2024 - 2025

|  |
| --- |
|  |
| Water Quality management system |
| Realisatiedocument |

Jens Geenen  
Student Bachelor in de Toegepaste Informatica – Internet Of Things

Inhoudsopgave

[1. Inleiding 3](#_Toc200646411)

[2. Analyse 4](#_Toc200646412)

[3. Uitwerking van het project 5](#_Toc200646413)

[4. Besluit 9](#_Toc200646414)

[Literatuurlijst 10](#_Toc200646415)

# Inleiding

Dit realisatiedocument geeft een overzicht van de uitvoering van mijn stageproject, dat gericht was op de ontwikkeling van een waterkwaliteitsmonitoringsysteem. De opdracht bestond erin een systeem te bouwen waarin waterkwaliteitsdata ingevoerd en geanalyseerd kan worden, waarbij automatisch een verslag gegenereerd wordt dat vervolgens verzonden wordt naar de verantwoordelijke organisatie van het waterschap in de regio Alentejo, Portugal.

De aanpak van het project werd bepaald tijdens de eerste anderhalve maand van mijn stage, waarin de functionele vereisten en de technische keuzes vastgelegd werden. Drie verschillende gebruikersrollen stonden hierbij centraal, elk met toegang tot een eigen dashboard dat afgestemd was op hun taken en bevoegdheden.

De uiteindelijke oplossing werd ontwikkeld als een Progressive Web App (PWA), met Django als backendframework. Voor de automatische PDF-generatie werd gebruikgemaakt van ReportLab.

In dit document worden achtereenvolgens de analyse van de gebruikte tools en technologieën besproken, de concrete realisatie van het systeem toegelicht, en het project geëvalueerd op basis van het verloop en de behaalde resultaten.

# Analyse

Voor de uitvoering van dit project zijn de belangrijkste technologische beslissingen van tevoren vastgesteld door het stagebedrijf. De taak omvatte het creëren van een Progressive Web App (PWA) met het Django-framework voor de backend en ReportLab voor het automatische genereren van PDF-rapporten. Hierdoor was er weinig gelegenheid voor een grondige vergelijkende analyse of het toepassen van bijvoorbeeld de Weighted Ranking Methode.  
  
Toch zijn deze beslissingen goed te rechtvaardigen binnen de context van het project:

* Django is een robuust en geavanceerd Python-webframework dat perfect is voor het snel ontwikkelen van veilige en schaalbare webapplicaties. Het levert een geïntegreerd beheerpaneel, biedt uitstekende ondersteuning voor gebruikersauthenticatie en sluit goed aan bij de behoefte aan rolgebaseerde toegang voor de drie verschillende gebruikersgroepen binnen het systeem.
* ReportLab is een betrouwbare Python-bibliotheek die gebruikt kan worden voor het dynamisch creëren van PDF-documenten. ReportLab biedt een hoge mate van flexibiliteit en nauwkeurigheid voor het automatiseren van waterkwaliteitsrapporten, die zijn uitgerust met grafieken en gestructureerde gegevens.
* PWA-technologie stelt gebruikers in staat om een webapplicatie te gebruiken als een native app, met mogelijkheden voor offline gebruik en mobiele toegang, wat cruciaal is voor gebruikers in het veld of in afgelegen gebieden zoals Alentejo.

Hoewel er weinig mogelijkheden waren voor alternatieve oplossingen, is er in de eerste weken van het project wel tijd besteed aan het verkennen, installeren en testen van deze technologieën. Dit stelde ons in staat om te doorgronden op welke manier de gekozen tools konden worden samengevoegd tot een gebruiksvriendelijke en technisch betrouwbare oplossing.

# Uitwerking van het project

**3.1 Overzicht van de toepassing**

De applicatie die ik heb gebouwd is een **Progressive Web App (PWA)** waarmee gebruikers waterkwaliteitsmetingen kunnen invoeren, opvolgen en beheren. De app is gemaakt met het Django-framework en bestaat uit drie verschillende dashboards, afgestemd op drie soorten gebruikers: **Employee**, **Manager** en **IT Administrator**.

Na het inloggen wordt de gebruiker automatisch naar het juiste dashboard gebracht, afhankelijk van zijn of haar rol.

**3.2 Gebruikersrollen en functies**

De app gebruikt **Role-Based Access Control (RBAC)**, wat betekent dat gebruikers enkel toegang krijgen tot wat ze echt nodig hebben. Elke rol heeft zijn eigen taken:

**🔹 Employee (medewerker in het veld)**

De employee voert metingen in via een formulier. Deze gegevens worden automatisch gelinkt aan de locatie van de medewerker.

