7-6-2025

[Muhammed (23130032)

HAAGSE HOGE SCHOOL

Logboek

Individuele opdracht Front-end en Back-end, project

# Inhoudsopgave

Inhoud

[Inhoudsopgave 1](#_Toc201340116)

[LOGBOEK FRONT-END 2](#_Toc201340117)

[Inleiding – front-end 2](#_Toc201340118)

[Leerdoelen formuleren: 2](#_Toc201340119)

[Activiteiten front-end 3](#_Toc201340120)

[Reflectie front-end 3](#_Toc201340121)

[LOGBOEK BACK-END 5](#_Toc201340122)

[Inleiding back-end 5](#_Toc201340123)

[Leerdoelen formuleren: 5](#_Toc201340124)

[Activiteiten back-end 6](#_Toc201340125)

[Reflectie back-end 7](#_Toc201340126)

[LOGBOEK EINDPRODUCT 8](#_Toc201340127)

[Inleiding EINDPRODUCT 8](#_Toc201340128)

[Leerdoelen formuleren: 8](#_Toc201340129)

[Activiteiten Eindproduct 9](#_Toc201340130)

[Reflectie Eindproduct 10](#_Toc201340131)

# LOGBOEK FRONT-END

## Inleiding – front-end

In dit logboek houd ik bij hoe ik heb gewerkt aan mijn herkansing voor de individuele front-end opdracht. Het doel van dit verslag is om te laten zien hoe ik het project heb aangepakt, welke keuzes ik heb gemaakt tijdens het bouwen, en wat ik heb geleerd.

Het verslag is opgebouwd uit verschillende onderdelen die samen het hele proces laten zien: van de eerste opzet en voorbereiding tot het afronden van de website. Ik beschrijf per fase wat ik heb gedaan, hoe ik het heb aangepakt en wat ik eventueel anders zou doen.

De opdracht was om een statische website te bouwen met HTML, CSS en JavaScript, zonder externe frameworks of bibliotheken. Op de site moest een verzameling van dertien objecten worden getoond, met mogelijkheden om te sorteren, filteren en bladeren door pagina’s.  
  
Ik had al wat ervaring met HTML en CSS, maar JavaScript vond ik in het begin lastig. Vooral het filteren van data en het dynamisch tonen van informatie op een pagina was nieuw voor mij. Tijdens het bouwen ben ik dat beter gaan begrijpen door veel te testen en rustig stap voor stap te werken. Ook het werken met Git, branches en commits was iets wat ik nu veel gestructureerder heb gedaan dan in eerdere opdrachten.  
  
In deze herkansing heb ik besloten om me te richten op alleen de verplichte onderdelen van de opdracht, zodat ik alles goed werkend kon opleveren zonder te veel extra functies toe te voegen.

## Leerdoelen formuleren:

Leerdoel 1 – Werken met versies en Git (onderzoekend & methodisch werken)  
Aan het eind van deze periode wil ik zelfstandig en correct gebruik kunnen maken van Git en GitHub voor versiebeheer, door bij elke functionele wijziging een aparte branch aan te maken, betekenisvolle commits te schrijven, en correct te mergen naar de dev-branch. Dit blijkt uit een duidelijke commitgeschiedenis en een overzichtelijke projectstructuur in mijn repository.  
Leerdoel 2 – Begrijpen en toepassen van JavaScript in dynamische content

Ik wil voor het einde van deze opdracht begrijpen hoe ik met JavaScript dynamisch content kan tonen op een webpagina, door producten vanuit een JS-array te filteren, sorteren en weer te geven met behulp van DOM-manipulatie. Ik weet dat ik dit doel heb bereikt als ik zelfstandig minimaal 13 producten via cards op een pagina toon met werkende sorteer- en filterfuncties.

## Activiteiten front-end

Om mijn leerdoelen te behalen en de opdracht succesvol af te ronden, heb ik verschillende stappen doorlopen. Hieronder beschrijf ik de belangrijkste activiteiten:

1. Opzetten van de mappenstructuur en repository  
Ik ben begonnen met het goed inrichten van mijn project, zodat alles overzichtelijk en gestructureerd bleef. In een aparte branch (structure) heb ik een logische mappenstructuur opgezet met aparte mappen voor HTML, CSS, JS, afbeeldingen en documentatie. Dit hielp me later om snel onderdelen terug te vinden en om conflicten in Git te voorkomen.  
  
