PROJECT COOLING DOWN THE CITIES

KWALITEITSEISEN DOCUMENT

INHOUD

1.0 TEAMLEDEN & ROLLEN

NAAM	ROL
MARTIN BORSJE	Product Owner (Stakeholder)
MOZES HAK	Smart Engineer
BILAL ACHRIFI	Software Developer, Databasebeheer (Microsoft Azure)
ENES TEKINBAS	Software Developer
RIAD ALHAKIM	Software Developer, Versiebeheer/Databasebeheer (Raspberry Pi), Versiebeheer (Microsoft Azure)
ROBIN VERVOORN	Scrummaster, Software Developer, Notulist

2.0 APPLICATIE

2.1 DOEL

DL-01 Het doel van de applicatie is om gegevens van diverse kleine weerstations te monitoren en te vergelijken

DL-02 In de applicatie zijn de volgende informatievoorzieningen belangrijk:

a. Gebruikersinformatie (Gebruikersnaam, Email, Wachtwoord.)
b. Weergegevens (Luchtvochtigheid, Koolstofdioxide, Temperatuur, Luchtkwaliteit, Luchtdruk en Vluchtig Organische Stoffen)
c. Vergelijking van gegevens (Uren, Dagen, Maanden, Jaren)

2.2 AARD

AA-01: De applicatie is een dashboard om data van weerstations te tonen en vergelijken
AA-02: Er zijn 5 andere scholen in Europa die een soort gelijke applicatie aan het maken zijn.
AA-03: Een webapplicatie dat hoogstwaarschijnlijk op deze applicatie zal lijken, is Buienradar? Hierbij kan men denken aan het opvragen van weeromstandigheden.

2.3 DOELGROEPEN

Omdat meerdere scholen aan hun variant bezig zijn, moeten mensen van die verschillende scholen erin kunnen komen om van elkaar te kunnen leren van hoe wij ons project hebben opgezet.

Natuurlijk is het ook de bedoeling om het klimaat van waar de kleine weerstations zich bevinden, te achterhalen en monitoren en ook om vergelijkingen te kunnen maken.

3.0 AGILE/SCRUM

3.1 VERANTWOORDELIJKHEDEN

Elke teamlid heeft zijn/haar eigen rollen waaronder verantwoordelijkheden die moeten nageleefd worden.

VER-01: Scrum Master: Is verantwoordelijk voor het op correcte wijze uitvoeren van de Scrum aanpak. Dit wordt gedaan door ervoor te zorgen dat het hele team de Scrum theorie, werkwijzen en regels naleeft. De Scrum Master communiceert met de Product Owner. Tevens heeft de Scrum Master op het moment ook de leiding over het versie deployment.

VER-02: Software Developers: Nemen op eigen wijze de ontwikkeling van de applicatie in handen.

VER-03: Databasebeheerder: Heeft eigenaarsrechten tot de Database Server.

VER-04: Versiebeheerder: Heeft eigenaarsrechten tot de Live omgeving.

VER-05: Repository-beheerder: Heeft eigenaarsrechten tot de GitHub-Repository.

VER-06: Smart Engineer: Maakt het fysieke weerstation met de benodigde sensors, onderdelen.

VER-07: Notulist: Houd belangrijke documentatie up-to-date waaronder Project Initiatie Document (PID), iteratiedocumentaties, Samenwerkingscontract, Tijdlijnen en eventueel verdere notities tijdens overleggen/gespreken.

3.2 DEFINITION OF DONE

Er moet altijd een akkoord zijn over wanneer een feature echt klaar is. Hiervoor gebruiken we de Definition of Done (DoD). De userstory is pas voltooid indien het aan de volgende criteria heeft voldaan:

DoD-1: Code is beoordeeld door andere programmeur op functionaliteit. (Alle must-have's zijn geïmplementeerd, code is gedocumenteerd met eventuele context, code voldoet aan layout/coding standards.)

DoD-2: De feature/bugfix is getest door minimaal één ander persoon.

- Teststappen worden beschreven
- Test wordt uitgevoerd
- Resultaat van test is beschreven

DoD-3: De code is gemerged via een pull-request in de development branch.

