|  |  |
| --- | --- |
| Opleidingsnaam | Informatica |
| Onderwijsblok | Blok 6 |
| Beroepsproductnaam | BP06 |
|  | |
| Studentnaam | Dominic Mol |
| Studentnummer | 2148084 |
|  | |
| Deelproductnaam  Deelproducten Blok 6 –  Klimaat Applicatie | |

Afbeelding met tekst, scherm

Door AI gegenereerde inhoud is mogelijk onjuist.

2025

Inhoud

[Voorwoord 3](#_Toc212378338)

[Inleiding 4](#_Toc212378339)

[2a.a Schermontwerp Front-end 5](#_Toc212378340)

[Afbeelding 1 Schermontwerp room overview, door D. Mol: 5](#_Toc212378341)

[Afbeelding 2 Schermontwerp create a room, door D. Mol: 6](#_Toc212378342)

[Afbeelding 3 Schermontwerp voorbeeld room page, door D. Mol: 7](#_Toc212378343)

[2e.1 Requirementsdocument 8](#_Toc212378344)

[Knelpuntanalyse 8](#_Toc212378345)

[Requirements Basis 9](#_Toc212378346)

[1: Identificatie: 9](#_Toc212378347)

[2: Kwalificatie: 10](#_Toc212378348)

[**3: Kwalificatie: Statische en Dynamische Requirements per tabel:** 10](#_Toc212378349)

[**4: Functionele requirements Klimaatapp** 14](#_Toc212378350)

[**5: Niet-functionele requirements** 17](#_Toc212378351)

[Aanvullende informatie niet functionele requirements 17](#_Toc212378352)

[Kwaliteitsrequirements 19](#_Toc212378353)

[**6: MoSCoW-overzicht met user stories en acceptatiecriteria** 20](#_Toc212378354)

[**7: Acceptatierequirements:** 23](#_Toc212378355)

[**8: Waardepropositie (Statement of Needs)** 25](#_Toc212378356)

[**9: Oplosstrategie (Qualification Strategy of Needs)** 26](#_Toc212378357)

[**11: CEM (Conceptual Enterprise Model)** 28](#_Toc212378358)

[**12: ERD (Entiteiten Relatie Diagram)** 29](#_Toc212378359)

[**13: UML Classdiagram (Klassendiagram)** 30](#_Toc212378360)

[**14: UML Usecasediagram** 31](#_Toc212378361)

[**15: UML Activitydiagram** 32](#_Toc212378362)

[16: Samenvatting Requirementsdocument 33](#_Toc212378363)

[4d.e Kwaliteitsontwerp Front-end 34](#_Toc212378364)

[Inleiding 34](#_Toc212378365)

[Gebruiksvriendelijkheid (Usability) 34](#_Toc212378366)

[Toegankelijkheid en devices 35](#_Toc212378367)

[Prestaties 36](#_Toc212378368)

[Betrouwbaarheid en beschikbaarheid 36](#_Toc212378369)

[Beveiliging en privacy 36](#_Toc212378370)

[Onderhoudbaarheid en uitbreidbaarheid 36](#_Toc212378371)

[Conclusie 37](#_Toc212378372)

[5a.2 Usabilitytest 38](#_Toc212378373)

[Inleiding 38](#_Toc212378374)

[Testopzet 38](#_Toc212378375)

[Resultaten en conclusie 40](#_Toc212378376)

[7d.1 – zelfstandig werken 41](#_Toc212378377)

[Inleiding 41](#_Toc212378378)

[Feedback en zelfreflectie 41](#_Toc212378379)

[Visuele presentatie en structuur 42](#_Toc212378380)

[Kritisch denken en beslissingen 42](#_Toc212378381)

[7d.2 – VerEvaRef 44](#_Toc212378382)

[Evalueren 44](#_Toc212378383)

[Verantwoorden 50](#_Toc212378384)

[Reflecteren 52](#_Toc212378385)

[Bibliografie 55](#_Toc212378386)

# Voorwoord

Tijdens dit project heb ik met veel plezier gewerkt aan het ontwikkelen van de Klimaatapp. Het was interessant om te zien hoe data van echte sensoren kan worden omgezet naar iets dat direct inzicht geeft in het binnenklimaat.

Ik wil graag mijn docenten bedanken voor hun hulp, feedback en tijd tijdens dit proces. Hun begeleiding heeft geholpen om dit project tot een goed einde te brengen.

Het was een leerzame opdracht waarin techniek, ontwerp en logica mooi samenkwamen. Ik heb er veel van geleerd en vond het leuk om alles uit te werken tot een werkend resultaat.

Veel leesplezier met het verslag!

— *Dominic Mol*

# Inleiding

Een goed binnenklimaat is belangrijk voor comfort, concentratie en gezondheid. Temperatuur, luchtvochtigheid en CO₂ spelen daarin een grote rol. Om deze waarden te laten zien, is een Klimaatapp ontwikkeld die live meetgegevens per kamer laat zien.

De app ontvangt data van klimaatsensoren die zijn gekoppeld aan *The Things Network (TTN)*. Via een eenvoudige verbinding met een webhook worden de meetgegevens doorgestuurd naar een webserver, opgeslagen in een cloud-database en vervolgens zichtbaar gemaakt op een overzichtelijk dashboard.

Het project is onderdeel van Blok 6 Frontend Development en het Basisprofiel. De focus ligt op het ontwerpen en bouwen van een duidelijke en betrouwbare webapplicatie die gebruiksvriendelijk is en goed werkt met actuele sensordata.

In dit verslag wordt stap voor stap uitgelegd hoe de applicatie is ontworpen, gebouwd en getest, en hoe de keuzes in techniek en ontwerp hebben bijgedragen aan het eindresultaat.

# 2a.a Schermontwerp Front-end

## Afbeelding 1 Schermontwerp room overview, door D. Mol:

Afbeelding met tekst, schermopname, nummer, software

Door AI gegenereerde inhoud is mogelijk onjuist.Dit scherm laat een overzicht zien van alle kamers die in het systeem zijn toegevoegd.  
Elke kamer heeft een eigen kaart waarop de status en de laatste update zichtbaar zijn.  
Op die manier is in één oogopslag te zien welke ruimtes actief zijn en welke niet.  
De interface is bewust rustig gehouden, met voldoende witruimte en herkenbare iconen.  
Links staat een eenvoudig menu waarmee snel tussen kamers kan worden gewisseld of een nieuwe kamer kan worden toegevoegd.  
Het doel was om het scherm zo overzichtelijk mogelijk te maken.

## Afbeelding 2 Schermontwerp create a room, door D. Mol:

Wanneer de gebruiker een nieuwe kamer wil toevoegen, verschijnt er een klein pop-upvenster in plaats van een apart scherm.  
Hiermee blijft de gebruiker in dezelfde omgeving en raakt die niet de context kwijt.  
De invoer is simpel gehouden: alleen de naam van de kamer hoeft ingevuld te worden.  
Daaronder staan twee duidelijke knoppen om op te slaan of te annuleren.  
Deze keuze maakt het proces snel en eenvoudig.

Afbeelding met tekst, schermopname, software, Multimediasoftware

Door AI gegenereerde inhoud is mogelijk onjuist.

## Afbeelding 3 Schermontwerp voorbeeld room page, door D. Mol:

Hier zullen meerdere pagina’s voor worden aangemaakt. Dus voor elke kamer één, maar deze gaan er allemaal hetzelfde uitzien.

In dit scherm wordt de actuele klimaatdata van één kamer weergegeven.  
De belangrijkste gegevens: temperatuur, luchtvochtigheid, lichtsterkte en aanwezigheid – zijn elk weergegeven in een eigen kaart.  
Deze kaarten zorgen voor overzicht en maken de data makkelijk te lezen.  
Bovenaan staat het tijdstip van de laatste meting, zodat de gebruiker direct ziet hoe recent de data is. De opzet is rustig en strak gehouden, zodat de aandacht volledig naar de waarden gaat.

Afbeelding met tekst, schermopname, software

Door AI gegenereerde inhoud is mogelijk onjuist.

# 2e.1 Requirementsdocument

In dit document worden alle eisen en wensen beschreven die horen bij de ontwikkeling van de Klimaatapp. Het doel van dit document is om duidelijk te maken wat de applicatie precies moet kunnen, hoe deze zich gedraagt en aan welke voorwaarden het systeem moet voldoen.

De Klimaatapp is een eenvoudige, maar praktische toepassing waarmee klimaatgegevens per kamer zichtbaar worden gemaakt. Door deze gegevens overzichtelijk te tonen, kunnen gebruikers snel zien hoe het binnenklimaat in verschillende ruimtes is zoals temperatuur, luchtvochtigheid, CO2, licht, aanwezigheid en geluid.

Het requirementsdocument vormt de basis voor het verdere ontwerp en de realisatie van de Klimaatapp. Het helpt om de ontwikkeling gestructureerd aan te pakken en ervoor te zorgen dat alle onderdelen logisch op elkaar aansluiten.

## Knelpuntanalyse

Net als elke digitale toepassing kan ook deze klimaatapp verschillende uitdagingen tegenkomen. Om verrassingen te voorkomen, zijn hieronder de belangrijkste knelpunten kort beschreven, telkens met een eenvoudige oplossing.

**Vertraging in data-overdracht**

Probleem: Meetgegevens worden niet altijd onmiddellijk doorgestuurd via het LoRa-netwerk. Daardoor kan de applicatie verouderde informatie tonen.

Oplossing: De app toont een tijdstempel bij elke waarde en kleurt de kaarten grijs als de data ouder is dan dertig minuten. Zo ziet de gebruiker meteen wanneer de laatste update is geweest.

**Device offline of niet aanwezig**

Probleem: Als een device is uitgeschakeld of nog niet aan een kamer is gekoppeld, worden er geen waarden weergegeven.

Oplossing: Er verschijnt een duidelijke melding bij kamers zonder actief device. Dit voorkomt verwarring en maakt het duidelijk dat er eerst een device gekoppeld moet worden.

**Onrealistische meetwaarden**

Probleem: Soms worden extreme of foutieve waarden doorgestuurd, bijvoorbeeld een CO2-waarde boven 5000 ppm.

Oplossing: De backend controleert de data op logische grenzen. CO2-waarden boven 5000 ppm worden als sensorfouten beschouwd en gefilterd uit de grafieken.

**Verplaatsing van het device**

Probleem: Wanneer een device wordt verplaatst naar een andere ruimte, moet de app weten in welke kamer hij nu staat. Anders kunnen de gegevens verkeerd toegewezen worden.

Oplossing: In de beheerinterface kan het device met één klik aan een andere kamer worden gekoppeld of ontkoppeld. Na het verplaatsen verschijnen de waarden automatisch op de juiste pagina.

**Onjuiste of dubbele kamernamen**

Probleem: Bij het aanmaken van kamers kan een naam per ongeluk tweemaal worden gebruikt of leeg worden gelaten.

Oplossing: De applicatie controleert of een naam al bestaat en of het veld niet leeg is. Bij een fout toont de app een korte melding en kan de gebruiker een andere naam kiezen.

**Onduidelijke knoppen en labels**

Probleem: Te technische termen of onduidelijke knoppen kunnen verwarrend zijn voor gebruikers die snel iets willen zien.

Oplossing: Alle knoppen, labels en meldingen zijn in heldere taal geschreven. Iconen en kleuren helpen om acties en statussen snel te herkennen.

## Requirements Basis

**1: Identificatie:**

**Tabel 1: rooms**

Binnen de verzameling van alle room\_id identificeert room\_id <room\_id> een specifieke room.

Binnen de verzameling van alle name identificeert name <n> een specifieke kamernaam.

Binnen de verzameling van alle created\_at identificeert created\_at <created\_at> een specifiek aanmaakmoment.

**Tabel 2: devices**

Binnen de verzameling van alle dev\_eui identificeert dev\_eui <dev\_eui> een specifiek device.

Binnen de verzameling van alle name identificeert name <n> een specifieke devicenaam.

Binnen de verzameling van alle room\_id identificeert room\_id <room\_id> een specifieke gekoppelde kamer.

Binnen de verzameling van alle last\_seen\_at identificeert last\_seen\_at <last\_seen\_at> een specifiek laatst-gezien-moment.

**Tabel 3: sensors**

Binnen de verzameling van alle dev\_eui identificeert dev\_eui <dev\_eui> een specifiek device.

Binnen de verzameling van alle channel identificeert channel <channel> een specifiek kanaalnummer.

Binnen de verzameling van alle type identificeert type <type> een specifiek sensortype.

Binnen de verzameling van alle unit identificeert unit <unit> een specifieke meeteenheid.

**Tabel 4: measurements**

Binnen de verzameling van alle id identificeert id <id> een specifieke meting.

