VRIJE
UNIVERSITEIT
BRUSSEL

Master of Science in de Industriële Wetenschappen:
Elektronica-ICT - netwerken

# HABIT TRACKER

## Webapplicaties

Robbe Vlaeminck

12 januari 2025

INGENIEURSWETENSCHAPPEN

## Inhoudsopgave

# Inleiding

Voor het vak Webapplicaties kregen we de opdracht om een webapplicatie te ontwikkelen, namelijk een habit tracker. Deze applicatie moest zowel een functionele backend als een interactieve frontend omvatten en voldoen aan een reeks specificaties.

De backend moest relevante API-endpoints bevatten om gebruikers te registreren en in te loggen, projecten en taken aan te maken, taken te beheren (status aanpassen, toewijzen, etc.), en gebruikersrollen te hanteren (normale gebruiker en administrator). Data werd opgeslagen in een database en toegankelijk gemaakt via een RESTful API. Authenticatie en beveiliging waren essentiële onderdelen van de backend.

De frontend diende te worden ontwikkeld met React, met nadruk op het hergebruik van componenten. Gebruikers moesten zich kunnen registreren, inloggen, projecten en taken aanmaken, en overzicht behouden over toegewezen taken en projecten. Naast de gebruikelijke functionaliteiten moest de applicatie informatief en interactief zijn, met aandacht voor responsive design en een degelijke opmaak, bij voorkeur ondersteund door CSS-frameworks zoals Bootstrap.

Verder waren er enkele extra vereisten, zoals het gebruik van git tijdens het project, het deployen van de applicatie met Docker, API-documentatie genereren, en ondersteuning bieden voor markdown bij taken (via Quill of Draft.js). Grote vereisten waren onder meer dat de data persistent opgeslagen werd, gebruikers hun profiel konden aanpassen, en integratie met third-party toepassingen zoals social media of OAuth.

In de volgende hoofdstukken zullen we de algemene opzet van het project bespreken, gevolgd door een gedetailleerde beschrijving van elke pagina in de applicatie, inclusief de toegepaste functionaliteiten en features.

# Projectstructuur en werkwijze

Voor de ontwikkeling van deze webapplicatie is gewerkt met een duidelijke projectstructuur, verdeeld in drie hoofdonderdelen: `backend`, `frontend`, en `database`. Deze structuur zorgt voor een logische scheiding tussen de verschillende onderdelen van de applicatie.

Het volledige project is beheerd via GitHub en is beschikbaar op de volgende repository: https://github.com/Vlaeminksken/Project_webapplicaties.git.

Voor het deployen van de applicatie is gebruikgemaakt van Docker, ondersteund door een `docker-compose.yml`-bestand.

## Projectstructuur

De projectmap is als volgt opgebouwd:

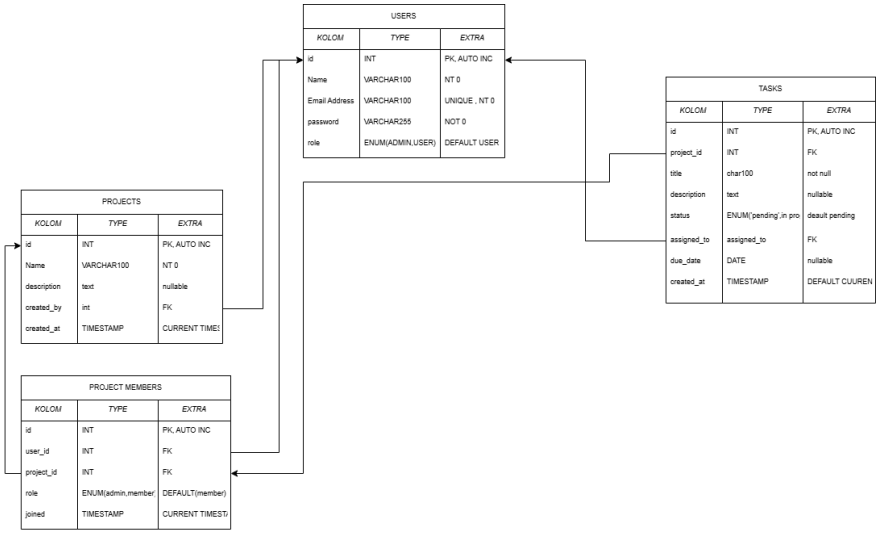
- **backend/**
  - `server.js`: Server die alle API's behandelt.
  - `node_modules/`: Bevat alle Node.js-afhankelijkheden.
  - `dockerfiles/`: Dockerconfiguratie voor de backend.
  - `package.json`: Backend-afhankelijkheden en scripts.
- **frontend/my-app/**
  - `dockerfiles/`: Dockerconfiguratie voor de frontend.
  - `npm_modules/`: Bevat alle Node.js-afhankelijkheden voor de frontend.
  - `src/`
    - `components/`: React-componenten voor alle pagina's.
    - `App.js`: Startpunt van de front-end.
  - `gitignore`: Uitsluitingen voor Git.
  - `package.json`: Frontend-afhankelijkheden en scripts.
- **database/**
  - `database.db`: Database schema (indien SQL gebruikt).
- `.gitignore`: Bestanden om uit Git te houden (zoals `node_modules`).
- `README.md`: Uitleg over het project.
- `docker-compose.yml`: Configuratiebestand voor Docker.