2. Plannen en faseren met branches  
Ik heb van tevoren bedacht in welke volgorde ik de onderdelen ging bouwen, en per stap een nieuwe branch gemaakt. Bijvoorbeeld: eerst index.html, daarna style.css, vervolgens overzicht.html met JS-functionaliteit. Door deze aanpak kon ik gefocust werken per onderdeel en tussentijds testen zonder alles tegelijk te moeten afmaken.  
  
3. Zelfstudie en tutorials  
Omdat ik nog niet zo vertrouwd was met JavaScript en DOM-manipulatie, heb ik meerdere tutorials bekeken over het maken van dynamische content met arrays, loops, en document.createElement(). Wat ik ook heb gedaan is de presenatie bekijken van de docent waarin ik handige linkjes vond om mijn front-end te testen.  
  
4. Coderen met constante feedback en testmomenten  
Tijdens het bouwen heb ik vaak getest in de Ik heb ook feedback gevraagd in mijn intervisiegroep over bijvoorbeeld het sorteren en pagineren.  
  
5. Gebruik maken van Git en GitHub

Ik heb voor elk onderdeel van mijn site aparte commits gemaakt met duidelijke berichten. Ook heb ik regelmatig gemerged naar een main branch om alles samen te brengen. In de README heb ik uitgelegd hoe het project is opgebouwd.

## Reflectie front-end

Situatie:  
Voor deze opdracht moest ik een statische front-end webapplicatie bouwen waarin een persoonlijke verzameling werd getoond. Ik koos ervoor om een collectie frisdranken te maken, met filter- en sorteeropties.

Taak:

Mijn doel was om JavaScript beter te leren gebruiken om dynamische content te tonen, en daarnaast om gestructureerd met versiebeheer (Git) te werken via branches en commits.

Actie:  
Ik heb het project opgedeeld in logische stappen, en per stap gewerkt in aparte branches zoals structure, overview-page, style-branch, data, enz. Voor JavaScript heb ik tutorials gevolgd, documentatie gelezen en veel getest om beter te begrijpen hoe filtering, sorteren en DOM-manipulatie werken.  
  
  
  
  
Resultaat:  
  
Beide leerdoelen heb ik behaald:   
De producten worden nu volledig via JavaScript opgebouwd vanuit een array   
Ik gebruik filtering, sortering, en paginanumers   
Mijn Git-geschiedenis is duidelijk, met correcte branches en commitberichten  
  
Reflectie:  
Deze manier van werken gaf me rust en overzicht. Ik heb geleerd dat structuur vooraf enorm helpt tijdens het coderen. Ook ben ik veel zekerder geworden in het toepassen van JavaScript en het begrijpen van functies en event-handlers.  
  
Transfer:  
  
Ik neem deze aanpak en vaardigheden zeker mee naar de volgende opdracht of sprint, ook als ik later met back-end of frameworks ga werken.

# LOGBOEK BACK-END

## Inleiding back-end

Dit logboek is opgesteld als onderdeel van de herkansing van de individuele back-end opdracht. Het doel van dit document is om de voortgang, aanpak en leerervaringen tijdens het ontwikkelen van een back-endoplossing voor een RESTful API in kaart te brengen. De opbouw volgt de verplichte onderdelen: oriëntatie, leerdoelen, activiteiten, reflectie en vervolgdoelen. De opdracht is uitgevoerd op basis van de sprinthandleiding uit Brightspace, met daarin de verplichte stappen 1 t/m 7 en 10.

Bij aanvang van dit project had ik geen ervaring met back-end development. Alles van het opzetten van een mock-server tot het bouwen van een Express-server – heb ik voor het eerst geleerd en toegepast tijdens dit traject. In het bijzonder wilde ik leren hoe API’s technisch werken, hoe je deze test met Postman, en hoe je foutafhandeling inbouwt. Daarnaast had ik extra begeleiding nodig bij het structureren van mijn back-endcode en documentatie. Van mijn teamgenoten.

## Leerdoelen formuleren:

**Leerdoel 1 (SMART):** Aan het eind van deze opdracht kan ik zelfstandig een eenvoudige RESTful API ontwerpen en implementeren met Node.js en Express, inclusief correcte foutafhandeling (statuscodes 404 en 400), zodat ik CRUD-functionaliteit kan leveren voor een verzameling frisdranken. Dit sluit aan bij de beroepstaak *"Ontwerpt en realiseert backend-functionaliteit"*. Zodat ik het duidelijker kan maken voor de eindgebruiker wat de foutmelding eigenlijk inhouden.