Binnen de verzameling van alle dev\_eui identificeert dev\_eui <dev\_eui> een specifiek device.

Binnen de verzameling van alle channel identificeert channel <channel> een specifiek kanaalnummer.

Binnen de verzameling van alle value identificeert value <value> een specifieke meetwaarde.

Binnen de verzameling van alle measured\_at identificeert measured\_at <measured\_at> een specifiek meettijdstip.

**2: Kwalificatie:**

**rooms**

Kamer <room\_id> heeft de naam <n> en is aangemaakt op <created\_at>.

**devices**

Device <dev\_eui> heeft de naam <n>, is gekoppeld aan kamer <room\_id> en werd voor het laatst gezien op <last\_seen\_at>.

**sensors**

Sensor van device <dev\_eui> op kanaal <channel> meet het type <type> in eenheid <unit>.

**measurements**

Meting <id> van device <dev\_eui> op kanaal <channel> heeft waarde <value> gemeten op <measured\_at>.

## 3: Kwalificatie: Statische en Dynamische Requirements per tabel:

In dit onderdeel wordt gekeken naar de eigenschappen en regels die bepalen hoe de gegevens in de applicatie worden opgeslagen en hoe ze zich gedragen tijdens het gebruik.

De statische requirements beschrijven welke gegevens in de database aanwezig zijn en hoe deze met elkaar in verband staan. Denk hierbij aan vaste onderdelen zoals kamers, devices, sensoren en metingen.

De dynamische requirements gaan over het gedrag van de applicatie: wat er gebeurt wanneer een gebruiker een kamer toevoegt, een device koppelt of wanneer er nieuwe meetgegevens binnenkomen.

Door deze twee onderdelen te combineren ontstaat een duidelijk beeld van zowel de structuur als de werking van de Klimaatapp.

**rooms**

|  |  |
| --- | --- |
| **SBVR Static rooms (1)** |  |
| **Kwalificatie** | 1. Kamer <room\_id> heeft de naam <n> en is aangemaakt op <created\_at>. |
| **Deel** | N.V.T. |
| **Hoofd** | room\_id is een hoofdverzameling van room\_id. name is een hoofdverzameling van name. created\_at is een hoofdverzameling van created\_at. |
| **Niet Leeg** | De invoer van room\_id mag niet leeg zijn. De invoer van name mag niet leeg zijn. De invoer van created\_at mag niet leeg zijn. |
| **Formaat** | Het formaat van room\_id is int (auto-increment). Het formaat van name is string met een maximum van 100 tekens. Het formaat van created\_at is datetime met standaardwaarde CURRENT\_TIMESTAMP. |
| **Uniciteit** | Elke invoer in rooms is met exact één room\_id geïdentificeerd. Elke invoer van name moet uniek zijn binnen de tabel rooms. |
| **SBVR Dynamic rooms (1)** |  |
| **Create** | Als de gebruiker een nieuwe kamer aanmaakt met een unieke name, dan wordt er een nieuw record toegevoegd aan tabel rooms (1) met een automatisch gegenereerd room\_id en de huidige datum/tijd als created\_at. |
| **Read** | Als de gebruiker het kameroverzicht opent, dan worden alle records uit tabel rooms (1) opgehaald en weergegeven met room\_id, name en created\_at. Als de gebruiker een specifieke kamer selecteert, dan worden de details van die kamer getoond inclusief gekoppelde devices. |
| **Update** | Als de gebruiker de naam van een kamer wijzigt naar een nieuwe unieke waarde, dan wordt het veld name in tabel rooms (1) bijgewerkt voor het betreffende room\_id. Het veld created\_at blijft ongewijzigd. |
| **Delete** | Als de gebruiker een kamer verwijdert en dit bevestigt, dan wordt het record met het betreffende room\_id verwijderd uit tabel rooms (1). Alle devices die aan deze kamer waren gekoppeld worden automatisch ontkoppeld (room\_id wordt NULL in tabel devices). |

**devices**

|  |  |
| --- | --- |
| **SBVR Static devices (2)** |  |
| **Kwalificatie** | 1. Device <dev\_eui> heeft de naam <n>, is gekoppeld aan kamer <room\_id> en werd voor het laatst gezien op <last\_seen\_at>. |
| **Deel** | room\_id is een deelverzameling van rooms.room\_id (foreign key relatie). |
| **Hoofd** | dev\_eui is een hoofdverzameling van dev\_eui. name is een hoofdverzameling van name. last\_seen\_at is een hoofdverzameling van last\_seen\_at. |
| **Niet Leeg** | De invoer van dev\_eui mag niet leeg zijn. De invoer van name mag leeg zijn (wordt dan automatisch gegenereerd). De invoer van room\_id mag leeg zijn (device is dan niet gekoppeld). De invoer van last\_seen\_at mag leeg zijn (wordt gevuld bij eerste data). |
| **Formaat** | Het formaat van dev\_eui is string met een maximum van 20 tekens (TTN device identifier). Het formaat van name is string met een maximum van 100 tekens. Het formaat van room\_id is int (nullable, foreign key naar rooms.room\_id). Het formaat van last\_seen\_at is datetime. |
| **Uniciteit** | Elke invoer in devices is met exact één dev\_eui geïdentificeerd. De dev\_eui is de primaire sleutel en moet uniek zijn. |
| **SBVR Dynamic devices (2)** |  |
| **Create** | Als een nieuw device voor de eerste keer data stuurt via de TTN-webhook en de dev\_eui nog niet bestaat in tabel devices (2), dan wordt er automatisch een nieuw record aangemaakt met de dev\_eui, een automatisch gegenereerde name (Device + laatste 4 karakters van dev\_eui), room\_id als NULL en last\_seen\_at als het huidige tijdstip. |
| **Read** | Als de gebruiker het deviceoverzicht opent of een kamerpagina bekijkt, dan worden de relevante records uit tabel devices (2) opgehaald en weergegeven met dev\_eui, name, room\_id en last\_seen\_at. De online/offline status wordt bepaald op basis van last\_seen\_at (offline indien ouder dan 30 minuten). |
| **Update** | Als een device data stuurt via de TTN-webhook, dan wordt last\_seen\_at bijgewerkt naar het huidige tijdstip. Als de gebruiker een device aan een kamer koppelt, dan wordt room\_id bijgewerkt naar het betreffende room\_id. Als de gebruiker een device ontkoppelt, dan wordt room\_id bijgewerkt naar NULL. |
| **Delete** | Als een device wordt verwijderd uit tabel devices (2), dan worden automatisch alle bijbehorende records in tabel sensors (3) verwijderd (CASCADE). Hierdoor worden ook alle metingen in tabel measurements (4) verwijderd die bij deze sensoren horen. |

**sensors**

|  |  |
| --- | --- |
| **SBVR Static sensors (3)** |  |
| **Kwalificatie** | 1. Sensor van device <dev\_eui> op kanaal <channel> meet het type <type> in eenheid <unit>. |
| **Deel** | dev\_eui is een deelverzameling van devices.dev\_eui (foreign key relatie). |
| **Hoofd** | dev\_eui en channel vormen samen de samengestelde primaire sleutel. type is een hoofdverzameling van type (temperature, humidity, presence, co2, light, noise). unit is een hoofdverzameling van unit (C, %, ppm, dB). |
| **Niet Leeg** | De invoer van dev\_eui mag niet leeg zijn. De invoer van channel mag niet leeg zijn. De invoer van type mag leeg zijn (wordt bepaald door channel mapping). De invoer van unit mag leeg zijn (wordt bepaald door channel mapping). |
| **Formaat** | Het formaat van dev\_eui is string met een maximum van 20 tekens. Het formaat van channel is int (waarde 1 t/m 6 volgens channel mapping: 1=humidity, 2=temperature, 3=presence, 4=co2, 5=light, 6=noise). Het formaat van type is string (temperature, humidity, presence, co2, light, noise). Het formaat van unit is string (C voor temperatuur, % voor humidity/presence/light, ppm voor co2, dB voor noise). |
| **Uniciteit** | Elke invoer in sensors is met exact één combinatie van dev\_eui en channel geïdentificeerd. De combinatie van dev\_eui en channel vormt de samengestelde primaire sleutel en moet uniek zijn. |
| **SBVR Dynamic sensors (3)** |  |
| **Create** | Als een device via de TTN-webhook data stuurt met een nieuw kanaalnummer dat nog niet bestaat voor deze dev\_eui in tabel sensors (3), dan wordt er automatisch een nieuw record aangemaakt met de dev\_eui, het channel nummer, en het type en unit volgens de channel mapping (1=humidity/%, 2=temperature/C, 3=presence/%, 4=co2/ppm, 5=light/%, 6=noise/dB). |
| **Read** | Als de gebruiker een kamerpagina bekijkt, dan worden alle sensoren van de gekoppelde devices opgehaald uit tabel sensors (3) en weergegeven met hun type en unit. Per sensor wordt ook de laatste meetwaarde opgehaald uit tabel measurements (4). |
| **Update** | Als de TTN-webhook data ontvangt voor een bestaande sensor (dev\_eui + channel combinatie), dan worden de velden type en unit bijgewerkt indien nodig (UPSERT operatie). Dit zorgt ervoor dat de channel mapping altijd actueel blijft. |
| **Delete** | Als een sensor wordt verwijderd uit tabel sensors (3), dan worden automatisch alle bijbehorende records in tabel measurements (4) verwijderd (CASCADE). Sensoren worden automatisch verwijderd wanneer het bijbehorende device wordt verwijderd. |

**measurements**

|  |  |
| --- | --- |
| **SBVR Static measurements (4)** |  |
| **Kwalificatie** | 1. Meting <id> van device <dev\_eui> op kanaal <channel> heeft waarde <value> gemeten op <measured\_at>. |
| **Deel** | dev\_eui en channel vormen samen een deelverzameling van sensors.(dev\_eui, channel) (foreign key relatie naar de samengestelde primaire sleutel van sensors). |
| **Hoofd** | id is een hoofdverzameling van id (auto-increment primaire sleutel). value is een hoofdverzameling van value. measured\_at is een hoofdverzameling van measured\_at. |
| **Niet Leeg** | De invoer van id mag niet leeg zijn (wordt automatisch gegenereerd). De invoer van dev\_eui mag niet leeg zijn. De invoer van channel mag niet leeg zijn. De invoer van value mag niet leeg zijn. De invoer van measured\_at mag niet leeg zijn. |
| **Formaat** | Het formaat van id is int (auto-increment primaire sleutel). Het formaat van dev\_eui is string met een maximum van 20 tekens. Het formaat van channel is int (waarde 1 t/m 6). Het formaat van value is decimal(10,2) voor nauwkeurige meetwaarden. Het formaat van measured\_at is datetime met het tijdstip van de meting. |
| **Uniciteit** | Elke invoer in measurements is met exact één id geïdentificeerd. Het id is de primaire sleutel en wordt automatisch gegenereerd. |
| **SBVR Dynamic measurements (4)** |  |
| **Create** | Als een device via de TTN-webhook meetdata stuurt, dan wordt voor elk kanaal in de payload een nieuw record aangemaakt in tabel measurements (4) met een automatisch gegenereerd id, de dev\_eui van het device, het channel nummer, de gemeten value en het tijdstip als measured\_at. |
| **Read** | Als de gebruiker de historie-pagina opent, dan worden de meest recente metingen opgehaald uit tabel measurements (4) en weergegeven in een tabel. De gebruiker kan filteren op kamer en sensortype. Voor grafieken worden de metingen gegroepeerd per uur en wordt het gemiddelde berekend per device. |
| **Update** | Metingen in tabel measurements (4) worden niet bijgewerkt na aanmaak. Elke nieuwe meting van een sensor wordt als een nieuw record opgeslagen om een volledige meethistorie te behouden. |
| **Delete** | Metingen ouder dan 4 dagen worden automatisch verwijderd door de cleanup-functie die wordt aangeroepen bij elke TTN-webhook aanroep. Dit voorkomt dat de database onnodig groot wordt en zorgt ervoor dat alleen recente, relevante data wordt bewaard. |

## 4: Functionele requirements Klimaatapp

Functionele requirements beschrijven wat de applicatie precies moet kunnen doen. Ze vormen de basis van de werking van de Klimaatapp en zorgen ervoor dat het systeem duidelijk afgebakende functies heeft die voor de gebruiker begrijpelijk en praktisch zijn.

In deze applicatie draait het om het ophalen, opslaan en weergeven van klimaatgegevens per kamer. Elke functie heeft een duidelijk doel, zoals het kunnen toevoegen van kamers, het tonen van sensorgegevens, het koppelen van devices en het bekijken van historische data in grafieken.

