**Ontwerpdocument BE**

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

Inhoudsopgave

[Inleiding 4](#_Toc201339993)

[Projectplan 5](#_Toc201339994)

[1. Oriëntatie en basiskennis 5](#_Toc201339995)

[2. Opzetten van een mock-server 5](#_Toc201339996)

[3. Ontwerpen en testen van de API 5](#_Toc201339997)

[4. Implementatie met Express.js 6](#_Toc201339998)

[5. Koppeling met front-end en documentatie 6](#_Toc201339999)

[Ontwerp back-end 7](#_Toc201340000)

[Code review 9](#_Toc201340001)

[Samenvatting van de bevindingen: 9](#_Toc201340002)

[Feedback ontvangen: 9](#_Toc201340003)

[Feedback gegeven: 9](#_Toc201340004)

[Verwerking van feedback: 9](#_Toc201340005)

[Bewijs van code review: 9](#_Toc201340006)

[Testplan 11](#_Toc201340007)

[Testuitvoering 12](#_Toc201340008)

[Resultaten 12](#_Toc201340009)

[Reflectie en Conclusies 13](#_Toc201340010)

[Bronnen 15](#_Toc201340011)

# Inleiding

Voor deze opdracht ontwikkel ik een back-end applicatie als uitbreiding op de front-end webapplicatie uit sprint 1. De back-end is gericht op het ontwerpen, implementeren en testen van een RESTful API die communiceert met een front-end verzameling van frisdranken.

De applicatie is bedoeld als individuele opdracht voor het toepassen van de basisprincipes van back-end development, waaronder het gebruik van Node.js, Express.js, JSON-server en Postman.

Het doel van de back-end is om in staat te stellen om frisdranken uit de collectie te bekijken, toevoegen, wijzigen en verwijderen via API-endpoints. De gegevens worden aangeboden in JSON-formaat, en zijn via de API benaderbaar op zowel op de gehele collectie als individueel  
  
De back-end wordt lokaal gedraaid en getest, met een duidelijke projectstructuur en uitgebreide documentatie van de endpoints.

# Projectplan

Om dit back-end project succesvol te voltooien, volg ik een gestructureerde aanpak bestaande uit meerdere fasen. De applicatie wordt ontwikkeld in een Visual Studio Code met gebruik van Node.js, Express.js, en Postman. Voor versiebeheer maak ik gebruik van Git en GitHub.

Het back-end projectplan bestaat uit de volgende vijf fasen:

## 1. Oriëntatie en basiskennis

De basisconcepten zoals HTTP, JSON en RESTful API's zijn belangrijk voor het begrijpen van communicatie tussen client en server. Door deze concepten eerst te bestuderen, kon ik gerichter werken in latere stappen. Het gaf me inzicht in hoe een API werkt, wat een endpoint precies is en hoe data wordt gestructureerd en uitgewisseld via JSON.

**Beslissing:** Eerst verdiepen in theorie zodat implementatie later efficiënter en correct verloopt.

## 2. Opzetten van een mock-server

Voordat ik begon met een echte server, heb ik gekozen om een JSON-server te gebruiken. Dit maakte het mogelijk om snel te oefenen met API-aanroepen zonder complexe logica of databaseconfiguraties. Het maken van een RESTful API met JSON-server bood een manier om het gedrag van endpoints en HTTP-methodes te begrijpen.

**Beslissing:** Gebruik van een mock-server voor snelle feedback en om basisfunctionaliteit te leren zonder de complexiteit van een echte backend.

## 3. Ontwerpen en testen van de API

Ik heb me gericht op het opzetten en testen van de vier belangrijkste CRUD-functionaliteiten:

* GET om alle drankjes of een specifiek drankje op te halen,
* POST om een nieuw drankje toe te voegen,
* PUT om een bestaand drankje te bewerken,
* DELETE om een drankje te verwijderen.

Deze functies zijn een voor een gebouwd en getest met Postman. Door meteen te testen na het bouwen van elke functie, kon ik snel zien of alles werkte zoals verwacht. Zo kon ik direct foutafhandeling toevoegen en controleren of de responses correct waren.

**Beslissing:** Gericht gewerkt op de basisfunctionaliteiten (CRUD) en deze direct getest met Postman voor om te zien of het werkt.

## 4. Implementatie met Express.js

Express is een populair framework binnen de Node.js-omgeving. Door hiermee te werken kreeg ik meer verstand over de serverlogica, foutafhandeling. In plaats van alleen een mock-server te gebruiken, bouwde ik een volledig werkende server met hardcoded data voor maximale controle.