**Leerdoel 2 (SMART):** Ik wil leren hoe ik een Postman-collectie kan gebruiken om alle endpoints van mijn API te testen op correctheid en stabiliteit. Mijn doel is om minimaal 9 tests uit te voeren (GET, POST, PUT, DELETE + fouten) en alle responses vast te leggen in een testplan. Dit sluit aan bij de beroepstaak *"Test en documenteert backend-functionaliteit"*. Aan de hand hiervan kan ik mijn applicatie testen waarbij dat nodig is en kan ik dit toepassen in een productie omgeving.

## Activiteiten back-end

1. **Oriëntatie via literatuur en tutorials**   
   Ik ben begonnen met het verdiepen in de kernconcepten HTTP, API, REST en JSON. Ik heb hoofdstuk 8.1 t/m 8.4 gelezen uit Stanford University's API Basics (Altman, Behrman, Wickham, 2021), en YouTube-video’s bekeken over API’s en JSON. Dit was nodig omdat ik geen voorkennis had en alles vanaf nul moest opbouwen.
2. **Opzetten JSON-server + testdata aanmaken**   
   Daarna heb ik Node.js geïnstalleerd en met npm een JSON-server opgezet. Ik heb een db.json bestand gemaakt met 13 drankjes die eerder in mijn front-end zaten. Dit maakte het mogelijk om snel endpoints te simuleren en te oefenen met GET/POST requests.
3. **Postman leren gebruiken en testplan opzetten** Ik heb een Postman-collectie opgezet met 9 testscenario’s, inclusief foutgevallen zoals een DELETE van een niet-bestaand ID. Door bij elke test het verwachte gedrag vast te leggen, kon ik gericht controleren of mijn Express-backend correct werkte. Deze collectie is geëxporteerd als bewijsmateriaal.
4. **Bouwen van de Express-backend (hardcoded)**   
   Vervolgens heb ik met Express een eigen server gebouwd (server.js). Hierin staan alle 13 drankjes hardcoded. Ik heb endpoints aangemaakt voor alle CRUD-methodes met correcte foutafhandeling en gebruik van middleware (cors, express.json). Ik heb besloten geen fetch toe te passen, omdat dit niet verplicht was.
5. **Code review, feedback en bijsturen**   
   Tijdens het bouwen heb ik feedback gevraagd aan een medestudent. Hij adviseerde me om foutmeldingen specifieker te maken (bijv. "Drankje niet gevonden" i.p.v. een algemene error). Ook wees hij erop dat mijn Postman-collectie nog foutscenario’s miste. Ik heb dit direct verwerkt en de foutmeldingen verbeterd in mijn Express-routes.

## Reflectie back-end

**Situatie:**  
Als herkansende student moest ik een complete back-end ontwikkelen die mijn front-end uit sprint 1 aanvult. De opdracht vroeg om het bouwen van een RESTful API met Node.js en Express, inclusief testen, documenteren en foutafhandeling.

**Taak:**  
Ik moest zelfstandig leren hoe back-end ontwikkeling werkt, een mock-server bouwen, een echte Express-server opzetten, alle functionaliteit testen en de backend documenteren in lijn met de eisen van de opleiding.

**Actie:**  
Ik heb gewerkt volgens de stappen uit de handleiding (1 t/m 7 en 10). Ik heb literatuur bestudeerd, externe bronnen geraadpleegd, zelfstandig code opgebouwd in Express, en Postman gebruikt om mijn werk te testen. Alle stappen zijn vastgelegd in een gedetailleerd logboek en ontwerpdocument.

**Resultaat:** Mijn API werkt correct. Alle endpoints geven de juiste responses, inclusief foutmeldingen. De backend is hardcoded, maar volledig functioneel. Ik heb mijn Postman-testplan afgerond, de collectie geëxporteerd en mijn project correct gestructureerd met GitHub en mappen voor API, JSON-server en documentatie.

**Reflectie:** Ik heb mijn leerdoelen behaald. Door het werken met Express en Postman begrijp ik hoe back-endtechnologieën functioneren. Ik ben zelfstandiger geworden in het oplossen van problemen, zoals het debuggen van endpoints en het verbeteren van foutafhandeling. Mijn denkvermogen heb ik gebruikt bij het analyseren van Postman-output.

