**Ontwerpdocument BE**

|  |  |
| --- | --- |
| **Productnaam** | Individuele Opdracht: Back-end |
| **Studentnummer en naam** | **23083344**  Khalid Alouat |
| **Groep** | Groep 3 |

# Inhoudsopgave

Contents

[Inhoudsopgave 2](#_Toc193839505)

[Inleiding 2](#_Toc193839506)

[Projectplan 3](#_Toc193839507)

[Ontwerp back-end 4](#_Toc193839508)

[Code review 5](#_Toc193839509)

[Testplan 7](#_Toc193839510)

[Reflectie en Conclusies 10](#_Toc193839511)

[Bronnen 11](#_Toc193839512)

# Inleiding

Deze back‑end applicatie vormt de kern van mijn motorverzameling, waarmee de front‑end data kan opvragen, toevoegen, wijzigen en verwijderen via een RESTful API.

**Functie van de applicatie:**

* Een CRUD‑API ontwikkelen in Express.js, die communiceert met een front‑end webapplicatie.
* Het werken met HTTP en JSON als intermediair, conform de principes van RESTful API’s.

**Vereisten:**

* Implementeer een API met endpoints voor het ophalen, toevoegen, wijzigen en verwijderen van motoren.
* Documenteer de API (met Swagger/OpenAPI).
* Zorg dat de front‑end applicatie via fetch‑calls data ophaalt en wijzigt in de back‑end.
* (Extra) Integreer persistente opslag met behulp van Sequelize en een SQLite‑database, zodat data behouden blijft na herstart.

# Projectplan

1. **Planning & Oriëntatie**

* Bestuderen van de literatuur over HTTP, RESTful API’s, en JSON.
* Bepalen welke endpoints nodig zijn voor CRUD‑functionaliteit (GET, POST, PUT, DELETE) en het ontwerpen van de API.

1. **Prototype met JSON-server**

* Een db.json bestand aangemaakt met mockdata (13 motoren) en een JSON-server opgezet om snel de werking van de API te testen.

1. **Implementatie Express API**

* Express en Cors geïnstalleerd, en een server (server.js) ontwikkeld met CRUD‑routes.
* Postman‑collectie gemaakt en getest om de endpoints te valideren.

1. **Documentatie met Swagger**

* Swagger (swagger-jsdoc en swagger-ui-express) geïntegreerd in server.js voor automatische API-documentatie op /api-docs.

1. **ORM-integratie met Sequelize**

* Persistente opslag toegevoegd door gebruik te maken van Sequelize met een SQLite‑database.
* Een Motor‑model gedefinieerd en een seeder-script (seeder.js) ontwikkeld om de oorspronkelijke mockdata uit db.json in de database te laden.

1. **Integratie met Front-end**

* De front‑end aangepast zodat deze via fetch‑calls met de back‑end API communiceert.
* De API is getest en werkt samen met de bestaande front‑end applicatie.

1. **Feedback en Testing:**

* Uitgebreide tests uitgevoerd met Postman en via de front‑end.
* Feedback verwerkt en problemen (zoals asynchrone issues en data persistentie) opgelost.

# Ontwerp back-end

**Architectuur van mijn applicatie:**

**Gebruikte technologieën:**

* Node.js, Express.js, Sequelize (ORM), SQLite, Swagger (voor documentatie).

**API Design:**

**Resource:** Motoren

**Alle endpoints:**

* **GET /** Welkomstbericht bij de root van de webapplicatie
* **GET /motoren** Haalt alle motoren op
* **GET /motoren/:id** Haalt een specifieke motor op
* **POST /motoren** Voegt een nieuwe motor toe
* **PUT /motoren/:id** Werkt een motor bij
* **DELETE /motoren/:id** Verwijdert een motor

**Communicatie:**

* Data-uitwisseling via JSON
* HTTP-methoden (GET, POST, PUT, DELETE) volgens REST-principes

**Data-opslag:**

* In de prototypefase werd een db.json gebruikt met mockdata.
* In de uiteindelijke implementatie wordt persistente opslag gerealiseerd met een SQLite‑database.
* Het Motor‑model bevat de attributen: naam (string, verplicht), bouwjaar (integer, verplicht), merk (string, verplicht), afbeelding (string, met een standaardwaarde) en alt (string).