Door deze functies te combineren ontstaat een eenvoudige maar complete toepassing die inzicht geeft in de actuele en historische omstandigheden in elke ruimte.

|  |  |
| --- | --- |
| **Dynamisch onderdeel** | **User Story** |
| rooms (1) | Als beheerder wil ik kamers kunnen aanmaken, bekijken, bewerken en verwijderen, zodat ik de ruimtes in het gebouw kan beheren. |
| devices (2) | Als beheerder wil ik devices kunnen bekijken en aan kamers koppelen of ontkoppelen, zodat de juiste metingen bij de juiste kamer worden getoond. |
| sensors (3) | Als systeem wil ik sensoren automatisch registreren per device en kanaal, zodat elke meetwaarde correct wordt opgeslagen met het juiste type en eenheid. |
| measurements (4) | Als systeem wil ik meetgegevens opslaan en na 4 dagen automatisch opschonen, zodat de database niet onnodig groot wordt. |

## 5: Niet-functionele requirements

Niet-functionele requirements beschrijven hoe de applicatie moet presteren. Het gaat hierbij om de kwaliteit, beveiliging en gebruikersbeleving van de Klimaatapp: snelheid, betrouwbaarheid, onderhoudbaarheid, gebruiksvriendelijkheid, responsiviteit, flexibiliteit en toegankelijkheid. Deze eisen zorgen ervoor dat de Klimaatapp niet alleen technisch goed werkt, maar ook veilig, prettig in gebruik en toekomstbestendig is.

|  |  |
| --- | --- |
| **Kwaliteit** | **Beschrijving** |
| **Prestaties** | Functioneel: De applicatie toont meetgegevens binnen enkele seconden na het openen van een pagina. Grafieken en tabellen laden snel en reageren direct op filteracties. |
|  | Op bedrijfsniveau: Gebruikers kunnen zonder wachttijd inzicht krijgen in het binnenklimaat, wat bijdraagt aan efficiënt gebouwbeheer en snelle besluitvorming. |
|  | Technisch: De TTN-webhook verwerkt binnenkomende data binnen 1 seconde. API-responses worden binnen 500ms geretourneerd. Grafieken laden binnen 2 seconden, ook bij 4 dagen aan data. |
| **Betrouwbaarheid** | Functioneel: De applicatie toont altijd de laatst bekende waarden en geeft duidelijk aan wanneer data verouderd of onbetrouwbaar is. |
|  | Op bedrijfsniveau: Beheerders kunnen vertrouwen op de getoonde data voor het nemen van beslissingen over klimaatbeheersing in het gebouw. |
|  | Technisch: De backend draait stabiel op Railway met minimale downtime. CO2-waarden boven 5000 ppm worden als sensorfouten gefilterd. Devices worden als offline gemarkeerd na 30 minuten zonder data. |
| **Onderhoudbaarheid** | Functioneel: Nieuwe kamers, devices en sensortypen kunnen worden toegevoegd zonder aanpassingen aan de bestaande functionaliteit. |
|  | Op bedrijfsniveau: De applicatie kan meegroeien met het gebouw en nieuwe ruimtes of sensoren kunnen eenvoudig worden toegevoegd. |
|  | Technisch: De frontend is opgebouwd met OOP-principes (ApiService, RoomManager, ChartManager). De backend gebruikt een modulaire architectuur. Code is voorzien van commentaar. |
| **Gebruiksvriendelijkheid** | Functioneel: De interface is intuïtief te gebruiken zonder handleiding. Alle acties geven directe feedback via toast-notificaties. |
|  | Op bedrijfsniveau: Medewerkers zonder technische achtergrond kunnen de applicatie zelfstandig gebruiken voor het monitoren van het binnenklimaat. |
|  | Technisch: De interface gebruikt duidelijke iconen, leesbare labels en consistente kleuren. Knoppen zijn groot genoeg voor touch-bediening. Foutmeldingen zijn in begrijpelijke taal. |
| **Responsiviteit** | Functioneel: De applicatie werkt goed op desktop, tablet en smartphone zonder functieverlies. |
|  | Op bedrijfsniveau: Beheerders kunnen het binnenklimaat monitoren vanaf elk apparaat, ook onderweg of tijdens rondgangen in het gebouw. |
|  | Technisch: De frontend gebruikt een responsive CSS-layout met flexbox en media queries. Minimale ondersteunde resolutie is 360px breed. Tabellen worden omgezet naar kaarten op kleine schermen. |
| **Flexibiliteit** | Functioneel: De applicatie ondersteunt 6 verschillende sensortypen en kan eenvoudig worden uitgebreid met nieuwe types. |
|  | Op bedrijfsniveau: Het systeem kan worden aangepast aan veranderende behoeften, zoals het toevoegen van nieuwe meetwaarden of locaties. |
|  | Technisch: Het channel mapping systeem (1=humidity, 2=temperature, 3=presence, 4=co2, 5=light, 6=noise) kan worden uitgebreid. De genormaliseerde databasestructuur ondersteunt nieuwe sensortypen zonder schema-wijzigingen. |
| **Beveiliging** | Functioneel: De applicatie verwerkt alleen klimaatdata en geen persoonsgegevens. Gevoelige configuratie is niet zichtbaar voor gebruikers. |
|  | Op bedrijfsniveau: De organisatie kan de applicatie gebruiken zonder zorgen over datalekken of ongeautoriseerde toegang tot gevoelige informatie. |
|  | Technisch: Communicatie verloopt via HTTPS. API-keys en database-credentials worden bewaard in environment variables aan de serverzijde. De frontend maakt geen directe database-connecties. |
| **Toegankelijkheid** | Functioneel: De applicatie is bruikbaar voor gebruikers met verschillende vaardigheden en beperkingen. |
|  | Op bedrijfsniveau: Alle medewerkers, ongeacht fysieke beperkingen, kunnen de applicatie gebruiken voor hun werkzaamheden. |
|  | Technisch: De frontend voldoet aan basisprincipes van toegankelijkheid: voldoende kleurcontrast, duidelijke focus-states, leesbare lettergroottes en logische tab-volgorde. |

**Aanvullende informatie niet-functionele requirements**

De Klimaatapp wordt getest op zowel functionele als niet-functionele eisen. De applicatie haalt data op via API-calls en toont deze in real-time op het dashboard en in de grafieken op de historie-pagina.

Daarnaast wordt aandacht besteed aan toegankelijkheid en bruikbaarheid: de interface is goed leesbaar op kleine schermen (minimale resolutie 360 px) en de status van devices wordt met kleur en tekst ondersteund (groen = online, grijs = offline).

Tot slot is er aandacht voor security en onderhoud. De frontend toont geen interne foutmeldingen en werkt uitsluitend via beveiligde API-calls. De OOP-architectuur (ApiService, RoomManager, ChartManager) maakt het eenvoudig om nieuwe functies toe te voegen.

**Kwaliteitsrequirements**

Naast de algemene niet-functionele eisen gelden er ook specifieke kwaliteitsvereisten voor de frontend, die hieronder zijn samengevat volgens het kwaliteitsontwerp (4d.2).

|  |  |
| --- | --- |
| **Kwaliteitsaspect** | **Beschrijving** |
| Gebruiksvriendelijkheid | De applicatie moet intuïtief te gebruiken zijn met een eenvoudige navigatie en duidelijke labels. |
| Toegankelijkheid en devices | De frontend schaalt automatisch mee op verschillende schermgroottes (desktop, tablet, smartphone). |
| Prestaties | De applicatie laadt snel en haalt alleen actuele data op. De interface reageert soepel op gebruikersacties. |
| Betrouwbaarheid | De app toont altijd de laatst beschikbare waarden en markeert oudere data automatisch als niet actueel. |
| Beveiliging en privacy | Alleen klimaatgegevens worden verwerkt. API-keys en andere gevoelige informatie blijven aan de serverzijde. |
| Onderhoudbaarheid | De frontend is modulair opgebouwd met OOP-classes, waardoor het eenvoudig is om nieuwe functies toe te voegen. |

## 6: MoSCoW-overzicht met user stories en acceptatiecriteria

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **User story** | **Prioriteit** | **Acceptatiecriteria** |
| US-001 | Create (beheerder) – Kamer Als beheerder wil ik een nieuwe kamer kunnen aanmaken, zodat ik een ruimte kan toevoegen aan het systeem. | Must | - De naam van de kamer is verplicht en maximaal 100 tekens. - De naam is nog niet in gebruik. - Na opslaan verschijnt de kamer direct in het overzicht. - Er wordt een bevestiging getoond via een toast-notificatie. |
| US-002 | Read (beheerder) – Kamers overzicht Als beheerder wil ik een lijst van alle kamers kunnen zien, zodat ik weet welke ruimtes zijn aangemaakt. | Must | - Het kameroverzicht toont alle aanwezige kamers. - Per kamer is de naam, het aantal devices en de online status zichtbaar. - Elk kamer-kaart is klikbaar om naar de detailpagina te navigeren. - Kamers worden alfabetisch gesorteerd. |
| US-003 | Read (gebruiker) – Kamer detail Als gebruiker wil ik de klimaatgegevens van een kamer kunnen bekijken, zodat ik inzicht krijg in het binnenklimaat. | Must | - De detailpagina toont alle 6 sensortypen: temperatuur, luchtvochtigheid, CO2, licht, aanwezigheid en geluid. - Elke waarde toont het tijdstip van de laatste meting. - Gekoppelde devices worden getoond met hun online/offline status. - Bij verouderde data (>30 min) wordt dit visueel aangegeven. |
| US-004 | Update (beheerder) – Kamer naam wijzigen Als beheerder wil ik de naam van een kamer kunnen wijzigen, zodat ik fouten kan corrigeren of de naam kan actualiseren. | Should | - De beheerder kan een kamer selecteren en een nieuwe naam invoeren. - De nieuwe naam moet uniek zijn binnen het systeem. - Na opslaan wordt de gewijzigde naam direct getoond. - Er wordt een bevestiging getoond via een toast-notificatie. |
| US-005 | Delete (beheerder) – Kamer verwijderen Als beheerder wil ik een kamer kunnen verwijderen, zodat ik ongebruikte ruimtes kan opruimen. | Should | - De beheerder ziet een verwijderknop bij elke kamer. - Voor verwijdering moet een bevestiging worden gegeven. - Na bevestiging verdwijnt de kamer uit het overzicht. - Gekoppelde devices worden automatisch ontkoppeld (niet verwijderd). |
| US-006 | Read (beheerder) – Devices overzicht Als beheerder wil ik alle beschikbare devices kunnen zien, zodat ik weet welke devices gekoppeld kunnen worden. | Must | - Op de kamerdetailpagina is een dropdown zichtbaar met niet-gekoppelde devices. - Per device is de dev\_eui (laatste 8 karakters) en naam zichtbaar. - De dropdown toont alleen devices die nog niet aan een kamer zijn gekoppeld. |
| US-007 | Update (beheerder) – Device koppelen Als beheerder wil ik een device aan een kamer kunnen koppelen, zodat de metingen van dat device bij de juiste kamer worden getoond. | Must | - De beheerder kan een device selecteren uit de dropdown. - Na klikken op 'Koppelen' wordt het device toegevoegd aan de kamer. - Het device verschijnt direct in de lijst van gekoppelde devices. - De sensorwaarden van het device worden getoond. |
| US-008 | Update (beheerder) – Device ontkoppelen Als beheerder wil ik een device van een kamer kunnen ontkoppelen, zodat ik het device kan verplaatsen naar een andere kamer. | Should | - Bij elk gekoppeld device is een 'Ontkoppelen' knop zichtbaar. - Na klikken wordt het device ontkoppeld van de kamer. - Het device verdwijnt uit de kamerdetailpagina. - Het device verschijnt weer in de dropdown voor niet-gekoppelde devices. |
| US-009 | Create (systeem) – Meting opslaan Als het systeem meetdata ontvangt via de TTN-webhook, dan moet deze worden opgeslagen zodat de data beschikbaar is voor weergave. | Must | - Bij elke TTN-uplink worden alle sensorwaarden per kanaal opgeslagen. - De channel mapping wordt correct toegepast (1=humidity, 2=temperature, etc.). - De meting bevat dev\_eui, channel, value en measured\_at. - Nieuwe devices worden automatisch aangemaakt. |
| US-010 | Delete (systeem) – Oude metingen opschonen Als systeem wil ik oude metingen automatisch verwijderen, zodat de database niet onnodig groot wordt. | Must | - Metingen ouder dan 4 dagen worden automatisch verwijderd. - De cleanup draait automatisch bij elke webhook-aanroep. - De verwijdering wordt gelogd in de server console. - Recente metingen (< 4 dagen) blijven behouden. |
| US-011 | Read (gebruiker) – Historie tabel Als gebruiker wil ik een tabel met recente metingen kunnen bekijken, zodat ik de ruwe meetdata kan inzien. | Should | - De historie-pagina toont een tabel met de 100 meest recente metingen. - Per meting is datum/tijd, kamer, device, sensortype en waarde zichtbaar. - De gebruiker kan filteren op kamer via een dropdown. - De gebruiker kan filteren op sensortype via een dropdown. |
| US-012 | Read (gebruiker) – Grafieken bekijken Als gebruiker wil ik grafieken kunnen bekijken, zodat ik trends in de klimaatdata visueel kan analyseren. | Should | - De historie-pagina toont per sensortype een grafiek. - Grafieken tonen gemiddelden per uur over de laatste 4 dagen. - Meerdere devices worden in dezelfde grafiek getoond met verschillende kleuren. - De filters voor kamer en sensortype werken ook op de grafieken. |
| US-013 | Read (gebruiker) – Device online status Als gebruiker wil ik kunnen zien of een device online of offline is, zodat ik weet of de data actueel is. | Could | - Devices die langer dan 30 minuten geen data hebben gestuurd worden als offline gemarkeerd. - De online/offline status wordt visueel aangegeven met kleur (groen/grijs). - De tekst 'Online' of 'Offline' wordt getoond bij het device. |
| US-014 | Read (gebruiker) – Tijd sinds laatste meting Als gebruiker wil ik kunnen zien hoe lang geleden de laatste meting was, zodat ik weet hoe actueel de data is. | Could | - Bij elke sensorwaarde wordt de tijd sinds de laatste meting getoond. - De tijd wordt in leesbaar formaat getoond (bijv. '5 minuten geleden'). - Bij verouderde data wordt dit visueel benadrukt. |
| US-015 | Create (beheerder) – Gebruikersaccounts Als beheerder wil ik gebruikersaccounts kunnen aanmaken met verschillende rechten. | Won't | - Niet geïmplementeerd in huidige versie. - De applicatie werkt zonder authenticatie. - Kan in toekomstige versie worden toegevoegd. |
| US-016 | Create (beheerder) – Notificaties instellen Als beheerder wil ik notificaties kunnen instellen bij extreme waarden. | Won't | - Niet geïmplementeerd in huidige versie. - Push-notificaties vallen buiten de scope. - Kan in toekomstige versie worden toegevoegd. |