**Functies:**

* Eenvoudig formulier met vragen over pH, temperatuur, troebelheid enz.
* Locatie wordt automatisch gekoppeld
* Met één klik worden de gegevens opgeslagen
* Bevestiging of foutmelding verschijnt meteen

Na het indienen wordt de data gecontroleerd. Als alles klopt, wordt er automatisch een rapport in PDF gemaakt voor de manager.

Afbeelding met tekst, software, schermopname, ontwerp

Door AI gegenereerde inhoud is mogelijk onjuist.

Afbeelding met tekst, schermopname, Lettertype, nummer

Door AI gegenereerde inhoud is mogelijk onjuist. Afbeelding met tekst, schermopname, nummer, Lettertype

Door AI gegenereerde inhoud is mogelijk onjuist.

**🔹 Manager**

De manager ziet de binnengekomen meetgegevens en keurt ze goed.

**Functies:**

* Dashboard met overzicht van ontvangen gegevens
* Kan PDF-rapporten bekijken
* Keurt metingen goed met één klik
* Rapport wordt dan automatisch via e-mail verzonden
* Kan goedgekeurde rapporten en historiek bekijken

Afbeelding met tekst, software, schermopname

Door AI gegenereerde inhoud is mogelijk onjuist.

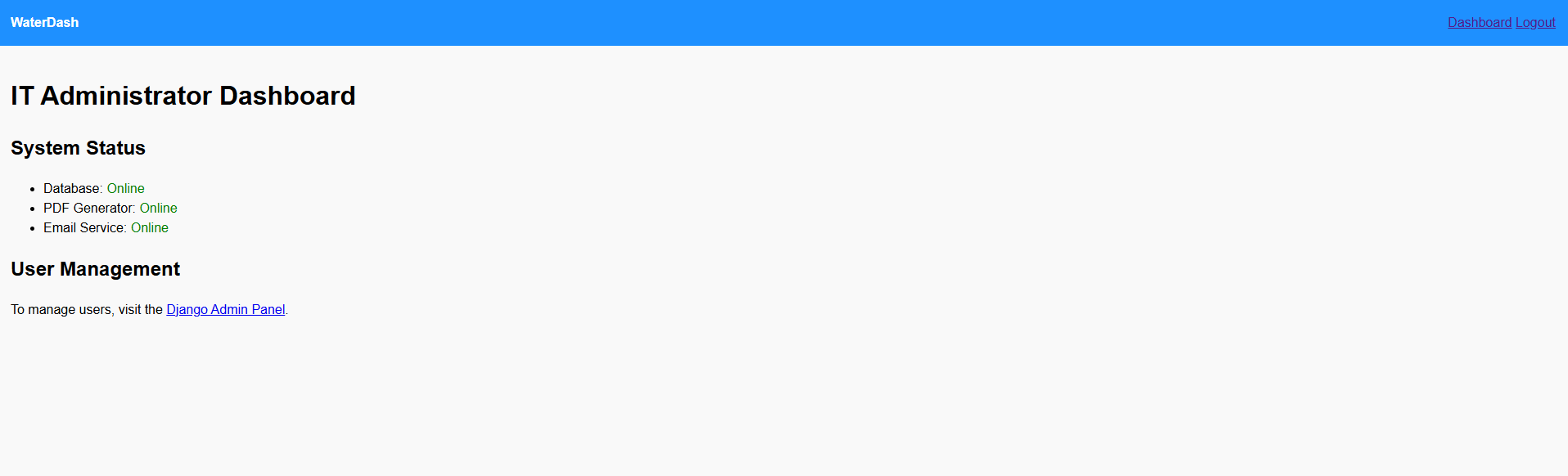
**🔹 IT Administrator**

De IT-beheerder zorgt ervoor dat alles technisch goed werkt.

**Functies:**

* Ziet de status van de database, rapportgenerator en e-mailverzending
* Kan gebruikers aanmaken, aanpassen of verwijderen
* Bepaalt wie wat mag doen
* Kan foutmeldingen en logboeken bekijken

**Beveiliging:**  
De app maakt gebruik van veilige login en sessies. Alleen bevoegde gebruikers kunnen gevoelige acties uitvoeren zoals goedkeuring of gebruikersbeheer.



Afbeelding met schermopname, software, tekst, Multimediasoftware

Door AI gegenereerde inhoud is mogelijk onjuist.