**Swagger Documentatie:**

* Door middel van Swagger zijn alle endpoints automatisch gedocumenteerd.
* De Swagger UI is beschikbaar via /api-docs en geeft de API-specificaties en voorbeeldresponses weer.

**Seeding:**

* Een seeder-script (seeder.js) is ontwikkeld om de initiële mockdata uit db.json in de SQLite‑database te laden.
* Hierdoor is de database vooraf gevuld met 14 motoren, zodat de API vanaf de start functioneel is.

# Code review

De code werd tijdens een gesprek op Teams gereviewd door mijn groepsgenoten.

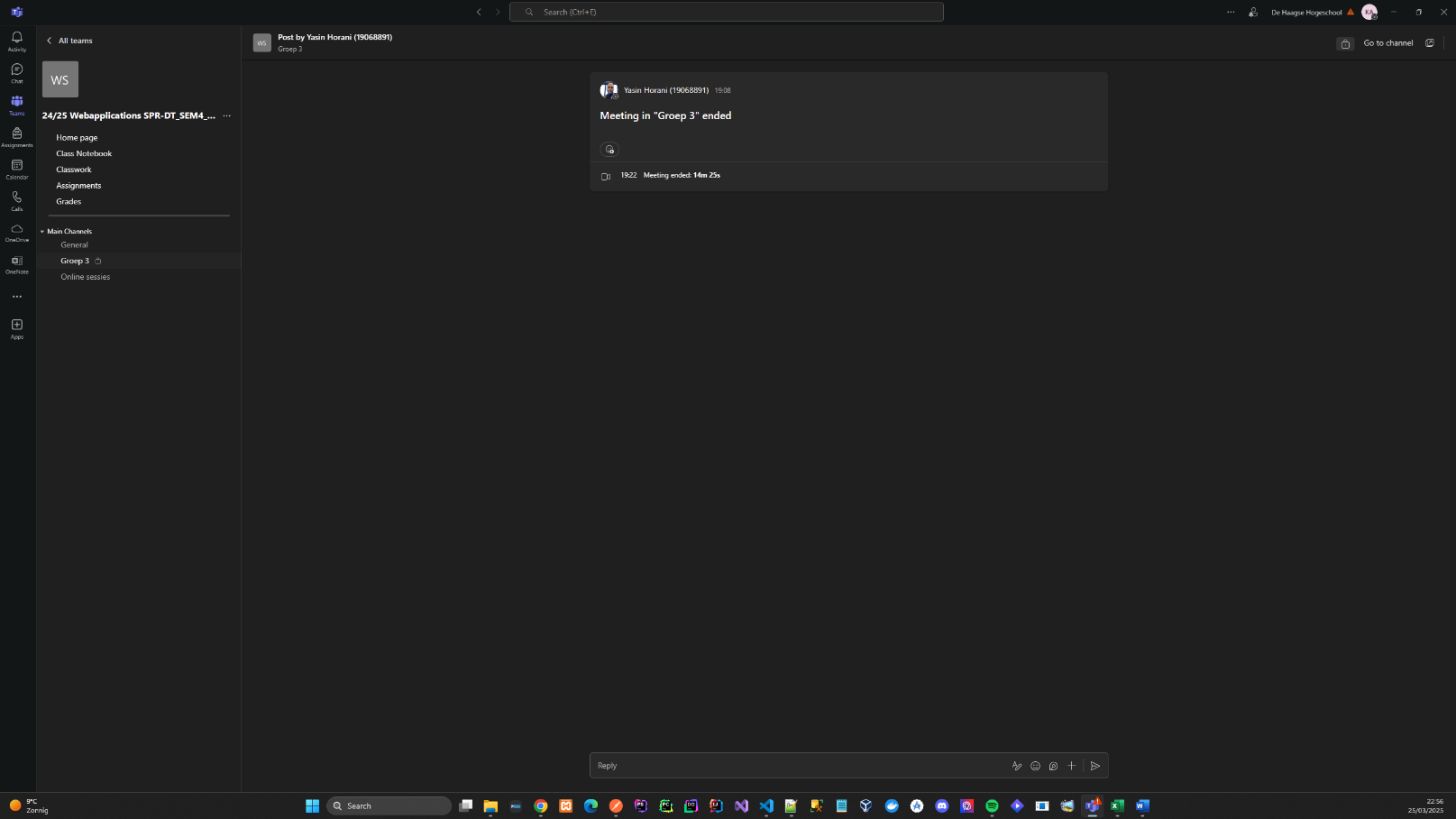
**De belangrijkste bevindingen waren:**

* Verbeteringen in de consistentie van asynchrone functies (async/await) in de CRUD-endpoints.
* Feedback op de implementatie van de Swagger-documentatie en de scheiding van de database‑logica.
* Aanpassingen om in-memory data volledig te vervangen door database-calls via Sequelize.

**Verwerkte Feedback:**

* Aanpassing van de volgorde van het laden van modules en de introductie van een seeder-script voor persistente data.
* Verbeterde error handling in de API-endpoints en consistent gebruik van async/await.
* Implementatie van duidelijke Swagger-annotaties in de code.

**Bewijs:**

Bewijs dat de Teams-call met mijn groep waarbij mijn code werd gereviewed om 10:08 uur is gestart en om 19:22 uur is beëindigd, met de datum rechtsonder weergegeven.

A screenshot of a computer

AI-generated content may be incorrect.Onderstaand vindt u een afbeelding als bewijs dat Swagger is geïntegreerd in mijn applicatie. Hieronder wordt de Swagger UI getoond.

# Testplan

**Teststrategie:**

* Alle endpoints zijn getest met Postman.

**Een lijst met kritische tests die ik hebt uitgevoerd in volgorden van belangrijk naar minder belangrijk:**

|  |  |  |  |
| --- | --- | --- | --- |