## 7: Acceptatierequirements:

Acceptatierequirements vormen een belangrijk onderdeel van het ontwikkelproces van de Klimaatapp. Ze beschrijven de criteria waaraan de applicatie moet voldoen om door zowel ontwikkelaars als gebruikers als 'geslaagd' te worden beschouwd.

Kort gezegd: de acceptatierequirements vormen de brug tussen ontwerp en realiteit. Ze zorgen ervoor dat de Klimaatapp niet alleen technisch klopt, maar ook praktisch bruikbaar, onderhoudbaar en toekomstbestendig is.

**Geïdentificeerd – Back-End-Applicatie**

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Acceptatie-eis (Back-End)** | **Resultaat** |
| 1 | De applicatie moet binnenkomende TTN-webhook payloads correct kunnen ontvangen en verwerken. | Ja/nee |
| 2 | Bij elke uplink moet de applicatie de sensorwaarden per kanaal opslaan in measurements met de juiste channel mapping (1=humidity, 2=temperature, 3=presence, 4=co2, 5=light, 6=noise). | Ja/nee |
| 3 | De applicatie moet devices correct koppelen aan kamers (via devices.room\_id) en deze relatie tonen in de frontend. | Ja/nee |
| 4 | Er moet een API-endpoint beschikbaar zijn waarmee de frontend per kamer de actuele meetwaarden kan ophalen (/api/rooms/:id). | Ja/nee |
| 5 | De backend moet CO2-waarden boven 5000 ppm filteren uit de grafiekdata als sensorfouten. | Ja/nee |
| 6 | De database moet CRUD-operaties ondersteunen voor rooms, devices, sensors en measurements. | Ja/nee |
| 7 | De cleanup-functie moet metingen ouder dan 4 dagen automatisch verwijderen bij elke webhook-aanroep. | Ja/nee |
| 8 | De backend moet 24/7 beschikbaar zijn voor het ontvangen van webhook-requests via Railway hosting. | Ja/nee |
| 9 | Wanneer een device langer dan 30 minuten geen data verstuurt, moet dit als offline worden gemarkeerd. | Ja/nee |
| 10 | Nieuwe devices moeten automatisch worden aangemaakt wanneer ze voor het eerst data sturen via TTN. | Ja/nee |

**Gekwalificeerd – Front-End-Applicatie**

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Acceptatie-eis (Front-End)** | **Resultaat** |
| 1 | De UI moet een lijst tonen van alle kamers met hun actuele temperatuur, luchtvochtigheid en online/offline status. | Ja/nee |
| 2 | De kamerdetailpagina moet alle 6 sensortypen tonen: temperatuur, luchtvochtigheid, CO2, licht, aanwezigheid en geluid. | Ja/nee |
| 3 | De beheerder moet devices kunnen koppelen aan en ontkoppelen van kamers via de interface. | Ja/nee |
| 4 | Kamers zonder gekoppeld device moeten een duidelijke melding tonen. | Ja/nee |
| 5 | De historie-pagina moet een tabel tonen met recente metingen, filterbaar op kamer en sensortype. | Ja/nee |
| 6 | De historie-pagina moet grafieken tonen per sensortype met gemiddelden per uur. | Ja/nee |
| 7 | De gebruikersinterface moet responsief zijn en werken op desktop, tablet en smartphone. | Ja/nee |
| 8 | Elke meting moet een tijdstempel tonen met de tijd sinds de laatste update. | Ja/nee |
| 9 | De lay-out moet gebruiksvriendelijk zijn met consistente benaming en toast-notificaties voor feedback. | Ja/nee |
| 10 | De frontend moet zijn opgebouwd met OOP-principes (ApiService, RoomManager, ChartManager). | Ja/nee |

## 8: Waardepropositie (Statement of Needs)

De waardepropositie beschrijft waar de Klimaatapp voor bedoeld is en welke behoeften die vervult. Het belangrijkste doel is simpel: in één oogopslag laten zien hoe het binnenklimaat in een ruimte ervoor staat. Geen ingewikkelde grafieken of technische termen, maar gewoon duidelijke waarden per kamer.

De app is bedoeld voor iedereen die inzicht wil in temperatuur, luchtvochtigheid, CO2, licht, aanwezigheid en geluid in verschillende ruimtes. De data komt binnen via sensoren die zijn aangesloten op The Things Network (TTN) en wordt getoond in een overzichtelijke interface. Zo kun je snel zien of een kamer te warm is, te vochtig, of dat de CO2-waarde aan de hoge kant is.

De opzet is bewust eenvoudig gehouden. Alle belangrijke informatie staat op één plek, zonder overbodige functies. Je kunt kamers toevoegen, bewerken of verwijderen, en devices koppelen of ontkoppelen. In elk kamerscherm zie je direct de meest recente metingen van alle zes sensortypen. Dat maakt de app praktisch en toegankelijk, ook voor mensen die niet technisch zijn.

Naast de actuele waarden biedt de app ook een historie-pagina. Daar kun je terugkijken naar de metingen van de afgelopen vier dagen, zowel in een tabel als in grafieken. Dit helpt om trends te herkennen: wordt een ruimte 's middags altijd te warm? Stijgt de CO2 tijdens vergaderingen? Dat soort patronen worden zo zichtbaar.

De interface is rustig en overzichtelijk. Elke waarde heeft zijn eigen kaart, en de kleuren en iconen helpen om snel te zien wat er aan de hand is. Of je nu één kamer bekijkt of meerdere devices beheert, de app blijft overzichtelijk.

Kortom: de Klimaatapp maakt klimaatdata begrijpelijk. Geen gedoe, gewoon duidelijke informatie over het binnenklimaat.

## 9: Oplosstrategie (Qualification Strategy of Needs)

Voordat er code werd geschreven, is eerst nagedacht over hoe de data het beste kon worden opgeslagen en getoond. De sensoren sturen hun meetwaarden via The Things Network naar een webhook, en die data moet ergens terechtkomen. Daarvoor is een databasestructuur ontworpen met vier tabellen: kamers, devices, sensoren en metingen. Elk onderdeel heeft zijn eigen plek, en de relaties ertussen zijn duidelijk vastgelegd.

Dit datamodel is uitgewerkt in een CEM (Conceptueel Entiteitenmodel) en een ERD (Entity Relationship Diagram). Het CEM beschrijft in zinnen hoe de gegevens zich tot elkaar verhouden, terwijl het ERD dit visueel maakt. Zo is in één oogopslag te zien dat een kamer meerdere devices kan bevatten, een device meerdere sensoren heeft, en elke sensor meerdere metingen oplevert.

Daarnaast zijn er statische en dynamische requirements opgesteld. De statische kant beschrijft wat er wordt opgeslagen: welke velden verplicht zijn (zoals dev\_eui en channel) en welke optioneel (zoals een devicenaam). De dynamische kant laat zien wat er gebeurt als een gebruiker een kamer aanmaakt, een device koppelt, of als er nieuwe meetdata binnenkomt via de webhook.

De functionele requirements beschrijven wat de app kan: kamers beheren, devices koppelen en ontkoppelen, actuele meetwaarden tonen en historische data bekijken in grafieken. De niet-functionele requirements gaan over hoe dat gebeurt: snel, betrouwbaar, en met een prettige interface die ook op een telefoon goed werkt.

Voor de frontend is een klassendiagram gemaakt dat laat zien hoe de code is opgebouwd. Er zijn drie classes: ApiService voor de communicatie met de backend, RoomManager voor het beheer van kamers en devices, en ChartManager voor de grafieken en historie. Door deze scheiding blijft de code overzichtelijk en makkelijk uit te breiden.

Tot slot zijn er acceptatietests opgesteld om te controleren of alles werkt zoals bedoeld. Hierbij wordt gekeken of de belangrijkste functies goed draaien, of de data correct wordt getoond, en of de app logisch aanvoelt voor de gebruiker.

Door deze aanpak is de Klimaatapp niet alleen technisch goed opgezet, maar ook begrijpelijk en prettig in gebruik.

## 10: Stakeholderanalyse (Belanghebbendelijst)

De stakeholderanalyse geeft een overzicht van de personen en partijen die direct of indirect betrokken zijn bij de ontwikkeling van de Klimaatapp.  
Bij dit type project is het belangrijk om een goed beeld te hebben van wie er belang heeft bij de werking van de applicatie en wie invloed kan uitoefenen op de ontwikkeling of het eindresultaat.  
De onderstaande tabel geeft weer welke stakeholders zijn betrokken en wat hun invloed is op het project.

|  |  |
| --- | --- |
| Stakeholder | Invloed |
| Projectbegeleider / Docent | Groot |
| Gebouwbeheerder of Facilitair medewerker | Groot |
| Softwareontwikkelaar (student) | Middel |
| Medestudenten / Testgebruikers | Middel |
| ICT-afdeling van de school of organisatie | Klein |
| Eindgebruikers (bijv. docenten, medewerkers) | Klein |

**1. Projectbegeleider/Docent (Groot)**

De projectbegeleider of docent begeleidt het gehele ontwikkelproces en beoordeelt of de applicatie aan de gestelde eisen voldoet.  
Hun feedback bepaalt in grote mate de richting van het project, zowel op technisch als inhoudelijk vlak.

**2. Gebouwbeheerder of Facilitair Medewerker (Groot)**

Deze persoon gebruikt de Klimaatapp om inzicht te krijgen in temperatuur, luchtvochtigheid en andere klimaatgegevens per ruimte.  
Voor hen is de betrouwbaarheid van de meetgegevens het belangrijkst, omdat ze de app gebruiken om het binnenklimaat te verbeteren.

**3. Softwareontwikkelaar (Student) (Middel)**

De ontwikkelaar (student) is verantwoordelijk voor het ontwerp, de functionaliteit en de technische uitvoering van de applicatie.  
Deze stakeholder heeft een directe invloed op hoe goed de applicatie werkt en hoe gebruiksvriendelijk deze is.

**4. Medestudenten/Testgebruikers (Middel)**

Tijdens de testfase helpen medestudenten bij het beoordelen van de bruikbaarheid van de app.  
Zij geven waardevolle feedback over de navigatie, duidelijkheid van informatie en het algemene gebruiksgemak.

**5. ICT-afdeling van de school of organisatie (Klein)**

De ICT-afdeling kan een rol spelen in de technische ondersteuning, bijvoorbeeld bij het hosten van de database of het beheren van netwerktoegang.  
Hun invloed is beperkt, maar hun hulp kan belangrijk zijn voor het online functioneren van de applicatie.