DoD-4: De feature/bugfix is geaccepteerd door Product Owner zonder verdere aanpassingen.

DoD-5: De feature/bugfix wordt gemerged in de main branch.

4.0 ARCHITECTUUR EN SOFTWAREPATRONEN

4.1 MVC

MVC staat voor **Model-View-Controller**. Het is een architectuur of een softwareontwerppatroon dat het maken van enorme applicaties eenvoudig maakt. Het behoort niet tot een specifieke programmeertaal of raamwerk, maar het is een concept dat je kunt gebruiken bij het maken van elke soort applicatie of software in elke programmeertaal.

Als u bijvoorbeeld een applicatie in Python ontwikkelt, kun je frameworks zoals Django of CherryPy gebruiken die MVC-architectuur gebruiken om je te helpen applicaties snel en eenvoudig te ontwikkelen.

4.1.1 VOORDELEN

a. De ontwikkeling van de applicatie gaat snel.
b. Makkelijk voor meerdere ontwikkelaars om samen te werken.
c. Makkelijker om de applicatie bij te werken.
d. Gemakkelijker te debuggen omdat er meerdere niveaus zijn binnen de applicatie.

4.1.2 NADELEN

a. In het begin is het vrij moeilijk om de MVC structuur te begrijpen.
b. Niet geschikt voor kleinere applicaties omdat het negatieve effecten kan hebben in de prestaties van de applicatie en het ontwerp ervan.

4.2 SPA

Ook wel een Single Page Application genoemd, is een webapplicatie of website die past op een enkele webpagina met het doel een vlotte gebruikerservaring te bieden. In een SPA wordt alle benodigde code - HTML, JavaScript en CSS - opgehaald met een enkele laadactie van de pagina, of de noodzakelijke middelen worden dynamisch geladen en aan de pagina toegevoegd zodra ze nodig zijn, meestal als reactie op acties van de gebruiker. De pagina wordt op geen enkel moment in het proces opnieuw geladen.

4.2.1 VOORDELEN

a. Reactief (illusie -> geen pagina laden).
b. Goede laadsnelheden tijdens gebruik (dit komt doordat alle content al van tevoren ingeladen is).
c. Goede gebruikersevaring.
d. Front-end en back-end zijn goed van elkaar gescheiden.

4.2.2 NADELEN

a. SEO blijft een uitdaging.
b. Beveiliging wordt bemoeilijkt (end points beveiligen i.p.v. individuele pagina’s).
c. Oudere browsers kunnen SPA’s vaak minder goed draaien.
d. Laden voor gebruik kan langzamer zijn (dit kan problemen zijn voor slechte internet verbindingen of oudere apparaten)

4.3 MPA

Ook wel een **Multi Page Application** genoemd. Deze maakt gebruik meerdere pagina's om de front-end met de back-end te laten communiceren. Voor elk request worden de HTML, CSS en scripts van een pagina opnieuw opgehaald en ingeladen.

4.3.1 VOORDELEN

a. Goede SEO mogelijkheden.
b. Eenvoudiger te doorgronden door natuurlijke structuur van pagina's.
c. Makkelijker te beveiligen door meerdere structuur lagen.
d. Beter beschikbaar voor verouderde apparaten en browsers.
e. Beter geschikt voor websites die weinig gebruikersactie en real-time updates vereisen.

4.3.2 NADELEN

a. Grote applicaties kunnen trager werken door het constant compleet herladen van pagina's.
b. MPA's duren langer te ontwikkelen dan SPA's vanwege complexiteit.
c. Server belading is generiek meer dan een SPA.

4.4 KEUZE ARCHITECTUUR EN SOFTWAREPATRONEN

Er is gekozen voor een Multi Page Application (MPA).

Een SPA is erg reactief. In die zin zou je kunnen zeggen dat het lijkt op een downloadbare mobiele of desktop app. Het reageert gelijk als je op iets klikt en het geeft je de illusie dat het geen enkele pagina opnieuw hoeft te laden. Laadsnelheden en gebruikersevaring zijn erg goed. Een MPA daarentegen kan trager werken, door het continu opnieuw laden van de pagina, maar vanwege het applicatie type is MPA meer geschikt voor deze applicatie.