waarbij de:

- `backend/`: Bevat de serverlogica en API's, inclusief alle vereiste afhankelijkheden.
- `frontend/`: Hierin bevindt zich de volledige React-gebaseerde front-end, met herbruikbare componenten en de hoofdtoepassing.
- `database/`: Bevat het databaseschema en beheert de persistentie van de data.

## 1.1 Databaseontwerp

De database speelt een cruciale rol in de applicatie. Onderstaande ER-diagram (Figuur 1.1) toont het ontwerp van de database en de relaties tussen de tabellen.




**Figuur 1.1:** ER-diagram

Het ER-diagram bevat de volgende tabellen:

- **Users**: Beheert de gebruikersinformatie, zoals naam, e-mailadres, wachtwoord en rol (admin of gebruiker).
- **Projects**: Houdt projectinformatie bij, zoals naam, beschrijving, aanmaker, en aanmaakdatum.
- **Project Members**: Bevat de koppeling tussen gebruikers en projecten, inclusief hun rol binnen een project (bijvoorbeeld admin of lid).
- **Tasks**: Beheert de taken binnen projecten, inclusief titel, beschrijving, status, toegewezen persoon, en deadlines.

De relaties tussen de tabellen:

- Eén gebruiker kan meerdere projecten aanmaken (`created_by` in `Projects` verwijst naar `Users`).
- Gebruikers kunnen aan projecten gekoppeld worden via `Project Members`.
- Elk project kan meerdere taken hebben (`project_id` in `Tasks` verwijst naar `Projects`).
- Taken kunnen worden toegewezen aan specifieke gebruikers (`assigned_to` in `Tasks` verwijst naar `Users`).

## 2.1 Inlogpagina

De inlogpagina maakt gebruik van React en biedt een eenvoudige interface voor gebruikers om in te loggen. De functionaliteit wordt beheerd via de `Logger.js`-component. Deze component communiceert met de backend via een POST-aanroep naar het `/login`-endpoint, zoals gedefinieerd in de backend (`server.js`).

### Gebruik van de code

De volgende functies en elementen worden gebruikt in de code van de inlogpagina:

- `useState`: Voor het bijhouden van de gebruikersnaam en het wachtwoord.
- `fetch API`: Voor het verzenden van inloggegevens naar de backend.
- Opslag van de `JWT-token` in `localStorage` voor beveiligde toegang tot verdere functionaliteiten.

Een belangrijk fragment uit de code:

```
const handleLogin = async (e) => {
    e.preventDefault();
    const response = await fetch('http://localhost:5000/login', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ name: username, password }),
    });
    const data = await response.json();
    if (response.ok) {
        localStorage.setItem('token', data.token);
        navigate('/home');
    }
};
```

### Lay-out en functionaliteiten

De pagina bevat een overzichtelijke lay-out, met een loginformulier dat bestaat uit invoervelden voor gebruikersnaam en wachtwoord. Daarnaast is er een link naar de registratiepagina voor nieuwe gebruikers. Een voorbeeld van de styling is opgenomen in een aparte CSS-bestand (`Form.css`), waarmee consistentie in ontwerp wordt gegarandeerd. De pagina toont foutmeldingen en successen op basis van serverreacties.