**Beslissing:** Express gekozen voor professionele ervaring met echte back-end ontwikkeling.

## 5. Koppeling met front-end en documentatie

Hoewel fetch-integratie optioneel was, heb ik wel getest hoe de front-end gegevens uit een API zou kunnen ophalen. Daarnaast is alles zorgvuldig gedocumenteerd, van de endpoints en testresultaten tot de opbouw van de projectmap.

**Beslissing:** Duidelijke documentatie gekozen zodat de werking en structuur van de API eenvoudig te begrijpen en controleren is.

# Ontwerp back-end

|  |  |
| --- | --- |
| Section | Description |
| Description | Deze API beheert een collectie van frisdranken. De API ondersteunt basisfunctionaliteiten: ophalen, toevoegen, bewerken en verwijderen van frisdranken. |
| Authentication | Niet van toepassing – er is geen authenticatie nodig. De API draait lokaal. |
| Base URL | http://localhost:3000 |
| Endpoints | GET /drinks – Haalt alle frisdranken op GET /drinks/:id – Haalt één specifieke frisdrank op POST /drinks – Voegt een nieuwe frisdrank toe PUT /drinks/:id – Past een frisdrank aan DELETE /drinks/:id – Verwijdert een frisdrank |
| Resource model | {  "id": 1,  "name": "Coca-Cola Original",  "category": "Cola",  "sugar": 35,  "image": "images/coca\_cola.jpg" } |
| Request Methods & Parameters | GET /drinks: geen parameters GET /drinks/:id: path parameter 'id' POST /drinks: body met JSON data Voorbeeld: {  "name": "Fanta Exotic",  "category": "Fruitdrank",  "sugar": 32,  "image": "images/fanta\_exotic.jpg" } PUT /drinks/:id: path parameter 'id' + body met JSON data DELETE /drinks/:id: path parameter 'id' |
| Response Format | JSON |
| Error Handling | 200 OK – Request succesvol uitgevoerd 201 Created – Nieuw item aangemaakt 400 Bad Request – Ongeldige data 404 Not Found – Resource niet gevonden |
| Voorbeeld Responses | GET /drinks response voorbeeld: [  {  "id": 1,  "name": "Coca-Cola Original",  "category": "Cola",  "sugar": 35,  "image": "images/coca\_cola.jpg"  },  {  "id": 2,  "name": "Fanta Orange",  "category": "Fruitdrank",  "sugar": 33,  "image": "images/fanta\_orange.jpg"  } ] etc. POST /drinks voorbeeld: Request body: {  "name": "New Test Drink",  "category": "Test",  "sugar": 99,  "image": "images/test.jpg" }  Response: {  "id": 14,  "name": "New Test Drink",  "category": "Test",  "sugar": 99,  "image": "images/test.jpg" } |

# Code review

Tijdens het ontwikkelen van de back-end applicatie heb ik regelmatig feedback gevraagd aan klasgenoten, met name tijdens het werken aan de Express-server en het testen van de CRUD-functionaliteiten. Er is een code review uitgevoerd door een medestudent die zelf al klaar was met zijn opdracht.

## Samenvatting van de bevindingen:

* De structuur van de Express-code was duidelijk en goed leesbaar.
* De foutafhandeling (bijv. bij een onbestaand ID) werkte correct.
* Alle CRUD-functionaliteiten waren aanwezig en functioneerden zoals verwacht.
* De keuze voor hardcoded data in plaats van fetch of database werd als prima beoordeeld, omdat het voldeed aan de eisen van de opdracht.

## Feedback ontvangen:

* De foutmeldingen (404 en 400) konden specifieker en duidelijker worden geformuleerd.
* Mijn postman kon wat test queries hebben.

## Feedback gegeven:

* Ik heb feedback gegeven aan een medestudent over het juist instellen van CORS in zijn Express-app.

## Verwerking van feedback:

* Alle wijzigingen zijn direct getest in Postman
* Error responses zijn geüpdatet met duidelijke foutboodschappen zoals "Drankje niet gevonden". In postman

## Bewijs van code review:

* Ik heb een export toegevoegd van postman met daarin een folder tests die ik heb uitgevoerd.
* Github commitgeschiedenis (<https://github.com/DirsDex/Back-end-individuele-opdracht-23130032>)
* Afbeelding met tekst, schermopname, software, Multimediasoftware

  Door AI gegenereerde inhoud is mogelijk onjuist.

# Testplan

Om de werking van de RESTful API te garanderen, heb ik alle CRUD-functionaliteiten getest met behulp van Postman. Elke test is ontworpen om een specifiek gedrag van de API te valideren, inclusief correcte responses, foutafhandeling en dataconsistentie.