**Feedback:** Feedback kreeg ik via peer review. De foutmeldingen waren aanvankelijk te generiek. Dit heb ik aangepast zodat elke foutmelding duidelijk en RESTful is. Ook heb ik extra testgevallen toegevoegd aan Postman zoals geadviseerd. Feedback heb ik verzameld door mijn project te laten zien aan een klasgenoot en hem concrete vragen te stellen.

**Toegepaste leerstrategieën:**

* Actief vragen stellen aan klasgenoten
* Literatuur analyseren en aantekeningen maken
* Zelf onderzoek doen naar HTTP en REST via Stanford
* Iteratief testen en verbeteren van eigen werk
* Reflecteren na elk afgerond onderdeel

**Problemen en oplossingen:**

* Ik had in het begin moeite met hoe ik Postman moest gebruiken. Door video’s te kijken en zelf te testen heb ik het onder de knie gekregen.
* Mijn API gaf in het begin geen correcte foutmeldingen. Door logs in te bouwen en de juiste statuscodes op te zoeken in de literatuur, kon ik dit oplossen.

**Kritisch denken:** Ik heb actief nagedacht over hoe ik mijn API zo RESTful mogelijk kon maken. Bijvoorbeeld door gebruik van juiste HTTP-methodes, foutafhandeling, en het teruggeven van JSON-data met de juiste structuur. Ik heb ook kritisch gekeken naar mijn projectstructuur en deze zo logisch mogelijk opgebouwd.

# LOGBOEK EINDPRODUCT

## Inleiding EINDPRODUCT

In dit logboek geef ik een overzicht van mijn persoonlijke bijdrage aan het eindproduct van de Afspraakplanner. Daarbij richt ik mij op mijn leerproces, de activiteiten die ik heb uitgevoerd en mijn inzichten tijdens het ontwikkelen van de backend, het inrichten van de hostingomgeving via Azure en mijn betrokkenheid bij het ontwerpen van de datastructuur en samenwerking met het front-end team.  
  
Hoewel ik al basiskennis had van Node.js en API-ontwikkeling, was het voor mij nieuw om te werken met geavanceerde foutafhandeling, meerdere gebruikersrollen en CI/CD-processen voor automatische uitrol naar Azure. Ik vond het belangrijk om mezelf te ontwikkelen in zowel backend als frontend om een breder begrip te krijgen van software developer.

## Leerdoelen formuleren:

Tijdens dit project wilde ik mij verder ontwikkelen in het opzetten van een backendapplicatie met een duidelijke structuur en een veilige REST API. In eerdere opdrachten merkte ik dat ik vaak bleef hangen in eenvoudige backendlogica zonder goed schaalbare routes, validatie of foutafhandeling. Daarom wilde ik dit keer leren hoe ik een backend kan opbouwen die bestaat uit losse modules met duidelijke verantwoordelijkheden, waarbij gebruikersbeheer, dataopslag en foutafhandeling goed zijn ingericht. Het doel was om een robuuste backend op te zetten die de functionele eisen van de Afspraakplanner ondersteunt en eenvoudig uitbreidbaar is.

Daarnaast wilde ik me verdiepen in het automatisch uitrollen van webapplicaties naar een hostingplatform, specifiek via GitHub Actions naar Azure. Voorheen deed ik alle deployments handmatig, wat foutgevoelig en tijdrovend was. Ik vond het belangrijk om te leren hoe ik met CI/CD-processen mijn backend en frontend automatisch kan bouwen en publiceren, zodat updates sneller en betrouwbaarder kunnen worden doorgevoerd. Mijn doel was om een volledige workflow in te richten waarin de applicatie automatisch gedeployed wordt zodra wijzigingen op de main-branch worden gepusht, inclusief de juiste configuratie, authenticatie en fallback-oplossingen.

## Activiteiten Eindproduct

Activiteiten Eindproduct

Ontwerpen van het ERD en de database-structuur

* Samen met mijn teamgenoten heb ik de entiteiten en relaties binnen het systeem uitgewerkt in een Entity Relationship Diagram (ERD).
* *Keuze:* Door dit eerst te structureren, hadden we een duidelijk fundament voor de backend en werd dataconsistentie gewaarborgd.

