaliteit zou daarbij ondersteunt worden door verschillende filters, sorteeropties en zoektermen. Hierdoor zouden gebruikers sneller en efficiënter de gewenste informatie kunnen vinden. Dit zal uiteindelijk de tevredenheid bevorderen van de gebruikers en daardoor zal de applicatie ook verbeteren.