Alles staat ook in de Postman die ik heb verstuurd zowel in de repository van git.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Testnr** | **Methode** | **Endpoint** | **Doel** | **status** | **respons** |
| 1 | GET | /drinks | Alle drankjes ophalen | 200 OK | Array van drank-objecten |
| 2 | GET | /drinks/1 | Specifiek drankje ophalen | 200 OK | Drankje met id: 1 |
| 3 | GET | /drinks/999 | Ongeldig ID ophalen | 404 Not Found | Foutmelding |
| 4 | POST | /drinks | Nieuw drankje toevoegen | 201 Created | Toegevoegd object met nieuw ID |
| 5 | PUT | /drinks/2 | Drankje bewerken | 200 OK | Aangepast object |
| 6 | PUT | /drinks/999 | Ongeldig ID bewerken | 404 Not Found | Foutmelding |
| 7 | DELETE | /drinks/3 | Drankje verwijderen | 200 OK | Verwijderd object |
| 8 | DELETE | /drinks/999 | Ongeldig ID verwijderen | 404 Not Found | Foutmelding |

**Test 4: POST /drinks**

Deze test controleert of de API correct een nieuw drankje kan toevoegen aan de bestaande collectie.  
  
**Dit is een goede test omdat:**

* Deze test controleert of de server correct input via JSON verwerkt
* Toont aan of de ID automatisch wordt gegenereerd
* Bevestigt of de nieuwe data correct wordt teruggegeven als bevestiging aan de gebruiker

Afbeelding met schermopname, tekst, software, Multimediasoftware

Door AI gegenereerde inhoud is mogelijk onjuist.

## Testuitvoering

* De tests zijn uitgevoerd met Postman, waarbij gebruik werd gemaakt van de Body-tab (JSON),
* Responses zijn gecontroleerd op correcte inhoud en formaat (JSON).

## Resultaten

* Alle GET/POST/PUT/DELETE-operaties werkten correct volgens specificaties.
* Error-handling werkte zoals verwacht bij ongeldige ID’s.
* Data bleef consistent tijdens meerdere bewerkingen.

# Reflectie en Conclusies

**Succesvol afgeronde eisen**

De meeste minimale eisen uit de back-end opdracht zijn succesvol geïmplementeerd:

* De applicatie bevat een Express-server met volledige CRUD-functionaliteit: ophalen, toevoegen, bewerken en verwijderen van frisdranken.
* Er zijn 13 hardcoded frisdranken opgenomen in een array in server.js, elk met de attributen: id, name, category, sugar en image.
* De API ondersteunt de HTTP-methodes: GET, POST, PUT en DELETE, en geeft correcte statuscodes en JSON-responses terug.
* De API is getest met Postman, waarbij alle endpoints correct functioneerden en ook foutgevallen (zoals onbestaande ID’s) werden getest.
* Er is een Postman-collectie opgesteld en geëxporteerd als bewijs.
* De API is gedocumenteerd met een duidelijke beschrijving van de endpoints, parameters, en voorbeeldresponses.
* De structuur van het project is helder, met mappen voor api, json-server, frontend, postman en documentatie.

**Niet-uitgevoerde extra’s:**

* De front-end is niet gekoppeld met fetch aan de back-end, omdat dit geen verplichte eis was. Er is gekozen voor een hardcoded Express-implementatie, zoals toegestaan in de opdracht.
* Werken met ORM (Object Relational Mapping)
* Deployen naar Azure

**Effectiviteit van het stappenplan**

Het opgedeelde stappenplan (oriëntatie → JSON-server → testen in Postman → Express API → documentatie) heeft goed gewerkt. Door stap voor stap te werken kon ik me goed focussen op de werking van elke fase.

Wat goed werkte:

* Testen in Postman tijdens het bouwen hielp bij het snel vinden en oplossen van fouten.
* De keuze voor hardcoded data maakte het eenvoudiger om logica te controleren en endpoints betrouwbaar te maken.

Wat eventueel beter had gekund:

* Meer tijd plannen voor documentatie; het schrijven van het ontwerpdocument en het verzamelen van bewijsmateriaal nam meer tijd in beslag dan verwacht.
* In een volgende versie zou ik willen oefenen met koppeling via fetch(), om front-end en back-end live met elkaar te laten communiceren.

# Bronnen

* De Haagse Hogeschool. (2025). *Backend Sprint Handleiding – Individuele opdracht backend (Stappen 1 t/m 7 en stap 10)* [Digitale leeromgeving Brightspace]. Geraadpleegd op 23 juni 2025, van <https://brightspace.hhs.nl/d2l/le/lessons/121906/topics/950277>
* Altman, S., Behrman, B., & Wickham, H. (2021, september 10). *Chapter 8: API Basics (Sections 8.1–8.4)*. Stanford University. Geraadpleegd op 20 juni 2025, van <https://dcl-wrangle.stanford.edu/api-basics.html>
* OpenAI. (2024). *ChatGPT*