| Testnummer | Testdoel | Beschrijving | Resultaat  (Geslaagd/mislukt) |
| 1 | GET alle motoren | Verifieer dat een GET http://localhost:4000/motoren alle seeded records teruggeeft | Geslaagd |
| 2 | |  | | --- | | POST nieuwe motor |  |  | | --- | |  | | |  | | --- | | Verstuur een geldige JSON body via POST http://localhost:4000/motoren en controleer of het record wordt toegevoegd |  |  | | --- | |  | | Geslaagd |
| 3 | |  | | --- | | GET motor by ID |  |  | | --- | |  | | |  | | --- | | Doe GET http://localhost:4000/motoren/1 en bevestig dat de juiste motor wordt teruggegeven |  |  | | --- | |  | | Geslaagd |
| 4 | PUT update motor | |  | | --- | | Update een bestaand record via PUT http://localhost:4000/motoren/1 en controleer of velden correct zijn aangepast |  |  | | --- | |  | | Geslaagd |
| 5 | |  | | --- | | DELETE motor |  |  | | --- | |  | | |  | | --- | | Verwijder een record via DELETE http://localhost:4000/motoren/1 en verifieer dat het niet meer bestaat (GET → 404) |  |  | | --- | |  | | Geslaagd |
| 6 | |  | | --- | | GET invalid ID |  |  | | --- | |  | | |  | | --- | | Doe GET http://localhost:4000/motoren/999 (niet-bestaand ID) → verwacht status **404 Not Found** |  |  | | --- | |  | | Geslaagd |
| 7 | |  | | --- | | POST invalid body |  |  | | --- | |  | | |  | | --- | | Verstuur een lege JSON body naar POST /motoren → verwacht **400 Bad Request** of error response |  |  | | --- | |  | | Geslaagd |
| 8 | |  | | --- | | DELETE invalid ID |  |  | | --- | |  | | |  | | --- | | Doe DELETE http://localhost:4000/motoren/999 → verwacht **404 Not Found** |  |  | | --- | |  | | Geslaagd |

**Voorbeeld Test (POST /motoren):**

**Waarom is dit een goede test?**

Het verifiëert dat de API correct nieuwe resources creëert en opslaat in de database. Dit is de kernfunctionaliteit van een CRUD-API.