**6. Eindgebruikers (Docenten of Medewerkers) (Klein)**

Dit zijn de uiteindelijke gebruikers die de data bekijken en interpreteren.  
Zij hebben weinig invloed op de ontwikkeling zelf, maar hun ervaring bepaalt wel of de app in de praktijk nuttig en prettig in gebruik is.

Kortom helpt de stakeholderanalyse om te begrijpen wie prioriteit heeft bij beslissingen tijdens het ontwikkelproces.  
Door de wensen van elke groep in kaart te brengen, kan de Klimaatapp zich richten op wat echt belangrijk is: een eenvoudige, betrouwbare en duidelijk werkende applicatie die inzicht biedt in het binnenklimaat.

## 11: CEM (Conceptual Enterprise Model)

Afbeelding met tekst, schermopname, nummer, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

Afbeelding met tekst, schermopname, nummer, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

Afbeelding met tekst, schermopname, nummer, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

Afbeelding met tekst, schermopname, nummer, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

### **12: ERD (Entiteiten Relatie Diagram)**

Afbeelding met tekst, schermopname, Lettertype, Parallel

Door AI gegenereerde inhoud is mogelijk onjuist.

### 

### **13: UML Classdiagram (Klassendiagram)**

**Afbeelding met tekst, schermopname, Lettertype, nummer

Door AI gegenereerde inhoud is mogelijk onjuist.**

### **14: UML Usecasediagram**

Afbeelding met tekst, diagram, lijn, schermopname

Door AI gegenereerde inhoud is mogelijk onjuist.

### **15: UML Activitydiagram**

Afbeelding met tekst, schermopname, diagram, lijn

Door AI gegenereerde inhoud is mogelijk onjuist.

16: Samenvatting Requirementsdocument  
  
In dit document is beschreven wat de Klimaatapp moet kunnen en hoe alles technisch in elkaar zit. De app toont per kamer de actuele klimaatgegevens: temperatuur, luchtvochtigheid, CO2, licht, aanwezigheid en geluid. Devices kunnen eenvoudig aan kamers worden gekoppeld of ontkoppeld, waardoor de meetwaarden altijd bij de juiste ruimte terechtkomen. Daarnaast biedt de historie-pagina inzicht in trends via een tabel met recente metingen en grafieken die gemiddelden per uur tonen.

De database is opgebouwd uit vier tabellen die logisch op elkaar aansluiten: kamers bevatten devices, devices bevatten sensoren en sensoren leveren metingen. Deze structuur is terug te zien in het ERD en het CEM. Het klassendiagram laat zien hoe de frontend is opgezet met drie classes die elk hun eigen verantwoordelijkheid hebben: ApiService voor communicatie met de server, RoomManager voor het beheer van kamers en devices, en ChartManager voor de grafieken en historie.

Door deze opzet blijft de app overzichtelijk en makkelijk uit te breiden. De usabilitytest helpt om te controleren of alles ook in de praktijk logisch aanvoelt voor de gebruiker. Zo ontstaat een applicatie die technisch goed in elkaar zit en tegelijkertijd prettig is om te gebruiken.

# 4d.e Kwaliteitsontwerp Front-end

### Inleiding

Dit kwaliteitsontwerp beschrijft waar de frontend van de Klimaatapp aan moet voldoen qua kwaliteit en gebruikservaring. Het gaat hierbij niet zozeer om wat de app doet, maar om hoe goed die dat doet. Denk aan snelheid, duidelijkheid, en of de app ook werkt op een telefoon of tablet.

Dit onderdeel bouwt voort op de niet-functionele requirements uit het requirementsdocument en gaat dieper in op het ontwerp van de interface. Hoe blijft de app overzichtelijk? Wat gebeurt er als een device even geen data stuurt? En hoe zorg je dat de code later nog te begrijpen en uit te breiden is? Dat soort vragen komen hier aan bod.

### Gebruiksvriendelijkheid (Usability)

De Klimaatapp is bedoeld om snel inzicht te geven in het binnenklimaat, zonder dat je er technische kennis voor nodig hebt. Daarom is gekozen voor een simpele opzet met duidelijke pagina's en weinig poespas.

*Duidelijke navigatie*

De app heeft drie hoofdpagina's: een overzicht van alle kamers, een detailpagina per kamer, en een historie-pagina met grafieken en een meettabel.

* Vanuit het kameroverzicht klik je direct door naar een specifieke kamer.
* De navigatiebalk bovenaan blijft op elke pagina hetzelfde, zodat je altijd weet waar je bent.
* Knoppen zijn groot genoeg om ook op een touchscreen goed te werken.

*Visuele opbouw en leesbaarheid*

Alle klimaatgegevens worden in kaarten weergegeven: temperatuur, luchtvochtigheid, CO2, licht, aanwezigheid en geluid. Elke kaart toont de waarde, de eenheid en hoe lang geleden de meting is gedaan.

* Er is bewust veel witruimte gebruikt zodat het niet druk oogt.
* Kleuren en iconen helpen om snel te zien wat er aan de hand is: groen voor online, grijs voor offline.
* Labels zijn kort en in gewone taal geschreven.

*Feedback en statusinformatie*

De gebruiker krijgt altijd te zien wat er gebeurt. Als je een kamer toevoegt of een device koppelt, verschijnt er een toast-notificatie die bevestigt dat de actie is gelukt.

* Bij fouten verschijnt er ook een melding, maar dan in het rood.
* Als een device langer dan 30 minuten geen data heeft gestuurd, wordt dit als offline gemarkeerd.
* De app toont nooit technische foutcodes, maar gewoon normale meldingen.

*Consistentie en eenvoud*

Alle pagina's gebruiken dezelfde stijl. Knoppen zien er overal hetzelfde uit, en de kleuren en lettertypes zijn consistent. Dit zorgt ervoor dat je niet hoeft na te denken over hoe iets werkt.

* Nieuwe devices verschijnen automatisch in de lijst zodra ze voor het eerst data sturen.
* Het koppelen en ontkoppelen van devices aan kamers gaat via een simpele dropdown en knop.

### Toegankelijkheid en devices

De Klimaatapp draait volledig in de browser en hoeft niet geïnstalleerd te worden. De layout past zich automatisch aan het schermformaat aan, of je nu op een desktop, laptop, tablet of telefoon zit.

* Op kleinere schermen worden de sensorkaarten onder elkaar geplaatst in plaats van naast elkaar.
* De navigatie blijft op elk apparaat hetzelfde, zodat de app altijd herkenbaar aanvoelt.
* Tekst en knoppen schalen mee, waardoor je nooit hoeft te zoomen.

De app is getest in Google Chrome en Microsoft Edge, de browsers die op school het meest worden gebruikt. De kleuren en contrasten zijn zo gekozen dat de app ook goed leesbaar is voor mensen met lichte visuele beperkingen.

### Prestaties

De app is zo gebouwd dat die snel laadt en soepel reageert, ook op wat oudere apparaten of tragere netwerken.

* Het kameroverzicht en de detailpagina laden binnen een paar seconden.
* De historie-pagina haalt grafiekdata op die al door de backend is gegroepeerd per uur, waardoor er minder data over de lijn hoeft.
* Er wordt alleen data opgehaald die echt nodig is. De app vraagt niet de hele database op, maar alleen de relevante metingen.

De grafieken worden gerenderd met Chart.js, een lichte library die goed presteert in de browser. Door de 4-dagen limiet op de data blijft de hoeveelheid informatie beheersbaar.

### Betrouwbaarheid en beschikbaarheid

De Klimaatapp is afhankelijk van data die binnenkomt via The Things Network. Als een device tijdelijk geen data stuurt, blijft de app gewoon werken met de laatst bekende waarden.

* Devices die langer dan 30 minuten geen data hebben gestuurd, worden als offline gemarkeerd.
* De backend filtert CO2-waarden boven 5000 ppm uit de grafieken, omdat dit sensorfouten zijn die het beeld zouden vertekenen.
* Zelfs als er even geen internetverbinding is, blijft de interface bruikbaar. Zodra de verbinding terug is, wordt de data vanzelf bijgewerkt.

### Beveiliging en privacy

De app verzamelt geen persoonsgegevens. Er worden alleen klimaatwaarden opgeslagen: temperatuur, luchtvochtigheid, CO2, licht, aanwezigheid en geluid. Daardoor is er weinig risico op datalekken of misbruik.

* Alle communicatie tussen de frontend en backend gaat via HTTPS.
* API-keys en andere gevoelige informatie staan alleen aan de serverkant en zijn niet zichtbaar in de browser.
* De frontend maakt geen directe verbinding met de database, maar communiceert alleen via de API.

### Onderhoudbaarheid en uitbreidbaarheid

De frontend is opgebouwd volgens OOP-principes met drie losse classes die elk hun eigen taak hebben:

* **ApiService**: verzorgt alle communicatie met de backend (GET, POST, PUT, DELETE).
* **RoomManager**: beheert de kamers, devices, navigatie en modals.
* **ChartManager**: regelt de grafieken en de historie-tabel.

Door deze scheiding is de code overzichtelijk en makkelijk te begrijpen. Als er later een nieuwe functie bij moet, hoef je niet de hele codebase door te spitten. Je voegt gewoon iets toe aan de juiste class.

De CSS is consistent toegepast, waardoor visuele aanpassingen snel door te voeren zijn. En doordat de backend al 6 sensortypen ondersteunt via channel mapping, kan de frontend eenvoudig worden uitgebreid als er nieuwe sensoren bijkomen.

### Conclusie

De Klimaatapp is gebouwd met oog voor duidelijkheid, snelheid en gemak. De interface werkt goed op verschillende apparaten en toont de klimaatgegevens op een overzichtelijke manier. Dankzij de OOP-structuur is de code makkelijk te onderhouden en later uit te breiden.

De app doet wat die moet doen: snel inzicht geven in het binnenklimaat, zonder gedoe. En dat is uiteindelijk waar het om draait.

# 5a.2 Usabilitytest

## Inleiding

De usabilitytest heeft als doel te onderzoeken of de Klimaatapp duidelijk en prettig te gebruiken is voor de eindgebruiker. Het draait hierbij niet om de techniek, maar om de ervaring: begrijpt de gebruiker wat er te zien is, en lukt het om zonder hulp de belangrijkste handelingen uit te voeren?

De test wordt uitgevoerd zodra de frontend van de applicatie beschikbaar is. Hierbij wordt gekeken naar de belangrijkste onderdelen van de app, zoals het bekijken van kamers, het inzien van klimaatgegevens en het herkennen van niet-actuele informatie. Door de test uit te voeren met één of meerdere proefpersonen wordt inzicht verkregen in hoe gebruiksvriendelijk en begrijpelijk de interface is. De resultaten van deze test zullen later worden toegevoegd zodra de werkende versie van de applicatie gereed is.

## Testopzet

De usabilitytest wordt gedaan volgens de **Hallway-testmethode**. Dat betekent dat één of meerdere proefpersonen de Klimaatapp gebruiken zonder dat er van tevoren uitleg wordt gegeven. Zo kun je goed zien of iemand de app begrijpt en weet wat hij moet doen zonder hulp. De focus ligt op de eerste indruk: is het duidelijk, voelt het logisch aan en kan iemand snel zijn weg vinden?

Tijdens de test krijgt elke deelnemer een paar korte opdrachten die aansluiten op de belangrijkste onderdelen van de app, zoals het bekijken van kamers, het zien van klimaatgegevens en het herkennen van oude data. De proefpersonen voeren deze taken zelfstandig uit, terwijl wordt gekeken hoe dit gaat en waar iemand eventueel vastloopt.

Bij elke test wordt gelet op drie dingen:

* **Duidelijkheid:** begrijpt de gebruiker wat er op het scherm staat?
* **Navigatie:** kan de gebruiker makkelijk de juiste functie vinden?
* **Feedback:** ziet de gebruiker of data nog actueel is of niet?

Alles wat opvalt wordt direct genoteerd: wat werkt goed, wat is verwarrend en wat zou beter kunnen. De test wordt uitgevoerd zodra de app online beschikbaar is. De resultaten worden later ingevuld in een overzichtstabel, zodat per user story te zien is hoe gebruiksvriendelijk de applicatie in de praktijk is.