Deze applicatie heeft weinig gebruikersinteractie nodig en real-time updates zijn niet vereist op de front-end vanwege de frequentie van updates (dit gebeurt ook al op de back-end). Daarnaast is er veel meer beveiliging mogelijkheden doordat de applicatie is opgedeeld in meerdere delen en lagen. Door deze meerdere delen en lagen zou het ook gemakkelijker zijn om aanpassingen aan te brengen.

5.0 FRAMEWORKS EN LIBRARIES

5.1 DJANGO

Django is een framework dat een snelle ontwikkeling van veilige en onderhoudbare websites mogelijk maakt. Het is het Python-framework/library gebouwd door ervaren developers. Waarom Django gebruiken? Het maakt de ingewikkeldheid van web development weg, zodat er gefocust kan worden op het schrijven van de applicatie zonder dat de technologie van web development met Python opnieuw te hoeven ontwikkelen.

5.1.1 VOORDELEN

a. Het maakt het proces van concept-tot-compleet sneller en effectiever.
b. Het is veilig en voorkomt veelgemaakte security fouten.
c. Het is gemaakt in Python wat op zichzelf al snel is en makkelijk leesbaar.
d. Django's design is DRY. (Do not Repeat Yourself)
e. Het is schaalbaar waardoor het elke soort hardware toevoegingen aankan

5.1.2 NADELEN

a. Django is monolithisch en kennis vooraf is vereist om ermee te werken

5.2 FLASK

Flask is een micro-webframework. "Micro" omdat Flask geen specifieke tools of bibliotheken vereist. Het staat bekend om zijn eenvoudige en minimalistische aanpak, wat een aantrekkelijke keuze is voor developers. Van 2018 tot 2021 was Flask gekozen als de beste framework onder Python developers.

5.2.1 VOORDELEN

a. Eenvoudig
b. Flexibiliteit (Amper beperkingen, etc)
c. Populaire keuze voor het maken van API's

5.2.2 NADELEN

a. Te veel flexibiliteit leidt tot inconsistente code en architectuur
b. Meer handmatig werk

5.3 JINJA

Jinja is een Python templating library die gebruikt kan worden met Django, Flask en Ansible frameworks om dynamische content toe te voegen aan webpagina's. Dit maakt het mogelijk om op de correcte manier data te tonen in een webpagina.

5.3.1 VOORDELEN

a. Het is een asynchrone proces waardoor grote datasets makkelijk en snel ingeladen kunnen worden.
b. Het is flexibel in hoe het gebruikt kan worden.
c. Het proces gaat direct door Python bytecode heen wat het sneller maakt.

5.3.2 NADELEN

a. Er zijn weinig packages beschikbaar, omdat meeste packages gebaseerd zijn op Django templating in plaats van Jinja templating

5.4 KEUZE FRAMEWORKS EN LIBRARIES

Er is gekozen voor Django met een MPA structuur.

Dankzij onze ervaring in Python, zal het voor ons als team gemakkelijk zijn om in Django aan het werk te gaan. Django is een minimalistisch, eenvoudig en door ervaren ontwikkelaars gecreëerde framework, dus we kunnen erop vertrouwen dat Django alle concepten van web development omvat.

Django heeft naast de flexibiliteit en efficiëntie, ook nog eens een grote community met allerlei ervaren ontwikkelaars en talloze bronnen die ondersteuning bieden in de vorm van tutorial, guides, etc.

