**Ontwerpdocument FE**

|  |  |
| --- | --- |
| **Productnaam** | Frisdranken Verzameling |
| **Studentnummer en naam** | 23130032 Muhammed Gursoy |
| **Groep** | 2 |

Inhoudsopgave

[Inleiding 2](#_Toc200223050)

[Projectplan 2](#_Toc200223051)

[Ontwerp front-end 4](#_Toc200223052)

[Code review 6](#_Toc200223053)

[Testplan 8](#_Toc200223054)

[Reflectie en Conclusies 8](#_Toc200223055)

[Bronnen 9](#_Toc200223056)

# Inleiding

Voor deze opdracht ontwikkel ik een statische front-end webapplicatie waarin een persoonlijke verzameling wordt gepresenteerd. De applicatie is bedoeld als opdracht voor HTML, CSS en JavaScript zonder gebruik van externe bibliotheken.

De gekozen verzameling bestaat uit frisdranken, met visuele en inhoudelijke informatie over elk product. Het doel van de applicatie is om gebruikers in staat te stellen producten te bekijken, sorteren op suikergehalte en filteren op categorie. Daarnaast wordt er gebruikgemaakt van paginanummering om de producten overzichtelijk weer te geven.

De website bestaat uit drie pagina’s:

* Homepagina: introductie van de collectie en navigatie
* Overzichtspagina: toont alle producten met sorteer- en filterfunctionaliteit
* Contactpagina: toont contactgegevens of een statisch formulier

De applicatie wordt gebouwd met een duidelijke mappenstructuur, gescheiden CSS en JavaScript-bestanden, en een overzichtelijke navigatie. De data wordt hardcoded in een aparte JavaScript-array en dynamisch weergegeven op de overzichtspagina met behulp van DOM-manipulatie.

# Projectplan

Om dit front-end project succesvol te voltooien, hanteer ik een gestructureerde aanpak bestaande uit meerdere fasen. Het project wordt ontwikkeld met behulp van HTML, CSS en JavaScript in een lokale ontwikkelomgeving (Visual Studio Code), en versiebeheer via Git/GitHub. Hieronder beschrijf ik de geplande aanpak in vijf duidelijke stappen.

**Stap 1 – Structuur & repository opzetten**

Het project start met het opzetten van een duidelijke mappenstructuur. Ik maak afzonderlijke mappen aan voor html, css, js, images en docs, zodat de bestanden logisch gescheiden zijn. Dit zorgt voor een overzichtelijke werkomgeving en maakt onderhoud eenvoudiger. Er wordt gewerkt met branches: elke functionele toevoeging krijgt zijn eigen branch (zoals structure, data, overview-page), en wijzigingen worden pas samengevoegd na afronding.

Keuze: werken in branches zorgt voor controle, voorkomt fouten en maakt het mogelijk om wijzigingen gescheiden te ontwikkelen en te testen.

**Stap 2 – Opbouw HTML-pagina’s**

De drie verplichte pagina’s worden opgezet:

* index.html (homepage)
* overzicht.html (verzameling met sorteer- en filteropties)
* contact.html (statische contactpagina)

Alle pagina’s worden aan elkaar gelinkt via een navigatiebalk. De titel van iedere pagina wordt duidelijk weergegeven in de browsertab.

Keuze: Door eerst de HTML-structuur vast te leggen, ontstaat een goede basis voor de styling en functionaliteit.

**Stap 3 – Styling met CSS**

Vervolgens wordt de layout gestyled met een extern CSS-bestand (style.css). De weergave van de producten wordt opgebouwd met behulp van cards (tegels). De navigatie, buttons, structuur en responsive gedrag worden allemaal vanuit CSS geregeld.

Keuze: Er wordt bewust geen gebruik gemaakt van externe frameworks zoals Bootstrap, omdat de opdracht vereist dat eigen CSS wordt toegepast. De layout wordt responsive ontworpen voor mobiel en desktop.

**Stap 4 – JavaScript-functionaliteit ontwikkelen**

In een apart bestand (app.js) wordt de functionaliteit voor sorteren, filteren, en pagineren gebouwd. De producten worden opgeslagen in een aparte JS-array (data.js) en met DOM-manipulatie weergegeven in de overzicht.html. De gebruiker kan producten sorteren op suikergehalte (op- en aflopend) en filteren op categorie. De weergave is gelimiteerd tot 5 producten per pagina, met knoppen voor het wisselen tussen pagina’s.