De onderstaande tabel wordt later ingevuld na uitvoering van de Hallway-test.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User story | Testopdracht | Doel van de test / Verwacht gedrag (acceptatiecriterium) | Observaties tijdens test | Resultaat (geslaagd / niet geslaagd) | Opmerkingen / Notities | Mogelijke verbeterpunten |
| US-001 | Maak een nieuwe kamer aan en controleer of deze in het overzicht verschijnt. | De gebruiker kan een kamer aanmaken met een unieke naam. Na het opslaan verschijnt de kamer direct in de lijst. |  |  |  |  |
| US-002 | Bekijk de klimaatgegevens van een kamer. | De gebruiker kan een kamer selecteren en ziet temperatuur, luchtvochtigheid, licht en aanwezigheid in één overzicht. |  |  |  |  |
| US-003 | Pas de naam van een kamer aan en controleer of de wijziging wordt opgeslagen. | De gebruiker kan een kamer selecteren, hernoemen en opslaan. De nieuwe naam verschijnt direct in het overzicht. |  |  |  |  |
| US-004 | Registreer een nieuwe sensor en koppel deze aan een kamer. | De gebruiker kan een sensor toevoegen met een unieke device-id en dev-eui, en deze koppelen aan een bestaande kamer. |  |  |  |  |
| US-005 | Controleer de sensorgegevens in een kamer. | De gebruiker ziet welke sensoren actief zijn, inclusief laatste update en status. |  |  |  |  |
| US-007 | Controleer of de laatste meting per sensor goed wordt weergegeven. | De app toont per sensor de juiste temperatuur, luchtvochtigheid, licht en aanwezigheid, met het tijdstip van de laatste update. |  |  |  |  |
| US-008 | Laat de gebruiker een sensor kiezen die geen recente data heeft en controleer of dit duidelijk wordt weergegeven. | De gebruiker ziet een melding of kleurverschil wanneer data ouder is dan 5 minuten (niet actueel). |  |  |  |  |
| US-009 | Bekijk een kamer waar geen sensor actief is. | De gebruiker krijgt een melding dat er geen actuele data beschikbaar is. |  |  |  |  |
| US-010 | Controleer of foutmeldingen (zoals dubbele namen) duidelijk worden weergegeven. | De gebruiker ziet een nette foutmelding bij ongeldige invoer, zonder technische termen. |  |  |  |  |

## Resultaten en conclusie

De resultaten van de usabilitytest worden later toegevoegd zodra de Klimaatapp online beschikbaar is. Tijdens deze test zal per user story worden gekeken of de gebruiker de taak zonder hulp kan uitvoeren en of de interface duidelijk genoeg is. De observaties worden genoteerd in de tabel, samen met eventuele opmerkingen of verbeterpunten.

De verwachting is dat de applicatie in grote lijnen goed te gebruiken zal zijn, omdat de structuur eenvoudig en herkenbaar is opgebouwd. Kleine verbeteringen zullen waarschijnlijk vooral gaan over details, zoals de plaats van knoppen of de duidelijkheid van bepaalde meldingen. De uiteindelijke conclusie zal worden gebaseerd op de ervaringen van de testpersonen en laat zien in hoeverre de Klimaatapp voldoet aan de belangrijkste eisen voor gebruiksgemak en duidelijkheid.

# 7d.1 – zelfstandig werken

|  |  |
| --- | --- |
| Wat? | Omschrijving |
| Inleiding | Tijdens dit blok heb ik gewerkt aan het ontwikkelen van de *Klimaatapp*, een project waarin sensordata via The Things Network (TTN) wordt verwerkt en weergegeven in een gebruiksvriendelijke webomgeving. In tegenstelling tot eerdere projecten, waar samenwerking met andere studenten centraal stond, heb ik deze opdracht volledig zelfstandig uitgevoerd. Dit kwam doordat ik als langstudeerder een alternatieve opdracht kreeg, waarin het doel was om te laten zien dat ik ook zonder begeleiding of groepsverband een compleet project kan organiseren en afronden.  Het zelfstandig uitvoeren van het project bracht andere uitdagingen met zich mee. Zo moest ik alle keuzes zelf maken: van het opzetten van de databank en architectuur tot het ontwerpen van de schermen en het schrijven van de documentatie. Er was geen projectgroep om op terug te vallen, waardoor ik extra kritisch moest kijken naar mijn eigen beslissingen en werktempo.  Tegelijkertijd bevond ik mij in een drukke periode: ik werkte parttime als webdeveloper bij een bedrijf dat met Craft CMS en WordPress werkt, en daarnaast zat ik midden in een verhuizing. Dit zorgde ervoor dat plannen soms lastig was en dat ik regelmatig taken vooruit schoof. Toch heb ik door middel van discipline, feedbackmomenten met mijn begeleider, en het gebruik van tools zoals Trello en weekplanning-sheets structuur kunnen aanbrengen in mijn werk. |
| Feedback en zelfreflectie | Omdat ik dit project volledig zelfstandig heb uitgevoerd, moest ik meer vertrouwen op mijn eigen oordeel en kritisch leren kijken naar mijn werk. Er was geen team dat dagelijks meekijkte, waardoor ik zelf verantwoordelijk was voor het opsporen van fouten en het beoordelen van de kwaliteit. Toch heb ik af en toe kleine feedbackmomenten opgezocht, bijvoorbeeld bij mensen in mijn omgeving of collega’s met technische kennis. Die korte gesprekken hielpen me om even afstand te nemen van mijn eigen werk en met een frisse blik naar bepaalde keuzes te kijken.  Het ontbreken van continue feedback opzoeken heeft me juist zelfstandiger gemaakt. Ik leerde om mijn eigen werk te evalueren, verbeterpunten te herkennen en deze direct door te voeren. Door vaker mijn eigen documentatie en ontwerp te herzien, merkte ik dat ik vanzelf kritischer werd. Kleine fouten of onduidelijkheden zag ik sneller, en ik kon deze direct verbeteren zonder op iemand anders te wachten. Dat heeft ervoor gezorgd dat ik veel bewuster en zelfstandiger te werk ging gedurende het hele project. |
| Visuele presentatie en structuur | Tijdens het project heb ik veel aandacht besteed aan de opbouw en uitstraling van zowel de Klimaatapp als dit verslag. Omdat ik het project alleen uitvoerde, vond ik het belangrijk dat alles duidelijk en overzichtelijk bleef. Ik heb daarom gekozen voor een vaste structuur in mijn document, met consistente opmaak, uniforme tabellen en duidelijke diagrammen. Door overal dezelfde stijl aan te houden, bleef het geheel rustig en beter leesbaar — niet alleen voor de beoordeling, maar ook voor mezelf tijdens het schrijven.  Ook in de applicatie zelf heb ik geprobeerd om de data zo overzichtelijk mogelijk weer te geven. Het dashboard laat per kamer de belangrijkste waarden zien, zoals temperatuur, luchtvochtigheid en lichtintensiteit. Door de informatie in duidelijke blokken te verdelen en simpele kleuren te gebruiken, is in één oogopslag te zien of een sensor actief is of niet. Deze visuele aanpak hielp mij niet alleen bij het ontwikkelen, maar ook bij het begrijpen van de logica achter de data. Het zorgde ervoor dat ik het overzicht kon behouden, zelfs wanneer het project complexer werd. |
| Kritisch denken en beslissingen | Bij het ontwikkelen van de Klimaatapp moest ik regelmatig keuzes maken die invloed hadden op de werking en eenvoud van het project. Omdat ik dit zelfstandig deed, was er niemand om beslissingen mee af te stemmen, dus moest ik zelf de voor- en nadelen goed afwegen. Een van de eerste keuzes ging over de manier waarop de sensordata binnen zou komen. Ik koos uiteindelijk voor een **webhook-oplossing** in plaats van MQTT, omdat dit beter paste bij de schaal van mijn project en eenvoudiger te testen was. De data werd zo automatisch doorgestuurd naar mijn backend zonder dat ik constant een verbinding hoefde te onderhouden, wat de structuur overzichtelijk hield.  Ook bij de database heb ik bewust gekozen voor **PostgreSQL** in plaats van SQLite. Hoewel SQLite eenvoudiger is, wilde ik een oplossing die cloud-based werkt en stabiel genoeg is om later uit te breiden. Dit gaf me de mogelijkheid om de app realistischer op te zetten, zoals dat ook in een professionele omgeving gebeurt. Verder koos ik ervoor om de app **minimalistisch te houden**. In plaats van veel extra functies toe te voegen, richtte ik me op de basis: het tonen van sensordata per kamer. Hierdoor bleef het project haalbaar binnen de beschikbare tijd en kon ik de kwaliteit beter bewaken.  Door al deze beslissingen zelfstandig te nemen, leerde ik kritisch kijken naar wat wel en niet nodig was. Ik toetste mijn keuzes regelmatig door kleine tests uit te voeren en logisch te redeneren of een bepaalde oplossing echt iets toevoegde. Dat proces heeft mij geholpen om niet alleen beter te plannen, maar ook verantwoordelijkheid te nemen voor mijn eigen technische beslissingen. |