5.5 HOOFCOMPONENTEN

De hoofdcomponenten zijn:

HC-01: Client

HC-02: Server

HC-03: Database

HC-04: Raspberry PI

5.6 SYSTEEMOMGEVING

SO-01: Django 4.2

SO-02: Microsoft Azure mysql database

SO-03: Microsoft Azure Live server

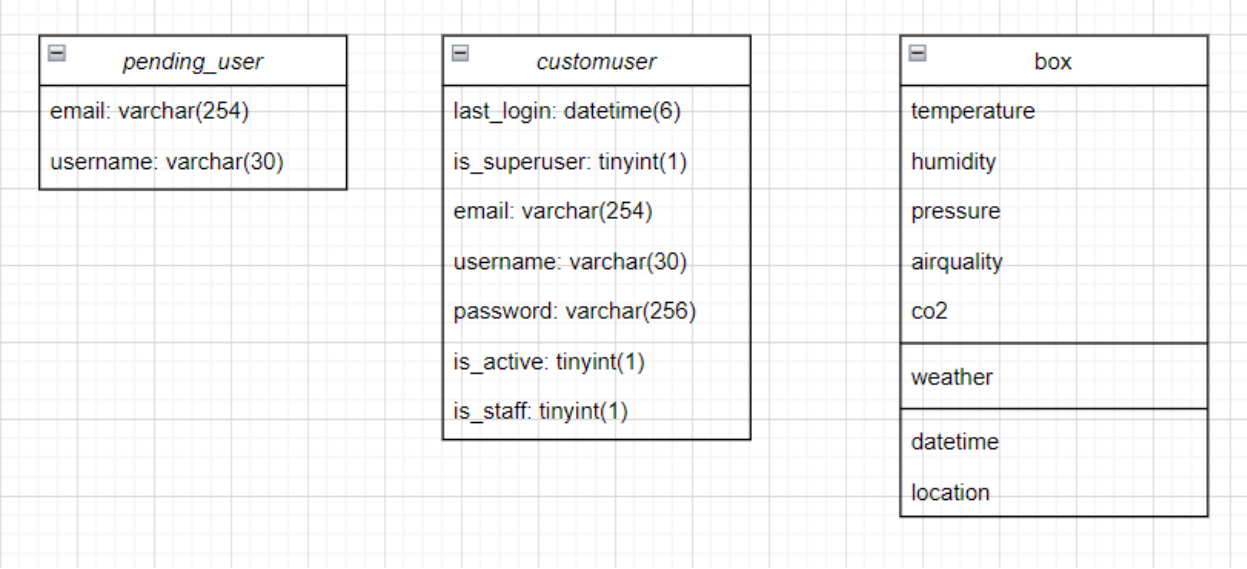
SO-04: Raspberry PI met database en live server

SO-05: Libraries die te vinden zijn in de requirements.txt

SO-06: .env bestand voor lokale omgeving gebruik

6.0 DATABASE

6.1 LOGISCHE DATASTRUCTUUR **(KAN NOG VERANDEREN!!!)**



6.2 GEKOZEN IMPLEMENTATIE EN ONDERBOUWING **(KAN NOG VERANDEREN!!!)**

De huidige database structuur is nog niet permanent en zal nog regelmatig updates krijgen!

7.0 CODING STANDARDS

Op 4-12-2023 zijn er Coding Standards gemaakt/geupdatet. Deze paragraaf gaat over de vastgelegde regels.

CS-01: **Alleen ingewikkelde code wordt gedocumenteerd met blok commentaar.** Zo kun je denken aan blok commentaar voor het beschrijven van functions en/of methods.

CS-02: **Inline-commentaar wordt alléén gebruikt voor reminders en verheldering van geschreven code.**

CS-03: **Variabele- en functie namen zijn altijd in het Engels.**

CS-04: **Gebruik van juiste tab-afstanden is vereist.** Bij iedere element/function/etc. is een tab inspringen verplicht. Een tab is 4 spaties!!!

CS-05: **Constanten altijd in UPPERCASE**

CS-06: **Classnamen in PascalCase**

CS-07: **Variabele en function namen in camelCase.**

CS-08: **Logisch gebruik van variabelen en constanten.**

CS-09: **Gebruik van DRY (Don't repeat yourself) principe.**

CS-10: **Gebruik van Single Responsibility principe op o.a. functions.**

CS-11: **Code moet herbruikbaar zijn wanneer mogelijk. (code reusability)**

CS-12: **Code wordt zo simpel en klein mogelijk gehouden.** Regelmatig refactoren a.d.v. feedback code reviews en eigen input na schrijven van code.