### Features

De inlogpagina houdt rekening met beveiliging door gebruik te maken van JWT-tokens voor authenticatie.

## 2.2 Registratiepagina

De registratiepagina maakt eveneens gebruik van React en biedt een interface voor nieuwe gebruikers om een account aan te maken. De functionaliteit wordt verzorgd door de `Register.js`-component. Net als bij de inlogpagina wordt er een POST-aanroep uitgevoerd, maar deze keer naar het `/register`-endpoint in de backend.

### Gebruik van de code

De volgende functies en elementen worden gebruikt in de code van de registratiepagina:

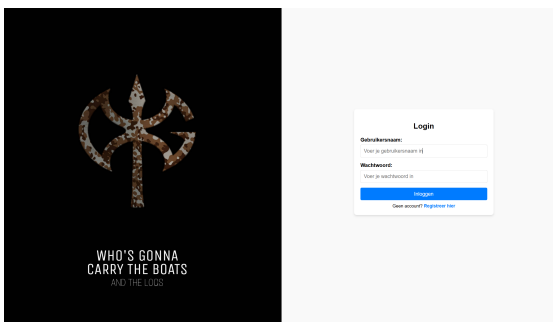
- `useState`: Voor het beheren van gebruikersnaam, e-mailadres, wachtwoord, en wachtwoordbevestiging.
- Validatie: Controleert of het wachtwoord en de bevestiging overeenkomen voordat het verzoek naar de server wordt verzonden.
- Gebruik van `fetch API`: Voor communicatie met de backend.

Een belangrijk fragment uit de code:

```
const handleRegister = async (e) => {
    e.preventDefault();
    if (password !== confirmPassword) {
        setNotification('Wachtwoorden komen niet overeen.');
        return;
    }
    const response = await fetch('http://localhost:5000/register', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ name: username, email, password }),
    });
    const data = await response.json();
    if (response.ok) {
        navigate('/');
    }
};
```

### Lay-out en functionaliteiten

De registratiepagina biedt een formulier met velden voor gebruikersnaam, e-mailadres, wachtwoord, en wachtwoordbevestiging. Daarnaast is er een link naar de inlogpagina. Net als bij de inlogpagina wordt de styling verzorgd door `Form.css`.

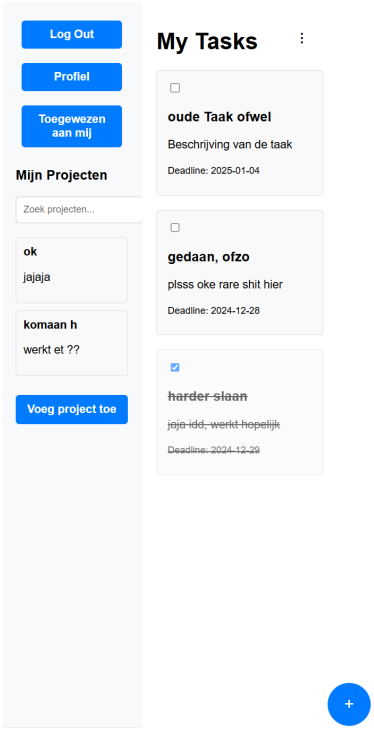



**Figuur 2.1:** inlogscherm

## 3.1 Lay-out

Het hoofdscherm van de applicatie biedt een overzichtelijke interface die is opgedeeld in drie secties:

- **Linkerzijbalk:** Deze bevat een lijst van alle projecten waaraan de gebruiker deelneemt, inclusief zoekfunctionaliteit en knoppen om projecten toe te voegen of te verwijderen. Gebruikers kunnen projecten selecteren om de bijbehorende taken weer te geven.
- **Middelste gedeelte:** Hier worden alle taken weergegeven die aan de gebruiker zijn toegewezen of bij het geselecteerde project horen. Taken kunnen bewerkt, verwijderd of als voltooid gemarkeerd worden. Daarnaast is er een knop in de vorm van een plus-teken waarmee nieuwe taken kunnen worden toegevoegd.
- **Dropdown-menu-optie van de projecten:** Een dropdown-menu biedt extra functionaliteiten zoals het bewerken of verwijderen van projecten en het toevoegen van personen aan projecten.