# 7d.2 – VerEvaRef

|  |  |
| --- | --- |
| Evalueren | |
|  | **Persoonlijke doelen** |
|  | Persoonlijke Doelen als Student  **Doel: Verbeteren van mijn zelfstandige werkhouding**   * User story: Als student in blok 6 wil ik leren om zelfstandig een project te plannen, op te zetten en af te ronden, zodanig dat ik zonder hulp van anderen een compleet product kan opleveren. In eerdere blokken werkte ik vooral in groepen en kon ik terugvallen op teamleden. Dit keer wil ik laten zien dat ik ook alleen de verantwoordelijkheid kan dragen voor een volledig traject, van analyse tot oplevering.   **Doel: Beter omgaan met planning en uitstelgedrag**   * User story: Als student in blok 6 wil ik een realistische weekplanning opstellen en deze beter volgen, zodanig dat ik mijn tijd efficiënter gebruik en minder werk tot het laatste moment bewaar. In eerdere periodes schoof ik taken vaak voor me uit, vooral als ik druk was met werk of privé. Deze periode wil ik meer structuur aanbrengen in mijn planning en bewuster omgaan met deadlines.   **Doel: Meer aandacht besteden aan overzicht en documentatie**   * User story: Als student in blok 6 wil ik consequenter werken aan de opbouw en duidelijkheid van mijn documentatie, zodanig dat mijn verslag en project ook voor anderen begrijpelijk zijn. In eerdere projecten was mijn verslag soms te technisch of onduidelijk opgebouwd. Dit keer wil ik bewuster schrijven, duidelijke uitleg geven bij diagrammen en alles goed onderbouwen zodat het geheel logisch en overzichtelijk blijft. |
| **Wat** | **Product** |
| \ | Afbeelding met tekst, schermopname, sneeuw, natuur  Door AI gegenereerde inhoud is mogelijk onjuist.  **Week 1**  In de eerste week lag de focus vooral op het technische fundament van de applicatie. Ik ben begonnen met het aanmaken van mijn TTN-account en het koppelen van de sensor, zodat de eerste data binnen kon komen. Nadat de verbinding succesvol tot stand was gebracht, heb ik verschillende tests uitgevoerd via MQTT Explorer om te controleren of de gegevens goed doorkwamen.  Tijdens dit proces merkte ik dat MQTT in de praktijk iets te omslachtig was voor wat ik nodig had. Daarom besloot ik al snel om over te stappen op een Webhook-implementatie — een beslissing die het systeem eenvoudiger en betrouwbaarder maakte. Vervolgens heb ik via Pipedream de eerste succesvolle test uitgevoerd om te verifiëren dat de Webhook live data kon ontvangen. Daarmee stond de basis van de dataverbinding stevig.  De week eindigde met het opstellen van een eerste ruwe schets van de database-structuur, waarin ik alvast nadacht over tabellen als *room*, *sensor* en *measurement*. Dit legde de basis voor de volgende sprint, waarin de database verder wordt uitgewerkt.  Afbeelding met tekst, schermopname, Multimediasoftware, berg  Door AI gegenereerde inhoud is mogelijk onjuist.  **Week 2**  In de tweede week heb ik me volledig gericht op het opzetten van de database en het logisch ordenen van de gegevensstructuur. De ruwe schets van vorige week heb ik uitgewerkt tot een volwaardig datamodel, waarin de tabellen *room*, *sensor* en *measurement* samen een duidelijke hiërarchie vormen.  Het was belangrijk om goed na te denken over de relaties tussen deze tabellen, zodat de data uit de Webhook straks eenvoudig kan worden opgeslagen en opgehaald. Na enkele schetsen heb ik het ERD-diagram uitgewerkt om deze structuur visueel weer te geven. Vervolgens heb ik met behulp van dit model een Conceptueel Entiteit Model (CEM) opgesteld, waarin ik alle kolommen, sleutels en relaties heb vastgelegd.  Aan het einde van de week heb ik alvast een begin gemaakt met het ontwerp van het scherm in Visily, om een eerste indruk te krijgen van hoe de data uiteindelijk in de frontend wordt weergegeven. De week verliep gestructureerd, al merkte ik dat het nog even wennen was om de datalogica goed te visualiseren.  Afbeelding met tekst, schermopname, Multimediasoftware, software  Door AI gegenereerde inhoud is mogelijk onjuist.  **Week 3**  In de derde week ben ik begonnen met het uitwerken van het schermontwerp. Hierbij was het doel om een eenvoudig, overzichtelijk design te maken waarin de kamers en sensoren duidelijk van elkaar te onderscheiden zijn. Ik heb gekozen voor een lay-out met aparte pagina’s per kamer, zodat het systeem makkelijk uit te breiden blijft.  Na het afronden van het ontwerp heb ik gewerkt aan het opstellen van de functionele en niet-functionele requirements. Hierbij heb ik nagedacht over wat de applicatie minimaal moet kunnen en welke kwaliteitsaspecten belangrijk zijn, zoals prestaties en onderhoudbaarheid.  Daarnaast heb ik de eerste versie van de waardepropositie en oplosstrategie geschreven, waarin ik de meerwaarde van het systeem uitleg en beschrijf hoe het technisch wordt opgebouwd. Deze week stond dus vooral in het teken van structuur en overzicht creëren, zodat de volgende sprints soepeler verlopen.  Afbeelding met tekst, schermopname, Multimediasoftware, sneeuw  Door AI gegenereerde inhoud is mogelijk onjuist.  **Week 4**  In week 4 lag de nadruk op de technische documentatie van de applicatie. Ik heb verschillende UML-diagrammen uitgewerkt om de structuur en werking van de Klimaatapp visueel duidelijk te maken. Eerst begon ik met het UML Class Diagram, waarin de tabellen en hun relaties zijn weergegeven. Daarna heb ik het UML Use Case Diagram toegevoegd om de interactie tussen de gebruiker en de applicatie te tonen.  Verder heb ik het UML Activiteitendiagram opgesteld om het proces van het inladen en tonen van klimaatgegevens te visualiseren. Dit hielp om het gedrag van de applicatie stap voor stap te begrijpen.  Aan het einde van de week heb ik een begin gemaakt aan het Kwaliteitsontwerp Frontend.  Hierin beschreef ik hoe de applicatie op verschillende apparaten en schermformaten goed bruikbaar blijft, en welke niet-functionele kwaliteitsaspecten belangrijk zijn, zoals prestaties en onderhoudbaarheid.  Afbeelding met tekst, schermopname, Multimediasoftware, landschap  Door AI gegenereerde inhoud is mogelijk onjuist.  **Week 5**  Tijdens week 5 heb ik het Kwaliteitsontwerp Frontend afgerond. Hierbij heb ik vooral gekeken naar hoe de applicatie reageert op verschillende schermgroottes en browsers, zodat de gebruikerservaring op alle apparaten goed blijft. Ook heb ik vastgelegd hoe de code onderhoudbaar blijft en hoe uitbreidingen in de toekomst eenvoudig door te voeren zijn.  Na het afronden van het kwaliteitsdocument ben ik gestart met het schrijven van de usabilitytest. Hierin heb ik beschreven hoe de gebruiksvriendelijkheid van de applicatie getest zal worden aan de hand van verschillende userstories. Ook heb ik alvast een testtabel gemaakt waarin later de resultaten ingevuld kunnen worden.  De week verliep goed, al merkte ik dat het lastig was om vooruit te plannen zonder al een werkende applicatie te hebben. Toch bood het schrijven van de usabilitytest een goed beeld van wat straks belangrijk zal zijn bij het testen van de eindversie.  Afbeelding met tekst, schermopname, Multimediasoftware, software  Door AI gegenereerde inhoud is mogelijk onjuist.  **Week 6**  Deze week lag de focus op het afronden van de documentatie en het schrijven van de hoofdstukken over zelfstandige werkwijze en persoonlijke reflectie. Omdat dit project individueel werd uitgevoerd, ging het vooral om het beschrijven van mijn eigen manier van werken, mijn leerproces en hoe ik beslissingen heb genomen zonder directe begeleiding van een team.  Ik heb de structuur van het verslag verbeterd, figuren gecontroleerd en gezorgd dat alle onderdelen consistent en logisch in elkaar overlopen. Daarnaast ben ik gestart met het opstellen van het VerEvaRef, waarin ik mijn groei als student en mijn persoonlijke leerdoelen verwerk. Dit hielp om terug te kijken op het hele proces en inzicht te krijgen in waar ik mezelf nog verder kan ontwikkelen.  De planning liep iets uit doordat ik langer bezig was met de opmaak van het verslag dan verwacht. Toch heb ik aan het eind van de week alles goed kunnen afronden en voorbereid voor de laatste twee weken, waarin ik de focus wil leggen op de technische uitwerking en afronding van het project.  f  **Week 7**  In week 7 ben ik begonnen met het technische gedeelte van het project. De eerste stap was het koppelen van de applicatie aan het TTN-account, zodat er een verbinding gelegd kon worden tussen de sensoren en de database. Na wat aanpassingen aan de instellingen is de koppeling uiteindelijk goed gelukt, waardoor de data van de sensoren in de cloudomgeving kon worden ontvangen.  Daarna heb ik de databaseconnectie opgezet en getest. Hierbij was het belangrijk dat de tabellen goed aansloten op de structuur van het ERD dat ik eerder had gemaakt. De eerste tests met de webhook-data verliepen niet helemaal soepel — de payload kwam niet altijd volledig door — maar na enkele aanpassingen in het testscript werkte het stabiel.  Tot slot heb ik een begin gemaakt met de opbouw van de basisfrontend, zodat de data op een eenvoudige manier kan worden weergegeven. Het was prettig om na weken documentatie eindelijk weer te werken aan iets technisch. Ondanks wat kleine obstakels voelde deze week als een belangrijke stap vooruit richting een complete applicatie. |
| **Hoe** | **Proces: leer- en regulatieactiviteit** |
| Affectie – *activiteit 1* | Verwachting kan worden gezien als een vooropgezette hoop of voorspelling over hoe iets zal verlopen.  **Hoe dit terugkwam in dit project:** Aan het begin van het project had ik de verwachting dat het ontwikkelen van de Klimaatapp overzichtelijk zou verlopen, omdat het concept op papier vrij eenvoudig leek. Toch wist ik ook dat het alleen werken aan een project altijd meer tijd kost dan je denkt. Ik verwachtte dat ik de meeste tijd kwijt zou zijn aan het koppelen van de data via The Things Network, en dat bleek uiteindelijk te kloppen. Doordat ik zelf verantwoordelijk was voor elke stap, merkte ik hoe belangrijk het is om realistische verwachtingen te hebben over wat haalbaar is binnen de beschikbare tijd.  **Week 8**  In de laatste week lag de nadruk volledig op afronding en controle. Ik heb alle onderdelen van het BP06-document samengevoegd en gecontroleerd op consistentie, bronverwijzingen en opmaak. Daarbij heb ik kleine aanpassingen gedaan aan de VerEvaRef en de Zelfstandige Werkwijze om alles beter op elkaar aan te laten sluiten.  Daarnaast heb ik de werking van de TTN-koppeling en de database nog één keer getest. De sensordata kwam stabiel binnen en kon correct worden weergegeven binnen de basisfrontend. De app was hierdoor technisch functioneel genoeg om het concept te presenteren  .  Tot slot heb ik de presentatie voorbereid. Ik heb een overzicht gemaakt van het hele ontwikkelproces van de eerste schetsen tot de technische realisatie en nagedacht over hoe ik dit op een duidelijke manier kan toelichten tijdens de eindpresentatie. Hoewel de week wat stressvol was door de hoeveelheid kleine details, gaf het een goed gevoel om het project volledig afgerond te zien. |
| Affectie – *activiteit 2* | Inspanning is de hoeveelheid energie en tijd die wordt besteed om een bepaald doel te bereiken.  **Hoe dit terugkwam in dit project:** Omdat ik dit project naast mijn werk en verhuizing uitvoerde, moest ik mijn tijd goed verdelen. De momenten waarop ik aan de app werkte, waren vaak in de avonduren of in het weekend. Ondanks dat het soms lastig was om de motivatie vast te houden, bleef ik doorwerken aan kleine stukjes per keer. Zo merkte ik dat constante, kleine inspanning uiteindelijk meer oplevert dan lange, intensieve sessies vlak voor de deadline. Het zelfstandig plannen en afronden van taken heeft mijn discipline en doorzettingsvermogen zichtbaar versterkt. |
| Cognitie – *activiteit 1* | Analyseren  Analyseren is het grondig onderzoeken en ontleden van iets om inzicht te krijgen.  **Hoe dit terugkwam in dit project:** Tijdens het ontwikkelen van de Klimaatapp analyseerde ik continu hoe de sensordata werd doorgestuurd en verwerkt. Als er iets misging met de verbinding tussen de sensor en de database, onderzocht ik stap voor stap waar het probleem lag. Ook keek ik kritisch naar mijn datamodel: kloppen de relaties tussen de tabellen en is de structuur logisch opgebouwd? Door dit soort analyses leerde ik beter begrijpen hoe de verschillende onderdelen van de applicatie met elkaar samenhangen. |
| Cognitie – *activiteit 2* | Selecteren is het kiezen van de beste of meest geschikte opties uit verschillende mogelijkheden.  **Hoe dit terugkwam in dit project:** Bij het ontwikkelen van de Klimaatapp moest ik meerdere keuzes maken tussen technologieën en methodes. Zo koos ik bewust voor het gebruik van **webhooks** in plaats van **MQTT**, omdat dit eenvoudiger te beheren was binnen de scope van het project. Ook selecteerde ik **PostgreSQL** als database, vanwege de betrouwbaarheid en cloudondersteuning. Door zelfstandig keuzes te maken, leerde ik beter afwegen wat echt nodig is voor de basis van een goed werkende applicatie. |
| Regulatie – *activiteit 1* | Plannen is het indelen van tijd in de toekomst voor een bepaalde werkzaamheid die je moet voltooien. Vaak zit er ook een deadline aan deze werkzaamheid.  **Hoe dit terugkwam in dit project:** Ik werkte met een aangepaste vorm van de **Agile Scrum-methode**, waarbij ik zelf wekelijks een kleine sprint opzette. Aan het begin van de week maakte ik een Trello-overzicht met de taken die ik wilde afronden. Soms liep dat plan anders door drukte op mijn werk, maar ik probeerde dan de week erna de focus terug te brengen. Door deze aanpak hield ik overzicht, ook al schoof ik taken soms wat op. Deze manier van werken paste goed bij mijn situatie, omdat het flexibel genoeg was om te combineren met andere verplichtingen. |
| Regulatie – *activiteit 2* | Toetsen kan gezien worden als het testen van iets en er dan feedback op geven.  **Hoe dit terugkwam in dit project:** Tijdens het project heb ik mijn applicatie regelmatig getest met dummydata om te controleren of de sensoren goed reageerden en of de databasewaarden klopten. Ook liet ik af en toe iemand meekijken om te zien of de interface logisch overkwam. Hierdoor ontdekte ik kleine verbeterpunten, zoals het duidelijker tonen van verouderde data of het aanpassen van de volgorde van de velden. Deze tests hielpen mij om het eindproduct betrouwbaarder te maken en gaven me meer vertrouwen in mijn eigen werk. |
| Verantwoorden | |
| Week 1  Week 2  Week 3  Week 4  Week 5  Week 6 | **Week 1 Beslismoment** In week 1 heb ik besloten om de focus van het project te leggen op de basisfunctionaliteit: het ophalen en weergeven van klimaatgegevens. Ik heb bewust niet direct met de vormgeving of extra functies begonnen, zodat ik eerst kon begrijpen hoe de data via The Things Network binnenkomt.  **Mogelijkheid** Ik had er ook voor kunnen kiezen om te starten met het ontwerpen van de interface, zodat ik sneller een visueel beeld had van de app. Dat zou motiverend zijn geweest, maar had waarschijnlijk tot vertraging geleid, omdat de technische basis nog niet stabiel genoeg was.  **Week 2 Beslismoment** In week 2 koos ik ervoor om de sensordata via een webhook te laten binnenkomen in plaats van met MQTT. Deze aanpak was makkelijker te testen en beter te combineren met een simpele backend.  **Mogelijkheid** Ik had kunnen kiezen voor MQTT, wat iets sneller werkt en vaker wordt gebruikt in IoT-projecten. Maar dat zou meer tijd kosten om goed op te zetten en te onderhouden. Voor dit project vond ik eenvoud belangrijker dan snelheid.  **Week 3 Beslismoment** In week 3 besloot ik om het datamodel klein te houden. Ik maakte slechts drie tabellen: room, sensor en measurement.Daarmee kon ik alle benodigde gegevens opslaan zonder dat de database onnodig ingewikkeld werd**.**  **Mogelijkheid** Ik had de data kunnen opsplitsen in meer tabellen, bijvoorbeeld voor type metingen of historische data. Dat zou later meer flexibiliteit geven, maar voor dit project was dat overbodig en had het vooral extra werk opgeleverd.  **Week 4 Beslismoment** In week 4 koos ik ervoor om de app minimalistisch te houden, met één dashboard dat alle kamers toont. Zo bleef de structuur overzichtelijk en kon ik sneller controleren of de data juist werd weergegeven.  **Mogelijkheid** Ik had een uitgebreidere app kunnen maken, met meer schermen en functies per sensor. Dat zou er indrukwekkender uitzien, maar had weinig extra waarde toegevoegd voor dit project. Door het simpel te houden bleef het overzicht behouden.  **Week 5 Beslismoment** In week 5 heb ik besloten om te werken met PostgreSQL als database. Dit leek mij betrouwbaarder dan SQLite en gaf me de mogelijkheid om de app later online te hosten.  **Mogelijkheid** Ik had SQLite kunnen gebruiken omdat dat sneller op te zetten is en geen aparte server nodig heeft. Toch koos ik voor PostgreSQL, omdat het beter aansluit bij hoe databases in de praktijk werken en ik er op die manier meer van leer.  **Week 6 Beslismoment** In week 6 koos ik ervoor om het front-end ontwerp in Visily te maken, omdat het snel resultaten oplevert en ik niet alles handmatig hoef te coderen. Dit hielp om het schermontwerp duidelijk te visualiseren.  **Mogelijkheid** Ik had dit ook volledig in Figma kunnen ontwerpen, maar dat zou meer tijd kosten en minder goed aansluiten bij mijn behoefte aan snelheid. Visily gaf me net genoeg mogelijkheden om de lay-out simpel, maar effectief te houden.  **Week 7 Beslismoment** In week 7 besloot ik om extra tijd te besteden aan het netjes afronden van de documentatie. Ik merkte dat er veel losse onderdelen waren, zoals diagrammen, tabellen en toelichtingen, die ik beter kon ordenen.  **Mogelijkheid** Ik had ervoor kunnen kiezen om die tijd te gebruiken om nog nieuwe functies toe te voegen aan de app. Toch koos ik bewust voor documentatie, omdat dat het geheel professioneler en begrijpelijker maakt.  **Week 8 Beslismoment** In week 8 koos ik ervoor om alles samen te voegen in één duidelijk verslag, inclusief diagrammen, toelichtingen en screenshots. Hierdoor kon ik het project als één geheel presenteren in plaats van losse onderdelen.  **Mogelijkheid** Ik had de onderdelen ook apart kunnen inleveren, wat sneller was geweest. Maar door alles te bundelen in één consistent document werd het verslag overzichtelijker en voelde het project afgerond. |
| Reflecteren | |
|  | **Onwenselijke overtuiging** |
|  | **Onwenselijke overtuiging: Moeite met hulp en feedback vragen**  **Gebeurtenis** Tijdens het werken aan projecten merk ik dat ik vaak alles zelf wil oplossen. In plaats van hulp te vragen, blijf ik zoeken naar mijn eigen oplossingen, ook als dat meer tijd kost. Ik stel het vragen van feedback vaak uit, omdat ik bang ben dat ik iets niet goed genoeg heb gedaan of dat ik onnodige kritiek krijg. Hierdoor mis ik soms kansen om sneller te verbeteren.  **Vroegste herinnering** Sinds mijn middelbare schooltijd heb ik dit gedrag al. Omdat ik enig kind ben, heb ik veel dingen altijd zelf moeten doen. Ik leerde om problemen zelfstandig op te lossen en raakte gewend aan het idee dat ik niemand nodig had om iets af te krijgen. Dat werkte vaak goed, maar het maakte het ook moeilijker om te leren samenwerken of om feedback echt te omarmen.  **Patroon** Het is geen bewuste keuze, maar een automatische reactie. Zodra ik vastloop, probeer ik het eerst zelf op te lossen in plaats van iemand te benaderen. Ik denk dan: “Ik vind het wel uit.” Dit zorgt ervoor dat ik soms te lang blijf hangen in kleine problemen die met één simpele vraag opgelost hadden kunnen worden. Het is een patroon dat terugkomt in bijna elk project dat ik doe.  **Trigger** Wanneer ik merk dat iets nog niet helemaal goed werkt, voel ik direct de drang om het zelf uit te zoeken. Ik wil het liefst eerst alles begrijpen voordat ik iemand anders erbij betrek. Vaak denk ik dat ik eerst iets moet verbeteren voordat ik het kan laten zien. Daardoor komt het moment van feedback vragen pas heel laat, of soms helemaal niet.  **Onwenselijk gedrag** Doordat ik zelden feedback vraag, duurt het vaak langer voordat ik vooruitgang boek. Ook twijfel ik vaker aan mijn eigen werk, omdat ik geen bevestiging of andere blik krijg. Soms voel ik me daardoor gestrest of onzeker, vooral als de deadline dichterbij komt. Het idee dat ik alles alleen moet kunnen, maakt het proces zwaarder dan nodig is.  **Verbetering**   1. Als student wil ik leren om eerder feedback te vragen, zodanig dat ik tijdig inzicht krijg in de kwaliteit van mijn werk. Dit voorkomt dat ik onnodig veel tijd steek in details die later aangepast moeten worden. 2. Als student wil ik mijn zelfstandigheid blijven gebruiken als kracht, maar combineren met open communicatie. Door kleine vragen te stellen of feedback te delen, kan ik sneller leren zonder mijn zelfstandigheid te verliezen. 3. Als student wil ik oefenen met het accepteren van feedback als iets positiefs. Door het te zien als onderdeel van groei in plaats van kritiek, kan ik met minder spanning mijn werk laten beoordelen.   Door deze stappen te volgen, hoop ik een betere balans te vinden tussen zelfstandig werken en het durven betrekken van anderen in mijn leerproces. Zo kan ik mijn werk effectiever uitvoeren en mijn zelfvertrouwen verder ontwikkelen. |
|  | **Wenselijke overtuiging** |
|  | **Wenselijke Overtuiging 1: Ik wil mijn professionele netwerk blijven uitbreiden**  Tijdens mijn studie heb ik geleerd hoe belangrijk het is om contact te leggen met mensen binnen en buiten de opleiding. Nu ik richting het werkveld ga, besef ik dat een sterk netwerk niet alleen kansen oplevert, maar ook helpt om te blijven leren. Door actief in contact te blijven met oud-docenten, medestudenten en professionals uit de branche, kan ik mijn kennis blijven verbreden. In de toekomst wil ik dit blijven doen door gesprekken aan te gaan, evenementen bij te wonen en ervaringen te delen met anderen. Zo bouw ik niet alleen aan een netwerk, maar ook aan een kring van mensen die elkaar inspireren en verder helpen.  **Wenselijke Overtuiging 2: Ik wil feedback zien als een onderdeel van groei**  In de afgelopen periode heb ik geleerd dat feedback vragen en ontvangen essentieel is voor persoonlijke ontwikkeling. Waar ik eerder geneigd was om alles zelf te willen doen, wil ik nu bewust feedback blijven gebruiken om te groeien. In mijn toekomstige werk wil ik regelmatig de mening van collega’s of leidinggevenden vragen, juist ook als iets nog niet perfect is. Door feedback te zien als een normaal en positief onderdeel van het leerproces, kan ik sneller verbeteren en meer vertrouwen krijgen in mijn werk. Ik wil leren om feedback niet te vermijden, maar het te gebruiken als richtingaanwijzer om mijn vaardigheden te blijven aanscherpen.  **Wenselijke Overtuiging 3: Ik wil een gezonde balans houden tussen werk en groei**  Tijdens mijn opleiding heb ik gemerkt dat ik soms te lang blijf doorwerken aan projecten, uit perfectionisme of omdat ik iets volledig wil begrijpen. In de toekomst wil ik daar bewuster mee omgaan. Ik wil leren om prioriteiten te stellen, grenzen te trekken en ruimte te laten voor rust en reflectie. Door een goede balans te vinden tussen werken, leren en ontspannen, blijf ik gemotiveerd en scherp. Dit helpt me niet alleen om beter werk te leveren, maar ook om met meer plezier en energie aan nieuwe uitdagingen te beginnen. |
|  | **Persoonlijke doelen** |
|  | **1. User story:**  Als student in blok 6 wil ik mijn zelfstandige werkhouding verder ontwikkelen, zodanig dat ik complexe projecten — zoals de Klimaatapp — volledig zelfstandig kan analyseren, ontwerpen en documenteren. Ik wil hierbij laten zien dat ik gestructureerd kan werken zonder constante begeleiding, en dat ik in staat ben om eigen keuzes te onderbouwen met logische argumenten en technisch inzicht.  **2. User story:**  Als student in blok 6 wil ik mijn vaardigheden in documentatie en verslaglegging verbeteren, zodanig dat mijn projectverslagen niet alleen technisch correct zijn, maar ook helder, consistent en prettig leesbaar. Ik wil leren om complexe technische keuzes op een toegankelijke manier uit te leggen, zodat ook niet-technische lezers mijn werk goed kunnen begrijpen.  **3. User story:**  Als student in blok 6 wil ik mijn vaardigheden in front-endontwikkeling verder aanscherpen, zodanig dat ik een gebruiksvriendelijke en overzichtelijke webapplicatie kan opleveren. Ik wil hierbij vooral focussen op de balans tussen eenvoud en functionaliteit — zodat de applicatie niet alleen goed werkt, maar ook logisch aanvoelt voor de gebruiker. |