Keuze: Door logica te scheiden in aparte bestanden blijft de code overzichtelijk en herbruikbaar. De gebruiker heeft volledige controle over wat zichtbaar is.

**Stap 5 – Testen, documenteren en afronden**

Tot slot wordt de applicatie getest op verschillende schermgroottes en browsers. De README en ontwerpdocumentatie worden aangevuld. De Git-branches worden samengevoegd naar de main branch en klaargemaakt voor oplevering als zip-bestand.

Keuze: Door testmomenten op te nemen in het plan wordt de kans op bugs verkleind. Documentatie zorgt ervoor dat de docent of beoordelaar begrijpt hoe de applicatie is opgebouwd.

# Ontwerp front-end

Afbeelding met tekst, schermopname, Lettertype, lijn

Door AI gegenereerde inhoud is mogelijk onjuist.

De startpagina dient als introductie van de website. Hier wordt kort uitgelegd wat de gebruiker op de site kan doen. De navigatiebalk bovenaan bevat links naar de drie pagina’s: Home, Overzicht en Contact.  
Het tekstvlak in het midden verwelkomt de gebruiker en legt uit dat men de frisdrankverzameling kan bekijken, filteren of sorteren via het overzicht.  
  
Afbeelding met tekst, diagram, Rechthoek, Parallel

Door AI gegenereerde inhoud is mogelijk onjuist.

Dit is de centrale pagina van de webapplicatie. Hier worden 13 producten uit de verzameling getoond in de vorm van tegels (cards). Elk product bevat:   
Een afbeelding   
De naam   
De categorie   
Het suikergehalte   
  
Boven de verzameling bevindt zich een dropdown om op categorie te filteren, en twee knoppen om het suikergehalte oplopend of aflopend te sorteren. Onder de producten zijn knoppen zichtbaar voor paginering (5 producten per pagina).

De gebruiker heeft controle over wat hij wil zien. Alles is duidelijk: filteren, sorteren en bladeren. De structuur is intuïtief en visueel eenvoudig, zodat de aandacht naar de producten gaat.

Afbeelding met tekst, schermopname, Lettertype, nummer

Door AI gegenereerde inhoud is mogelijk onjuist.

De contactpagina bevat een kort bericht en een eenvoudig formulier waarmee bezoekers contact kunnen opnemen.   
Het formulier bevat de velden:   
Naam   
E-mailadres   
Bericht   
Verzendknop   
Het formulier is eenvoudig gehouden, zoals verwacht bij een statische website. De contactpagina versterkt het idee dat de site echt door iemand persoonlijk is opgebouwd, en biedt gebruikers een kanaal om vragen te stellen over de collectie.

# Code review

**Uitvoering van de code review**

Tijdens het ontwikkelproces heb ik regelmatig code reviews uitgevoerd, zowel zelfstandig als met hulp van anderen. De feedbackmomenten vonden plaats binnen mijn intervisiegroep, via ChatGPT en tijdens persoonlijke gesprekken met Marciano Hardoar. Om structuur te behouden, werkte ik steeds in losse Git-branches per onderdeel. Hierdoor kon ik feedback snel toepassen zonder dat dit invloed had op andere onderdelen van de applicatie.  
  
**Belangrijkste bevindingen & beslissingen**

Het scheiden van de productdata in een apart bestand (data.js) bleek overzichtelijker dan deze direct in app.js te plaatsen.

Er werd opgemerkt dat de <title> van de pagina per ongeluk door JavaScript werd aangepast. Dit is aangepast naar een statisch element in de HTML zelf.

De opmaak van de navigatiebalk en de kaarten werd geoptimaliseerd voor leesbaarheid en consistentie, o.a. via suggesties uit de review.  
  
**Feedback ontvangen & gegeven**

Van Marciano Hardoar kreeg ik de tip om bij het schrijven van JavaScript de logica beter te scheiden in functies (zoals apart sorteren, filteren). Dit heeft mij geholpen om de code eenvoudiger te testen en minder foutgevoelig te maken.

Marciano benadrukte ook het belang van een consistente structuur in mijn CSS, zoals het vermijden van dubbele regels en het hanteren van een duidelijke layout.