**Figuur 3.1:** Home pagina

De opmaak is responsief en is ontworpen om goed te werken op zowel desktop- als mobiele apparaten. De styling wordt verzorgd door CSS en houdt rekening met gebruiksvriendelijkheid en consistentie.

## 3.2 Features

Het hoofdscherm biedt de volgende functionaliteiten:

- **Lijst van projecten:** Gebruikers kunnen projecten zoeken, bekijken, toevoegen en verwijderen.
- **Weergave van taken:** Alle taken binnen een geselecteerd project worden weergegeven, inclusief de status, beschrijving en deadline.
- **Taakbeheer:** Gebruikers kunnen:
  - De status van taken aanpassen (bijvoorbeeld van ïn behandeling"naar "voltooid").
  - Taken verwijderen.
- **Dropdown-menu:** Dit biedt toegang tot functies zoals het bewerken of verwijderen van projecten en het toevoegen van personen aan projecten.
- **Knop voor je profiel te zien:** Als je deze knop klikt dan kom je naar een pagina waar je de info van je profiel kunt zien en aanpassen.
- **Knop voor je opdrachten die aan jezelf zijn toegewezen te zien:** Als je deze knop klikt dan kom je naar een pagina waar je alle taken kunt zien die tot jezelf zijn toegewezen.
- **Knop voor nieuwe taken:** Een duidelijk zichtbare plus-knop stelt gebruikers in staat om snel nieuwe taken toe te voegen.
- **Uitloggen:** Gebruikers kunnen uitloggen, waarbij hun authenticatietoken wordt verwijderd.
- **filter:** er is een filter voor te zoeken op project.
- **knop filter:** Als je duwt op 1 van de project dan wordt er op het scherm enkel de taken getoond van dat project.
- **taken bewerken:** als je op de taken duwt dan komt er een extra scherm waarbij je de eigenschappen van de taak kan aanpassen.

Hieronder volgt een code voor het weergeven van de lijst met projecten en taken:

```
useEffect(() => {
    fetch('http://localhost:5000/projects', {
        method: 'GET',
        headers: { Authorization: localStorage.getItem('token') },
    })
    .then((response) => response.json())
    .then((data) => setProjects(data))
    .catch(() => setProjects([]));

    fetch('http://localhost:5000/tasks', {
        method: 'GET',
        headers: { Authorization: localStorage.getItem('token') },
    })
    .then((response) => response.json())
    .then((data) => setTasks(data));
}, []);
```

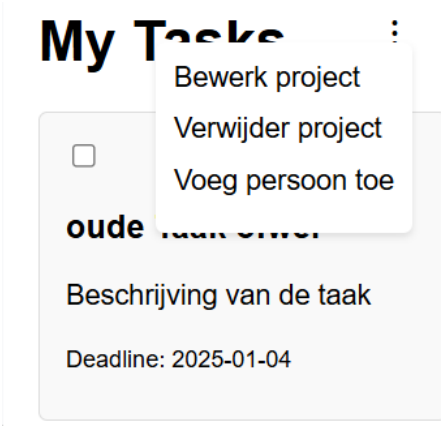
## 3.3 DROPDOWN

Wanneer je op de 3 bolletjes-teken anklikt voor de dropwdown menu, krijg je de volgende 3 mogelijkheden:

- bewerk project
- verwijder project
- voeg persoon toe

Dit zijn de 3 mogelijkheden wat je met elk project kunt doen. Eerst en vooral moet je wel een project aanduiden anders werkt deze dropdown menu niet en geeft hij u een melding.

Bij de keuzes, voeg persoon toe en bewerk project, word je herleidt naar een andere pagina waar je de respectievelijke bewerkingen kunt doen.