CS-13: **Objectnamen altijd in camelCase.**

7.1 BRANCHING & COMMITS

7.1.1 BRANCH TYPES

Er zijn diverse branches waar op gecommit kan worden.

BR-01: De **main branch** waar (voorlopige) releases op komen.

BR-02: De **dev branch** waar voltooide features samengevoegd worden en getest worden door andere developers.

BR-03: De **Individuele branches** waar features worden gebouwd voordat deze op de **dev branch** worden samengevoegd.

7.1.2 BRANCH STANDARDS

BR-04: **Individuele branches** mogen pas verwijderd worden nadat alles in de betreffende branch gereviewd is, getest is en de desbetreffende sprint afgelopen is.

BR-05: Pas aan het einde van een sprint worden alle commits op de **dev branch** samengevoegd met de **main branch** als een heel pakket. Dit is met uitzondering van Hotfixes.

BR-06: **Individuele branch** namen moeten verwijzen naar de betreffende feature in het Trello-Board.

BR-07: **Individuele branches** mogen nooit direct samengevoegd worden met de **main branch**.

BR-08: Alleen met speciale uitzondering mogen commits direct worden toegevoegd in **main branch** in plaats van de gewoonlijke route van **dev branch** naar **main branch.**

7.1.3 COMMITS

CO-01: Alle commits zijn voorzien van een korte beschrijving/samenvatting van de code veranderingen.

CO-02: Alle commits zijn voorzien van een modulenaam om te refereren voor welke module de commit is gericht.

CO-03: Per merge-commit wordt er een korte samenvatting beschreven van alle changes die er worden gemerged.

7.2 OVERIGE STANDARDS

7.2.1 STYLE CSS CONVENTIES

SC-01: Het hoofdcomponent blijft genaamd ‘app.css’ en blijft direct in de css-map.

SC-02: De submappen voor de css-map zijn:

a. abstracts (met variables, functions etc.)
b. base (met animations, base, typography, etc.)
c. components (met buttons, images etc.)
d. layout (met header, footer etc.)
e. pages (overige css voor pagina’s)

7.2.2 OVERIGE STANDARDS

OS-01: Iedere html pagina moet alleen benaderd kunnen worden op groot scherm. Verdere formaten zijn later pas van toepassing

8.0 EISEN USER INTERFACE

8.1 LOOK & FEEL

De klant wilt dat de app er overzichtelijk uitziet. Dit betekent dat hij de layout wilt net zoals andere weervergelijking websites en apps. Voor stijlvoorbeelden zie de websites en apps van andere weerstations. Volgens de klant is er geen voorkeur voor stijl en kleurenstijl. Deze keuze is overgelaten bij de ontwikkelaars.

9.0 ORGANISATIE

9.1 ROLVERDELING

- **Scrum Master:** Robin Vervoorn
- **Databasebeheerder:** Bilal Achrifi (Microsoft Azure) / Riad Al Hakim (Raspberry Pi)

- **Versiebeheer:** Riad Al Hakim (Microsoft Azure) / Riad Al Hakim (Raspberry Pi)
- **Contactpersoon:** Robin Vervoorn
- **Notuleren:** Robin Vervoorn

9.2 BEREIKBAARHEID

• Robin Vervoorn	99066333@mydavinci.nl	06 42697468
• Bilal Achrifi	99066318@mydavinci.nl	06 13701123
• Enes Tekinbas	99040669@mydavinci.nl	06 28135555
• Riad Al Hakim	99054420@mydavinci.nl	06 20483236

9.3 WERKAFSPRAKEN

In principe zijn we altijd aanwezig bij de lessen. Buiten de lessen om is het een eigen keuze of je aanwezig/online bent of niet. Bij doktersafspraken e.d. op tijd (minimaal 24u) van tevoren doorgeven aan teamgenoten.

Bij afwezigheid kan er gecommuniceerd worden via Microsoft Teams / WhatsApp / Discord.

Indien er onvoldoende inzet wordt getoond tijdens het project of de regels in het onderlinge samenwerkingscontract niet worden nageleefd, dan volgt er een waarschuwing en daarna een strike. Bij drie strikes worden docenten eerst betrokken.