**Input Dataset:**

* Naam: Suzuki V-Strom
* Bouwjaar: 2022
* Merk: Suzuki
* Afbeelding: images/dummy.jpg
* Alt: Suzuki logo

**Waarom zijn deze gegevens representatief voor de test?**

Deze gegevens bevatten alle verplichte velden volgens het schema en simuleren een realistische toevoeging van een nieuw motor-object.

**Verwachte Uitkomst:**

* HTTP status 201 Created
* Een JSON-response met een nieuw gegenereerd ID en alle opgegeven data.
* Er wordt een nieuwe record opgeslagen voor de motor in de database.

**Resultaat**: De test is geslaagd. Zie bewijs op de volgende pagina.

**Bewijs:**

Onderstaand vindt u een afbeelding als bewijs dat ik de test daadwerkelijk heb uitgevoerd en succesvol heb afgerondA screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.Zoals hieronder weergegeven, is de motor succesvol toegevoegd als nieuw record in de database.

# Reflectie en Conclusies

**Behaalde Eisen:**

* De API ondersteunt alle CRUD-operaties conform de REST-principes.
* De API is gedocumenteerd met Swagger en getest met Postman.
* Persistentie is gerealiseerd via Sequelize en SQLite, waardoor de data behouden blijft na herstart van de server.
* De front‑end is succesvol gekoppeld aan de back‑end door fetch‑calls.

**Niet behaald:**

* Zover ik weet zijn alle minimale eisen behaald.

**Uitdagingen:**

* Aanvankelijk was het moeilijk om de juiste volgorde van asynchrone functies te bepalen.
* Het inladen en seeden van de mockdata vereiste extra documentatie en feedback.
* Feedback was nodig om consistentie te waarborgen tussen in-memory data en database-calls.

**Wat ik heb geleerd:**

* Het opzetten van een volledige RESTful API in Express met ondersteuning voor CRUD-operaties.
* Hoe ik Swagger kan integreren om mijn API te documenteren.
* Het belang van persistente opslag met behulp van een ORM, zodat data niet verloren gaat bij serverherstart.
* Het effectief toepassen van feedback en code review om een robuuste back‑end oplossing te bouwen.

**Effectiviteit van het stappenplan:**

Mijn stapsgewijze aanpak werkte heel goed om stapsgewijs complexiteit toe te voegen.

**Wat goed ging:** snelle validatie met JSON‑server; heldere API‑documentatie via Swagger; soepele migratie naar Sequelize.

**Wat ik anders zou doen:** de ORM-integratie eerder plannen en direct correcte statuscodes implementeren. Daardoor had ik minder refactor‑werk nodig.

# Bronnen

Coyier, C. (2024, 12 augustus). *CSS Flexbox Layout Guide | CSS-Tricks*. CSS-Tricks. <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

*DOM scripting introduction - Learn web development | MDN*. (2025, 19 maart). MDN Web Docs. <https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/DOM_scripting>

Scottie. (z.d.). *GitHub - scottie1984/swagger-ui-express: Adds middleware to your express app to serve the Swagger UI bound to your Swagger document. This acts as living documentation for your API hosted from within your app.* GitHub. <https://github.com/scottie1984/swagger-ui-express>

*Sequelize*. (z.d.). Feature-rich ORM For Modern TypeScript & JavaScript. https://sequelize.org/

Surnet. (z.d.). *GitHub - Surnet/swagger-jsdoc: Generates swagger/openapi specification based on jsDoc comments and YAML files.* GitHub. <https://github.com/Surnet/swagger-jsdoc>