# Bibliografie

*Technische bronnen*

1. Chart.js. (2024). Simple yet flexible JavaScript charting library. Retrieved from <https://www.chartjs.org/docs/latest/>
2. Express.js. (2024). Fast, unopinionated, minimalist web framework for Node.js. Retrieved from <https://expressjs.com/>
3. GitHub Docs. (2025). Version control and collaboration. Retrieved from <https://docs.github.com/>
4. MDN Web Docs. (2025). JavaScript reference and tutorials. Retrieved from <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
5. MySQL. (2024). MySQL 8.0 reference manual. Retrieved from <https://dev.mysql.com/doc/refman/8.0/en/>
6. Node.js. (2024). Node.js documentation. Retrieved from <https://nodejs.org/en/docs/>
7. npm. (2024). npm documentation. Retrieved from <https://docs.npmjs.com/>
8. Railway. (2024). Railway documentation: Deploy and host applications. Retrieved from <https://docs.railway.app/>
9. Stack Overflow. (2025). Community discussions about TTN webhooks and data mapping. Retrieved from <https://stackoverflow.com/>
10. The Things Network. (2024). Getting started with webhooks. Retrieved from <https://www.thethingsnetwork.org/docs/integrations/webhooks/>
11. The Things Network. (2024). Payload formatters and decoders. Retrieved from <https://www.thethingsnetwork.org/docs/devices/payload-formatters/>
12. W3Schools. (2025). HTML, CSS, and JavaScript tutorials. Retrieved from <https://www.w3schools.com/>
13. W3Schools. (2024). CSS Flexbox and responsive layouts. Retrieved from <https://www.w3schools.com/css/css3_flexbox.asp>

*Ontwerp en usability bronnen*

1. Nielsen Norman Group. (2023). 10 usability heuristics for user interface design. Retrieved from <https://www.nngroup.com/articles/ten-usability-heuristics/>
2. Smashing Magazine. (2023). Accessible web design: Best practices for developers. Retrieved from <https://www.smashingmagazine.com/>
3. Visily. (2024). Create wireframes and prototypes easily. Retrieved from <https://www.visily.ai/>
4. W3C. (2023). Web accessibility initiative (WAI) guidelines. Retrieved from <https://www.w3.org/WAI/>

*Reflectie en persoonlijke ontwikkeling bronnen*

1. CareerFoundry. (2024). How to give and receive feedback effectively. Retrieved from <https://careerfoundry.com/>
2. Coursera. (2024). Self-regulated learning: How to learn more effectively. Retrieved from <https://www.coursera.org/>
3. Education Corner. (2023). What are affective, cognitive, and psychomotor learning domains? Retrieved from <https://www.educationcorner.com/>
4. Indeed. (2024). How to practice critical thinking skills at work. Retrieved from <https://www.indeed.com/>
5. MindTools. (2024). Gibbs reflective cycle explained. Retrieved from <https://www.mindtools.com/>
6. MindTools. (2024). Kolb's experiential learning cycle. Retrieved from <https://www.mindtools.com/>