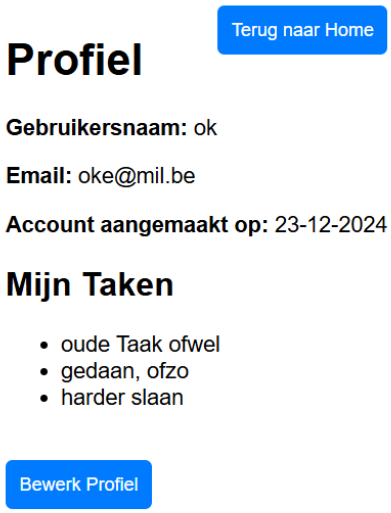

**Figuur 3.2:** Dropdown

# Profielpagina

De profielpagina biedt gebruikers de mogelijkheid om persoonlijke gegevens te bekijken en te beheren. Daarnaast worden alle taken weergegeven die aan de gebruiker zijn toegewezen.

## 4.1 Functionaliteiten

- **Weergave van gebruikersinformatie:** Op de profielpagina kunnen gebruikers hun gebruikersnaam, e-mailadres en aanmaakdatum van hun account zien.
- **Lijst van taken:** Alle taken die aan de gebruiker zijn toegewezen worden in een lijst weergegeven.
- **Profiel bewerken:** Door op de knop "Bewerk Profiel"te klikken, kunnen gebruikers hun gegevens aanpassen, zoals hun naam en e-mailadres.




**Figuur 4.1:** Profielpagina

## 4.2 Code

De profielpagina maakt gebruik van de volgende codefragmenten:

### Backend: ophalen van profielinformatie

In de backend is het `/profile`-endpoint verantwoordelijk voor het ophalen van de gebruikersinformatie en taken. Hieronder een voorbeeld van hoe de gegevens worden opgehaald:

```
// Backend endpoint (server.js)
app.get('/profile', authenticateToken, (req, res) => {
    const query = 'SELECT name, email, created_at FROM USERS WHERE id = ?';

    db.get(query, [req.user.id], (err, row) => {
        if (err) {
            console.error(err.message);
            return res.status(500).json({ message: 'Fout bij het ophalen van profielgegevens!' });
        }

        if (row) {
            const queryTasks = 'SELECT title FROM TASKS WHERE assigned_to = ?';
            db.all(queryTasks, [req.user.id], (err, tasks) => {
                if (err) {
                    console.error(err.message);
                    return res.status(500).json({ message: 'Fout bij het ophalen van taken!' });
                }

                res.status(200).json({ user: row, tasks: tasks.map((task) => task.title) });
            });
        } else {
            res.status(404).json({ message: 'Gebruiker niet gevonden!' });
        }
    });
});
```

### Frontend: weergeven van profielgegevens en taken

De `Profile.js`-component haalt de profielinformatie en taken op bij het laden van de pagina en toont deze aan de gebruiker:

```
useEffect(() => {
    const token = localStorage.getItem('token');
    if (!token) {
        navigate('/');
        return;
    }

    fetch('http://localhost:5000/profile', {
        method: 'GET',
        headers: { Authorization: token },
    })
        .then((response) => response.json())
        .then((data) => {
            setProfile(data.user);
            setTasks(data.tasks);
        })
        .catch(() => {
            navigate('/');
        });
}, [navigate]);

```

```
22 if (!profile) {
23     return <h2>Profiel laden...</h2>;
24 }
```

## 4.3 Gebruikservaring

Wanneer gebruikers de profielpagina openen, zien zij een overzicht van hun persoonlijke gegevens. Taken worden weergegeven in een overzichtelijke lijst. Door op "Bewerk Profiel"te klikken, worden gebruikers doorgestuurd naar een pagina waar zij hun gegevens kunnen aanpassen.

# Toegewezen aan mij

De pagina "Toegewezen aan mij"geeft gebruikers een overzicht van projecten en taken waaraan ze zijn toegewezen. Deze pagina maakt gebruik van een overzichtelijke lay-out en biedt verschillende functionaliteiten om taken en projecten te beheren.

## 5.1 Functionaliteiten

- **Lijst van projecten:** Een overzicht van alle projecten waaraan de gebruiker is toegewezen, met hun naam en beschrijving. Gebruikers kunnen projecten selecteren om de bijbehorende taken weer te geven.
- **Lijst van taken:** Toont alle taken die bij het geselecteerde project horen. Gebruikers kunnen details bekijken, zoals titel, beschrijving, status en deadline.
- **Bewerkoptie:** Gebruikers met de juiste rol (bijvoorbeeld öperator") kunnen taken bewerken door op een knop te klikken.
- **Navigatie:** De gebruiker kan eenvoudig terugkeren naar de hoofdpagina.