Opzetten van backendstructuur en API’s

* Lukas en ik heb RESTful API-endpoints gebouwd voor gebruikers, klassen, afspraken en oudercodes.
* *Keuze:* Modularisatie maakte het eenvoudiger om per rol gerichte functionaliteiten te bouwen.

Foutafhandeling en validatie implementeren

* Voor elke route implementeerde ik server-side validatie en duidelijke foutmeldingen.
* *Keuze:* Zo zorgde ik voor meer stabiliteit en veiligheid van het systeem.

CI/CD-pipeline configureren voor deployment

* Met GitHub Actions heb ik scripts geschreven om zowel de backend als frontend automatisch uit te rollen naar Azure App Services.
* *Keuze:* Deze aanpak verhoogt betrouwbaarheid en bespaart tijd bij updates.

Afstemming en ondersteuning front-end integratie

* Tijdens het integreren van front-end componenten heb ik ondersteuning geboden bij API-aanroepen en meegekeken bij problemen.
* *Keuze:* Samenwerking bevordert consistentie en voorkomt mismatch in dataformaten of logica.

## Reflectie Eindproduct

**Situatie (S):**Binnen het project Afspraakplanner was ik verantwoordelijk voor de backendontwikkeling. Dit omvatte het opzetten van de API, het structureren van de backendlogica, het configureren van de databaseverbindingen, en het inrichten van de automatische deployment naar Azure.

**Taak (T):**Mijn taak was om een betrouwbare, schaalbare en goed gedocumenteerde backend te bouwen die veilig en efficiënt communiceert met de front-end. De backend moest gebruikersrollen ondersteunen, foutafhandeling bevatten en aansluiten op de ERD en projecteisen.

**Actie (A):**Ik heb RESTful API’s opgezet voor onder andere gebruikersbeheer, afspraken en oudercodes. Daarnaast heb ik validatie- en foutafhandeling toegevoegd, en met GitHub Actions een CI/CD-pipeline ingericht voor automatische uitrol naar Azure. Tijdens integratie met de front-end heb ik ondersteuning geboden bij het testen en verifiëren van API-functionaliteit. Bij technische problemen heb ik documentatie geraadpleegd, tutorials gevolgd en hulp gevraagd aan teamleden

**Resultaat (R):**De backend is stabiel, modulair opgebouwd en bevat alle benodigde endpoints voor de applicatie. Dankzij CI/CD is de applicatie automatisch te deployen, wat zorgt voor een efficiënte workflow. De API sluit goed aan op de front-end en alle datastromen functioneren zoals bedoeld.

**Reflectie (R):**Mijn doelen zijn behaald. Ik heb geleerd hoe ik een professionele backendstructuur opzet en hoe ik deze automatisch kan deployen via Azure. Daarnaast heb ik ervaring opgedaan met het samenwerken aan een project, waarin ik ook inzicht kreeg in de eisen en werking van de front-end en backend.

**Toegepaste leerstrategieën:**

* API-documentatie en foutcodes opgezocht (o.a. bij MDN en OWASP)
* Feedback gevraagd aan front-endteam over API-integratie
* Testscenario’s opgesteld en gecontroleerd met Postman

**Feedback en toepassing:**

* ***Feedback:*** De API werkt wel, maar responses zijn niet overal even duidelijk.  
  Ik heb beschrijvingen en foutcodes toegevoegd aan responses, inclusief statuscodes (200, 400, 401,)
* **Feedback:** “De CI/CD faalt soms bij deployment.”  
   Opgelost door de juiste padstructuur aan te houden en environment variables correct in te stellen via Azure en GitHub Secrets.

**Problemen:**

* **Probleem:** deployment werkte niet vanwege ontbrekende build-stappen  
  Opgelost door middleware toe te voegen voor role-based access control
* **Probleem:** Verkeerde data-opslag door inconsistente key-namen

Opgelost door backendtypes te herzien en tests te automatiseren in Postman

**Transfer naar het volgende semester:**

Volgend semester wil ik me richten op automatisch testen van backendfunctionaliteit met Jest of Supertest, zodat ik fouten sneller kan detecteren zonder handmatig testen. Ook wil ik mijn kennis van state management aan de front-endzijde verdiepen, zoals Context API of Redux, zodat ik beter begrijp hoe data-uitwisseling tussen front-end en back-end.