**3.3 Technische werking**

De backend is gebouwd met Django. Elke gebruiker krijgt een rol toegewezen, en afhankelijk daarvan wordt het juiste dashboard geladen.

De gegevens worden opgeslagen in een PostgreSQL-database. De tabellen bevatten onder andere gebruikers, metingen, locaties en rapporten.

De workflow:

1. De medewerker voert data in
2. De gegevens worden gecontroleerd en opgeslagen
3. Bij goedkeuring door de manager wordt een PDF-rapport aangemaakt
4. Het rapport wordt automatisch via e-mail verzonden

**3.4 Rapporten en e-mail**

De PDF-rapporten worden gemaakt met **ReportLab**. In de afbeelding hier onder staan de ingevoerde waarden en informatie over de gebruiker en locatie.

Voorbeeld:

Afbeelding met tekst, schermopname, Lettertype, nummer

Door AI gegenereerde inhoud is mogelijk onjuist.

Na goedkeuring wordt het rapport als bijlage verstuurd via e-mail. Dit gebeurt automatisch met Django’s e-mailfunctie.

Afbeelding met tekst, schermopname, software, Multimediasoftware

Door AI gegenereerde inhoud is mogelijk onjuist.

**3.5 Frontend en PWA**

De voorkant van de app is gebouwd met Django-templates, HTML en CSS . De app werkt goed op zowel computers als mobiele apparaten, en kan ook offline functioneren dankzij de PWA-instellingen.

**Belangrijke eigenschappen:**

* Mobielvriendelijk ontwerp
* Snelle laadtijden

**3.6 Structuur van de code**

* **Gebruikte tools:** Django, ReportLab
* **Database:** PostgreSQL
* **Beveiliging:** Gebruikerslogin met rechten per rol
* **Opbouw:** Aparte onderdelen voor inloggen, metingen, rapporten
* **Ontwikkeling:** Lokaal ontwikkeld, klaar voor hosting

# Besluit

Tijdens mijn stage heb ik gewerkt aan een Progressive Web App (PWA) om waterkwaliteit te monitoren. Het doel was om een systeem te bouwen waarmee medewerkers metingen konden invoeren, waarna er automatisch een rapport werd aangemaakt en doorgestuurd naar het agentschap in Portugal. Dat systeem is intussen klaar als een werkend proefproject (proof of concept), en de belangrijkste doelen zijn daarmee bereikt.

Ik ben best tevreden met het eindresultaat, omdat het systeem goed werkt en de basisfuncties aanwezig zijn. Toch had ik graag nog wat meer gedaan. In het begin van mijn stage was het project nog niet helemaal duidelijk, waardoor ik later van start kon gaan. Ook verliep de communicatie met de opdrachtgever in Portugal wat moeilijk, vooral omdat hij geen Engels sprak en niet alles technisch begreep. Daardoor kreeg ik niet altijd snel genoeg duidelijke feedback of opvolging.

Wat wel heel goed ging, was het leren van nieuwe dingen. Ik had nog nooit met Django of PWA gewerkt, maar ik heb in korte tijd veel bijgeleerd en heb die kennis direct kunnen toepassen in een echte toepassing. Dat vond ik erg waardevol.

Als ik dit project opnieuw zou doen, zou ik er eerst voor zorgen dat het project meteen duidelijk is. Dan kan je vanaf het begin gericht werken en meer uit je tijd halen.

Voor de toekomst zijn er zeker mogelijkheden om het systeem verder te verbeteren. Bijvoorbeeld:

* Het design van de app kan gebruiksvriendelijker en aantrekkelijker gemaakt worden.
* Het zou handig zijn als het systeem zelf al controleert of de ingevoerde meetwaarden goed of slecht zijn, door ze te vergelijken met verwachte richtwaarden.
* Ook kan er gedacht worden aan extra functies, zoals grafieken, filters of analyses om trends in waterkwaliteit te ontdekken.

Het systeem dat ik nu gebouwd heb, vormt een goede basis waarop het stagebedrijf volgend jaar eventueel verder kan bouwen. Het project is technisch klaar voor uitbreiding en heeft aangetoond dat het idee werkt.

Literatuurlijst

Django Project. (z.d.). *Django documentation*. <https://docs.djangoproject.com/>

ReportLab. (z.d.). *ReportLab User Guide*. <https://www.reportlab.com/docs/reportlab-userguide.pdf>

Google Developers. (z.d.). *What is a Progressive Web App (PWA)*. <https://developer.chrome.com/docs/web/progressive-web-apps/>