## 5.2 Code

De pagina "Toegewezen aan mij"maakt gebruik van de volgende codefragmenten:

### Backend: ophalen van toegewezen projecten en taken

De backend biedt endpoints voor het ophalen van projecten en taken die aan de gebruiker zijn toegewezen. Een voorbeeld van het `/assigned-projects`-endpoint:

```
// Backend endpoint (server.js)
app.get('/assigned-projects', authenticateToken, (req, res) => {
    const userId = req.user.id;

    const query = `
        SELECT p.id, p.name, p.description
        FROM PROJECTS p
        INNER JOIN PROJECT_MEMBERS pm ON p.id = pm.project_id
        WHERE pm.user_id = ?
    `;

    db.all(query, [userId], (err, rows) => {
        if (err) {
            console.error('Fout bij het ophalen van toegewezen projecten:', err.
    message);
            return res.status(500).json({ message: 'Fout bij het ophalen van
    projecten.' });
        }

        res.status(200).json(rows || []);
    });
});
```

### Frontend: weergave van projecten en taken

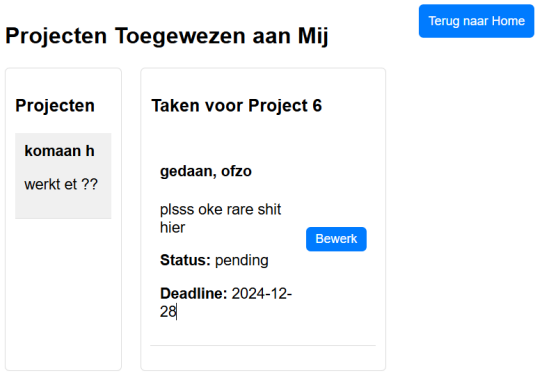
De `AssignedToMe.js`-component haalt projecten en taken op bij het laden van de pagina. Hier is een voorbeeld van hoe projecten worden opgehaald:

```
useEffect(() => {
    const token = localStorage.getItem('token');
    if (!token) {
        navigate('/');
        return;
    }

    fetch('http://localhost:5000/assigned-projects', {
        method: 'GET',
        headers: { Authorization: token },
    })
        .then((response) => {
            if (!response.ok) {
                throw new Error('Fout bij het ophalen van toegewezen projecten.'
    );
            }
            return response.json();
        })
        .then((data) => setProjects(data))
        .catch((err) => {
            console.error(err.message);
            alert('Fout bij het ophalen van toegewezen projecten.');
        });
}, [navigate]);
```

## 5.3 Gebruikservaring

De pagina toont eerst een lijst van alle projecten. Wanneer een project wordt geselecteerd, worden de bijbehorende taken weergegeven in een gescheiden sectie. Taken kunnen worden bewerkt als de gebruiker de juiste rol heeft. Dit zorgt voor een duidelijke workflow en efficiënt taakbeheer.Indien de gebruiker je geen toegang gegeven heeft dan kan je deze dus niet aanpassen.




**Figuur 5.1:** Toegewezen-aan-mij-pagina

## 6.1 Wat is API-documentatie?

API-documentatie biedt een overzicht van de beschikbare endpoints van een applicatie, inclusief hun functionaliteit, vereiste parameters, en verwachte antwoorden. Het dient als een handleiding voor ontwikkelaars om eenvoudig met de API te kunnen werken.

In dit project is er een API-documentatie gegenereerd om duidelijk te maken hoe externe systemen en gebruikers kunnen communiceren met de backend. De documentatie is opgesteld met behulp van `Swagger`, een populaire tool voor het beschrijven en testen van RESTful API's.