Ik gaf zelf feedback aan anderen over het correct structureren van hun HTML en het toevoegen van alt-teksten bij afbeeldingen (belangrijk voor toegankelijkheid).

Via ChatGPT kreeg ik advies over DOM-manipulatie en hoe ik producten dynamisch kon genereren met JavaScript.

**Verwerking van feedback**

De code is opgeschoond en voorzien van duidelijke functies per functionaliteit.

De paginering is visueel verbeterd op basis van review-opmerkingen, waarbij de actieve pagina nu duidelijker gemarkeerd is.

CSS is herschreven om herhalingen te vermijden en componenten te hergebruiken via klassen.

**Bewijs van code review**

In mijn Git-geschiedenis zijn commits te vinden zoals refactor: functies gesplitst voor sorteren/filteren of fix: title element verplaatst naar HTML.

Mijn repository bevat meerdere branches die corresponderen met losse ontwikkelonderdelen, wat terug te zien is in de commitstructuur.

# Testplan

|  |  |  |  |
| --- | --- | --- | --- |
| Testnummer | Testdoel | Beschrijving | Resultaat  (geslacht /mislukt) |
| 1 | Paginering werkt correct met 13 producten | Controleer of 13 producten correct verdeeld worden over 3 pagina’s (5-5-3). | Geslaagd |
| 2 | Sorteerfunctie werkt op suikergehalte | Test of sorteren (op- en aflopend) op suikergehalte correct wordt toegepast. | Geslaagd |
| 3 | Filteren op categorie werkt | Test of alleen producten uit de geselecteerde categorie worden weergegeven. | Geslaagd |

De gebruiker opent de overzichtspagina. Op basis van de ingestelde itemsPerPage = 5 worden de producten in het data.js bestand dynamisch weergegeven. De test valideert of:

Pagina 1 → 5 producten toont

Pagina 2 → 5 producten toont

Pagina 3 → 3 producten toont

Navigatie tussen pagina’s werkt via de knoppen  
  
Dit werkt door: const itemsPerPage = 5; // ingesteld in app.js

# Reflectie en Conclusies

**Succesvol afgeronde eisen**

De meeste minimale eisen uit de opdracht zijn succesvol geïmplementeerd:

* De applicatie bestaat uit drie gelinkte pagina’s (Home, Overzicht, Contact), met een duidelijke en werkende navigatiebalk.
* Er zijn 13 hardcoded producten opgenomen in een aparte JavaScript-array in data.js.
* De producten worden weergegeven in de vorm van cards, met elk ten minste vier attributen: naam, suikergehalte (sorteerveld), categorie (filterattribuut) en een afbeelding met alt-tekst.
* De app toont vijf producten per pagina, met correcte paginering over drie pagina’s.
* De gebruiker kan producten sorteren op suikergehalte, oplopend of aflopend, met visuele knoppen.
* De gebruiker kan producten filteren op categorie via een dropdownmenu.
* De website is responsive en past zich correct aan op verschillende schermgroottes.

Niet-uitgevoerde extra’s:  
  
Gebruikers kunnen geen eigen producten toevoegen. Deze keuzes zijn gemaakt om binnen de beschikbare tijd de minimale eisen kwalitatief goed uit te voeren.  
  
**Effectiviteit van het stappenplan**Het opgedeelde stappenplan (planning → structuur → HTML → CSS → JS → testen → documentatie) werkte over het algemeen goed. Doordat ik steeds met losse branches werkte per onderdeel (zoals structure, data, overview, styling, readme), bleef het overzicht behouden en kon ik gefocust werken aan één ding tegelijk.

Wat eventueel beter had gekund:

* De planning had iets ruimer mogen zijn bij het testen en de styling; daar ging meer tijd in zitten dan vooraf gedacht.

# Bronnen

Mathis, L. (2011). *Designed for use: Create usable interfaces for applications and the web*. Pragmatic Bookshelf. (Beschikbaar als e-book via de O'Reilly bibliotheek van De Haagse Hogeschool)

Duckett, J. (2014). *HTML, CSS, and JavaScript all in one* (2nd ed.). Pearson Education. (Beschikbaar via de O'Reilly bibliotheek van De Haagse Hogeschool)

OpenAI. (2024). *ChatGPT*