## 6.2 Waar kan de documentatie worden gevonden?

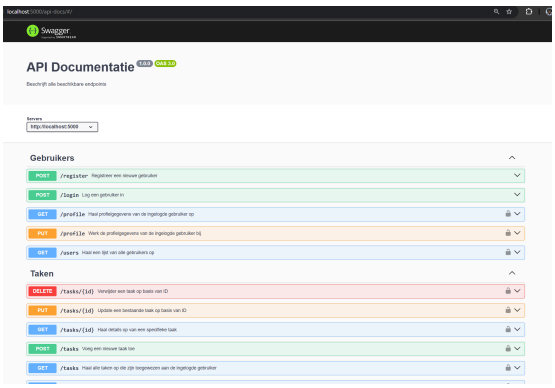
De API-documentatie is geïntegreerd in de applicatie en kan worden benaderd via de volgende URL:

- `http://localhost:5000/api-docs`

Wanneer deze URL in een browser wordt geopend, wordt een interactieve interface weergegeven waarin alle beschikbare API-endpoints worden gedocumenteerd. Deze interface stelt gebruikers in staat om API-aanroepen direct vanuit de browser te testen.

## 6.3 Hoe is de documentatie gemaakt?

De documentatie is gemaakt met behulp van `swagger-jsdoc` en `swagger-ui-express`, beide krachtige tools voor het genereren van documentatie vanuit de code. Het proces omvat het toevoegen van `Swagger`-annotaties in de backend-code en het instellen van een Swagger-configuratie in `server.js`.




**Figuur 6.1:** Swagger pagina

Hier is een fragment van de configuratie in de `server.js`:

```
// Import Swagger bibliotheken
const swaggerJsDoc = require('swagger-jsdoc');
const swaggerUi = require('swagger-ui-express');

// Configuratie van Swagger
const swaggerOptions = {
    swaggerDefinition: {
        openapi: '3.0.0',
        info: {
            title: 'API Documentatie',
            version: '1.0.0',
            description: 'Beschrijft alle beschikbare endpoints',
        },
        servers: [{ url: 'http://localhost:5000' }],
    },
    apis: ['./server.js'], // Geef het pad naar je bestand
};

// Genereer Swagger-documentatie
const swaggerDocs = swaggerJsDoc(swaggerOptions);
app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerDocs));
```

## 6.4 Wat kun je zien in de documentatie?

De documentatie biedt een gedetailleerd overzicht van de volgende elementen:

- **Beschikbare endpoints:** Elk endpoint wordt weergegeven met een korte beschrijving van de functionaliteit.
- **HTTP-methoden:** Zoals `GET`, `POST`, `PUT`, en `DELETE`.
- **Parameters:** Details over vereiste en optionele parameters, inclusief datatypen.
- **Voorbeelden van verzoeken en antwoorden:** Inclusief voorbeeldpayloads en voorbeeldreacties.
- **Beveiliging:** Informatie over vereiste authenticatie, zoals het gebruik van een JWT-token.

Een voorbeeld van een gedocumenteerd endpoint voor het registreren van een gebruiker:

```
/**
 * @swagger
 * /register:
 *   post:
 *     summary: Registreer een nieuwe gebruiker
 *     tags: [Gebruikers]
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             type: object
 *             properties:
 *               name:
 *                 type: string
 *                 description: Naam van de gebruiker
 *               email:
 *                 type: string
 *                 description: E-mailadres van de gebruiker
 *               password:
 *                 type: string
 *                 description: Wachtwoord van de gebruiker
 *     responses:
 *       200:
 *         description: Gebruiker succesvol geregistreerd
 *       400:
 *         description: Validatiefout (bijvoorbeeld ontbrekende velden of al
    bestaande email)
 *       500:
 *         description: Interne serverfout
 */
```

## 6.5 Doel van de API-documentatie

Het doel van de API-documentatie is om:

- **Gemak:** Ontwikkelaars eenvoudig toegang te bieden tot informatie over beschikbare endpoints.
- **Testen:** API-aanroepen direct in de browser te kunnen testen.
- **Consistentie:** Een consistente bron van waarheid te bieden voor de implementatie van de API.
- **Samenwerking:** Het ontwikkelingsproces te vergemakkelijken door duidelijke communicatie tussen backend- en frontend-teams.

## 6.6 Conclusie

De API-documentatie speelt een essentiële rol in het begrijpelijk maken van de backend-functionaliteiten. Het biedt ontwikkelaars een krachtig hulpmiddel om efficiënt en foutloos met de API te integreren. In toekomstige uitbreidingen kan de documentatie worden uitgebreid met meer details over nieuwe endpoints en geavanceerde functionaliteiten.